TU Clausthal

INVESTIGATING THE EFFECTS OF NON-UNIFORM INPUT DATA WINDOWING ON ELECTRICAL LOAD DISAGGREGATION PERFORMANCE

MASTER'S THESIS

presented by

ABDUL RAHMAN HAKMEH

Energy Informatics group
Department of Informatics
Technische Universität Clausthal

EI-M023

Abdul Rahman Hakmeh: *Investigating the effects of non-uniform input data windowing on electrical load disaggregation performance*

STUDENT ID
462491

ASSESSORS
First assessor: Priv.-Doz. Dr.-Ing. habil. Andreas Reinhardt
Second assessor: Prof. Dr. Steffen Herbold

DATE OF SUBMISSION
March 25, 2022

EIDESSTATTLICHE VERSICHERUNG

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, wurden als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsstelle vorgelegt.

Clausthal-Zellerfeld, den 25. März 2022

Abdul Rahman Hakmeh

PUBLICATION AGREEMENT

Ich erkläre mich mit der öffentlichen Bereitstellung meiner Master's Thesis in der Instituts- und/oder Universitätsbibliothek nicht einverstanden.

Clausthal-Zellerfeld, den 25. März 2022

Abdul Rahman Hakmeh

ABSTRACT

Non-intrusive Load Monitoring (NILM) is an application of smart meters to decompose the aggregated electrical load consumption into individual appliance profiles. State-of-the-art NILM methods based on neural networks showed remarkable disaggregation accuracy results. Internally, the network input data are sampled based on a windowing technique with a fixed window length limiting the number of input data, affecting the disaggregation performance. In addition, the use of different window length configurations per device increases the complexity of the model.

In this thesis, a windowing method is proposed using nonlinear algebraic functions to capture the input data points. The goal is to investigate the effect of expanding the time interval of the input window on load disaggregation without influencing the model complexity. Non-equidistant temporal sampling technique splits the traditional window into two sequences according to a fraction: a high-resolution sequence and an non-equally spaced sequence generated by means of non-linear algebraic functions, allowing the input window to contain more historical information without having to adjust the window length parameter. For evaluating the proposed technique, different variants of non-equidistant temporal sampling are suggested, the inverse and the linear downsampling function. The inverse generates an entirely non-equally spaced window, and the linear function creates a fraction of the window with equally time-spaced samples. Both functions cover the same effective window length as the proposed method.

The results confirm a positive impact of aggregating the historical information on load disaggregation. The promising potential of the proposed technique shown during the evaluation can make a step forward in the Non-Intrusive Load Monitoring (NILM) field, as collecting historical input data according to a non-equidistant or linear manner ensure an improvement in the performance without affecting the model complexity.

ZUSAMMENFASSUNG

Nonintrusive Load Monitoring (NILM) ist eine Methode zur Zerlegung von aggregierten Lastsignalen in individuelle Verbrauchsprofile. Moderne NILM-Methoden, die auf neuronalen Netzen basieren, haben bemerkenswerte Ergebnisse bei der Disaggregation der Energiedaten gezeigt. Intern werden die Netz-Eingangsdaten auf der Grundlage einer Fenstertechnik mit einer festen Fensterlänge abgetastet, wodurch die Anzahl von den Datenpunkten begrenzt wird, was die Disaggregationsleistung beeinträchtigt. Darüber hinaus erhöht die Verwendung unterschiedlicher Fensterlängen pro Gerät die Komplexität des Modells.

In dieser Arbeit ist eine Fensterungstechnik vorgeschlagen, die nichtlineare algebraische Funktionen zur Erfassung der Datenpunkte verwendet. Ziel ist es, die Auswirkung der Erweiterung des Zeitintervalls des Eingangsfensters auf die Disaggregation der Energiedaten zu untersuchen, ohne die Modellkomplexität zu beeinflussen. Diese Technik teilt das Fenster in zwei Sequenzen entsprechend einem vordefinierten Parameter auf: eine hochauflösende Sequenz und eine nicht-äquidistante zeitlich abgetastete Sequenz, die mit Hilfe nichtlinearer algebraischer Funktionen erzeugt wird, so dass das Eingabefenster mehr historische Informationen enthalten kann, ohne dass der Parameter für die Fensterlänge angepasst werden muss. Zur Bewertung der vorgeschlagenen Technik werden zwei Varianten der nichtäquidistanten zeitlichen Abtastung vorgeschlagen, die inverse und die lineare Downsampling Funktionen. Die inverse Funktion erzeugt ein völlig äquidistantes Fenster und die lineare Funktion erzeugt einen Bruchteil des Fensters mit gleichmäßig verteilten Samples. Beide Funktionen decken die gleiche effektive Fensterlänge ab wie die vorgeschlagene Methode.

Die Ergebnisse bestätigen eine positive Auswirkung der Aggregation von historischen Informationen auf die Disaggregation der Energiedaten. Das versprechende Potenzial der vorgeschlagenen Technik, das während der Evaluierung aufgezeigt wurde, kann einen Schritt nach vorne im Bereich NILM machen, da das Sammeln historischer Eingangsdaten auf eine uneinheitliche oder lineare Weise eine Verbesserung der Leistung gewährleistet, ohne die Modellkomplexität zu beeinträchtigen.

ACKNOWLEDGMENTS

During the preparation of this thesis, I have received valuable help and support. Therefore, I would like to thank my thesis supervisor, Prof. Andreas Reinhardt, for giving me the opportunity to join his research group and work on this thesis under his supervision, besides the insightful recommendations he provided regarding my research and during writing this thesis.

Finally, I would like to thank my family wholeheartedly for their unwavering encouragement and support during my years of study.

CONTENTS

1 INTRODUCTION 1
1.1 Motivation . . . 3
1.2 Aim and Scope . . . 4
1.3 Thesis Outline . . . 4

2 BACKGROUND 7
2.1 Electrical Engineering . . . 7
2.1.1 Basic Concepts of Electricity . . . 7
2.1.2 Appliance Monitoring . . . 9
2.1.3 Frequency Domain Analysis . . . 10
2.1.4 Window Functions in Signal Processing . . . 11
2.2 Time Series Data Analysis . . . 12
2.2.1 Characterizations . . . 13
2.2.2 Sampling Rate . . . 13
2.3 Deep Neural Networks . . . 14
2.3.1 Input window Length . . . 14
2.3.2 Learning Models . . . 15
2.4 Non-Intrusive Load Monitoring . . . 16
2.4.1 Evaluation Metrics . . . 17
2.4.2 Non-Intrusive Load Monitoring Toolkit . . . 19
2.5 Household Energy Datasets . . . 19

3 LITERATURE REVIEW 23
3.1 Window Length Parameter Selection . . . 23
3.2 Related Work to Data Preprocessing . . . 25

4 PRELIMINARY STUDY 27
4.1 Input Dataset . . . 27
4.2 Input Data Scaling and Downsampling . . . 28
4.3 Window Technique Analysis . . . 29
4.3.1 Window Functions . . . 30
4.3.2 Symmetric Window . . . 32
4.3.3 Asymmetric Window . . . 33
4.4 Study Results . . . 34
4.5 Summary . . . 36

5 METHODOLOGY 37
5.1 Contribution Overview . . . 37
5.2 Non-linear Time Spacing Functions . . . 38
5.2.1 Quadratic Spacing . . . 38

5.2.2 Exponential Spacing . . . 39
5.3 Preprocessing Input Data . . . 41
5.3.1 Windowing Parameterization . . . 41
5.3.2 Non-equidistant Sampling . . . 43
5.3.3 Data Sampling Functions . . . 45
5.4 Non-equidistant Sampling Variants . . . 46
5.4.1 Inverse Function . . . 47
5.4.2 Linear Downsampling Function . . . 48
5.5 Experiment Design . . . 48
5.5.1 Environment Setup . . . 49
5.5.2 Model Parameters . . . 50

6 RESULTS AND EVALUATION 53
6.1 Define Results Margin . . . 53
6.2 Sequence-to-Sequence Results . . . 55
6.2.1 Quadratic Function . . . 55
6.2.2 Exponential Function . . . 58
6.3 Sequence-to-Point Results . . . 59
6.3.1 Quadratic Function . . . 59
6.3.2 Exponential Function . . . 60
6.4 Summary . . . 61

7 CONCLUSION AND FUTURE WORK 63
7.1 Conclusion . . . 63
7.2 Future Work . . . 64

BIBLIOGRAPHY 65

A SEQUENCE-TO-SEQUENCE RESULTS 69
A.1 Quadratic Function . . . 69
A.2 Exponential Function . . . 74

B SEQUENCE-TO-POINT RESULTS 79
B.1 Quadratic Function . . . 79
B.2 Exponential Function . . . 85

LIST OF FIGURES

Figure 1.1 Greenhouse gases emission covered by the UN framework [GEA] . . . 1
Figure 1.2 Household electricity savings by feedback type [EMDL+10] . . . 2
Figure 2.1 Sine wave presented by voltage cycles showing signal properties . . . 8
Figure 2.2 Schematic diagram of appliances and smart meter distribution comparison between submetering monitoring and NILM . . . 10
Figure 2.3 Practical example of spectral leakage phenomenon [KJ09] . . . 12
Figure 2.4 Single neuron in Recurrent Neural Network (RNN) structure and its unfold version [GBC16] . 15
Figure 4.1 A practical example of appliance's operation cycles extracted using Non-Intrusive Load Monitoring Toolkit (NILMTK) . . . 29
Figure 4.2 The Hamming and the Hann window functions 31
Figure 4.3 The Blackman-Harris and the Flat Top window functions . . . 31
Figure 4.4 A practical example elucidates applying symmetric window functions . . . 32
Figure 4.5 An example of applying asymmetric window functions . . . 33
Figure 4.6 Results of applying symmetric window functions for microwave . . . 34
Figure 4.7 Results of applying asymmetric window functions for microwave . . . 35
Figure 5.1 Sampling the data points based on quadratic functions elucidates the ability to collect historical information compared to the traditional uniform windowing t^1 . . . 39
Figure 5.2 Sampling the data points based on exponential functions . . . 40
Figure 5.3 The results of initial experiments show performance degradation when using a history aggregation factor (HAF) greater than 0.15 . . . 42
Figure 5.4 An example of non-equidistant sampling shows differences in the effective window length when using different spacing functions . . . 45

Figure 5.5 Statistical functions are applied on a sample of mains 46
Figure 5.6 The system designed to automate the experiments execution over four GPU-machines 50
Figure 6.1 Quantile-Quantile plot indicates slight deviation of 100 NILM runs from the normal distribution (red line) 54
Figure 6.2 Normal distribution curve fitted to the histogram of 100 NILM rus 54
Figure 6.3 Comparison of different sampling functions on Sequence-to-Sequence (S2S) 55
Figure 6.4 Performance of $t^{1.2}$ function with non-equidistant techniques 56
Figure 6.5 Results summary of quadratic function on S2S; The x-axis shows the exponents in quadratic function (first line), and the corresponding exponents used in the inverse function (second line) 57
Figure 6.6 Results summary of exponential function on S2S 58
Figure 6.7 Comparison of four different sampling functions on Sequence-to-Point (S2P) 59
Figure 6.8 Results summary of quadratic function on S2P. The x-axis indicates the evaluated exponents and the corresponding exponents used in the inverse function to produce non-equally spaced window 60
Figure 6.9 Results summary of four exponential functions on S2P 61
Figure A.1 Evaluating the function $t^{1.1}$ performance with the baseline 69
Figure A.2 Evaluating the function $t^{1.3}$ performance with the baseline 70
Figure A.3 Evaluating the function $t^{1.4}$ performance with the baseline 70
Figure A.4 Evaluating the function $t^{1.5}$ performance with the baseline 71
Figure A.5 Evaluating the function $t^{1.6}$ performance with the baseline 71
Figure A.6 Evaluating the function $t^{1.7}$ performance with the baseline 72
Figure A.7 Evaluating the function $t^{1.8}$ performance with the baseline 72
Figure A.8 Evaluating the function $t^{1.9}$ performance with the baseline 73

Figure A.9 Evaluating the function $t^{2.0}$ performance with the baseline . . . 73
Figure A.10 Evaluating the function 1.1^t performance with the baseline . . . 74
Figure A.11 Evaluating the function 2^t performance with the baseline . . . 74
Figure A.12 Evaluating the function 1.2^t performance with the baseline . . . 75
Figure A.13 Evaluating the function 1.3^t performance with the baseline . . . 75
Figure A.14 Evaluating the function 1.4^t performance with the baseline . . . 76
Figure A.15 Evaluating the function 1.5^t performance with the baseline . . . 76
Figure A.16 Evaluating the function 1.6^t performance with the baseline . . . 77
Figure A.17 Evaluating the function 1.7^t performance with the baseline . . . 77
Figure A.18 Evaluating the function 1.8^t performance with the baseline . . . 78
Figure A.19 Evaluating the function 1.9^t performance with the baseline . . . 78
Figure B.1 Evaluating the function $t^{1.1}$ performance with the baseline . . . 79
Figure B.2 Evaluating the function $t^{1.2}$ performance with the baseline . . . 80
Figure B.3 Evaluating the function $t^{1.3}$ performance with the baseline . . . 80
Figure B.4 Evaluating the function $t^{1.4}$ performance with the baseline . . . 81
Figure B.5 Evaluating the function $t^{1.5}$ performance with the baseline . . . 81
Figure B.6 Evaluating the function $t^{1.6}$ performance with the baseline . . . 82
Figure B.7 Evaluating the function $t^{1.7}$ performance with the baseline . . . 82
Figure B.8 Evaluating the function $t^{1.8}$ performance with the baseline . . . 83
Figure B.9 Evaluating the function $t^{1.9}$ performance with the baseline . . . 83
Figure B.10 Evaluating the function t^2 performance with the baseline . . . 84
Figure B.11 Evaluating the function 1.1^t performance with the baseline . . . 85

Figure B.12 Evaluating the function 1.3^t performance with the baseline 86
Figure B.13 Evaluating the function 1.4^t performance with the baseline 86

ACRONYMS

BSS	Blind Source Separation
NILM	Non-Intrusive Load Monitoring
ANN	Artificial Neural Networks
GPU	Graphics Processing Unit
S2P	Sequence-to-Point
S2S	Sequence-to-Sequence
FFT	Fast Fourier Transform
UK-DALE	United Kingdom Domestic Appliance-Level Electricity
RNN	Recurrent Neural Network
CNN	Convolution Neural Network
S2SS	Sequence-to-Short-Sequence
DRED	Dutch Residential Energy Dataset
EMF	Electromotive Force
VA	Volt-Ampere
Hz	Hertz
DC	Direct Current
FT	Fourier Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
NILMTK	Non-Intrusive Load Monitoring Toolkit
CO	Combinatorial Optimization
HMM	Hidden Markov Model
MDF	Multirate Digital Filtering
REDD	Reference Energy Disaggregation Dataset
kHz	kilohertz
DRED	Dutch Residential Energy Dataset
REFIT	Personalised Retrofit Decision Support Tools For UK Homes Using Smart Home Technology
CNN	Convolutional Neural Network
HMM	Hidden Markov Model
RMSE	Root Mean Square Error
NAPE	Normalized Allocated Power Error

1

INTRODUCTION

In recent decades, the attention to energy consumption saving has expanded. The focus is no longer solely on the finite nature of fossil fuels. It is also about the environmental consequences of producing million tons of carbon dioxide, which is the main reason behind climate change and global warming. Many countries around the world have already started investing in technologies to develop energy efficiency concepts in parallel with the clean energy transition. Words like sustainability and renewable energy shape the energy market in the modern societies.

Germany - as an example - has set itself the goal of significantly reducing CO_2 production by 2030. Figure 1.1 provides an overview of the development of emissions in Germany since 1990 and the planned reduction. It shows that Germany has already achieved a reduction of about 40 % compared to 1990. Another fact that emerges from the statistics is that households have a share in emissions production. This highlights the need to develop special methods for households to ensure energy efficiency, which must go hand in hand with the shift to clean resources in order to achieve a significant improvement in CO_2 reduction.

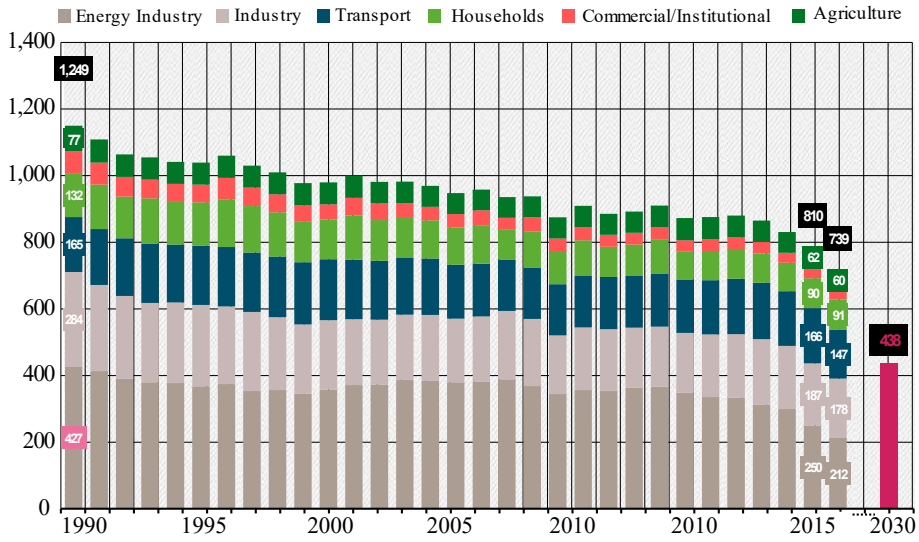


Figure 1.1: Greenhouse gases emission covered by the UN framework [GEA]

In this context, the use of smart meters has opened the door to potential energy savings by using a central sensor that can report the aggregate electricity consumption of a household. It emerged as an additional use case beyond the capability to collect meter readings digitally. The use of smart meters enables additional demand-side management and on-site monitoring. These concepts mean, on one hand, keeping track of power grids and, on the other hand, providing detailed information about end-user consumption, such as the available power supply and demand must line up. In addition, on-site monitoring offers customers the opportunity to reduce their energy consumption when they receive detailed information about it. The flexible loads is important to make supply and demand meet. However, the smart electricity meter can only provide information over the total consumption during certain periods, such as daily or weekly. Unfortunately, this method is not sufficient to improve energy savings. Several studies in the same areas have shown that optimal energy savings can be achieved when real-time, appliance-level information is provided to the customer [EMDL+10]. Figure 1.2 shows the improvement of annual percent savings from 3.8 % to 12 % when real-time information plus feedback down to the appliance-level is provided.

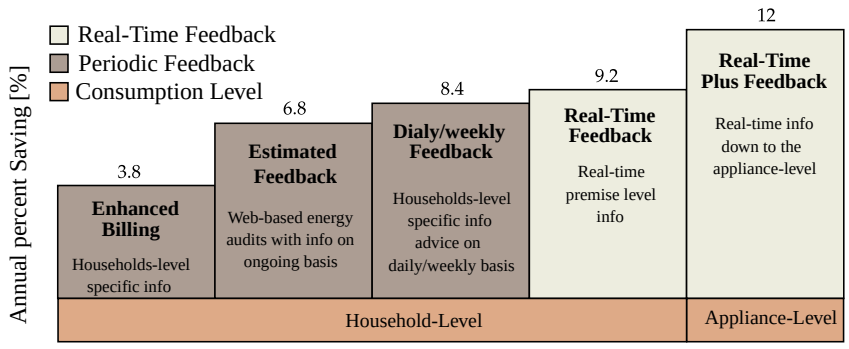


Figure 1.2: Household electricity savings by feedback type [EMDL+10]

Thus, new approaches are needed that focus on decomposing aggregated data to achieve optimal energy savings. This task of separating a signal from mixed channels based on the source is a common problem in numerous scientific fields such as signal and image processing, referred to as Blind Source Separation (BSS). To tackle this problem, NILM was invented in the early 1980s. NILM defined a methodology to break down aggregated consumption signals from various devices into individual consumption profiles. It can be used to engage the end user in the energy management plan by providing detailed device-level information. This enables the analysis of run

times and identification of energy-hungry devices, improving energy consumption efficiency.

1.1 MOTIVATION

Since the invention of NILM, numerous scientific publications have methodically and technically improved the concept. Nowadays, contributions focus on conventional algorithms based on Deep Learning with Artificial Neural Networks (ANN). Besides their dominant performance, the development of Graphics Processing Unit (GPU) hardware has significantly accelerated the computation, which has brought more and more attention to the above methods. Recently, and due to their remarkable performance in disaggregation accuracy, S2S and S2P represent the stat-of-art NILM [RK20].

Approaches based on neural networks usually rely on the preparation of data before it is fed into the network the so-called preprocessing of data. The process consists of several phases, such as filtering the data by removing the outliers, handling the non-values, defining the number of input samples to be fed into the network (window length) and splitting the dataset into training and test data, besides setting the configuration parameters (preprocessing and neural network hypoparameters). One of these parameters is crucial in this work, namely the window length. This parameter limits the number of data points in the window affecting the disaggregation accuracy. This effect is confirmed in a previous study [RB20], where the investigation showed an impact of the parameter variations on the results using S2S and S2P learning model.

Two techniques for setting the window length parameter are observed in the literature. The one-for-all technique, where a fixed value is assigned and only one network is trained to predict the operating conditions of all devices. Another technique is the per-device technique, in which operational duration of the targeted device is averaged and assigned to the parameter. In this case, a separate network must be trained per target appliance. Recent research has empirically demonstrated that the device-based design increases disaggregation accuracy [RB20]. It turns out that the optimal values have nothing to do with the operational duration. The study reported the optimal value by conducting a comparative examination of F1-scores resulting from performing experiments on real-world data (DRED dataset). However, simplicity of configuration is still a challenge in using this technique. In contrast, improved disaggregation results must be sacrificed when the one-for-

all technique is used.

In this work, we encounter the dilemma of finding a middle ground between simplicity and improved results. Therefore, we introduce a novel windowing technique based on non-linear algebraic functions. In this technique, a different temporal distance between sampled data in the same window is established, resulting in an asymmetric fixed-length window that fits all devices, allowing the window to aggregate more samples from the device's operational history and make them available to the network. In addition, apply statistical functions to the non-uniformed sampled data to improve the representation of the the corresponding device events.

1.2 AIM AND SCOPE

In this thesis, the effect of expanding the time interval of the input window on load disaggregation is investigated without influencing the model complexity. Firstly, a preliminary study is conducted to measure the impact of symmetric and asymmetric windows on disaggregation results while maintaining simplicity by using one-for-all strategy for setting the parameter. According to the methodology used in Fast Fourier Transform (FFT) for resolving the problem of having discontinued waveform, numerous window functions are applied to the network input to exclude additional measurements from the learning process and maintain a fixed window length for all appliances. Thereafter, a windowing technique is proposed, in which the sampled window is partially divided into uniform and non-uniform sampled sequences. The non-linearity is generated by two algebraic functions, one exponential and one quadratic. This allows the windowing to include historical data points beyond the time boundary of the traditional window. The goal is to find a fixed window length that ensures improved overall disaggregation accuracy.

The NILMTK for comparing disaggregation algorithms is used for perform experiments, as it provides the implementation for the S2S and S2P optimization methods used to evaluate the approach proposed above.

1.3 THESIS OUTLINE

Based on the previously stated research objective, this thesis is organized as follows. In chapter 2, the basic knowledge for this work is presented by introducing fundamental electrical quantities related to

our work, followed by a brief overview of the characteristics of time series and deep neural networks. Later, a detailed insight into the NILM approach is given, introducing disaggregation methods and the Non-Intrusive Load Monitoring Toolkit (NILMTK). Then, an overview of the state of the art and window length selection methods is provided to highlight the research gap into which our work should fit.

Previous contributions in the field of NILM are mentioned in chapter 3 to define the state of the art and highlight the research gap, where our contribution should fit.

In chapter 4, a preliminary study is conducted to analyze the symmetric and asymmetric window functions and their effects on disaggregation accuracy when applied to network input. Based on the results of this study, the investigation is inversed from the direction of hiding samples to the direction of aggregating data points into the input window.

Chapter 5 handles detailed information on the proposed non-equidistant temporal sampling using the quadratic and the exponential functions, followed by an explanation of the comparison between different variants of non-equidistant sampling, including the inverse and the uniform downsapling functions. Furthermore, the settings for the experiments, such as the energy dataset used and the environment parameters selection, are mentioned.

The results of the experiments, the corresponding analysis of the parameterization impact on the disaggregation accuracy, and the overall performance evaluation are discussed in chapter 6.

A summary of our work containing the conclusion and an outlook on prospects to expand this research in the future is covered in chapter 7.

2 BACKGROUND

After introducing the motivation and the goal of this study in the previous chapter, the reader should familiarize themselves with the terms and methods mentioned before delving deeply into the proposed methodology. Therefore, this chapter delivers the background of the work by explaining the concept of NILM giving an overview of the methods used to disaggregate the load consumption and the commonly used energy real-world datasets.

2.1 ELECTRICAL ENGINEERING

Electrical engineering is one of the branches of engineering whose concerned origins date back to the nineteenth century. This branch evolved from the development of electrical circuits for electricity and telephony to various fields of study. Originally, electricity was used for lighting systems and, to a lesser extent, to run loads such as electric motors. Later, more industries began to use electricity to run their businesses instead of using oil. This increased demand for electricity has promoted the need to measure electricity consumption. Nowadays, research is being accomplished in the field of electrical signal processing to efficiently process and transmit data. These data come from various sources such as sounds and measurements by electrical sensors [RK09].

In this section, load monitoring is presented as the driving force for the invention of NILM and its advantages in energy saving, followed by an overview of frequency domain analysis, which is the basis of the coming chapter. Before that, some basic terminologies of electricity are explained.

2.1.1 *Basic Concepts of Electricity*

In any subject related to electricity, it is necessary to talk about the basic quantities that make up electrical signals used to transmit energy or transfer information: current and voltage.

Voltage (V), it refers to the amount of energy required to move one unit (*volt*) from the more negative origin to the more positive destination. It is also defined as Electromotive Force (EMF). Practically, voltage

is what the circuit user applies to push current to flow [HHR89].

Current (I), it is defined as the movement of electrons through an object, such as a wire, driven by an electrical pressure and measured in *amperes* or A. Frequency, measured in Hertz (Hz), is the rate at which the direction of the current changes. Although the flow of current in a circuit is considered to be a movement from a more positive point to a more negative point, the actual flow of electrons is in the opposite direction, unlike Direct Current (DC) which flows in the same direction [HHR89].

Active power is the full watt power consumed for different purposes, such as mechanical work (operate motors). In contrast, Reactive Power (Q) keeps bouncing uselessly back and forth between the load and the energy source (battery). This movements can be useful in some applications, such as generating magnetic fields in motors.To express the total power in a circuit (active and reactive power), the term apparent power (S) is used. The apparent power is composed of the dissipated and absorbed power measured in Volt-Ampere (VA) [HHR89].

Most types of information can be presented by one of the fundamental quantities in electrical engineering: the signal. There are two types of signals in electrical engineering; Analog signals with continuous values and digital signals where discrete values represent the signal. The independent variable of the signal can be time, as in signals generated by speech or space, as in images [RK09]. The most vital signal in elec-

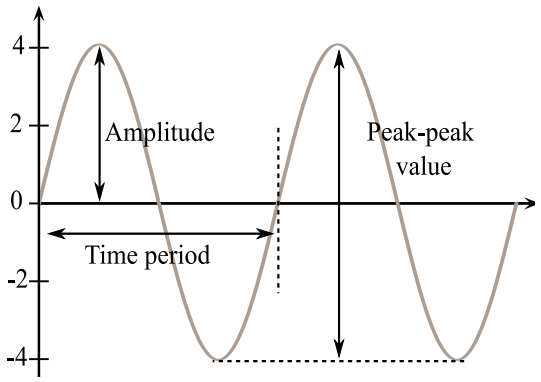


Figure 2.1: Sine wave presented by voltage cycles showing signal properties

trical engineering is the sine wave, also referred to as *sinusoid*. signals in electrical have different properties that are commonly mentioned in the signal analysis domain. *Amplitude* or peak voltage is the maximum voltage that the signal reaches and is measured in *volts* (for voltage signals). *Period duration* is the time it takes a signal to complete a cy-

cle [RK09]. Figure 2.1 illustrates these properties. Note that the above properties apply to any signal with a continuously repeating shape.

2.1.2 *Appliance Monitoring*

The idea of appliance monitoring is driven by the fact, that when users receive feedback on how their habits affect the energy consumption, they can take the first step toward energy-saving behavior by learning and adapting when they are informed about how consumption statistics respond to their actions [EMDL+10].

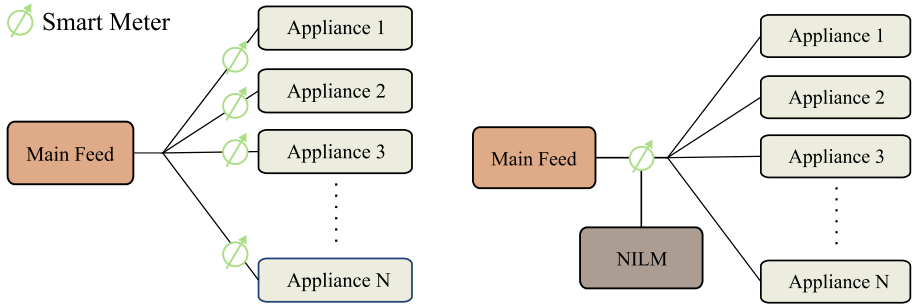
To explain the importance of detailed feedback on energy consumption and how it makes intuitive sense, we give an real-life example: The receipt that one usually receives in supermarkets after shopping encloses the total amount, broken down into the cost of each item. This information gives the customer more insight into the possible savings they can make. Utilities and energy providers, however, typically provide consumers with only a short summary of the energy consumed during a given period, without any details. This deprives consumers of important information, such as appliance runtime analysis, which could help the used users to optimize the runtime of the appliances that consume the most energy.

There are different types of feedback that offer the end-user consumption transparency, but the most relevant are *direct* and *indirect feedback* [EMDL+10]. The first is to provide feedback in real time, where the consumer can read the energy meters through a monitor or any terminal device, so that the feedback is provided on demand. Despite its effectiveness, this type of feedback is still a challenge. Developing and installing such systems in households is a time-consuming task with high associated costs. In addition, providing complex information in real-time might be difficult for some people to understand.

In contrast to direct feedback, the indirect option is to provide regular feedback in the form of invoices with disaggregated information on equipment consumption generated by the energy provider. This approach could reduce complexity and cost, but is still not immediately available. The study mentioned in chapter 1, which examined different types of feedback, shows that direct feedback results in savings of 12 %, while indirect feedback results in savings of only 8 % [EMDL+10].

Two main alternatives are available for measuring energy consumption at the device level; NILM and the sub-meters. The use of sub-meters offers a decentralized solution with a separate sensor. Each device in

the system is connected to a sub-meter, and the entire system depends on the functionality of all sub-meters. [Liu19].



(a) Sub-metering monitoring (b) Non-Intrusive Load Monitoring

Figure 2.2: Schematic diagram of appliances and smart meter distribution comparison between sub-metering monitoring and NILM

NILM is the more cost-effective centralized solution because it uses a single sensor connected to all devices and the breakdown of consumption is done using various disaggregation methods. Moreover, This procedure does not need to intervene in the homes of consumers to measure the energy consumption of various devices [Liu19]. Both structures are schematically shown in figure 2.2.

2.1.3 *Frequency Domain Analysis*

Electrical signals are usually analyzed by plotting the signal in the time or frequency domain. The time domain captures the current and voltage as a function that varies with time. In the frequency domain, time is ignored, and the signals are conveyed in terms of frequency instead of time. The frequency domain is powerful for analyzing linear systems because it simplifies the mathematical formulation and helps determine the correction values resulted by noise, because noise is simply an undesired frequency [RK09]. The transformation of signals into the frequency domain uses different transformation methods. Laplace, Wavelet, and the Fourier transformation. In this section, we concentrate on the last method, as it represents the basics of the preliminary study later in chapter 4.

Fourier Transform (FT) is one of the most adopted methods in signal processing as it transforms the signal from the time domain to the frequency domain, where the harmonics of the signal, such as phases and amplitudes, become visible. In this transformation, each signal is decomposed into a sum of sinusoidal functions; each

of these functions has a different frequency [BB86]. The following equation (2.1) emphasizes the mathematical formulation of FT as an integrable function $f : \mathbb{R} \to \mathbb{C}$, where ω represents frequency and $i = \sqrt{-1}$. The equation (2.2) is the inverse transform with t representing the time [BB86].

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i t \omega}\, dt \qquad \forall\, \omega \in \mathbb{R} \tag{2.1}$$

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\omega)\, e^{2\pi i t \omega}\, d\omega \qquad \forall\, t \in \mathbb{R} \tag{2.2}$$

In both equations the complex exponential $e^{i\theta}$ is defined by:

$$e^{i\theta} = \cos\theta + i \sin\theta \tag{2.3}$$

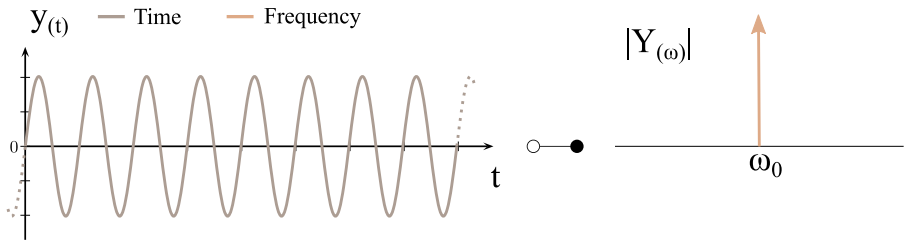
The FT converts a signal in the time domain into a continuous spectrum consisting of multiple sinusoids. When analyzing signals with a discrete number of samples, a finite number of sinusoids is required; in these cases, the Discrete Fourier Transform (DFT) can be used. The FFT is an optimized application of the DFT, which requires less computational effort [BB86].

2.1.4 *Window Functions in Signal Processing*

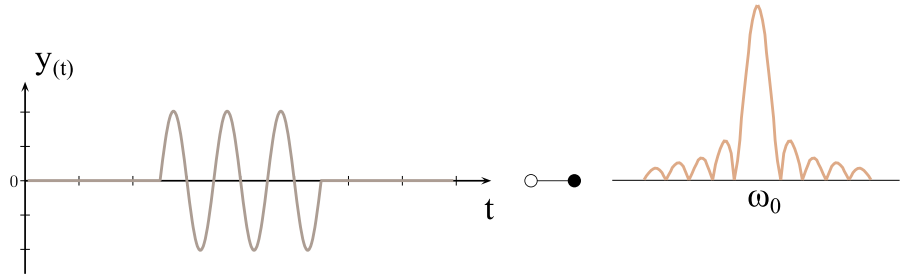
Applying the FFT helps troubleshoot signals by removing undesired frequencies. However, there are limitations. In this subsection, we present these limitations and show how window functions can be used improve the clarity of the signal in FFT [KJ09]. Mentioning Fourier transformation in the next paragraph is motivated by the fact that the preliminary study adopted the same methodology used in FFT to merge discontinued waveform before converting the signal into the frequency domain.

The FFT assumes that the used data is a finite quantity, a complete continuous waveform representing one period of a periodic signal. If no integer number of periods can be found within the signal, the finite nature of the signal under study can result in a truncated waveform called aliasing. Aliasing appear after application of the FFT as high-frequency elements. which are not present in the original signal, so that the displayed spectrum does not correspond to the actual spectrum of the original signal. This phenomenon is anointed as spectral leakage [KJ09]. The Figure 2.3 illustrates the phenomenon by showing theoretical case of an infinitely long sine and the corresponding

transformation in the frequency domain, where $Y(\omega)$ represents the Fourier transforms of $y(t)$. Note that the displayed spectrum in frequency domain (b) has nothing to do with the actual spectrum (a).



(a) Theoretical case of an infinitely long sine wave in time domain resulting in absolute value of the spectrum after transformation into frequency domain



(b) Non-integer number of periods produce a spectrum that does not correspond to the actual spectrum of the original signal

Figure 2.3: Practical example of spectral leakage phenomenon [KJ09]

The transformation of a signal with a non-integer number of periods into the frequency domain using FT presents a technical problem that can be solved by means of window functions. Applying these windows on the signal decreases the amplitude, where discontinuities occur. When applying window functions the sequence representing the signal under study is multiplied by a window function of the same length. The amplitude of this window goes toward zero at both sides of the window, which can merge the discontinuities and produce a a continuous waveform [KJ09].

2.2 TIME SERIES DATA ANALYSIS

Time series data is a collection of infinite number of data points recorded at equal time intervals. Time series analysis involves strategies for analyzing time series data to extract meaningful information and other features from the given data. Time series forecasting utilizes learning models to estimate future data based on readings collected in the past. while the regression in time series aims to test relationships

between one or more different time series channels [WML05]. This study focuses on regression analysis because it aims to predict the consumption of different appliances individually based on single time series channel that represents the total consumption of a particular household. This section provides an overview of the characterization of time series, followed by an explanation of sample rate in time series.

2.2.1 *Characterizations*

A time series can be explained as a collection of data points obtained by repeated measurements over a period. Time series can be be mathematically defined as a timestamp vector t_i and associated measurements x_i, which are sampled at a constant sampling period $\triangle t$ resulting an compact vector $x = (x_1, x_2, \ldots, x_N)$ where N data points have been sampled at times $t = (0, \triangle t, 2 \triangle t, \ldots, (N-1) \triangle t)$ [Ful17]. Time series exhibit a wide range of temporal patterns and different types of structures. The historical data and its patterns are considered when analyzing a time series because of the temporal relationship between data points. Therefore, the analysis is different from other types of data, besides having special properties like the distribution of values and the autocorrelation, where the values in a time series are correlated with themselves over time. Another property called stationarity measures how the statistical properties change during a record [Ful17].

2.2.2 *Sampling Rate*

The most important feature of time series, indicating how observations are distributed over time, is the sampling frequency or sampling rate. The sampling frequency is the number of uniformly distributed samples per unit time, measured in Hz, equal to $1/t$ [Ful17]. In deep learning applications, the sampling rate is considered one of the most important parameters that must be deliberately chosen in the preprocessing phase of the input data. Since the neural network in S2S and S2P takes sequences of data points as input, the parameter defines the time interval covered by the sequence, which affects the results of the network.

In general, the sampling rate can, on one hand, be increased by inserting additional samples within a time series. This method is called upsampling. On the other hand, downsampling is the elimination of intermediate samples. Both procedures belong to a bunch of filters are known as Multirate Digital Filtering (MDF) [KA72]. This group contains

numerous mechanisms for increasing or decreasing the sampling rate, e.g., downsampling by averaging and the downsampling by an integer factor [RG75].

2.3 DEEP NEURAL NETWORKS

Deep learning is a specific subcategory of machine learning. Unlike machine learning, deep learning builds computational models at multiple processing levels to learn a multi-level abstraction of data representations. These learning techniques have achieved remarkable improvements in visual object recognition, signal processing, and other areas. Artificial Neural Networks can be theoretically simulated as a directed graph to simplify its structure. The nodes in a directed graph represent the artificial neurons, the main component of the neural network, while the edges enable information to flow from neuron/node to another one. The network used in S2S and S2P usually rely on the backpropagation algorithm to discover complex data structures. This function tells the network how to tune its internal parameters during the learning process. [GBC16].

This section presents two deep learning optimization methods that have achieved remarkable results in load disaggregation, before this the input sequence length parameter is explained.

2.3.1 *Input window Length*

Neural networks expect input data to be converted into a vectorized representation. This vectorization allows the code to perform matrix operations efficiently. Creating a vector of sequences sampled from the physical measurement is one of the essential steps in the preprocessing phase of the input data. A sequence is a series of finite number of data points taken by sliding a window over the dataset. The window length parameter defines this value, limiting the number of samples made available to the neural network.

The setting of sequence length parameter must be done, taking into account the sampling rate of the input data. This relationship between the two parameters must be considered, because both parameter defines the actual duration of historical data, effecting the performance of the neural network.

2.3.2 *Learning Models*

Neural networks are a promising approach to solving the BSS problem. Utilizing deep learning application for NILM tasks was presented in 2015 by [Kel] (using Sequence-to-Sequence), the proposed model takes a non-linear regression between a sequence of vectorized readings of a single device and a sequence of data points collected from the aggregated reading, both sequences have the same time stamp. The architectures of Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) are usually used in this method [Kel; Zha+18]. RNN differ from other network structures in that they allow cycles in the network graph in which neuron n_1 at time step t in layer l reports output via a weighted edge V to every neuron in layer l, including n_1, at time step $t+1$ [Kel]. The Figure 2.4 shows the structure of a single neuron in RNN and its unfolded version (on the right), where x, o, h are the input, the output, and the hidden states respectively. The weights of the network are U, V, and W.

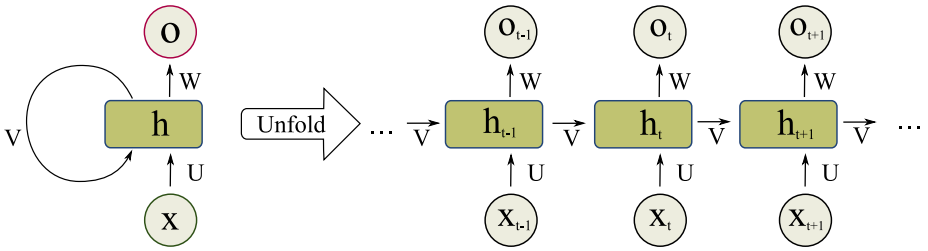


Figure 2.4: Single neuron in RNN structure and its unfold version [GBC16]

CNNs have unique layers that distinguish them from RNNs. By using a convolutional layer, the input is transformed before it is passed on to the next layer. Basically, it is like applying filters to the time series and then slide the filter over it. A multivariate time series results from applying multiple filters on the time series. Extracting multiple distinguishing features is helpful for the regression task. It is done by applying multiple filters to an input. The convolution can automatically learn the distinguishing features by using a discriminative classifier or during a pooling operation. The pooling operation works as an averaging process, where the length of a time series is reduced by an accumulation process resulting in a single real value. [IF+19].

Mathematically, S2S defines a neural network as a function F_S, where a sliding window of length W $Y_{t:t+w-1}$ from the aggregated power (mains) is mapped to a corresponding window $X_{t:t+w-1}$ from

the appliance submeter data, resulting in the model $X_{t:t+w-1} = F_S(Y_{t:t+w-1}) + \omega$, where ω represents gaussian random noise. Hence, the loss function of train the network with data pair (X,Y) is defined by equation (2.4) [Zha+18], where ψ_S are the network parameters.

$$L_S = \sum_{t=1}^{T-W+1} \log p(X_{t:t+W-1} | Y_{t:t+W-1}, \psi_S) \tag{2.4}$$

In Sequence-to-Point the network is trained to predict only the center of this window, rather than predicting a window of measurements like in S2S. Thus, the input is a window of measurements and the output is the center of the corresponding window of the target device. S2P dose not considers the entire output window but the center of it as a non-linear regression to the input window [Zha+18].

The mathematical model of S2P shows the differences to S2S, as the network F_P estimates the midpoint x_τ of the output window $X_{t:t+w-1}$ from the input window $Y_{t:t+w-1}$ resulting in the following training function $x_\tau = F_P(Y_{t:t+w-1}) + \omega$. Equation (2.5) shows the loss function, where ψ_P are the network parameters.

$$L_P = \sum_{t=1}^{T-W+1} \log p(x_\tau | Y_{t:t+W-1}, \psi_P) \tag{2.5}$$

To handle the endpoints of the full input sequence in S2P, a padding operation is performed with $[W/2]$ zeros at both sequence sides. In S2S the multiple predictions of one point are averaged [Zha+18].

2.4 NON-INTRUSIVE LOAD MONITORING

Non-Intrusive Load Monitoring is a an application of smart meter whose goal is to break down the total load measurements from single sensor and assign the broken-down signals to individual devices. Ideally, all detected devices add up to a total signal [Har92].

NILM can be presented as a Blind Source Separation (BSS) problem where more than one source must be extracted from a single observation, leading to difficulties in identification [Zha+18]. This is the same process in power disaggregation. The power system signal consists of multiple sources of uncertainty such as background power noise, numerous devices with essentially the same power consumption, and multiple devices that turn on or off simultaneously. In addition, the exact power consumption of each device in each household is unknown. Therefore, predictions must be made based on the aggregate

signal. This problem can be formulated mathematically with equation (2.6) at any time t, where $\omega(t)$ stands for unaccounted devices and noise. The equation defines the relationship between the main power consumption $X = \{X_1, X_2, \ldots, X_T\}$ and the corresponding devices $N = \{1, 2, \ldots, T\}$, so the goal is to extract the contribution y_t^i of device i at time t, where $i \in \{1, 2, \ldots, T\}$ [Fau+17].

$$X_t = \sum_{i=1}^{N} y_t^i + \omega(t) \tag{2.6}$$

In NILM the event detection is the feature extraction phase in which the collected data is processed and analyzed to identify individual device events. There are two classes of feature extraction in NILM: the steady-state signature, which analyzes the signal amplitude, and the transient feature, which represents the changes in the waveform of the stream as a unique pattern followed by a device and is used to identify the different devices [Arm+13].

The device classification and disaggregation phases are responsible for feature selection and extraction to create an energy consumption profile for each usage pattern. Based on these patterns, classification approaches can be used to identify devices by comparing observed features of a given power consumption with previously known features of a typical device in the database [Arm+13]. Later in the next section, the common disaggregation algorithms for NILM are presented.

2.4.1 *Evaluation Metrics*

In this subsection, a set of metrics [Bat+14] is introduced that combines both general detection metrics, especially those related to the electrical load disaggregation.

- ▷ Normalized Allocated Power Error (NAPE) is the sum of differences between the actual power y and the allocated power $\hat{y}$ of the device under evaluation n in each period t.

$$\frac{\sum_t \left| y_t^{(n)} - \hat{y}_t^{(n)} \right|}{\sum_t y_t^{(n)}} \tag{2.7}$$

- ▷ Root Mean Square Error (RMSE) is defined as standard deviation of the prediction errors. In other words, it shows the difference

between actual power and the given power of appliance n in each time slice t.

$$\mathrm{RMSE}_{(n)} = \sqrt{\frac{1}{T}\sum_{t}\left(y_t^{(n)} - \hat{y}_t^{(n)}\right)^2} \tag{2.8}$$

▷ The confusion matrix contains the number of periods in which each estimated state of the targeted device was rightly classified or mixed up with each other state. In the following are the terms used in confusion matrix and their explanation depending on the estimated state and the actually state of the targeted device. True positives TP means that the device was correctly classified as on (TP) or as off (TN), while false negatives FN and false positives FP refer to the situation when the device was classified as on while it was off or vice versa. The following equations [Bat+14] are the corresponding mathematical definitions, where n is the appliance indices.

$$\mathrm{TP}^{(n)} = \sum_{t} \mathrm{AND}\left(x_t^{(n)} = \mathrm{on}, \hat{x}_t^{(n)} = \mathrm{on}\right) \tag{2.9}$$

$$\mathrm{FP}^{(n)} = \sum_{t} \mathrm{AND}\left(x_t^{(n)} = \mathrm{off}, \hat{x}_t^{(n)} = \mathrm{on}\right) \tag{2.10}$$

$$\mathrm{FN}^{(n)} = \sum_{t} \mathrm{AND}\left(x_t^{(n)} = \mathrm{on}, \hat{x}_t^{(n)} = \mathrm{off}\right) \tag{2.11}$$

$$\mathrm{TN}^{(n)} = \sum_{t} \mathrm{AND}\left(x_t^{(n)} = \mathrm{off}, \hat{x}_t^{(n)} = \mathrm{off}\right) \tag{2.12}$$

▷ Precision presents the proportion of periods in which a device under evaluation was off but it is correctly estimated to be on. In the same way, Recall presents the proportion of periods in which a device under evaluation was correctly estimated to be on while it was on.

$$\textit{Precision} = \frac{\mathrm{TP}}{(\ \mathrm{TP} + \mathrm{FP}\ )} \quad \textit{Recall} = \frac{\mathrm{TP}}{(\ \mathrm{TP} + \mathrm{FN}\ )} \tag{2.13}$$

▷ F1-Score is a combination between Precision of and Recall. It refers to the weighted average. It thus takes into account false

positives and false negatives. The F-measure is accurate in evaluating the effectiveness of a model, mainly when observations number in all classes is unbalanced, or the data have an uneven category distribution.

$$F\text{-}Score = \frac{2 \,.\, Precision \,.\, Recall}{Precision \,+ Recall} \tag{2.14}$$

2.4.2 *Non-Intrusive Load Monitoring Toolkit*

Although there are valuable contributions to NILM field, there are obstacles that prevent a direct comparison of the proposed approaches; experimental results are difficult to reproduce due to differences between studies in household selection, devices, time periods, and evaluation metrics. Nonetheless, there are no benchmark algorithms to compare the new contributions. The obstacles mentioned led the way to present Non-Intrusive Load Monitoring Toolkit. The presented tool is an open source application offers the research the ability of comparing their work to the-state-of-art NILM. The tool offers also comparison over different algorithms written in Python and different energy datasets [Bat+14].

NILMTK provides a comparison of energy disaggregation algorithms for different datasets, as it provides implementation of disaggregation algorithms such as Hidden Markov Model (HMM) and Combinatorial Optimization (CO), as well as preprocessing functions such as voltage normalization and downsampling functions. Later, several state-of-the-art disaggregation algorithms were added to NILMTK to keep the tool up to date. Implementations of deep learning approaches such as Sequence-to-Sequence and Sequence-to-Point are also included [Bat+19].

2.5 HOUSEHOLD ENERGY DATASETS

In the previous section, the NILM concept and modern algorithms for breaking down energy consumption were explained. In this section, the household datasets to evaluate disaggregation algorithms and their important aspects are addressed. Since 2011, numerous datasets have been publicly available to researchers, allowing objective comparison between studies. Prior to this, most approaches are evaluated using their own dataset, which made a comprehensive evaluation difficult. Publicly available datasets can be divided into either datasets that contain aggregated measurements for the entire house or datasets that provide readings for individual devices in addition to aggregated

data. However, supervised machine learning methods, where labeled datasets are used to train the model to predict outcomes accurately, must be trained with actual data; therefore, an individual signal representing the entire house is not ideal for training the network. The datasets mentioned in this section contain the mains and the appliance-level data.

Reference Energy Disaggregation Dataset (REDD) is energy datasets utilized for disaggregation algorithms published by Massachusetts institute of technology in the USA. It provides measurements of whole house appliances (9-24 appliances). The data are collected from 6 households at a high sampling frequency equal to 15 kilohertz (kHz). REDD also provides sub-metered data of each house at a frequency rate of 3 seconds [KJ11].

Another dataset has the same characteristics as REDD, but it includes more household, United Kingdom Domestic Appliance-Level Electricity (UK-DALE) dataset provides a total power consumption of five households at a sampling rate of 16 kHz. In addition, the power consumption of each appliance is measured at a sampling rate of 1/6 Hz (54 appliances) . This dataset was collected in the UK over 655 days [KK15b].

Long duration interval is provided by the Personalised Retrofit Decision Support Tools For UK Homes Using Smart Home Technology (REFIT) dataset. It contains electric load measurements from 20 households for a continuous period of about two years (2013-2015). In addition to the aggregated measurements, the consumption of nine individual appliances is also provided in 8-second intervals. Collecting data from numerous households in one dataset provides different electricity consumption habits, which increases the diversity of the data [MSS17].

In 2015 the Dutch Residential Energy Dataset (DRED) was released, consisting of whole-house and individual appliance electricity consumption for a household over a six-month period (July-December 2015) collected in the Netherlands. It includes aggregate consumption and appliance readings. Nine individual circuits are located in the targeted house, where appliance readings were collected at a frequency of 1 Hz: refrigerator, washing machine, central heating, microwave, oven, stove, blender, toaster, and laptop. In addition to consumption data, the dataset also includes environmental parameters such as wind speed and humidity. DRED also contains household metadata such as appliance-location association and number of occupants, which can be

useful for NILM [UNRLP15].

DRED provides total energy measured at the device level in a high percentage for all days. According to the dataset publisher, most other datasets do not include all appliances located in the household in the monitoring system, so 50 % of the total consumption data is not collected. The variability of this uncollected data can significantly impact the disaggregation accuracy. DRED has about 75 % of the appliance-level data and other mentioned datasets have about 45 % [UNRLP15]. In this the DRED deadest is used for performing experiments.

3 LITERATURE REVIEW

In the previous chapter, we laid out the basis of our work by introducing the basic concepts of electricity and deep learning neural networks, followed by a detailed explanation of NILM concept and energy breakdown methods. In this chapter, we review previous contributions in this area with a special focus on the preprocessing phase of the input data. Since the sequence length lies at the core of this study, the related works is divided into two subsections, one covering the parameter selection methods and the other providing an overview of the studies in which various methods were used to prepare the input data.

3.1 WINDOW LENGTH PARAMETER SELECTION

The interest in studying NILM has increased over the past decade, probably due to the remarkable results of consumption disaggregation through deep learning. Neural NILM is a study by Kelly and Knotenbelt [KK15a] that used three deep learning network architectures for NILM purposes using the UK-DALE dataset, which provides a sampling rate of 1/6 Hz. we skip directly to the method for selecting the window length value. In the study in [KK15a], the window length was selected based on a device-based strategy. After looking at the published implementation of the study [Kel], we were able to locate the function in which the window length is hard coded for each device. The values were defined as follows: 128 samples for the kettle, 288 samples for the microwave,1024 for the washing machine, 512 for the fridge, and 1536[1] samples for the dishwasher. The defined lengths thus have a time interval from 13 minutes to 2.5 hours. It was mentioned that the authors concluded experimentally that increasing the window length negatively affected the disaggregation results. However, they have not mentioned what methodology they used to determine the values. The same values have been used in a later work published by Liang [Lia+19] presenting the Sequence-to-Short-Sequence (S2SS) optimization method using UK-DALE dataset.

A study by Zhang and Zhong [Zha+18] introduced the Sequence-to-Point CNN to train the model using UK-DALE and REDD dataset, as the model receives input in the form of a window of aggregate con-

1 It was coded as $1024 + 512$

sumption and reports a signal point of the target device as output. In contrast to the previously mentioned paper, this study applied the one-for-all strategy, assigning a single window length of 599 samples for all devices. The author used the same value for both datasets despite having different sampling rates. Thus, 599 samples cover different time intervals, namely 599 seconds (REDD) and 1 hour (UK-DALE). The same value for the window length parameter was adopted later in [YLL21; Son+21; DSZ19]. Another study by Faustine and Pereira [Fau+20] proposed the use of a neural network architecture to detect the state of the devices and estimate their power consumption using UK-DALE dataset. The authors set the window length parameter for all devices to a value of 100 samples (10 minutes).

Later, a paper was published by Krystalakos and Nalmpantis [KNV18], who proposed a RNN architecture for disaggregating consumption using UK-DALE dataset. To predict the current data point, a sliding window of past aggregated data provided by the dataset is presented to the network as an input. When choosing the window length, the author decided to run simulations to define optimal value with an initial length of 50 samples (5 minutes). Then two more values were empirically defined , namely 100 and 200 samples corresponding to 10-20 minutes. Experimental assignment of the window length parameter is used in other studies. Ding and Zhang [Din+21] decided to measure the effects of varying the window size per device. They conducted experiments with window lengths according to the following ranges using REDD dataset: [100,500] samples for the fridge, [200,1201] samples for the washer dryer and dishwasher, and [10,55] samples for the microwave. The results confirmed the model's sensitivity (CNN) responding to the parameter variations.

All of the previously mentioned papers have used different strategies to select the input window length, ignoring the fact that the sampling rate of the dataset used also affects the time interval of the historical data. A recent paper by Reinhardt and Bouchur [RB20] focused on the impacts of window size on S2S and S2P optimization methods using the DRED. The study considered the per-device and one-for-all strategies, as well as the default window size value used in the NILMTK framework (99 samples). The results showed the optimal fixed window size worth 561 samples (5614 seconds). Moreover, it turned out that per-device strategy leads to better disaggregation results, while the optimal values unrelated to the operating durations. On the one side, the simplicity of such a strategy is still a challenge, since assigning different values means training an individual network per target device, which requires more configurations and increases the complexity of the model. On

the other side, using a fixed value for all devices means sacrificing disaggregation accuracy. This dilemma (accuracy vs. simplicity) must be resolved by developing a methodology that provides the simplicity of the one-for-all strategy while guaranteeing better results.

3.2 RELATED WORK TO DATA PREPROCESSING

Potential gaps in the data stream can be found due to transmission errors or outliers due to the noise originating from the interfering magnetic fields. Therefore, the preprocessing of the input data is a critical part of the learning process and has a significant impact on the results [ZC18]. Since windowing is a phase of data preprocessing, we decided to include the related works that contributed to this phase.

The author of the Neural NILM paper [KK15a] decided to increase the training set by generating synthetic total data by randomly combining extracted operating cycles of different devices. Then they trained the network with both real and synthetic data in a 50:50 ratio. In the testing phase, only real data was used. Filling the gaps in the data is done according to the duration of the gap. If the gap is shorter than 3 minutes, it is filled by forward-filing otherwise, it is filled with zeros. In the work by Zhou [Zho+20], a different approach was used to generate the training set. They manipulated the mains to increase the proportion of on-state samples by randomly excluding the samples representing the off-state of the device resulting a balanced dataset to increase the training performance.

The same concept of filtering the sequences representing off state is adopted on the wash machine in a later study by Chen and Wang [Che+18]. In addition, the consumption of the devices is divided by the maximum power demand to normalize the data points. Another strategy to normalize the data is used in [DSZ19; YLL21; Son+21]. The consumption of the devices is divided by standard deviation after subtracting the mean, both are calculated over the mains.

4 PRELIMINARY STUDY

This chapter presents the preliminary analysis in which an initial study is conducted to investigate the impact of applying window functions on input data. In this study, two different categories of window functions are examined; the symmetric and the asymmetric windows. The objective is to measure the disaggregation performance when a part of the input window is faded out. The remainder of this chapter lists the preliminary results of using window functions, followed by a discussion of possible improvements to the method based on our observations.

The S2S and S2P present the learning models of the state-of-art NILM. Both methods are based on training a neural network to estimate the targeted device from an aggregated signal. Internally in the neural network, the input data is sampled based on a windowing technique with a fixed window length that converts the input data into vectorization representation, limiting the number of samples available to the network, affecting the disaggregation performance. Two strategies for setting the parameter can be found in the literature: one-for-all and per-device, which compromise simplicity and improved disaggregation results. The motivation is to develop a windowing technique that enables the researcher to train one individual neural network for NILM purposes without using different window length per each device. We investigates the effect of different windowing techniques where the data is non-uniformly sampled. The goal is to maintain the simplicity of the one-for-all strategy by assigning a fixed value for the window length for training while modifying the sampling technique to improve the disaggregation task.

4.1 INPUT DATASET

Since this work aims to improve the disaggregation performance by applying non-uniform windowing techniques, we use an initial fixed window length at which the disaggregation accuracy of all considered devices is maximal. This can help to narrow down the solution space. To obtain this value, a preliminary investigation is required to examine an interval of different window lengths using the one-for-all strategy. A similar investigation [RB20] is already performed in the literature

using DRED to measure the effects of sequence length variations. The most effective window size for all devices is found at 561. The same value for the window length is adopted in this work as well as the targeted devices, subsequently the chosen dataset is DRED and the studied device is the microwave. The period from 27 July to 9 August is defined for training the network, and from 10 to 16 August is for testing. The date range specified is based on [RB20], where the best-scored window length using the one-for-all strategy on DRED is found.

4.2 INPUT DATA SCALING AND DOWNSAMPLING

Due to the nature of electrical appliances, each appliance type has a different power consumption. The electrical components of this device determine the power required. For example in a given model, the refrigerator consumes a maximum of 300 *watts* through the operation, while the microwave requires 3000 *watts* and the washing machine 2500 *watts* leading to extreme variations between data points. Since the deep learning model updates its small weights depending on the error between predictions and expected values, unscaled data can lead to an unstable or slow learning process, especially in regression problems where massive deviations between values (a spread of thousands of *watts*) can lead to fail the learning process.

Many techniques are used to rescale input and output data, such as standardization and normalization. The normalization means converting the data into values in the range $[0, \ldots, 1]$, and the standardization aims to standardize the data with a standard deviation of one and a mean of zero. In our case, the standardization procedure is applied to the data because there is a wide range of values.

Standardization requires an accurate estimate of the standard deviation and mean of the observed values, and then each value is standardized according to equation (4.1), where σ is the standard deviation and $\bar{x}$ is the mean. Both are calculated as equation (4.2)

$$x_{stand} = \frac{x_i - \bar{x}}{\sigma} \tag{4.1}$$

$$\bar{x} = \frac{\sum_i x_i}{n} \qquad \sigma = \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{n-1}} \tag{4.2}$$

In addition to rescaling the data, a downsampling procedure is applied (`reduction factor`). Setting the parameter is done according to [RB20].

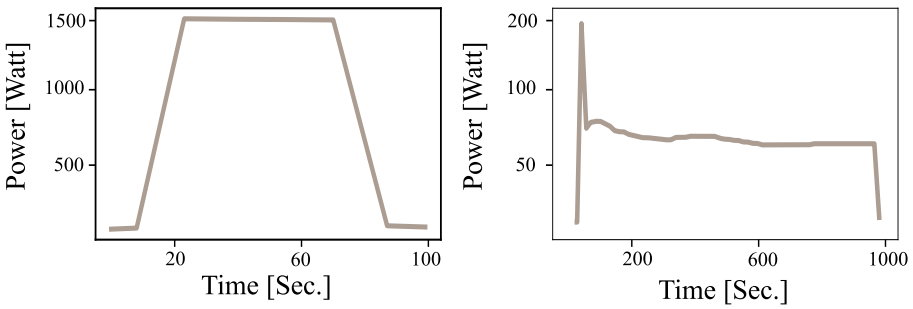
Table 4.1: Data preprocessing parameters and the assigned values

PARAMETER	VALUE
σ	600
$\overline{x}$	1800
reduction factor	10
sequence length	561

4.3 WINDOW TECHNIQUE ANALYSIS

This section covers the windowing technique used in deep learning disaggregation approaches on DRED. The sequences extracted by linear sampling are analyzed. Then, a methodological approach used to handle the spectral leakage problem in signal processing is proposed. Different window functions are applied to the input window of the network. In the end, the results of the study are listed and discussed.

DRED provides the total consumption of all devices (mains). The current values change depending on the device type and state, with an operation cycle drawn for each state. Extracting those cycles can be done using a function (*getActivations()*) provided by NILMTK. The function finds strictly consecutive samples above given threshold power. An operation cycle can be caused by switching the appliance On/Off or starting a new function such as drying in a washing machine. The figure 4.1 illustrates two operation cycles of different devices, which differ significantly in their duration.



(a) Microwave operation cycle (b) Fridge operation cycle

Figure 4.1: A practical example of appliance's operation cycles extracted using NILMTK

This aggregated consumption is the input to the neural network. The data is re-scaled using the standardization method introduced before

to convert the aggregated data into values in the range $[0, \ldots, 1]$. As mentioned in subsection 2.3.1, deep learning networks expect the input data to be transformed into a vectorized representation. Therefore, a sliding window is used to capture the data in sequences in the same length. Each sequence represents the previous time steps to predict the next time step. The sequence length parameter controls the time interval covered in a sequence. As presented in section 3.1 two strategies are usually used to set this parameter; the per-device and the one-for-all strategy.

In the preliminary study, an investigation is performed to evaluate the disaggregation performance when part of the sequence is masked out by multiplying the input sequence by a window function. This operation reshapes the signal so that the amplitude varies uniformly toward zero at both edges. Lower values at both edges can have positive impact on the performance by reducing the affect of these values on the neural network weighting process. For simplicity, a single network is used for training the devices under consideration. Therefore, a fixed window length (561) is defined during the study for sampling the input data according to the one-for-all strategy.

4.3.1 *Window Functions*

Window functions present sequences of data points with finite lengths. The signal amplitude approaches zero uniformly at both edges for most window functions. Window functions are utilized in signal processing to eliminate the spectral leakage mentioned in subsection 2.1.4 by merging the endpoints of the waveform, resulting in a continuous waveform without discontinuities.

There are several window functions and each has its own properties. Choosing window functions for this study is according to the motivation of examining different effects on the input window; therefore, the chosen window functions must have different sidelobes. The cosine sum window family contains two commonly used functions; Hamming and Hann. They are used to reduce spectral leakage. Both have low sidelobes and a broad peak. The Hann window function approaches zero at both ends of the window eliminating discontinuities. In contrast to Hann, the Hamming window does not entirely approach zero showing slight discontinuity. The figure 4.2 illustrates the Hamming window and the Hann window.

Another window functions are Blackman-Harris and the Flat Top shown in figure 4.3.The Blackman-Harris window is similar to the pre-

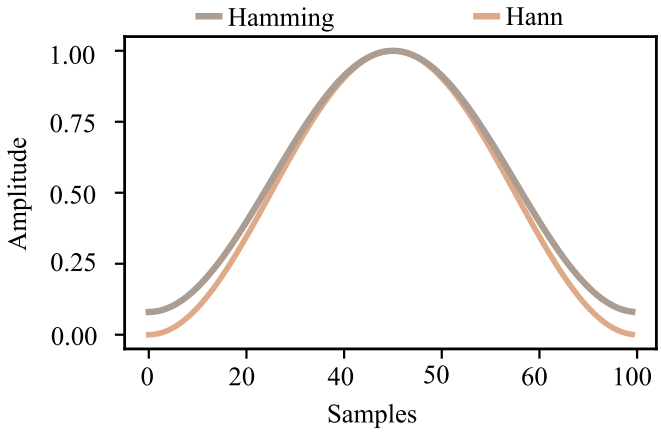


Figure 4.2: The Hamming and the Hann window functions

viously presented Hamming and Hann windows functions. Its spectrum has a sidelobe compression. A sinusoidal shape can be observed in the Flat Top window with narrow sidelobes. The application of these windows, due to their characteristics, fades out more data points on the sides than the Hamming and Hann windows, which can have a greater impact on learning performance.

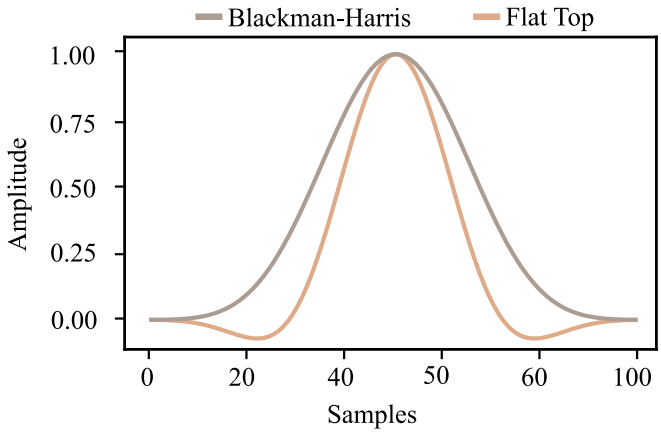
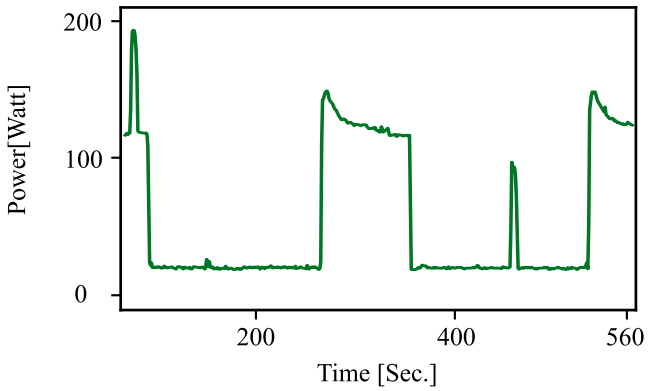


Figure 4.3: The Blackman-Harris and the Flat Top window functions

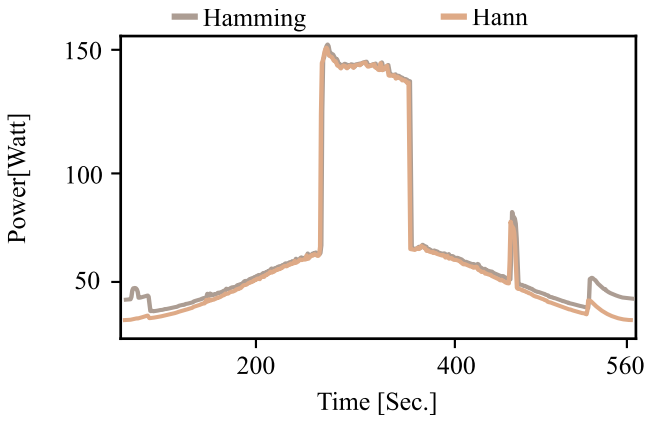
The literature does not provide a universal approach for selecting a window function. Moreover, the effect of applying the window function on the neural network input has not been studied yet. Therefore, a group of the most commonly used window functions were selected to conduct experiments. However, the chosen strategy is to compare the performance of applying different window functions with the baseline, where no window function is used. Note that the length of the used window function are the same as the assigned sequence length 561. To perform the multiplication operation, each data point of the window is multiplied by one point of the window function.

4.3.2 *Symmetric Window*

Window functions are applied by multiplying a sequence of input data by a window. A window represents a series of weights. In this way, each input value can be weighted according to its time.



(a) A sequence of aggregated consumption



(b) Application of two different window functions on the sequence

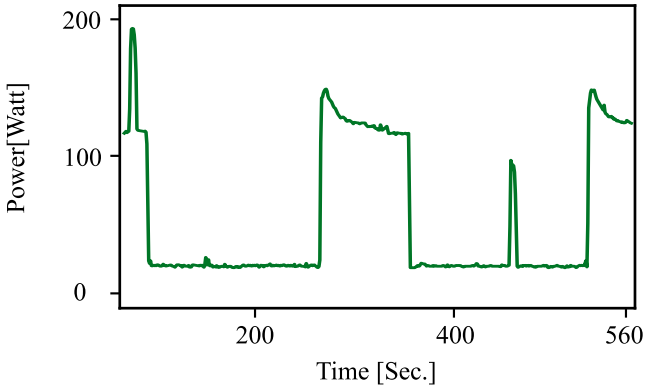
Figure 4.4: A practical example elucidates applying symmetric window functions

For symmetric windows, where a symmetric shape is observed around its center, the points on either side has a lower weight than the points in the center. Assuming that the center of the windows is the zero point on the x-axis (axis of symmetry), symmetric windows satisfy the symmetry condition: $x(t) = x(-t)$, where $x(t)$ is a value of the signal where t is positive and $x(-t)$ is a value of the signal where t is negative. Figure 4.4 shows the effect of window functions after multiplying a mains sample by two different window functions, the Hann and Hamming windows. The events on the sides of the sequence are hidden after the window is applied. The plot also shows

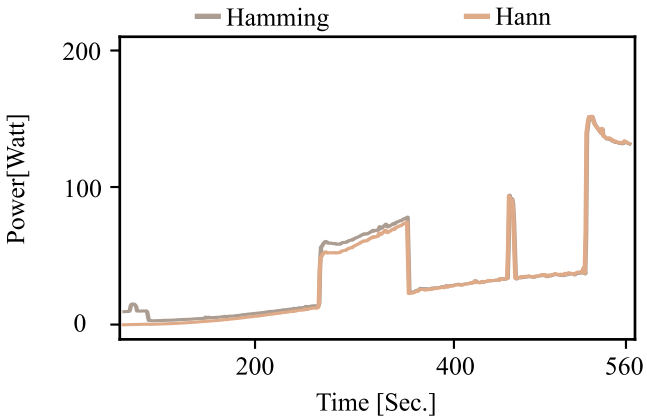
that, because of the small side lobes of the Hann window, the event is completely masked out compared to the Hamming window. In this way, these parts of the window have less impact on the weighting process during training. The effect of the window function is clearly visible for the historical data points, but not for the current points in the window.

4.3.3 *Asymmetric Window*

Based on the observation that symmetric windows can eliminate the most recent data point of a predicted value, the use of asymmetric window functions is proposed. To create asymmetric windows, new



(a) A sequence of aggregated consumption



(b) Apply asymmetric window functions on the sequence

Figure 4.5: An example of applying asymmetric window functions

data points are computed within the points range of the half window, subsequently the length of the input and the window are aligned to fulfill the multiplication operation.

Unlike symmetric windows, asymmetric windows do not have a symmetric shape, which provides the ability to selectively target a particular portion of the signal. Previously, both ends of a sequence are multiplied by values close to zero. Asymmetric windows target the historical points. The importance of each value is based on how far it is from the most current value in the sequence. In this way, the most recent information is weighted more than other points. The figure 4.5 illustrates a practical example of applying asymmetric window functions on a mains sample.

4.4 STUDY RESULTS

This section provides the results of experiments performed using the S2S algorithm on DRED. NILMTK is used as an execution environment.

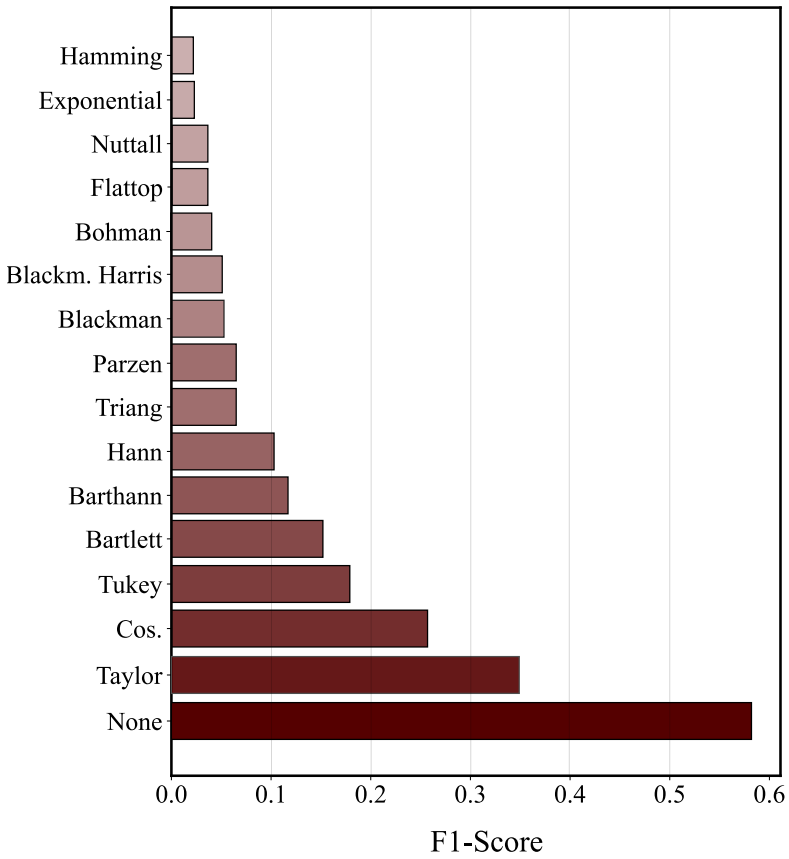


Figure 4.6: Results of applying symmetric window functions for microwave

All hyperparameters of the neural network are set to default values. The number of epochs refers to the number of complete runs through the training dataset. In our case is increased to 30 epochs as [RB20] for

the stabilization of the results.

Figure 4.6 contains the F1-score of the microwave disaggregation after applying symmetric windows to the input sequence. The evaluation also provides a baseline where no window function is applied. The results confirm the effect of the window function on the network's performance. However, overall, no improvement is observed compared to the baseline. All window functions have scored significantly worse than the default setting. The S2S have achieved near 0.6 F1-score with no window function, and the percentage goes dramatically near three percent when the window function Hamming is applied. Between those two values, the outcomes of the other windows are spirited. The Taylor window results is the nearest to baseline.

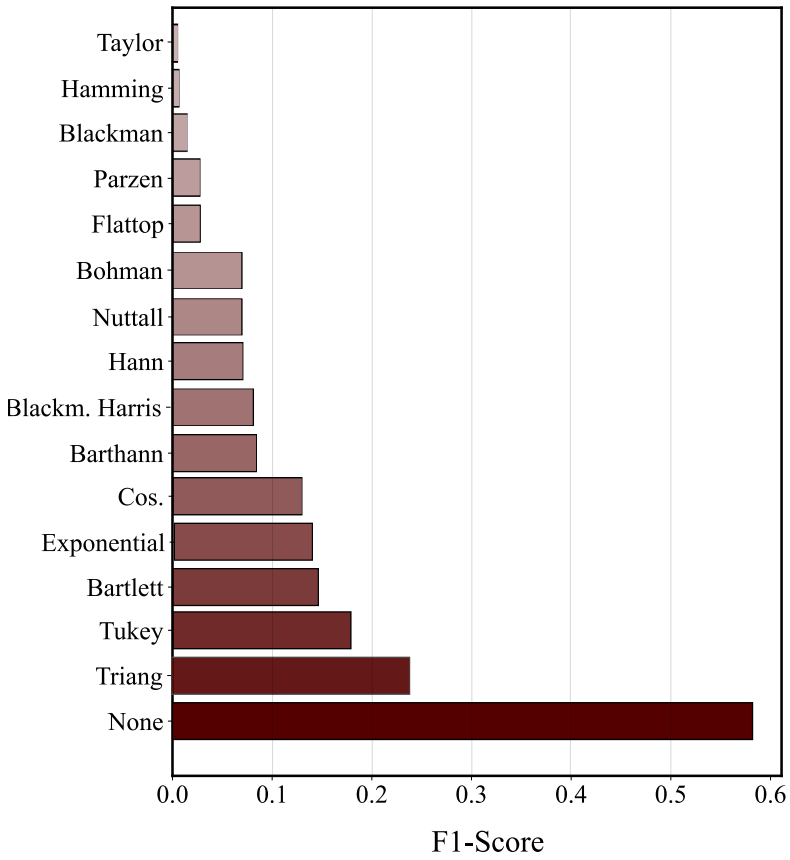


Figure 4.7: Results of applying asymmetric window functions for microwave

Figure 4.7 presents the results of applying asymmetric window functions on the input. The evaluation elucidates weak performance compared to the baseline. However, the applied windows reported a different response to the asymmetric windows. For example, the best-

scored function in symmetric windows (Taylor) has scored worse in asymmetric windows. The reader can visually observe that the results of the symmetrical windows are on average better than those of the asymmetrical ones, but still far from the baseline.

4.5 SUMMARY

This chapter presented the preliminary study in which an initial investigation was conducted on the effects of applying symmetric and asymmetric window functions on the input data. we estimated the disaggregation performance when a part of the input window is hidden or has a small impact on the weighting process in the neural network. The goal is achieved by multiplying the input window by a series of values that touch zero on the sides.

Applying window functions on aggregated data have shown that events and device activation on the sides of a sequence are faded out (as shown in figure 4.4). The asymmetric window is proposed to target one side of the input window representing the historical values to fade out the less relevant data.

The results showed that the application of the window fraction has unsatisfactory performance compared to the baseline. The disaggregation results deviated from the baseline by more than 0.3 F1-score in the symmetric and asymmetric windows.

A study is made in this chapter to investigate the effect of fading out samples from the input window on load disaggregation. The goal is to improve the load disaggregation performance while using the one-for-all strategy for setting the window length parameter to maintain simplicity. Two variants of window functions are used to target a different part of the window. The symmetric windows target the samples on both sides of a sequence, while the asymmetric windows aim to fade out the historical samples. After analyzing the results, it turns out that hiding the information by window functions leads to degradation in disaggregation performance. Therefore, we decided to reverse the investigation from hiding the information in the input window and feed the window with more historical information without changing the overall window length of the studied devices.

5 METHODOLOGY

This chapter covers information on the proposed non-equidistant temporal sampling technique using time spacing functions. First, a summary of our contribution is given, followed by an explanation of the proposed methodology to reach the research goal. Next, different variants of non-equidistant temporal sampling are presented for evaluating the methodology used. The remaining sections deal with the design of the experiments, such as preparing the execution environment and selecting the model parameters.

5.1 CONTRIBUTION OVERVIEW

In the preliminary study, we aimed to improve the disaggregation performance by reducing the impact of some samples in the input window on the neural network training process. The results confirmed a negative impact on the results. Therefore, we changed the research direction from hiding samples to aggregating more data points in the window to extend the time interval covered by the window by adding more historical information.

This chapter proposes using a non-equidistant temporal sampling technique to sample the input data from the aggregated signal representing the appliances consumption of a household through two algebraic functions: the quadratic and the exponential. The motivation is to develop a windowing technique that enables the researcher to train one individual neural network for NILM purposes without using different window length per device. The goal is to investigate the impact on the load disaggregation when the input window can represent more historical information.

This technique splits the traditional uniform window into two parts according to a fraction: a high-resolution part and unequally spaced part taken by means of non-linear algebraic functions. Using non-linear algebraic functions creates non-equidistant temporal space between samples allowing the input window to contain more historical information without having to adjust the window length parameter. For ensuring a fair evaluation of the proposed method, different variants are suggested—the inverse and the linear downsampling functions. The first employs an inversed equation to generate an

entirely non-equally spaced window, and the last creates a fraction of the window with equally time-spaced samples. Both functions cover the same effective window length as the proposed method.

5.2 NON-LINEAR TIME SPACING FUNCTIONS

As mentioned before, non-linear algebraic functions are utilized to expand the time spacing between the samples in the input window. In this section, the quadratic and the exponential functions are introduced.

It is important to first explain the linear function before demonstrating the non-linear function. Visually, a linear function is one whose curve draws a line. In other words, the slope of the curve between any two points is the same. Algebraically, it can be defined as a polynomial function whose highest exponent is equal to 1, or as a horizontal line ($z = t$, where z is a constant). Non-linear functions have the opposite property of a linear function. The curve of a non-linear function has a slope that varies between points. It can be defined as a polynomial function whose smallest exponent is greater than one.

The non-linear algebraic functions are employed as time spacing functions to create non-equidistant temporal space between data points in the input window. This space is controlled by the function used. Utilizing different spacings functions is driven by the fact that different linear functions have differences in the temporal distance between samples and the effective window length, which refers to the time interval covered by the window. In addition, the samples collected by different functions have various distributions over the timeline due to the nature of the function used. This was a reason to include different functions in this study. The following subsections introduce various non-linear functions used in this study, showing their differences.

5.2.1 *Quadratic Spacing*

The quadratic function is a non-linear function. It can be mentioned as the second degree of the polynomial function. The equation (5.1) presents the basic form of such functions, where $a, b, c \in \mathbf{N}$.

$$f(x) = ax^2 + bx + c \tag{5.1}$$

We use the quadratic function as a time spacing function to create non-equidistant temporal between samples in a window. In other words, the data points in a window are sampled with non-equidistant temporal spacing made by the quadratic function. Thus, there is a temporal

distance between each two samples corresponding to the relation t^n, where t is the temporal horizontal of the sampled window and n is a rational number representing the exponent of the function in the range $[1, \ldots, 2]$. Obviously, the exponent of one refers to a uniform spacing where there are no time gaps between data points.

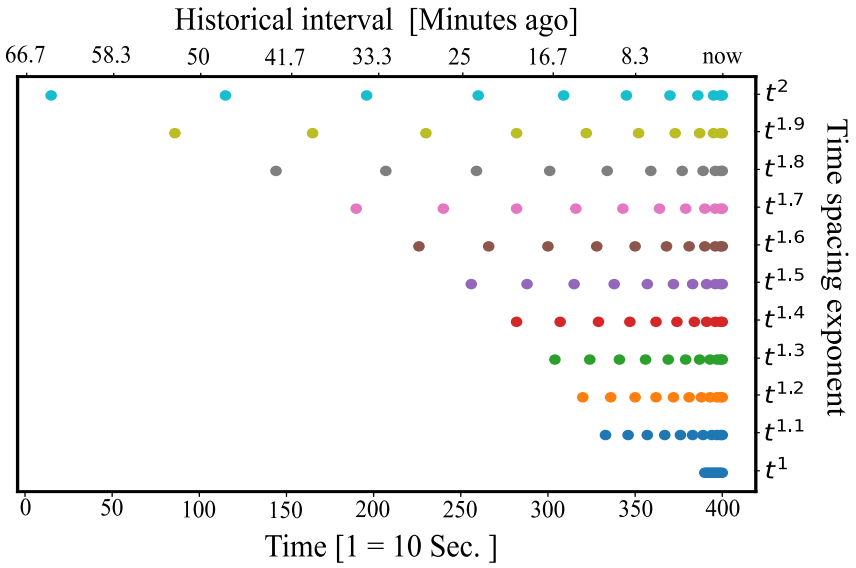


Figure 5.1: Sampling the data points based on quadratic functions elucidates the ability to collect historical information compared to the traditional uniform windowing t^1

The goal of using non-linear functions is to obtain a windowing mechanism to collect historical samples of the corresponding output window with a fixed window length. Figure 5.1 draws a practical example of sampling data points by using non-linear functions with different exponent values. This demonstrates the ability to collect historical information compared to the traditional windowing (Spacing function t^1). The plot of values distribution shows the property of quadratic function, as the time spacing between points started relatively small and then increased quickly after a period.

5.2.2 *Exponential Spacing*

A different non-linear function is proposed for non-equidistant windowing. Exponential functions have a number in the base greater than one. It has a basic form shown in equation (5.2) where a represents the coefficient.

$$f(y) = ab^x + c \tag{5.2}$$

The exponential function is also utilized as a time spacing function to create non-equidistant temporal between samples corresponding to the relation n^t, where t is the temporal horizontal of the sampled window and n is the exponent of the function in the range $[1, \ldots, 2]$. The range is defined to represent different non-equidistant temporal behavior of different spacing functions. Numerous exponents lie within the defined range to explore the differential distribution of samples across the time axis while also exploring the sensitivity of S2S and S2P to the variation in effective window lengths that results from using different exponents. Note that using high values as exponents aggregates historical information that the used data set does not provide. Note that using high values as exponents may aggregate history that the dataset used does not provide, e.g., aggregating data points going back two weeks while the dataset contains aggregated data for only one week. The figure 5.2 visually demonstrates the samples distribution collected according to the exponential function.

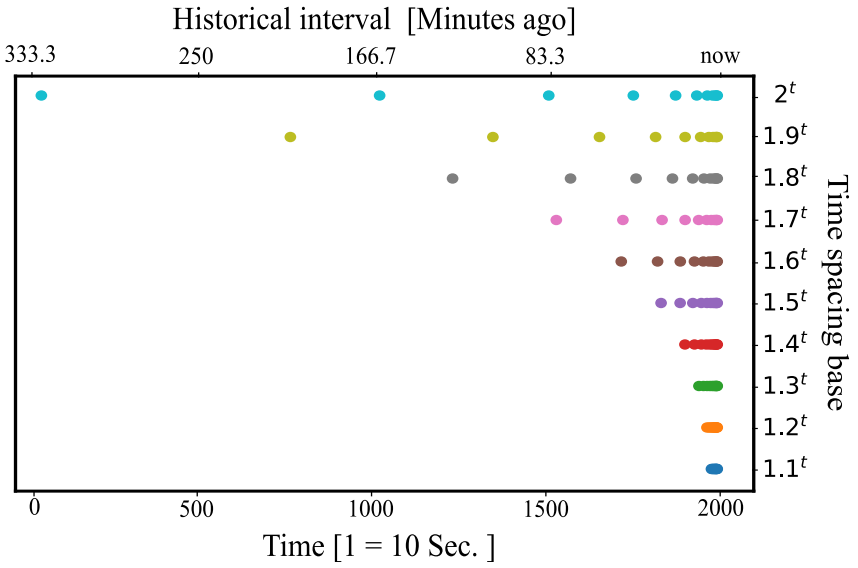


Figure 5.2: Sampling the data points based on exponential functions

The previously mentioned figures indicate the different properties of quadratic and exponential functions. The time interval between exponentially sampled points started relatively small and then increased rapidly after a period. If this pattern is followed in sampling data, more historical data are collected than quadratic sampling. To illustrate this, consider the sample distribution of the function 2^t. It covers about 5 hours compared to one hour for the quadratic function t^2.

The motivation behind highlighting these differences is to build the knowledge necessary to analyze the behavior of each function in the

results. The use of different functions in the experiments is driven by the desire to explore more areas in the non-equidistant sampling space.

5.3 PREPROCESSING INPUT DATA

Some gaps in the data stream can be found due to transmission errors or outliers. Therefore, the preprocessing of the input data is a critical part of the learning process and has a significant impact on the results [ZC18].

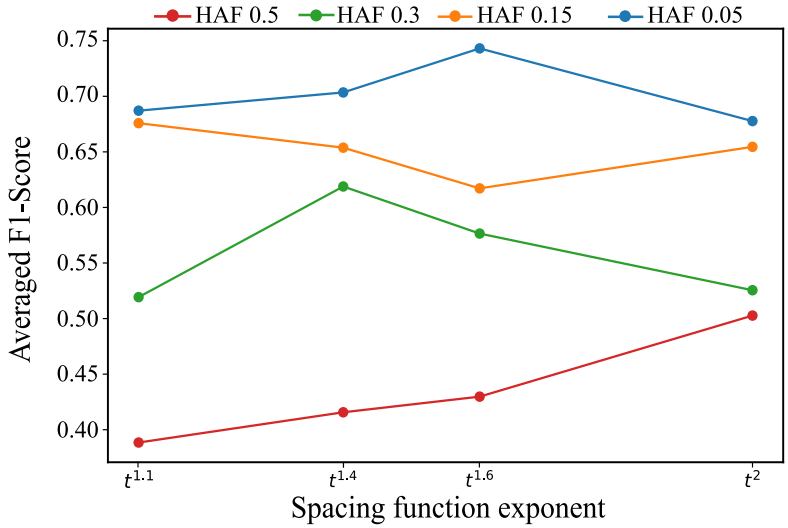
After introducing the concept of generating non-linear distances between samples, this section deals with the preprocessing phase of the input data, where the aggregated data is downsampled and normalized to be converted into a vectorized representation. This section also provides more technical details about the non-equidistant temporal sampling technique, such as the definition of the window parameterization and the application of mathematical functions to the collected samples.

The exact configurations used in the preliminary study are adopted in the final experiments, in terms of the parameters used in data standardization and the reduction factor (**??**). The DRED dataset was selected to run experiments focused on the period from 27 July to 9 August for training the network (14 days) and from 10 August to 16 August 2015 for testing (7 days) as suggested in [RB20]. The appliances considered are the fridge, the microwave, the cooker and the washing machine. The data is standardized and downsampled by factor 10. The used window length is 561.

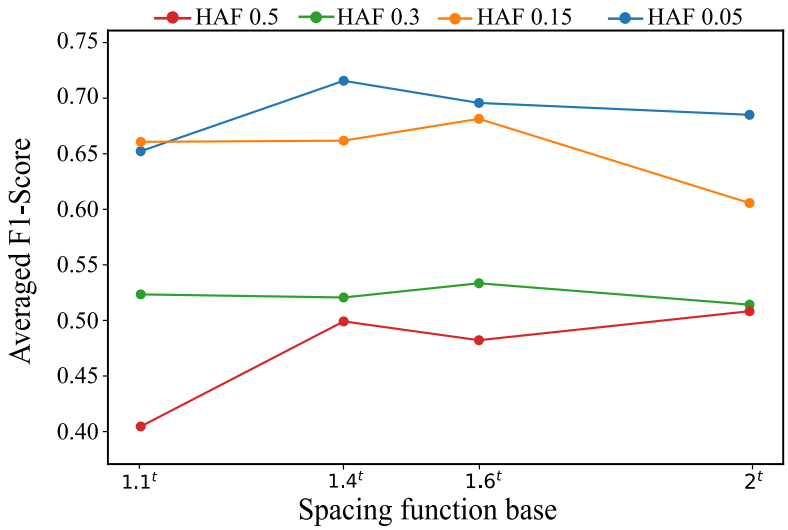
5.3.1 *Windowing Parameterization*

Two parameters are associated with windowing operation: the *spacing function value* and the *history aggregation factor*. The first one refers to the value of the exponent in the quadratic function or the base value of the exponential function and belongs to the range $[1, \ldots, 2]$. This parameter controls the behavior of the function within the part where a non-equidistant time interval is created; the larger the value, the larger the historical interval from which the function can aggregate information. The division of the window into high resolution and non-equidistant parts is based on the history aggregation factor. Multiplying the window length by the aggregation factor gives the length of the non-equally spaced part. we give an example to clarify the parameters. Given a window with 561 samples, with a spacing function value of 2

and an aggregation factor of 0.05. This means that the window contains 533 samples at full resolution and 28 samples aggregated with the function t^2. Since the factor has a wide range $[0, \ldots, 1]$, corresponding to $[2, \ldots, 559]$ non-equidistant samples, exploring the range in small steps is a time-consuming task. Therefore, initial experiments are conducted. The goal is to visually define an upper bound. Beyond this value, the disaggregation performs worse. The experiments are performed on both proposed spacing functions with aggregation factors $\{0.05, 0.15, 0.3, 0.5\}$ and four spacing function values $\{1.1, 1.4, 1.6, 2\}$.



(a) Quadratic spacing function



(b) Exponential spacing function

Figure 5.3: The results of initial experiments show performance degradation when using a history aggregation factor (HAF) greater than 0.15

The figure shows the results of the initial experiments to establish an upper limit for the aggregation factor. It can be seen that both functions perform worse beyond the factor 0.15 (orange curve). Thus, the upper limit is 15 % which defines a new aggregation range [0.003, 0.15] for performing the final experiments.

5.3.2 *Non-equidistant Sampling*

As the reader notice, we seek to improve load disaggregation performance by using non-linear functions to create non-equidistant temporal part in the input window, where distances between data points is established so that the window can provide more historical information to the neural network. The proposed technique divides the classical uniform window into two parts according to the history aggregation factor: a high-resolution part, where all samples are uniformly distributed, and a sequence with non-equidistant spaced samples, which are collected using nonlinear algebraic functions controlled by spacing function value parameter.

Since the resulting window is non-linear, calculating the effective sequence length (time covered) is not intuitive. With non-equidistant windowing, the window length used in sampling the data do not represent the effective length. Therefore, we have introduced the algebraic formula to calculate this value for the used spacing functions. Equation (5.3) gives the covered time for the quadratic function and the equation (5.4) for the exponential function.

$$ESL_{quad} = \left(UPL + \sum_{t=1} t^{SFV} \right) * DF \tag{5.3}$$

$$ESL_{expo} = \left(UPL + \sum_{t=1} SFV^{t} \right) * DF \tag{5.4}$$

wehre:
ESL = effective sequence length
UPL = length of the uniformed part
DF = data downsampling factor
SFV = spacing function value

Multiplying the window length by the downsampling factor gives the value of UPL. SFV represents the exponent in the quadratic and the base in the exponential functions.

Algorithm 1 Non-equidistant Temporal Sampling Technique

Require: An array contains the aggregated consumption (mains)
Ensure: Vector of sequences that are non-equidistantly sampled

```
1:  Initialize the parameters: mainsLength, UPL, HAF, TSF
2:  for i = 0,... mainsLength do
3:      Sequance_indices ⟵ GenerateSequancIndexes(i, i+N)
4:      equallyPart_indices ⟵ SplitWindow(Sequance_indices, HAF)
5:      for i = 1,...,UPL do
6:          Timestep ⟵ CreateTimeStep(TSF, i, equallyPart_indices)
7:          if timestep > mainLength then
8:              break
9:          end if
10:         collectedSamples ⟵ getDataByTimestep(mains,timestep)
11:         nonUniWin ⟵ ApplySamplingFunction(collectedSamples)
12:     end for
13:     uniWin ⟵ getDatabyIndex(equallyPart_indices)
14:     sequence⟵ concatenate(uniWin,nonUniWin)
15:     traningSet ⟵ append(sequence)
16: end for
17: return traningSet
```

In algorithm 1, the pseudocode represents the operation of non-equidistant temporal sampling. The algorithm takes an array of aggregate consumption as input and reports a vector of sampled sequences using the non-equidistant temporal sampling technique. In line 1, the parameters for windowing; the length of the uniform part UPL, the history aggregation factor HAF, and the time spacing function TSF are initialized.

Lines $3-4$ creates an array of indices and splits this array according to the aggregation factor. Line 6 creates a nonlinear time step based on the specified distance function. Lines $7-8$ ensure that no timestep exceeds the total length of the available history. Lines $9-10$ collect the data points from the mains according to the calculated timestep and apply a sampling function to the collected samples. Using the sampling functions is to increase the representation of a single sample for the time interval generated, without listing each value in the window, e.g., the max function. Lines $12-13$ collect the data points for the uniform part and concatenate the two resulting parts into one sequence. Line 14 adds the sequence to the training set.

Figure 5.4 gives a practical example of non-equidistant sampling, where three input windows are sampled according to three different sampling techniques. The sequence at the bottom is sampled based

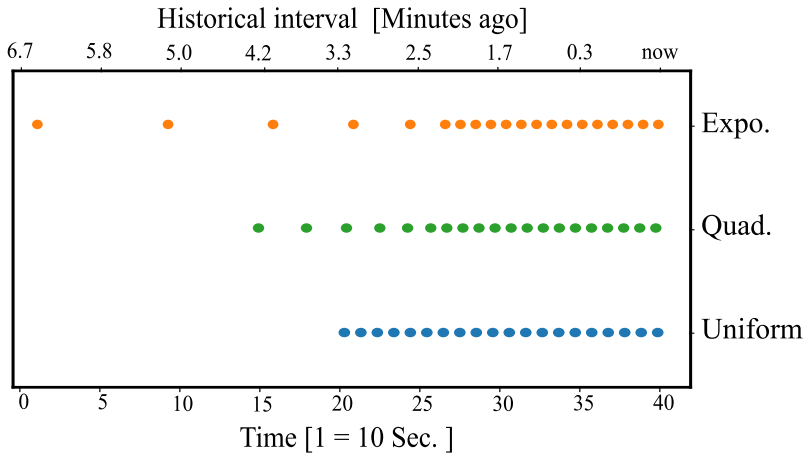


Figure 5.4: An example of non-equidistant sampling shows differences in the effective window length when using different spacing functions

on uniform windowing presenting the traditional technique. The following window is based on non-equidistant sampling using the quadratic function, and the last on top is based on the exponential function. It is clear that the windows have the same window length, but each has different effective window lengths. For example, the exponential function can aggregate data covering the double interval of the traditional windowing.

5.3.3 *Data Sampling Functions*

non-equidistant temporal sampling generates time gaps between sampled data points. The generated distances are nonlinear due to the algebraic properties of the functions used. For example, the exponential function grows rapidly, so a longer time interval between samples can reach hours. This behavior may lead to the loss of information about the events and the activations that occurred in this interval.

To avoid this problem, the application of statistical functions is presented. The goal is to increase the representation of a single sample for the time interval produced by spacing functions so that a summary of the data is given without listing each value. The max, min, average, and median functions are proposed to cover the samples to apply the function to each non-linear time step.

The use of the statistical functions may affect the input sequence differently. The max function outputs the maximum value of a series of samples. This function selects an already presented value, which

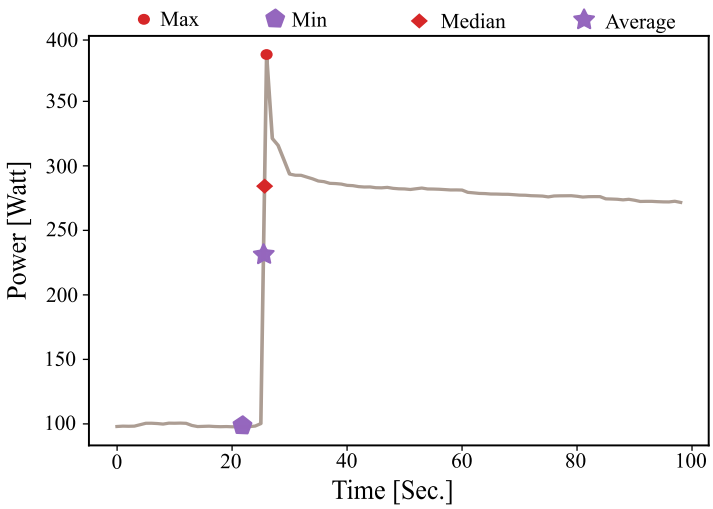


Figure 5.5: Statistical functions are applied on a sample of mains

means that the value belongs to the data set for training and testing. No further calculation is required. In addition, the Max function can catch a possible activation of the same device under consideration, which could improve the learning process by providing historical information about the device under study. However, the function could capture an outlier or noise that does not represent the entire sequence.

Compared to the maximum value, the average and median functions better represent the values within a sequence because all values are considered in the calculation and can affect the output. Both functions are intended to give a typical or mean value of a series of samples. The average value is given by adding up all the individual values and dividing this sum by the total number of observations. The median value is calculated by taking middle value where half of the observations are lower and half are higher. Nevertheless, the median generally has an advantage over the average when there are extreme values within the sequence. The median function excludes the extreme values from the calculation and gives a more accurate reflection. Figure 5.5 elucidates a practical example, where the mentioned statistical functions are applied on a sample of mains signal.

5.4 NON-EQUIDISTANT SAMPLING VARIANTS

As mentioned earlier, ensuring a fair evaluation of the proposed windowing technique is essential in the study. Since the non-equidistant sampling and uniform sampled windows have a significant variation in the covered time per window (a range of hours), comparing their performance may not be objective. Therefore, different variants of non-

equidistant sampling are proposed; the inverse and uniform downsampling functions. Both functions can cover the same time interval covered by the proposed method. This section introduces those functions and addresses their properties

5.4.1 *Inverse Function*

In contrast to non-equidistant temporal sampling, the inverse function has a different manner of collecting samples from the aggregated signal. Instead of splitting the window into two parts to create a non-non-equidistant sampled sequence, the inverse function creates this part over the entire window in total length. This means that no split is used and the data points are collected according to the spacing function (quadratic or exponential), resulting in a total window with non-equally sampled window and the aggregation factor is set in this case to one to target the entire window.

In order to obtain a fair evaluation, the generated window with unequal spacing must cover the same time interval as the windows with equal and unequal proportions. Using the previously presented equations (equation (5.3) and equation (5.4)) to generate such a window might be possible by inverting the question so that the input to the equation is the total time covered and the output is the exponent of the spacing function function. The resulted exponent can be fed into the equation to generate full unequally spaced window.

$$\sum_{t=1}^{n} t^x = \left(t_1^x + t_2^x + \ldots + t_n^x - \mathrm{ELS}_{quad}\right) \tag{5.5}$$

$$\sum_{t=1}^{n} x^t = \left(x^{t_1} + x^{t_2} + \ldots + x^{t_n} - \mathrm{ELS}_{expo}\right) \tag{5.6}$$

Equation (5.5) shows the inverse of the quadratic and the function where the input is the exponent x and the output is the effective sequence length. Equation (5.6) also shows the corresponding inverse of the exponential function, where n is the window length. Solving the equation means finding an exponent/base that generates a non-equally spaced window with the same effective window length as the non-equidistant temporal sampling. calculating the exponent/base is possible numerically. Therefore, a function is written to calculate the value of the spacing function value, where the search starts with a small value and keeps increasing in short steps. The search continues until the exponent found can cover the same effective window length as the original equation.

5.4.2 *Linear Downsampling Function*

Another variant of non-equidistant sampling for comparison is proposed—the uniform downsampling function. The core idea is taking samples in uniform time step as a part of the window.

The linear downsampling function retains the technique of splitting the classical window based on a predefined factor, resulting in two parts: a uniformly sampled part where the data points are evenly spaced and there is no time interval between samples, and the other linearly spaced part where the samples are collected linearly after a fixed step. The equation (5.7) shows the mathematical formulation of the linear downsampling function.

$$\mathrm{ESL}_{\mathrm{uniform}} = \left(\mathrm{UPL} + \sum_{t=1} (t + \mathrm{FTS}) \right) * \mathrm{DF} \tag{5.7}$$

wehre:
FTS = fixed time step
ESL = effective sequence length
UPL = length of the uniformed part
DF = data downsampling factor
SFV = spacing function value

The motivation for presenting numerous variants of the non-equidistant sampling technique is to evaluate the proposed method from various aspects. On the one hand, the inverse function offers a comparison in which no splitting technique is used and the samples are drawn non-equidistantly. On the other hand, the linear function maintains the splitting technique, but the sampling is done with a fixed linear step.

5.5 EXPERIMENT DESIGN

In order to investigate the effects of the proposed method on load disaggregation performance, numerous experiments are planned to be executed with a range of parameters. The table 5.1 shows a set of the parameter used in the experiments and the corresponding values.

The final experiments explore the non-equidistant windowing technique against the linear and the inverse functions with a wide range of aggregation factors and multiple sampling functions using different optimization algorithms. More than 14.000 experiments are planned to cover all parameters combinations. The estimated time to complete

Table 5.1: Windowing parameters used in the experiments and the corresponding ranges

PARAMETER	VALUE / RANGE	INTERVAL
Algorithm	[S2S , S2P]	-
Time spacing function	[non-equidistant, linear, inverse]	-
Sampling function	[max, average, median, min]	-
spacing function value	{0.1, . . . , 2}	0.1
history aggregation factor	{0.003, . . . , 0.15}	0.002

the planned experiments is 300 days (15 minutes per each). A parallel execution with several machines is decided. The parameters are divided into four groups, and each group is assigned to one machine. The estimated time with parallel machines is reduced to 75 days. GPU computing is used to speed up the calculations. All hardware used have the same specifications: processor i5 — 7500 3.4GHz, 16GB RAM and GPU NVIDIA Geforce-GTX 1060.

5.5.1 *Environment Setup*

NILMTK is used as an execution platform installed in the Anaconda environment. The CUDA toolkit and Tensorflow-GPU are essential for enabling GPU computing features. The goal of using NILMTK is to use the already available implementations to evaluate new aggregation methods against the state-of-art methods.Running a large number of experiments in parallel on many GPUs demonstrates the need to automate execution so that each GPU machine has a list of experiments assigned to it. Automation is also responsible for robustness to unexpected terminations or interruptions. Thus, the computer runs the experiment that was running before the interruption, and there is no need to re-run previously executed experiments.

Figure 5.6 shows a flowchart of the developed system for automating the execution of the experiments. After specifying the parameter ranges, the system calculates all combinations and splits them into several CSV files. The files contain the model and windowing parameters. Each experiment has a status. When the execution is started, a shell function reads the file and activates a prepared Anaconda environment to call the NILMTK implementation and pass the parameters. After the execution is finished, the results are saved, and the experiment's status is updated in the file.

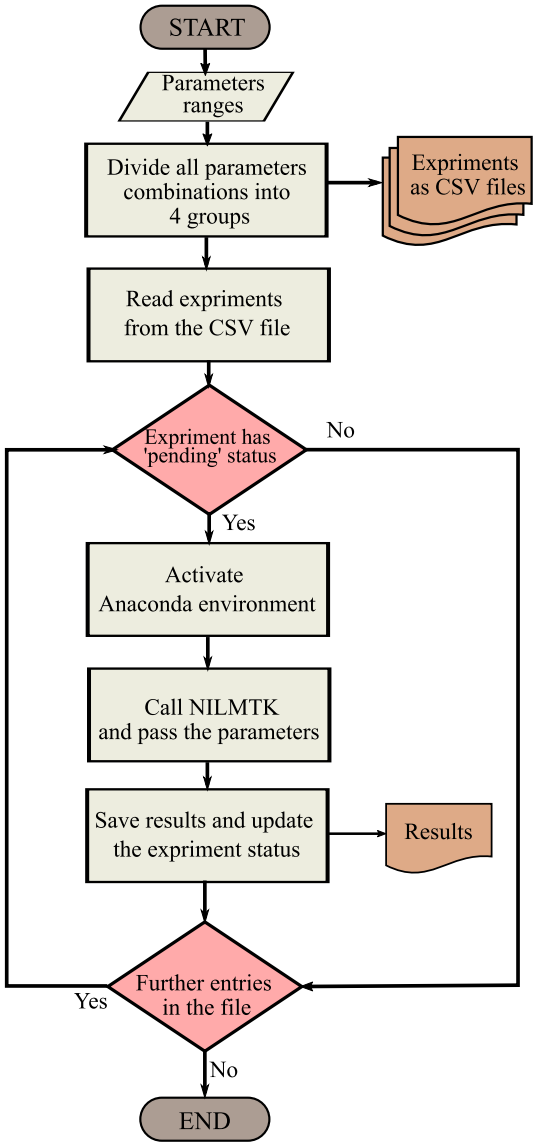


Figure 5.6: The system designed to automate the experiments execution over four GPU-machines

5.5.2 *Model Parameters*

Neural networks use numerous optimization methods to tune internal parameters till they can perform well against error measurements, e.g., the mean square error. The epoch number and the batch size are hyperparameters of the model. The epoch number defines the number of complete runs on the training set. the batch size parameter handles the number of samples that are run before internal parameters are updated. All the model parameters are set to their default values provided by NILMTK. using the standard configurations of NILMTK ensures that any

changes in the model performance (improvement or deterioration) are mainly due to changing the windowing technique, and it is not relevant to tune the parameters or use different model configurations. Therefore, the batch size is maintained as the default setting with 512. The number of epochs is increased from the default value (10) to 30 as suggested in [RB20].

The following is the network architecture used by S2S and S2P. The architecture is adopted originally from [Zha+18] and implemented in NILMTK. Except the output layer, the ReLU activation is applied in all layers. The network architecture consist of input layer, output layer, fully connected layer (1024), and 5 1D-Convolution layers that have 30,30,40,50 and 50 filters respectfully.

6 RESULTS AND EVALUATION

Previously in chapter 5, the non-equidistant temporal sampling technique based on the exponential and quadratic spacing functions was introduced. The technique was developed based on a preliminary study in chapter 4, where the effect of applying window functions to the input window was investigated. The goal is to improve the disaggregation performance with a fixed the sequence length parameter according to the one-for-all strategy explained in the chapter 2. Due to fluctuations observed in estimations, an analysis is performed in this chapter to define a results margin, then evaluate the findings of performing experiments using non-equidistant temporal sampling on S2S and S2P optimization methods.

6.1 DEFINE RESULTS MARGIN

F1-score fluctuations were observed when analyzing the experimental results (figure 6.4 is an example for the observed fluctuations). Consequently, evaluating the performance was challenging. It was not clear whether the observed differences in the results were actually differences in performance or a consequence of the randomness used in initializing the model parameters of the neural network. Although the seed of python *Numpy* and *Tensorflow* libraries used to generate the initial weights of the neural network is set to a static value (1234), the parallelization of the GPU introduces randomness during the computation. Therefore, a results margin is defined. Beyond its value, the difference can be diagnosed with confidence as a difference in performance.

The margin value represents the standard deviation, which shows the variability of the individual values to the mean. In a normal distribution, there is acceptable probability that most measurements are within the standard deviation around the mean [AKS14]. Therefore, statistical normality test is employed to ensure normality distribution of the findings; qq-plot (Quantile-Quantile plot). The qq-plot is a typical graphical method used for comparing two probability distributions against each other, providing the comparison in the form of a scatter plot. A line of points shows a perfect fit to the distribution at a 45 degree angle. Deviations of the values from the drawn line indicate a deviation from the normal distribution.

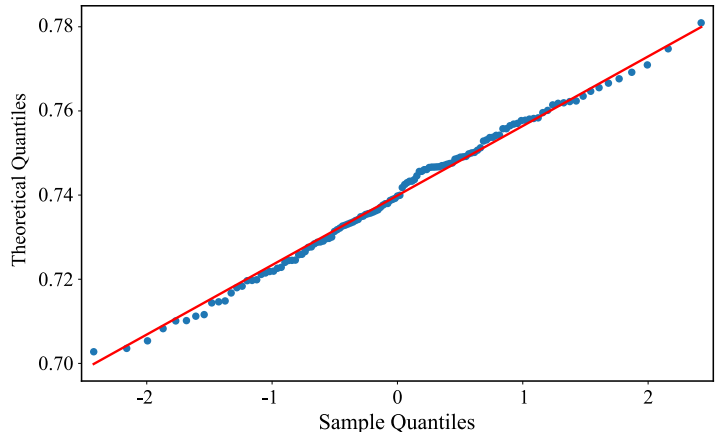


Figure 6.1: Quantile-Quantile plot indicates slight deviation of 100 NILM runs from the normal distribution (red line)

For performing the normality test, 100 NILM runs are performed with the same configurations. Figure 6.1 shows a slight deviation of the runs outputs from the normal distribution plotted using qq-plot. The standard deviation can be defined after confirming the normal distribution of the samples. For this purpose, a curve is fitted to a histogram in which the samples are sorted into bins. Each one contains the number of observations as figure 6.2 illustrates. The resulted curve has a standard deviation of 0.01523 (1.5 %). The margin is defined to be the resulted standard deviation value.

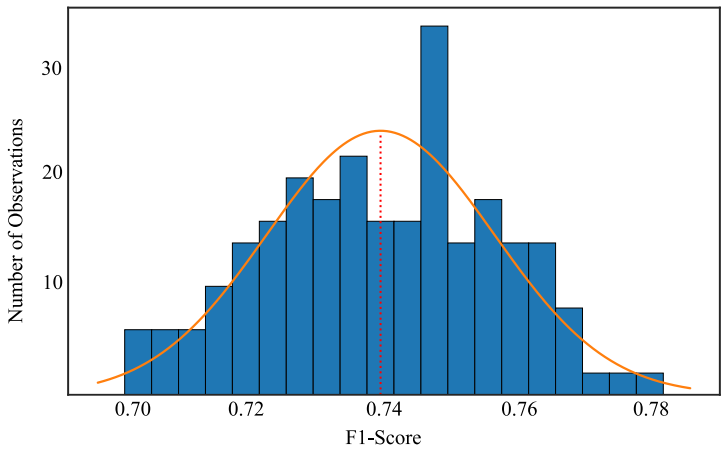


Figure 6.2: Normal distribution curve fitted to the histogram of 100 NILM rus

6.2 SEQUENCE-TO-SEQUENCE RESULTS

This section reports the results of load disaggregation using the Sequence-to-Sequence model. The results are separated into two categories based on the used time spacing function. Because of the wide range of function values evaluated, numerous figures are provided to outline the results. To ensure a smooth reading experience, a function value is discussed. This is followed by a summary of the spacing functions. The remaining results of the quadratic function are in appendix A.1 and in appendix A.2 are the remaining results of the exponential function.

6.2.1 *Quadratic Function*

In the following, we present a comparison between the different sampling functions in Sec. 5.3.3 and then go into the results of the non-equidistant temporal sampling using the quadratic function.

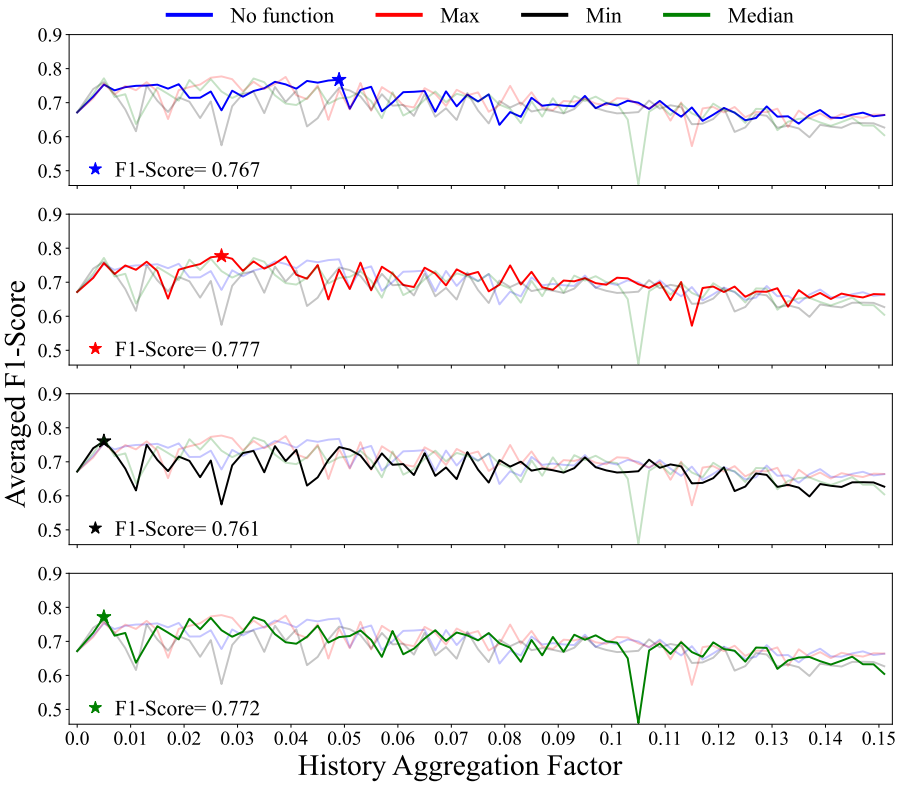
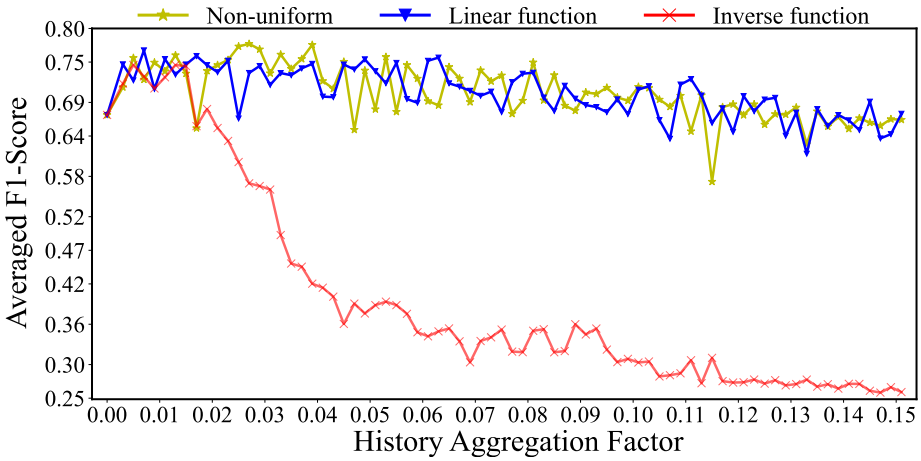


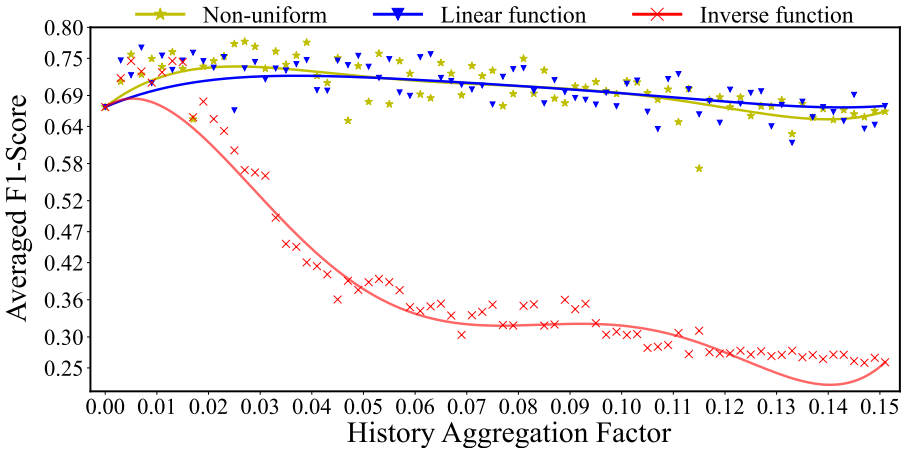
Figure 6.3: Comparison of different sampling functions on S2S

Figure 6.3 shows the performance of four sampling functions used with non-equidistant temporal sampling (Max, Min, Average, Median). Each curve represents a sampling function. The x-axis is

the aggregation factor, and the y-axis is the averaged F1-score of four devices: fridge, cooker, microwave, and washing machine. All functions are drawn transparently in each plot except the one in focus. Therefore, four graphs are displayed to demonstrate the behavior of the sampling functions. The best F1-score obtained by each curve is also annotated and the value is mentioned. The sampling functions scored F1-score in the range of {0.6, ..., 0.8} responding to the factors in the range of {0.003, ..., 0.15}. The highest scores are achieved with a factor of less than 0.5 (5 %). This means that an improvement can be achieved with narrow aggregated history. The function that has the best F1-score is the Max function, with a difference of 0.03 F1-score from the curve where no sampling function is applied. This behavior confirms the assumption that the Max function better represents a sequence of samples, since it can capture the events of the devices that affect the learning process.



(a) $t^{1.2}$ performance compared to the different variants of non-equidistant sampling



(b) Corresponding approximation utilizing a Bézier interpolation

Figure 6.4: Performance of $t^{1.2}$ function with non-equidistant techniques

Figure 6.4 compares the quadratic function $t^{1.2}$ as an example and the windowing functions proposed in Sec. 5.4; the linear and inverse functions. The curves show different behaviors. The linear function (blue) shows similar performance to the non-equidistant temporal sampling (green), while the inverse function converges to zero with a solid correlation to the aggregation factor showing a steep drop in F1-score from 0.75 to 0.4 within factor range of {0.2, ..., 0.4}. This behavior can be interpreted by recapitulating the main structure of the inverse function. The function creates an entirely non-equally spaced window, this could lead to a lack of information about the recent events due to the time intervals generated. The lack of information becomes apparently significant when higher aggregation factors are chosen, which explains the decrease in performance. The results are approximated and shown in figure 6.4 (b). Bézier interpolation is used to improve the clarity of the results. The approximation shows a peak at the beginning made by the non-equidistant temporal sampling scoring better than the other functions.

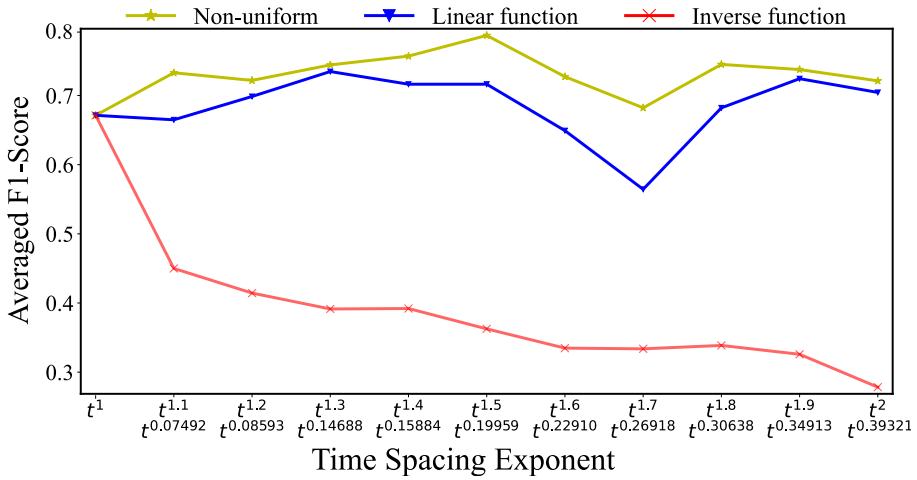


Figure 6.5: Results summary of quadratic function on S2S; The x-axis shows the exponents in quadratic function (first line), and the corresponding exponents used in the inverse function (second line)

Figure 6.5 illustrates the performance summary of the non-equidistant windowing using the quadratic function compared to the inverse and linear functions. The summary is obtained by collecting the F1-score from all exponents according to the best-scored parameter value (0.041 equivalent to 4 %). The x-axis handles the spacing function value representing the exponent in quadratic function (first line). The axis also shows the corresponding exponents used in the inverse function to produce a full non-equally spaced window using quadratic function, which has an equivalent effective window length (second line). The summary shows satisfactory performance of non-equidistant

temporal sampling over the linear and inverse functions exceeding the specified margin (1.5 %) at most exponents. It is necessary to mention that the function t^1 on the x-axis presents the uniform windowing, as it shows the difference in performance between the traditional uniform windowing used by default in NILMTK and the proposed technique. The aggregation performance is enhanced when the window presents additional historical data points to the learning model.

6.2.2 *Exponential Function*

Parallel to the quadratic function, in the following, the results of the experiments on non-equidistant temporal sampling are presented, showing the impact of providing the S2S algorithm with historical information available collected according to the exponential function.

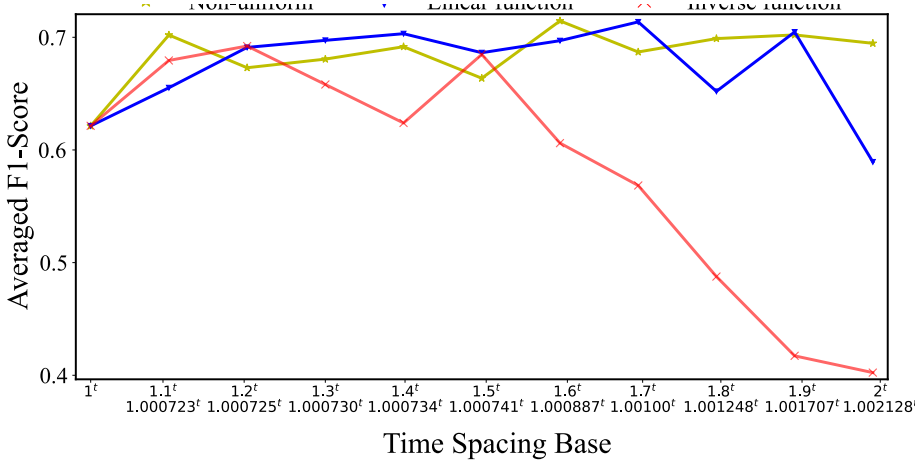


Figure 6.6: Results summary of exponential function on S2S

Figure 6.6 summarizes the exponential function results performed on range of the base values $\{0.1, \ldots, 2\}$ using S2S. The x-axis shows this range and the corresponding base values used in the inverse function to generate an entire non-equally spaced window with equivalent ESL. In contrast to the previously introduced summary of the quadratic function findings, the F1-score of the non-equidistant windowing based on exponential function falls under the linear function (blue curve) at most base values. Generally speaking, no dominant performance can be observed. The reason may lie in the nature of the spacing function used. The exponential function grows rapidly compared to the quadratic function, which leads to large distances between samples that a single value cannot represent even when sampling functions are used, while the linear function has fixed time spacing between samples.

6.3 SEQUENCE-TO-POINT RESULTS

In this section, we report the load disaggregation findings of employing the non-equidistant technique for sampling input data points when the S2P optimization method is used. Two categories show the summary of results according to the time spacing function used. The remaining results of the function values are shown in the appendix B.1 for the quadratic and in appendix B.2 for the exponential functions.

6.3.1 *Quadratic Function*

Figure 6.7 illustrates a comparison of four sampling functions. As before, each curve represents a sampling function. According to the annotations, the maximum F1-score over all is achieved by a factor of less than 0.6 (6 %). This confirms the assumptions regarding the history interval required to achieve an improved F1-score. Using small values for the aggregation factor, aggregation results can be improved. The highest scoring function is the Max function with a difference of 0.02 F1-score compared to the curve, where no function is applied, otherwise, all the functions are similar in terms of the best F1-score.

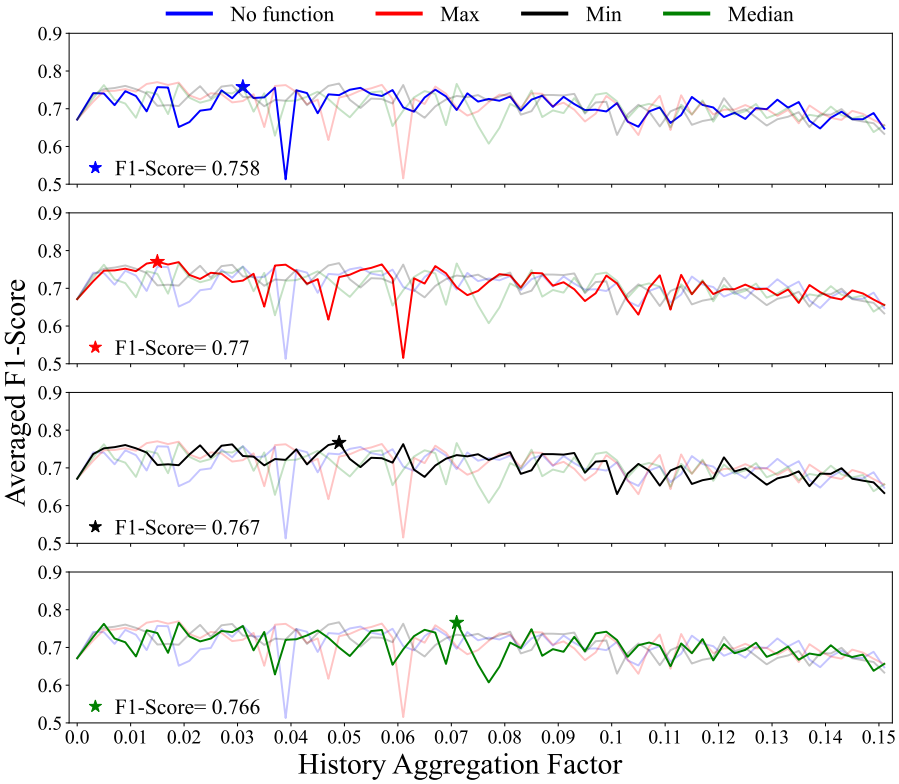


Figure 6.7: Comparison of four different sampling functions on S2P

Figure 6.8 demonstrates the summary of results from experiments in which the Sequence-to-Point algorithm was fitted with non-equidistant spaced windows using the quadratic function. The y-axis represents the averaged F1-score of load disaggregation, while the x-axis indicates the evaluated exponents and the corresponding exponents used in the inverse function to produce non-equally spaced window that has the same effective sequence length. The summary is obtained by collecting the F1-score of all exponents according to the best-scored factor. The inverse function exhibits similar behavior observed in the S2P results concerning the correlation between the F1-score and the function exponent.

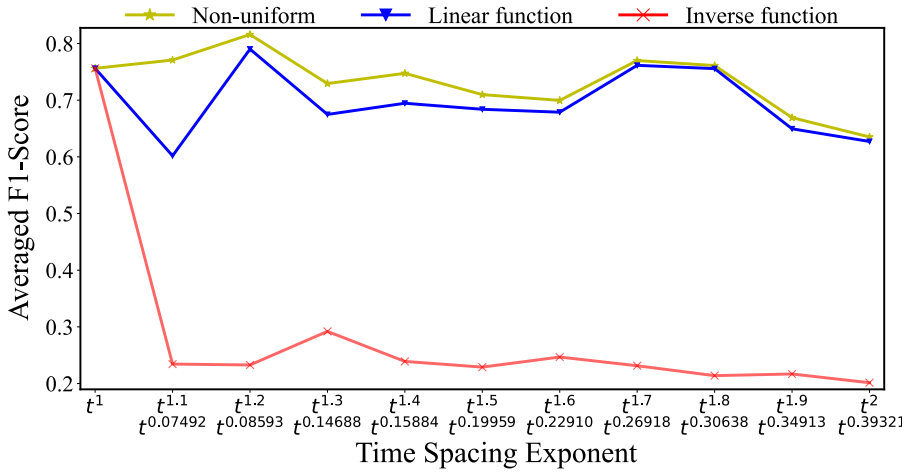


Figure 6.8: Results summary of quadratic function on S2P. The x-axis indicates the evaluated exponents and the corresponding exponents used in the inverse function to produce non-equally spaced window

The plot shows also F1-score differences between the blue and green lines representing linear and the non-equidistant temporal sampling. Later, the dissimilarity becomes smaller at higher exponents (less than the margin of 1.5 %). On the one hand, this confirms the assumption that there is no dominant performance with a significant F1-score; both lines generally show a slight difference in performance. On the other hand, considering the exponent t^1, which stands for the traditional uniform windowing used in NILMTK, proves the improvement obtained by employing time spacing functions as a windowing technique.

6.3.2 *Exponential Function*

The evaluation of the experiments based on non-equidistant temporal sampling with S2P is given below. Due to the six-month deadline for submission the thesis, not all findings of the exponential base values can be reported. The summary of the results is presented in the follow-

ing while the detailed findings of the individual function values can be found in appendix B.2.

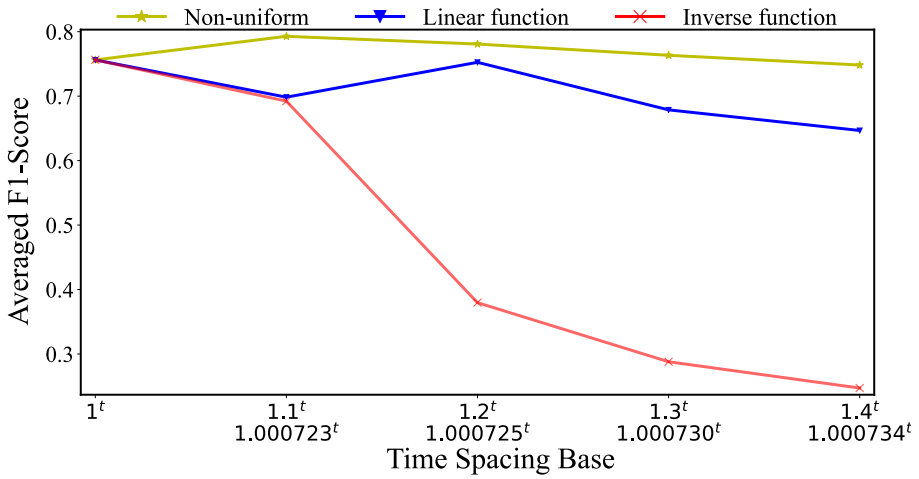


Figure 6.9: Results summary of four exponential functions on S2P

The figure shows the F1-score of performing experiments with the exponential function on S2P. Four function values are shown on the x-axis with the corresponding function base used in the inverse function. Despite the significant performance over the linear and inverse functions obtained with the non-equidistant temporal sampling (green line), it is difficult to conclude that the exponential function performs remarkably better due to the lack of evaluation of remaining function values on S2P.

6.4 SUMMARY

This chapter describes experiments performed on DRED, in which non-equidistant spaced data points are collected according to various non-linear algebraic functions. The results are divided based on the optimization method used to train the neural network. A comparison is also made to measure the effects of using statistical functions, such as max and min functions, on disaggregation performance. The performance of the experiments is evaluated using the F1-score metric. In general, the results have shown that the non-equidistant temporal sampling technique has an impact on the disaggregation of the load in the S2S and S2P learning models. The results revealed no dominant performance of any of the evaluated functions, while the max function applied to the time intervals achieved a better representation of the samples compared to other statistical functions.

The Sequence-to-Sequence results showed F1-score improvement (approximately 0.1) in load disaggregation over the uniform windowing used in NILMTK. The non-equidistant temporal sampling with the quadratic function generally performed better than other variants of non-equidistant sampling. Using the exponential function provided similar performance to the linear function, while a clear correlation was observed between the aggregation factor and the inverse function.

Using the quadratic function with Sequence-to-Point reports satisfactory but not a significantly better performance compared to the other variants of non-equidistant sampling, as the results show coherence with the linear function at high aggregation factors. In parallel to the S2S results, there is an F1-score improvement (approximately 0.06) for non-equidistant sampling over the traditional windowing technique, where no time spacing between samples is established.

As a summary, numerous Experiments are performed on S2S and S2P learning models. We proposed modifying the traditional windowing technique to allow the window to have more historical information using non-linear algebraic functions. The proposed method is parametrized and compared to different variants of non-equidistant sampling to ensure fair evaluation. After analysing the results, it is clear that aggregating historical data points within fixes window length can ensure load disaggregation improvement and maintain the model complexity. However, the data can be sampled through quadratic or linear functions since no dominant behavior was observed.

We can say that the proposed methodology achieved the research objective. Presenting historical data points to the network improved the results while using the one-for-all strategy for setting the window length parameter; nonetheless, the non-equidistant manner of collecting the data points has contributed less to the improvement of the results since the fixed step used in the linear function shows similar performance to the quadratic function.

7 CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

This thesis proposes the non-equidistant temporal sampling technique to sample the input data from the aggregated load signal representing the appliances consumption. The technique combines the input data into two sequences according to a predefined fraction; uniform and non-equidistant time-spaced. The non-uniformity of the the time series is established by means of non-linear algebraic functions. The goal is to investigate the impact on the load disaggregation when the input window contains historical data points. In addition, the use of statistical functions is presented to increase the representation of the samples located within the time interval produced by the spacing functions.

Moreover, two functions are presented to ensure a fair evaluation of the proposed method: the inverse function, which creates a window with non-equally time spacing between samples, and the linear downsampling, where a fraction of the window is formed with uniformly distributed samples. Both functions cover the same effective window length as the proposed technique.

The experiments are conducted to compare the performance of the different windowing techniques on DRED, which was first preprocessed and converted to a vectorized representation using a parameterized windowing technique. It was then fed into a deep learning model to estimate the individual consumption signal from the aggregated load. The F1-score metric is used to measure model performance optimized by Sequence-to-Sequence and Sequence-to-Point methods. The investigation is conducted among a predefined range of spacing function exponent and aggregation factor to explore possible combinations and study their effects. The experiments were distributed and automated to run on multiple GPU machines to accelerate the computation.

The comparative assessment of the results confirm a positive impact of the aggregating of historical information on load disaggregation. The promising potential of the proposed technique shown during the evaluation can make a step forward in the NILM field, as collecting historical input data according to a non-equidistant temporal sampling

or linear manner ensure up to 0.1 improvement in F1-score without affecting the model complexity.

7.2 FUTURE WORK

For short-term research, the effect of non-equidistant temporal sampling on a different dataset can be investigated, where the training and testing phases can be performed on different buildings. Since the window length used in this study has been previously defined empirically in the literature through experiments on DRED, a study can be conducted in the long term to describe a procedure to theoretically define an appropriate window length when a different energy dataset is used.

BIBLIOGRAPHY

[AKS14] M. Ahsanullah, B. Kibria, and M. Shakil. "Normal distribution". In: *Normal and Student st Distributions and Their Applications*. Springer, 2014, pp. 7–50.

[Arm+13] K. C. Armel, A. Gupta, G. Shrimali, and A. Albert. "Is disaggregation the holy grail of energy efficiency? The case of electricity". In: *Energy Policy* 52 (2013), pp. 213–234.

[Bat+14] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava. "NILMTK: An open source toolkit for non-intrusive load monitoring". In: *Proceedings of the 5th international conference on Future energy systems*. 2014, pp. 265–276.

[Bat+19] N. Batra, R. Kukunuri, A. Pandey, R. Malakar, R. Kumar, O. Krystalakos, M. Zhong, P. Meira, and O. Parson. "Towards reproducible state-of-the-art energy disaggregation". In: *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*. 2019, pp. 193–202.

[BB86] R. N. Bracewell and R. N. Bracewell. *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York, 1986.

[Che+18] K. Chen, Q. Wang, Z. He, K. Chen, J. Hu, and J. He. "Convolutional sequence to sequence non-intrusive load monitoring". In: *Journal of Engineering* 17 (2018), pp. 1860–1864.

[Din+21] D. Ding, J. Li, K. Zhang, H. Wang, K. Wang, and T. Cao. "Non-intrusive load monitoring method with inception structured CNN". In: *Applied Intelligence* (2021), pp. 1–18.

[DSZ19] M. D'Incecco, S. Squartini, and M. Zhong. "Transfer learning for non-intrusive load monitoring". In: *IEEE Transactions on Smart Grid* 11.2 (2019), pp. 1419–1429.

[EMDL+10] K. Ehrhardt-Martinez, K. A. Donnelly, S. Laitner, et al. "Advanced metering initiatives and residential feedback programs: a meta-review for household electricity-saving opportunities". In: American Council for an Energy-Efficient Economy Washington, DC. 2010.

[Fau+17] A. Faustine, N. H. Mvungi, S. Kaijage, and K. Michael. "A survey on non-intrusive load monitoring methodies and techniques for energy disaggregation problem". In: *arXiv preprint arXiv:1703.00785* (2017).

[Fau+20] A. Faustine, L. Pereira, H. Bousbiat, and S. Kulkarni. "UNet-NILM: A deep neural network for multi-tasks appliances state detection and power estimation in NILM". In: *Proceedings of the 5th international workshop on non-intrusive load monitoring*. 2020, pp. 84–88.

[Ful17] B. D. Fulcher. "Feature-based time-series analysis". In: *arXiv preprint arXiv:1709.08055* (2017).

[GEA] German-Environment-Agency. *Emission of greenhouse gases covered by the UN Framework Convention on Climate.* [Online: accessed 10-Feb-2022]. URL: https://www.umweltbundesamt.de/en/data/environmental-indicators/indicator-greenhouse-gas-emissions#at-a-glance.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[Har92] G. W. Hart. "Nonintrusive appliance load monitoring". In: *Proceedings of the IEEE* 80.12 (1992), pp. 1870–1891.

[HHR89] P. Horowitz, W. Hill, and I. Robinson. *The art of electronics*. Vol. 2. Cambridge university press, 1989.

[IF+19] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. "Deep learning for time series classification: a review". In: *Data mining and knowledge discovery* 33.4 (2019), pp. 917–963.

[KA72] E Kan and J Aggarwal. "Multirate digital filtering". In: *IEEE Transactions on Audio and Electroacoustics* 20.3 (1972), pp. 223–225.

[Kel] J. Kelly. *Neural NILM: Implementation repository e578.py*. [Online: accessed 04-Feb-2022]. URL: https://github.com/JackKelly/neuralnilm/tree/master/experiment_definitions.

[KK15a] J. Kelly and W. Knottenbelt. "Neural nilm: Deep neural networks applied to energy disaggregation". In: *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*. 2015, pp. 55–64.

[KK15b] J. Kelly and W. Knottenbelt. "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes". In: *Scientific data* 2 (2015), p. 150007.

[KJ09] U. Kiencke and H. Jäkel. *Signale und Systeme*. Oldenbourg Wissenschaftsverlag, 2009.

[KJ11] J. Z. Kolter and M. J. Johnson. "REDD: A public data set for energy disaggregation research". In: *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*. Vol. 25. CiteSeerX. 2011, pp. 59–62.

[KNV18] O. Krystalakos, C. Nalmpantis, and D. Vrakas. "Sliding window approach for online energy disaggregation using artificial neural networks". In: *Proceedings of the 10th hellenic conference on artificial intelligence*. 2018, pp. 1–6.

[Lia+19] J. Liang, Z. Ren, L. Wang, B. Tang, J. Liu, and Y. Liu. "Deep neural network in sequence to short sequence form for non-intrusive load monitoring". In: *3rd Conference on Energy Internet and Energy System Integration (EI2)*. IEEE. 2019, pp. 565–570.

[Liu19] H. Liu. *Non-intrusive Load Monitoring: Theory, Technologies and Applications*. Springer Nature, 2019.

[MSS17] D. Murray, L. Stankovic, and V. Stankovic. "An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study". In: *Scientific data* 4.1 (2017), pp. 1–12.

[RG75] L. R. Rabiner and B. Gold. "Theory and application of digital signal processing". In: *Prentice Hall* (1975).

[RB20] A. Reinhardt and M. Bouchur. "On the impact of the sequence length on sequence-to-sequence and sequence-to-point learning for nilm". In: *Proceedings of the 5th international workshop on non-Intrusive load monitoring*. 2020, pp. 75–78.

[RK20] A. Reinhardt and C. Klemenjak. "How does load disaggregation performance depend on data characteristics? insights from a benchmarking study". In: *Proceedings of the eleventh ACM international conference on future energy systems*. 2020, pp. 167–177.

[RK09] G. Rizzoni and J. Kearns. *Fundamentals of electrical engineering*. McGraw-Hill New York, 2009.

[Son+21] J. Song, H. Wang, M. Du, L. Peng, S. Zhang, and G. Xu. "Non-Intrusive Load Identification Method Based on Improved Long Short Term Memory Network". In: *Energies* 14.3 (2021), p. 684.

[UNRLP15] A. S. Uttama Nambi, A. Reyes Lua, and V. R. Prasad. "Loced: Location-aware energy disaggregation framework". In: *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*. 2015, pp. 45–54.

[WML05] Q. Wang, V. Megalooikonomou, and G. Li. "A symbolic representation of time series". In: *Proceedings of the eighth international symposium on signal processing and its applications*. Vol. 2. CiteSeerX. 2005, pp. 655–658.

[YLL21] M. Yang, X. Li, and Y. Liu. "Sequence to point learning based on an attention neural network for nonintrusive load decomposition". In: *Electronics* 10.14 (2021), p. 1657.

[Zha+18] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton. "Sequence-to-point learning with neural networks for non-intrusive load monitoring". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[ZC18] A. Zheng and A. Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.

[Zho+20] G. Zhou, Z. Li, M. Fu, Y. Feng, X. Wang, and C. Huang. "Sequence-to-Sequence Load Disaggregation Using Multiscale Residual Neural Network". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2020), pp. 1–10.

A

SEQUENCE-TO-SEQUENCE RESULTS

A.1 QUADRATIC FUNCTION

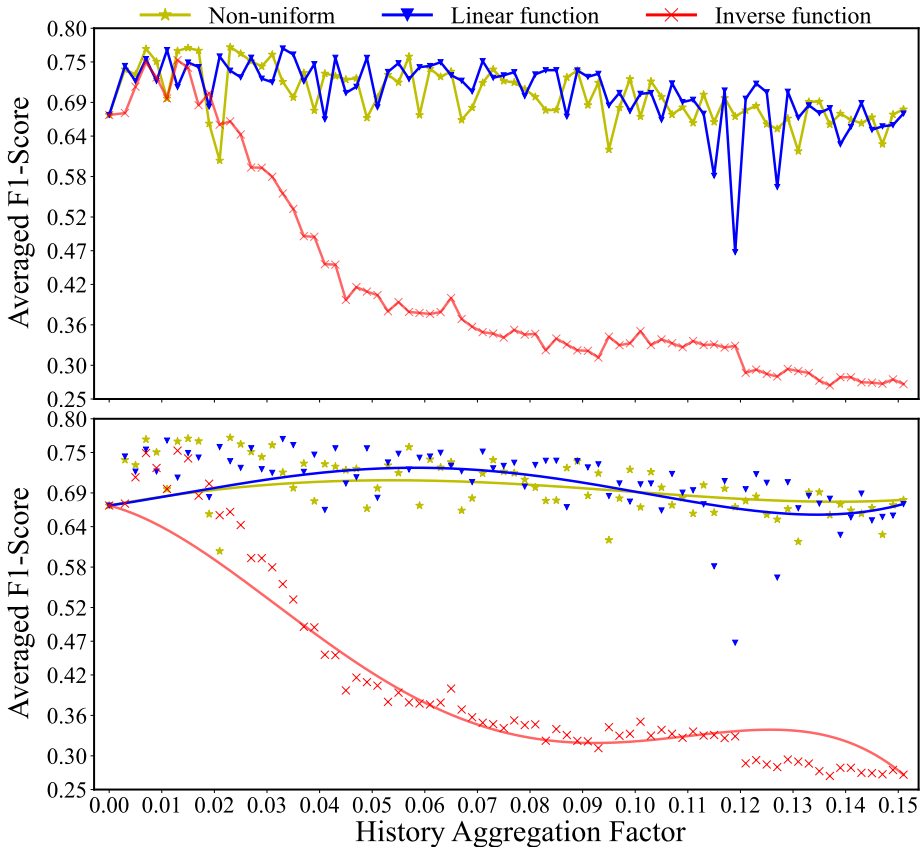


Figure A.1: Evaluating the function $t^{1.1}$ performance with the baseline

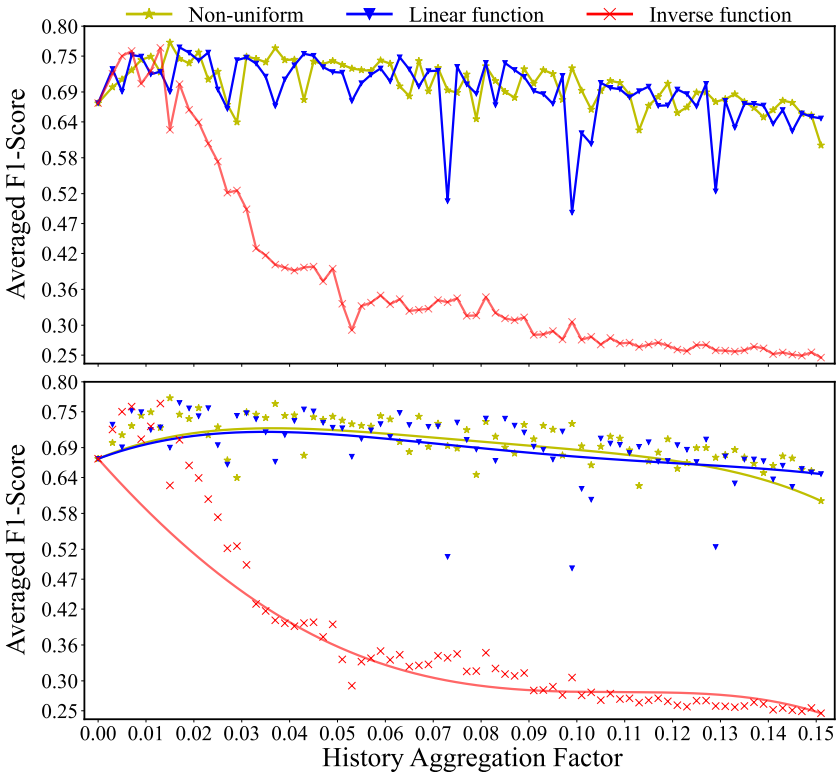


Figure A.2: Evaluating the function $t^{1.3}$ performance with the baseline

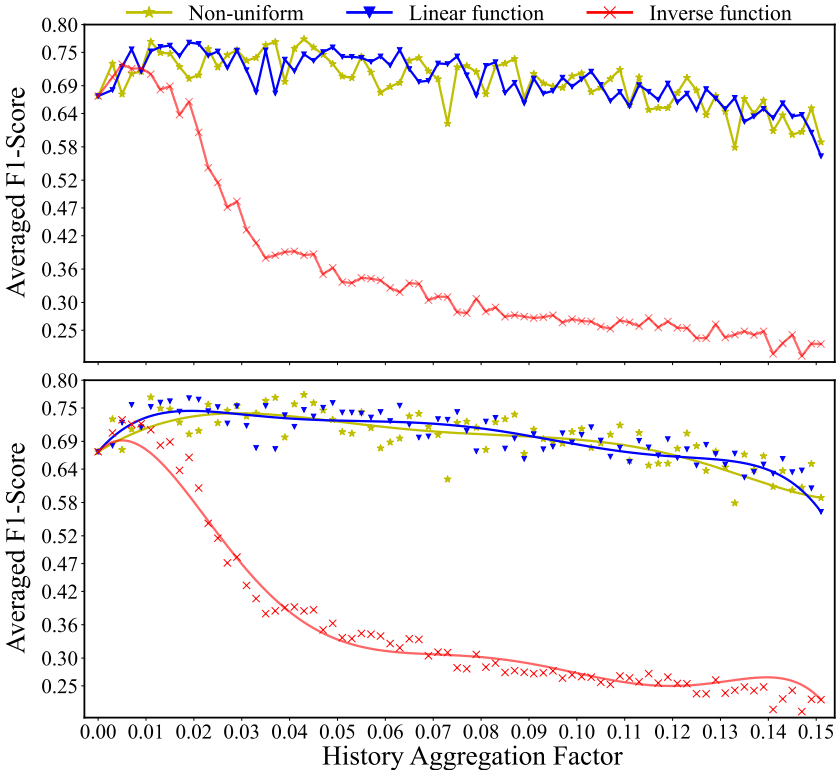


Figure A.3: Evaluating the function $t^{1.4}$ performance with the baseline

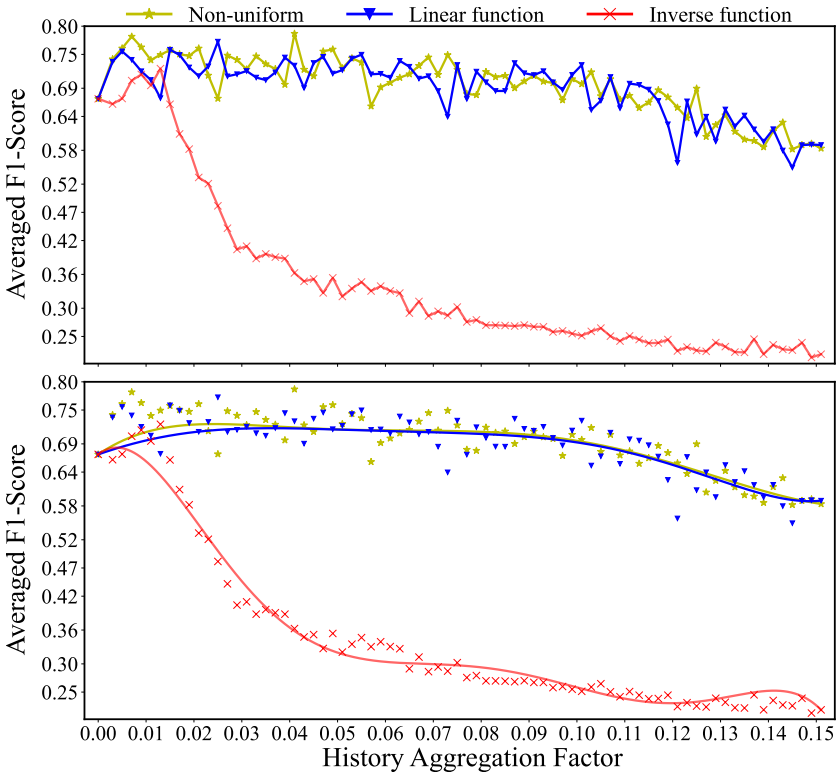


Figure A.4: Evaluating the function $t^{1.5}$ performance with the baseline

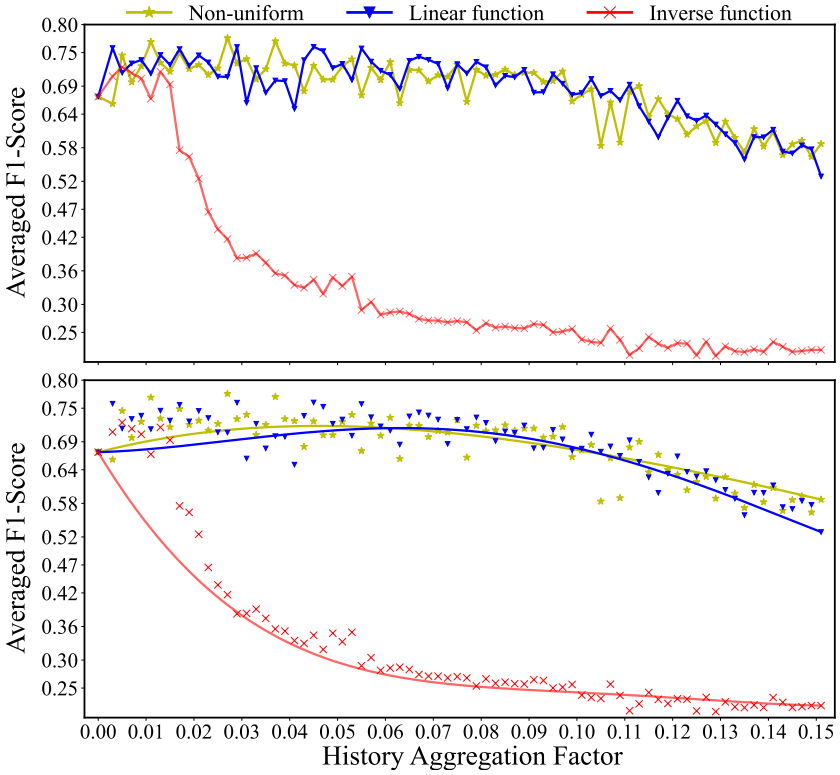


Figure A.5: Evaluating the function $t^{1.6}$ performance with the baseline

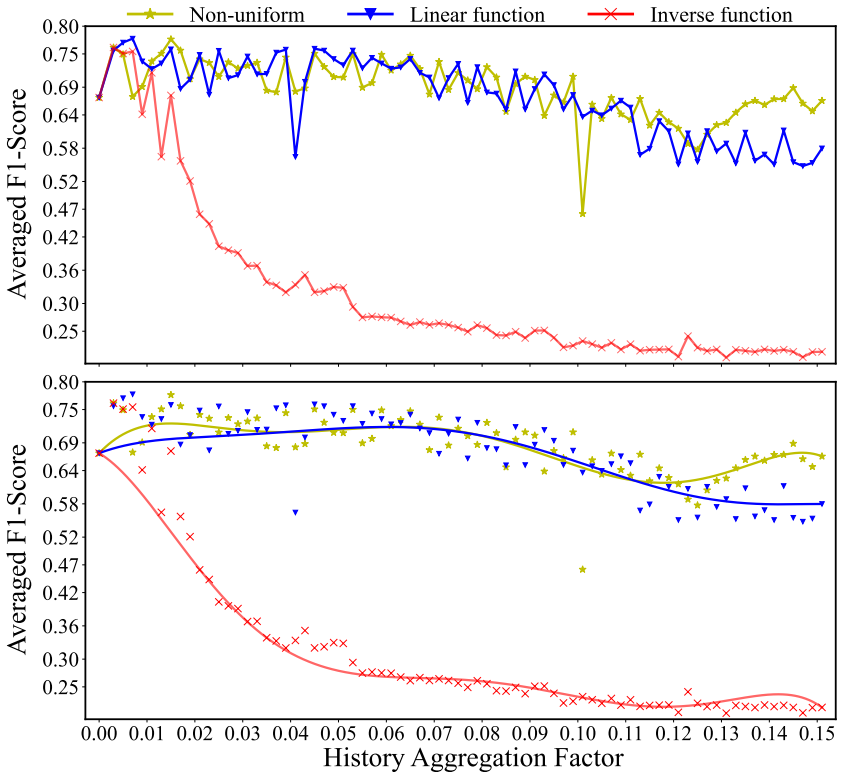


Figure A.6: Evaluating the function $t^{1.7}$ performance with the baseline

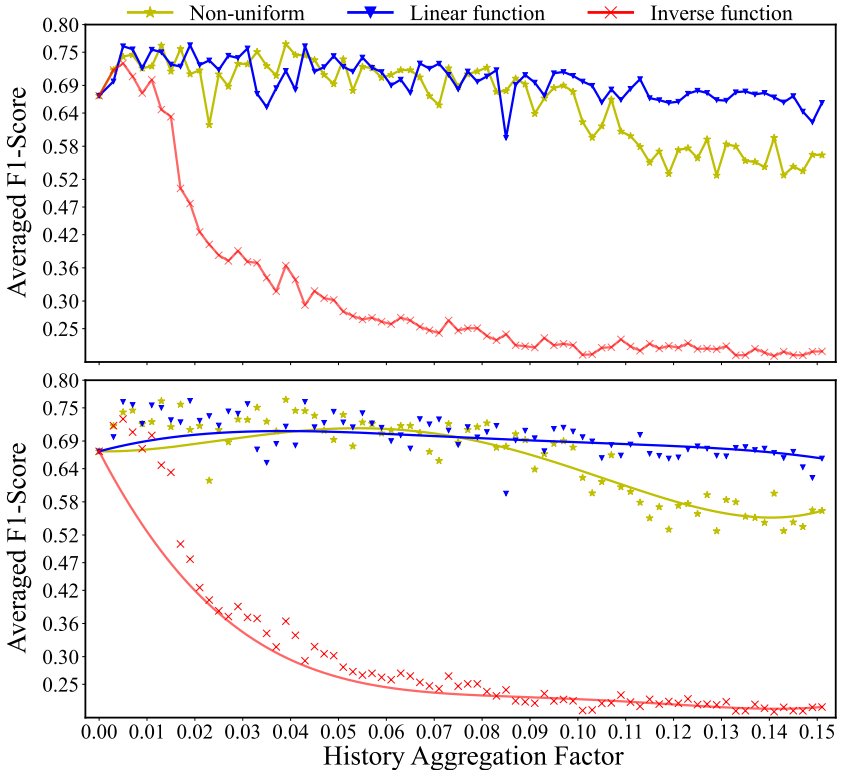


Figure A.7: Evaluating the function $t^{1.8}$ performance with the baseline

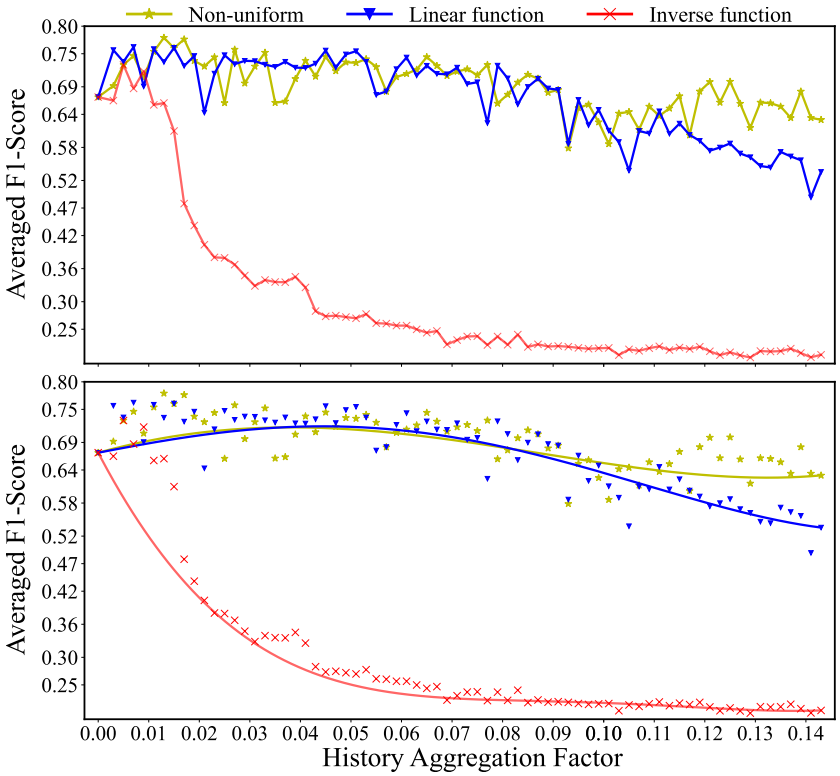


Figure A.8: Evaluating the function $t^{1.9}$ performance with the baseline

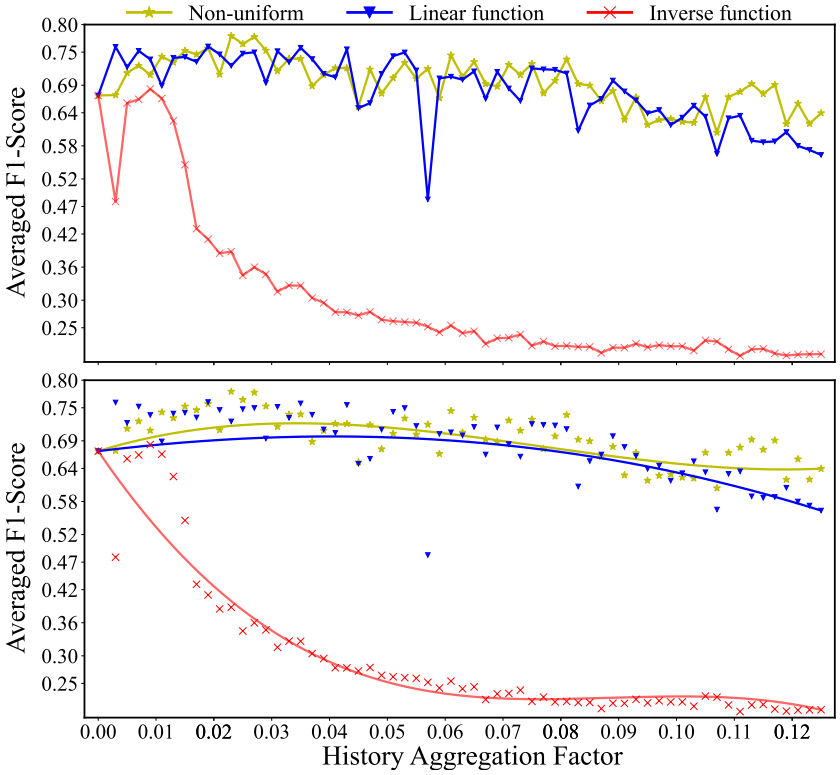


Figure A.9: Evaluating the function $t^{2.0}$ performance with the baseline

A.2 EXPONENTIAL FUNCTION

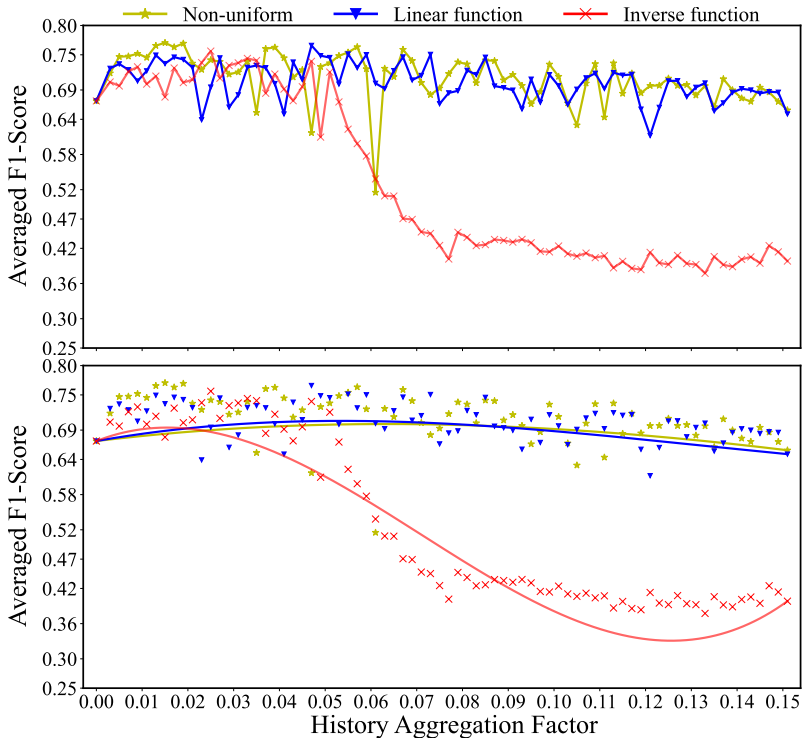


Figure A.10: Evaluating the function 1.1^t performance with the baseline

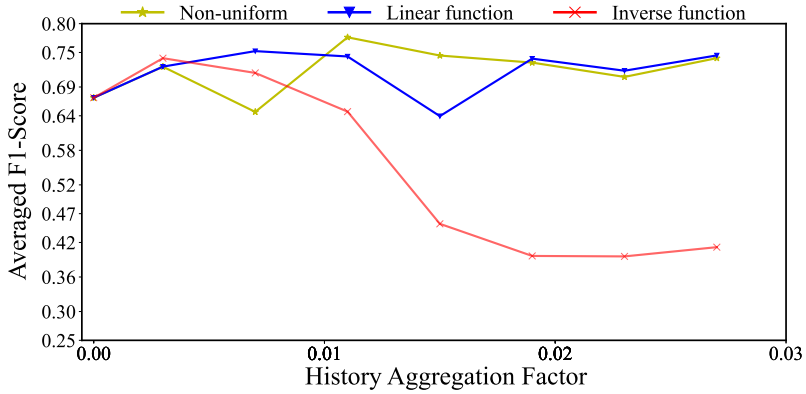


Figure A.11: Evaluating the function 2^t performance with the baseline

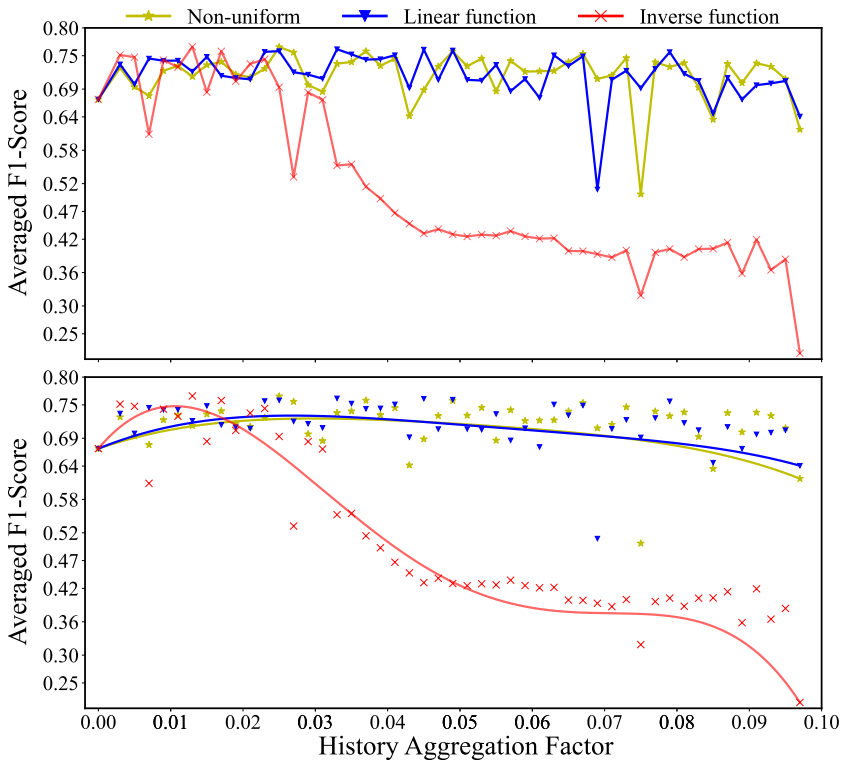


Figure A.12: Evaluating the function 1.2^t performance with the baseline

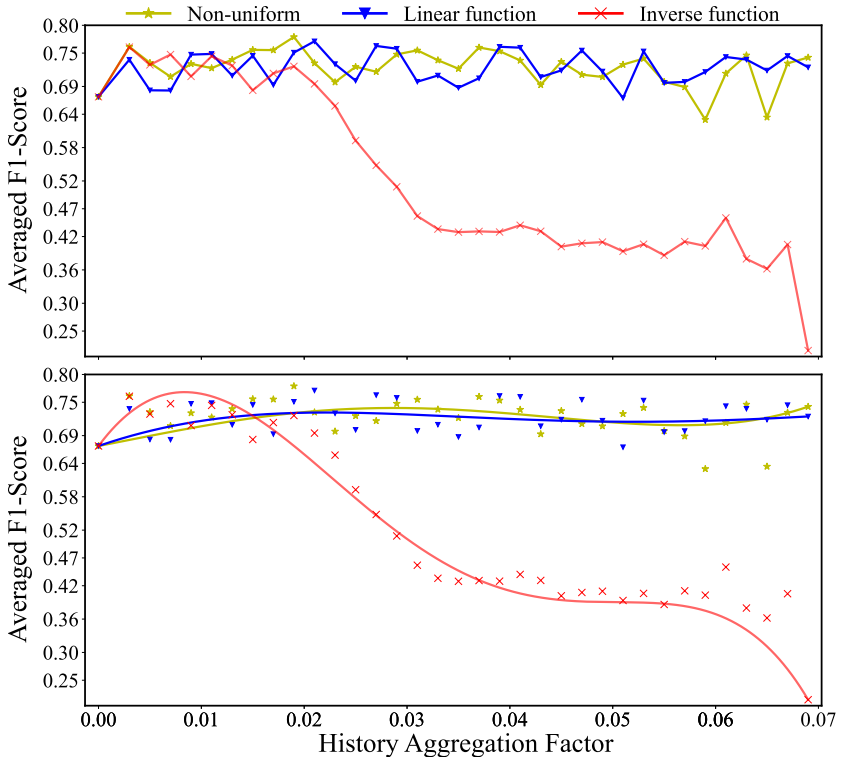


Figure A.13: Evaluating the function 1.3^t performance with the baseline

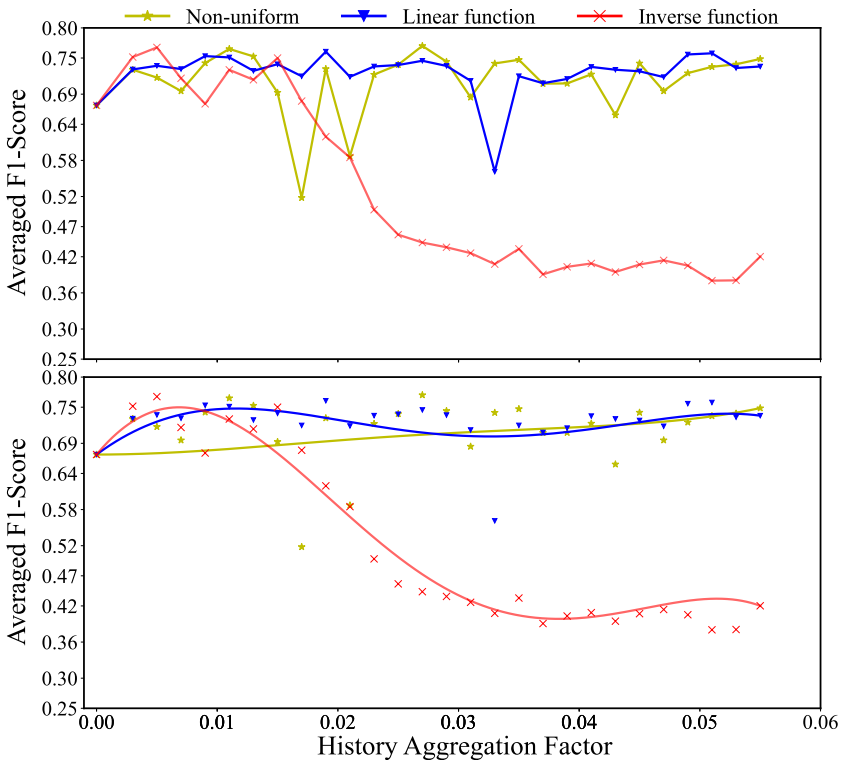


Figure A.14: Evaluating the function 1.4^t performance with the baseline

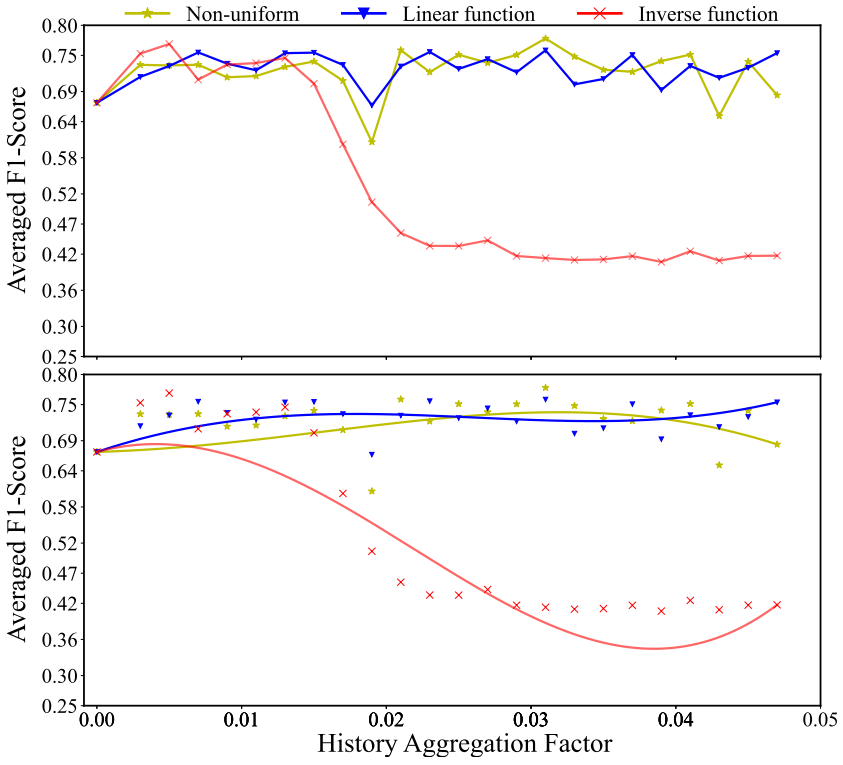


Figure A.15: Evaluating the function 1.5^t performance with the baseline

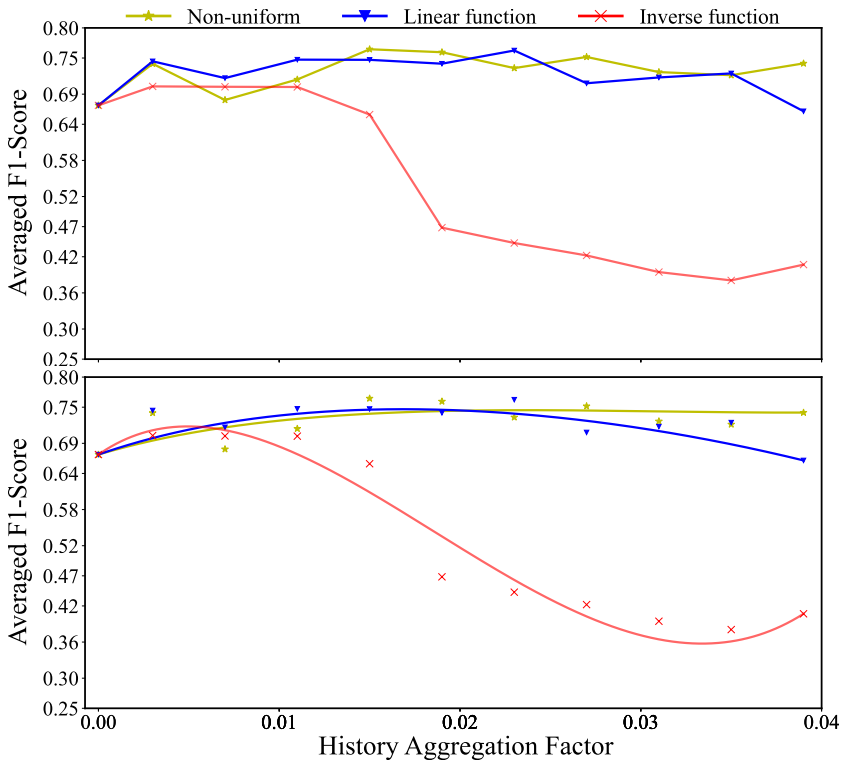


Figure A.16: Evaluating the function 1.6^t performance with the baseline

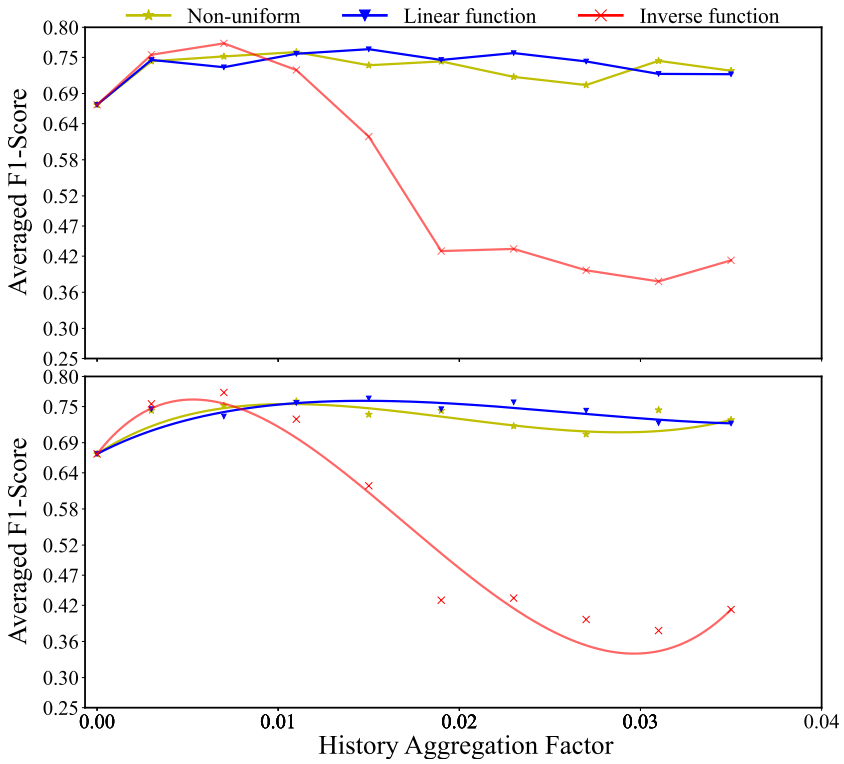


Figure A.17: Evaluating the function 1.7^t performance with the baseline

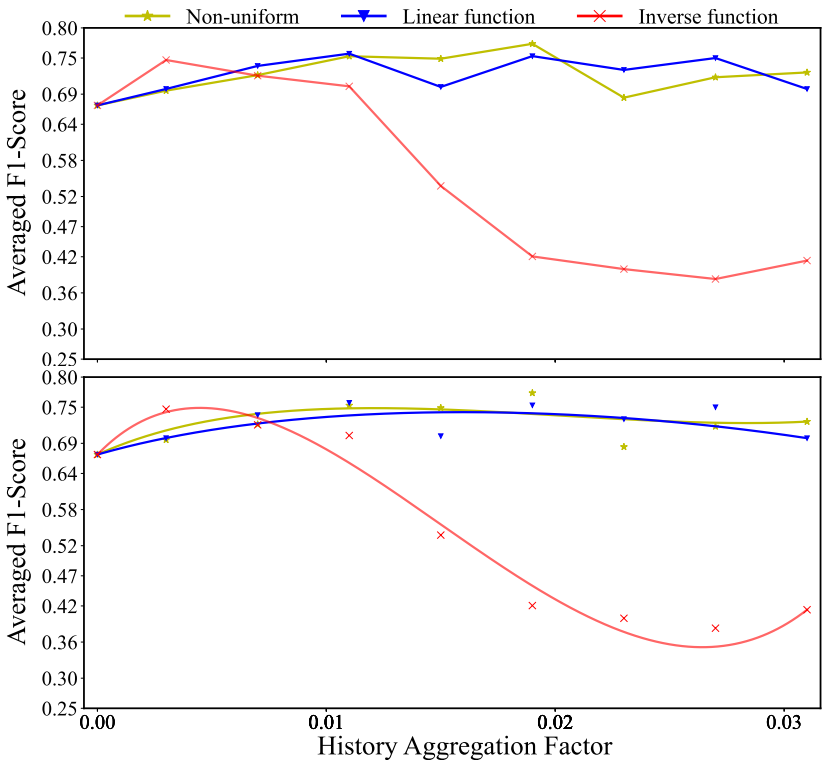


Figure A.18: Evaluating the function 1.8^t performance with the baseline

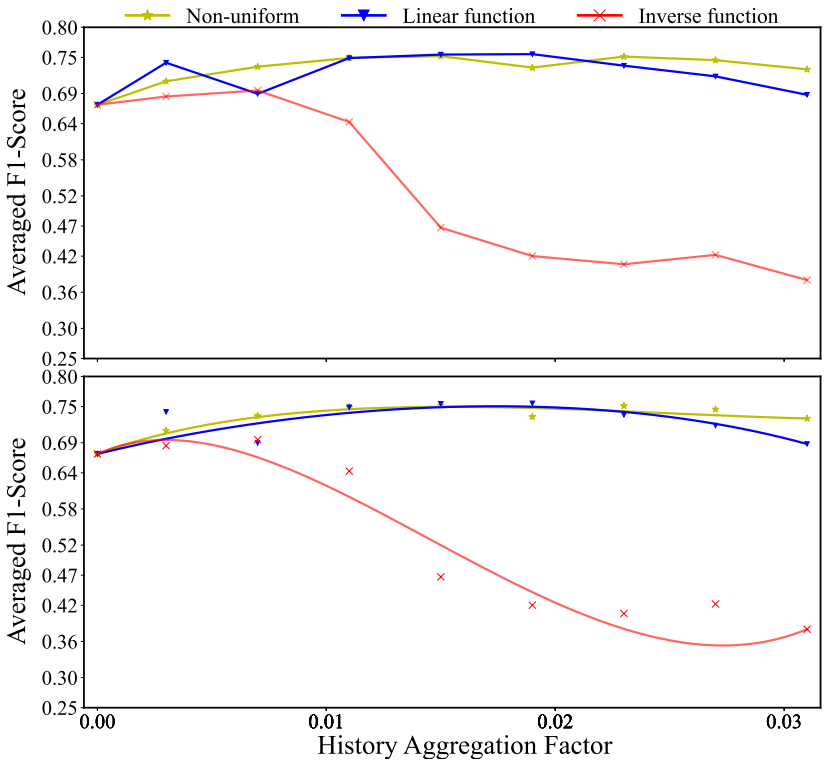


Figure A.19: Evaluating the function 1.9^t performance with the baseline

B

SEQUENCE-TO-POINT RESULTS

B.1 QUADRATIC FUNCTION

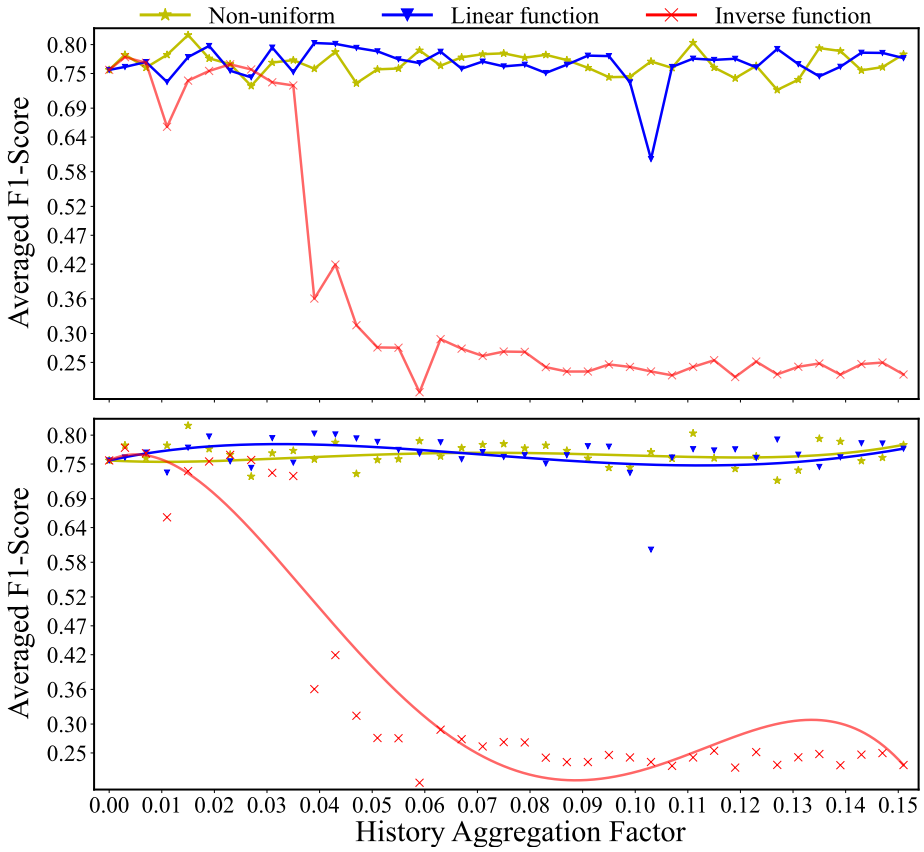


Figure B.1: Evaluating the function $t^{1.1}$ performance with the baseline

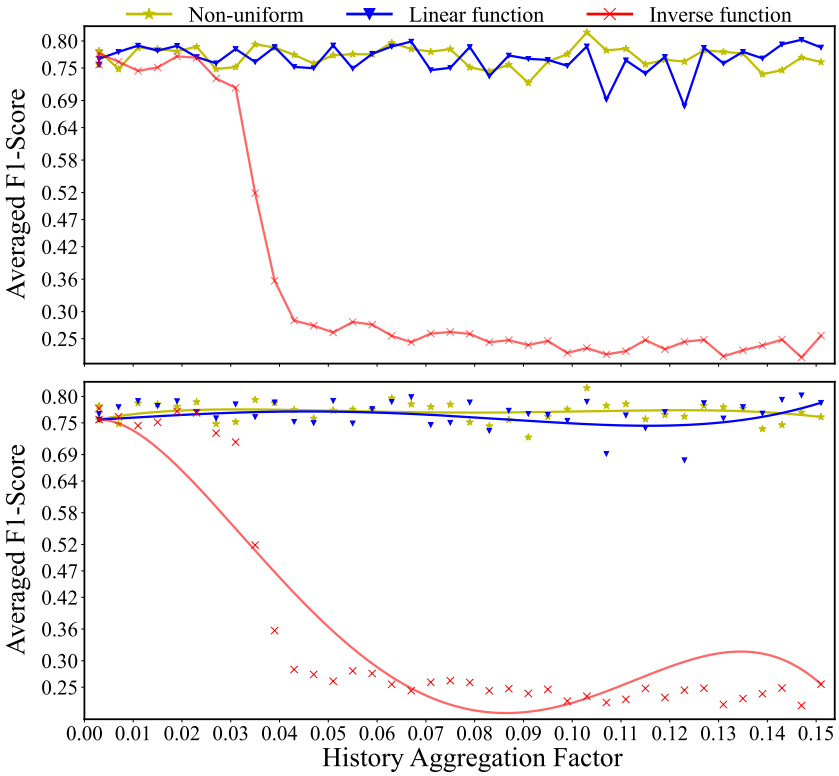


Figure B.2: Evaluating the function $t^{1.2}$ performance with the baseline

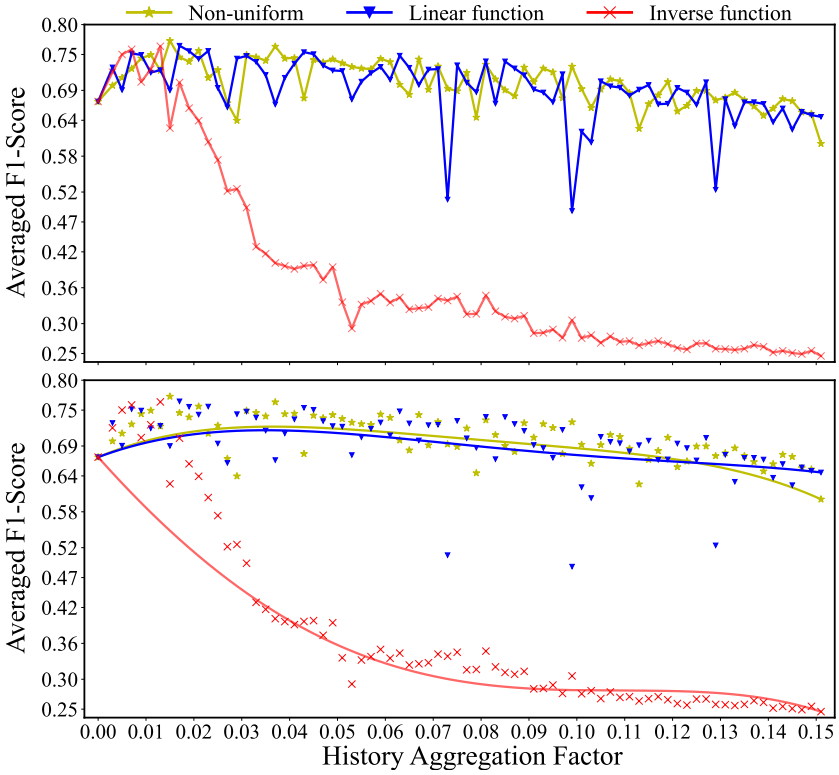


Figure B.3: Evaluating the function $t^{1.3}$ performance with the baseline

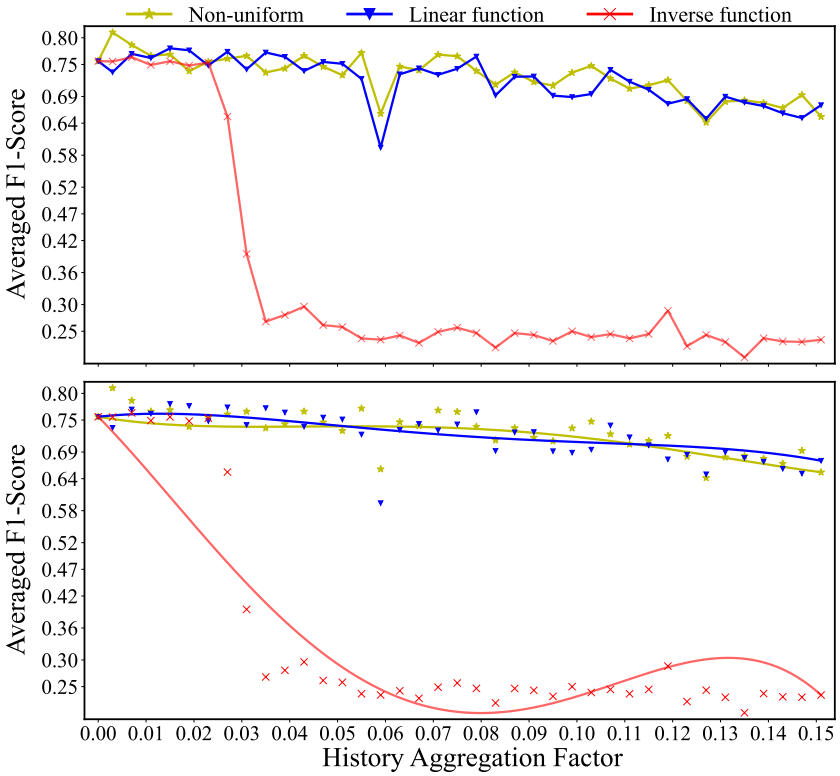


Figure B.4: Evaluating the function $t^{1.4}$ performance with the baseline

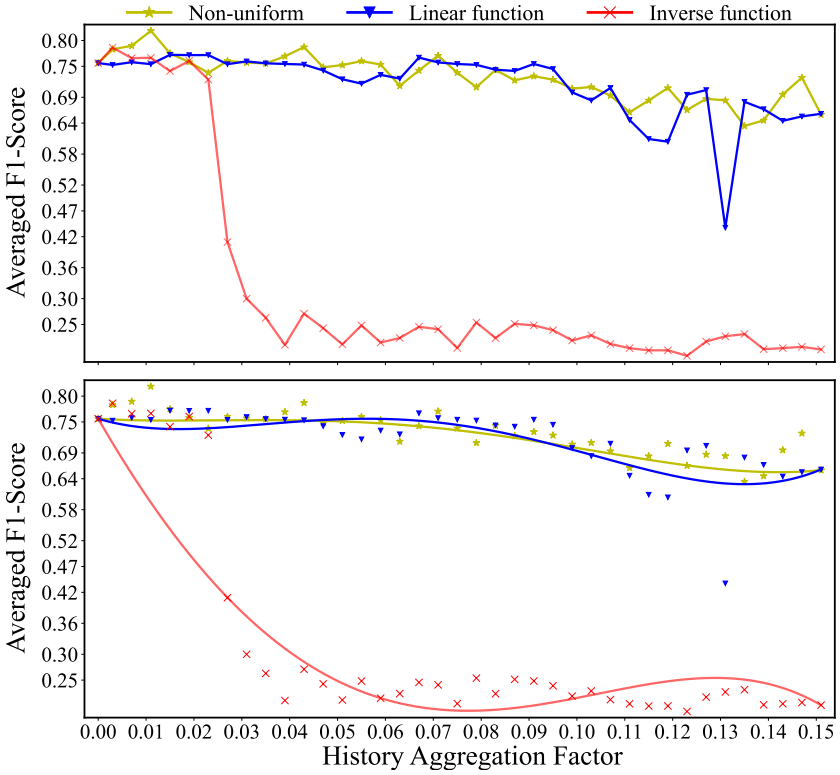


Figure B.5: Evaluating the function $t^{1.5}$ performance with the baseline

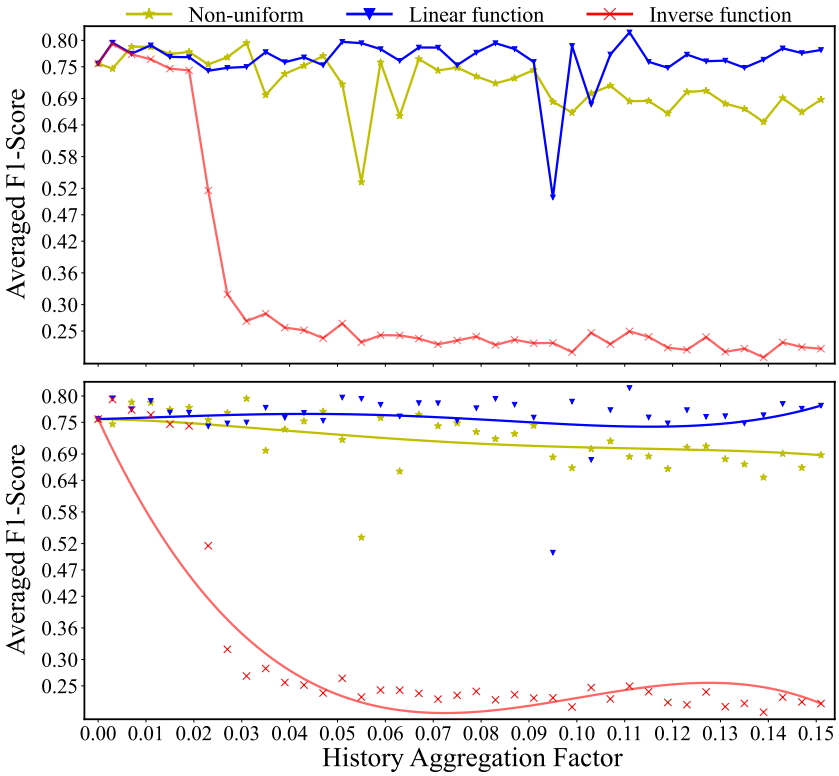


Figure B.6: Evaluating the function $t^{1.6}$ performance with the baseline

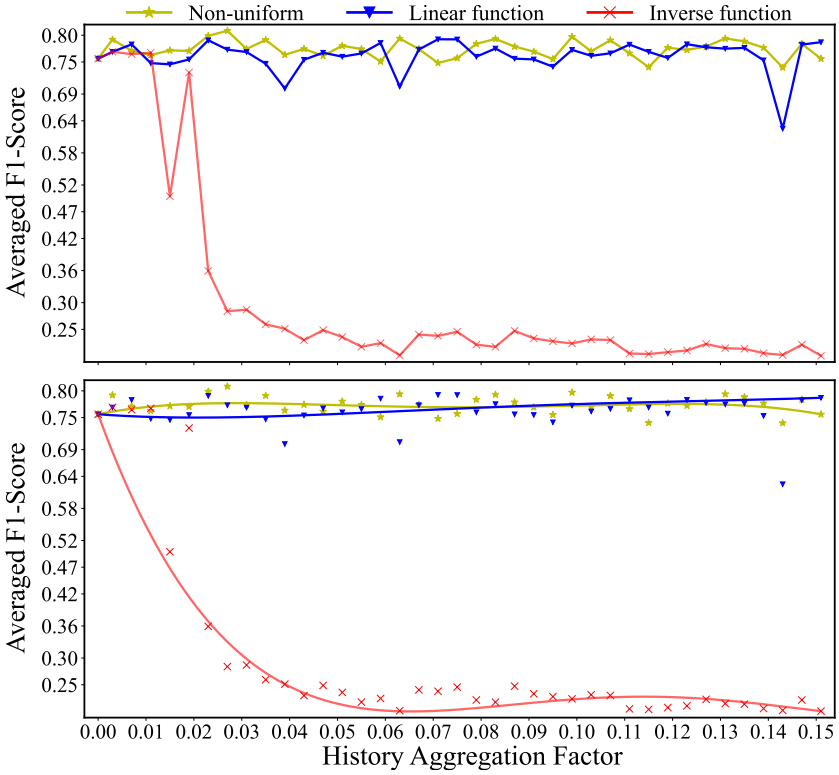


Figure B.7: Evaluating the function $t^{1.7}$ performance with the baseline

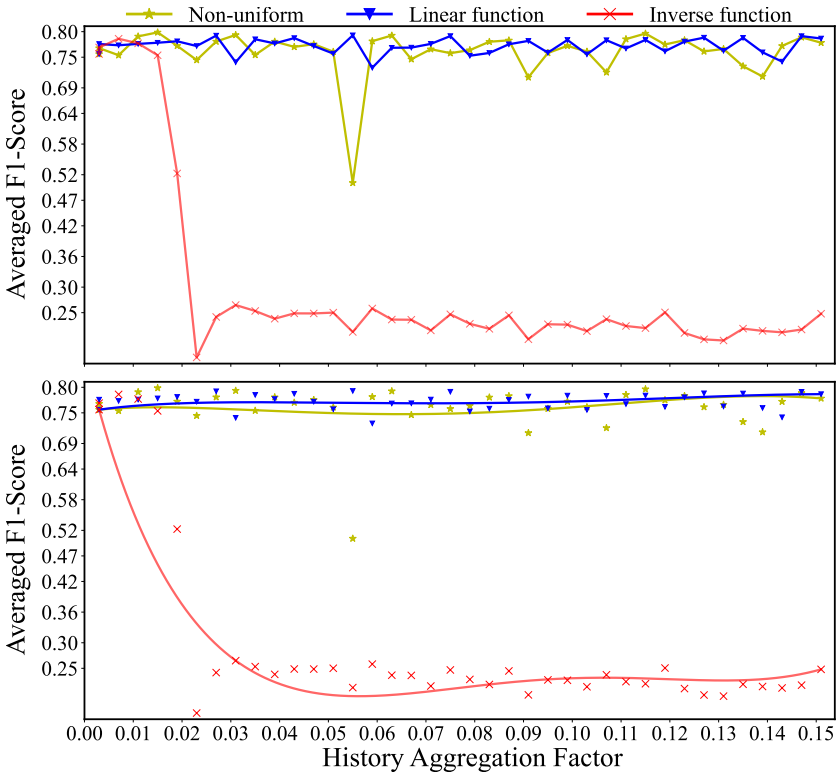


Figure B.8: Evaluating the function $t^{1.8}$ performance with the baseline

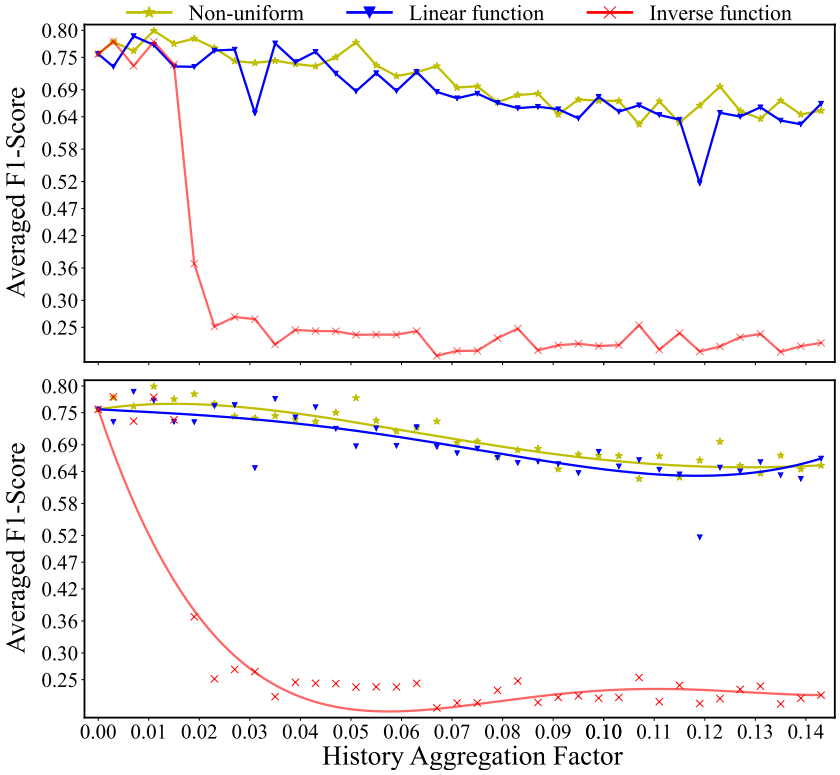


Figure B.9: Evaluating the function $t^{1.9}$ performance with the baseline

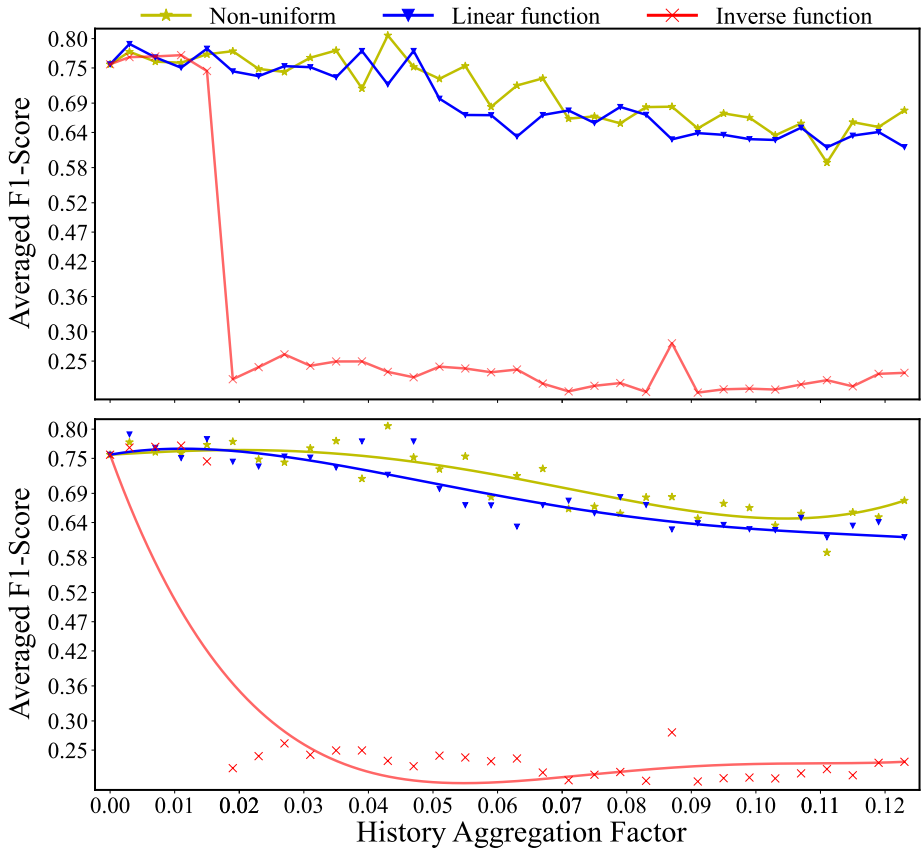


Figure B.10: Evaluating the function t^2 performance with the baseline

B.2 EXPONENTIAL FUNCTION

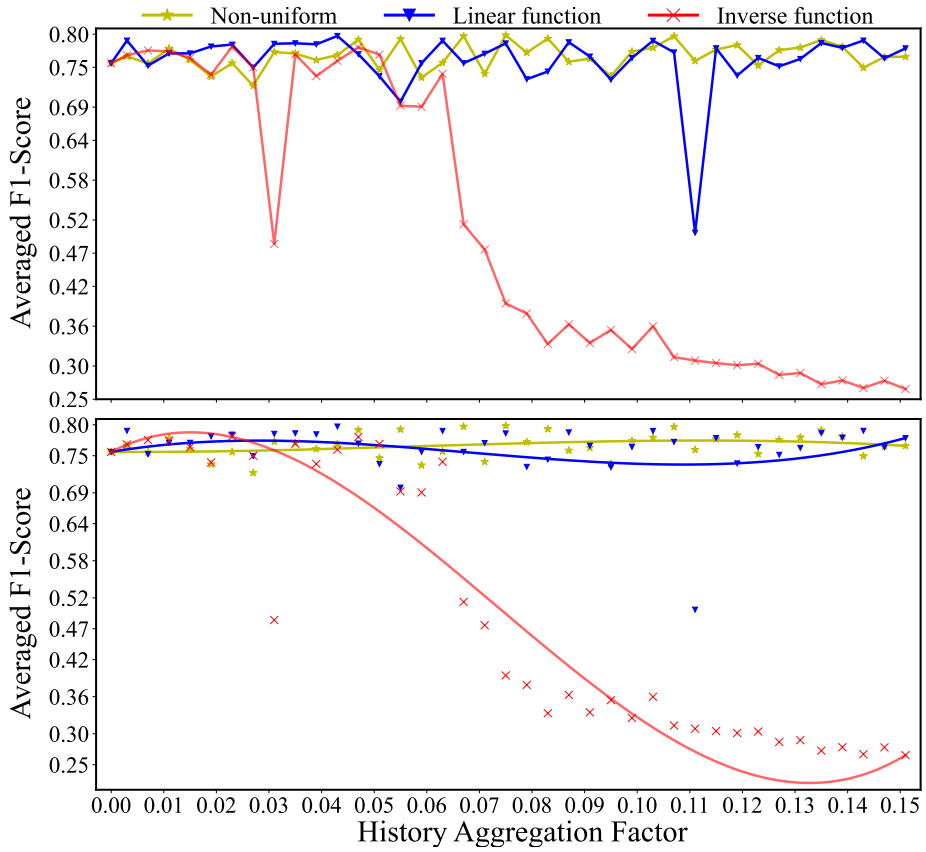


Figure B.11: Evaluating the function 1.1^t performance with the baseline

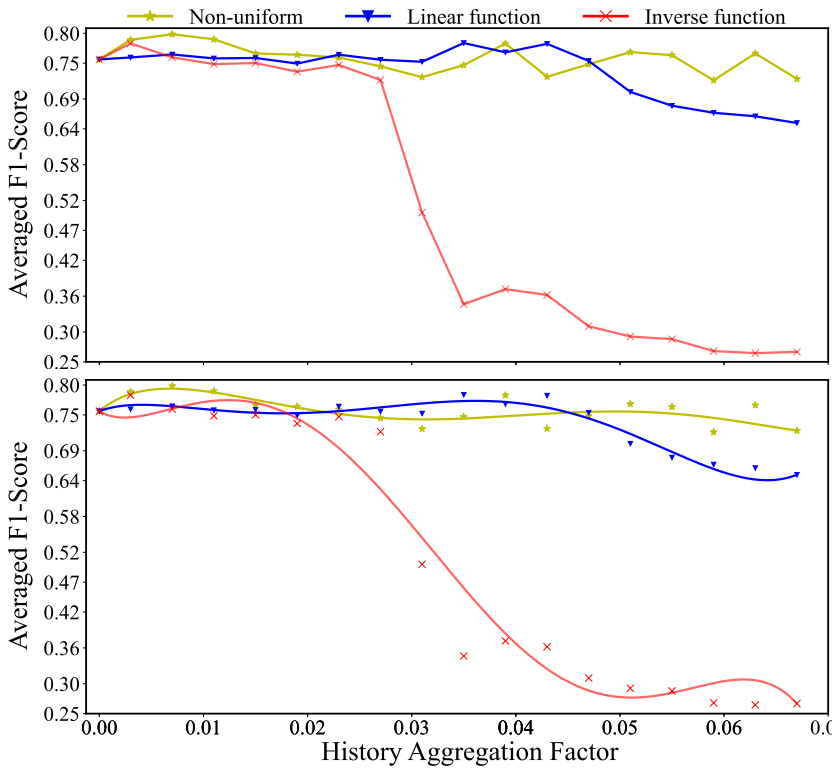


Figure B.12: Evaluating the function 1.3^t performance with the baseline

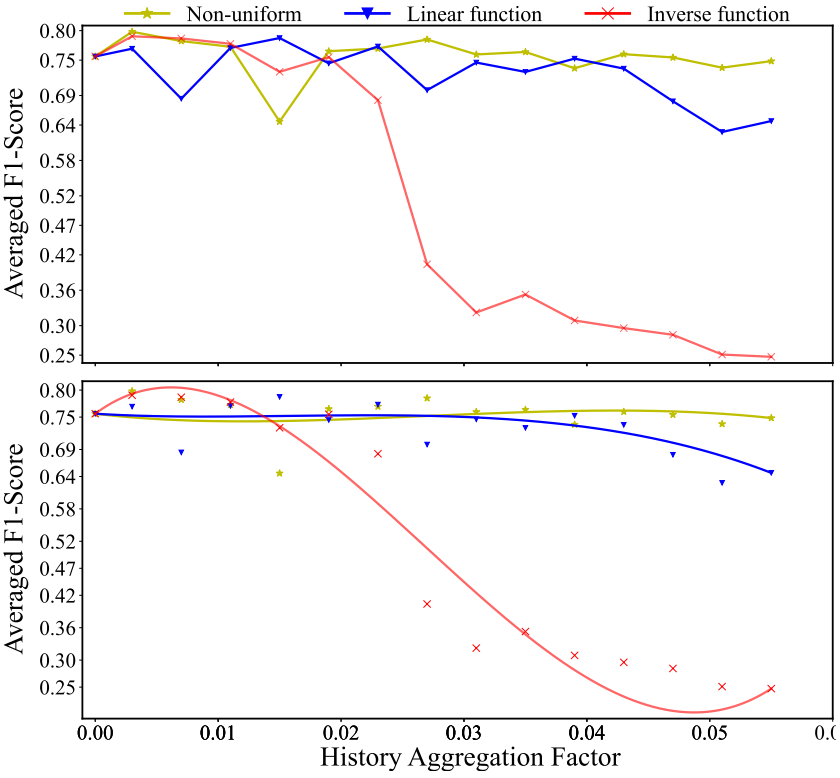


Figure B.13: Evaluating the function 1.4^t performance with the baseline