

Detection of plasma instabilities in tokamaks with machine learning

Casper Haems, Mateo Van Damme

Student numbers: 01804998, 01705169

Supervisor: Prof. dr. Geert Verdoolaege

Counsellor: Jerome Alhage

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2022-2023

Preface

The authors give permission to make this master's dissertation available for consultation and to copy parts of this master's dissertation for personal use.

In all cases of other use, the copyright terms have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this master dissertation.

01/01/2024

Acknowledgments

We would like to express our sincere gratitude to the supervisor, Prof. Dr. Geert Verdoolaege, and the counselor, Jerome Alhage, for their expertise in the field and their assistance and guidance in completing this master's dissertation. Their knowledge was essential in understanding the characteristics of Edge-localized modes (ELMs). Without their expertise, it would not have been possible to develop accurate ELM detection methods.

Casper Haems

I would like to extend a special thanks to my colleague, Mateo Van Damme, with whom I collaborated closely on this joint master's dissertation. His knowledge in the field of computer vision directly translated into significant improvements in the results.

Furthermore, I would like to thank my parents and my significant other for their continuous support throughout this study. Without them, it would have been more challenging.

Lastly, I would like to express my gratitude to my friends and family for providing the advice and support necessary for completing this dissertation.

Mateo Van Damme

I would like to express my sincere gratitude to my colleague, Casper Haems, for our close cooperation throughout this project. I am particularly thankful for his valuable contribution in developing a grid search algorithm that efficiently identifies the optimal parameters for our ELM detection methods. His dedication and expertise significantly enhanced the quality of our research.

I would also like to extend my appreciation to Complex Systems Modelling - UiT for their generosity in sharing their Deconvolution method as an open-source implementation on GitHub.

Explanation in connection with the master's dissertation

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Confidentiality Notice

Confidential up to and including 01/01/2024

Important

This master's dissertation contains confidential information and/or confidential research results proprietary to Ghent University or third parties. It is strictly forbidden to publish, cite or make public in any way this master's dissertation or any part thereof without the express written permission of Ghent University. Under no circumstance may this master's dissertation be communicated to or put at the disposal of third parties. Photocopying or duplicating it in any other way is strictly prohibited. Disregarding the confidential nature of this master's dissertation may cause irreparable damage to Ghent University. The stipulations mentioned above are in force until the embargo date.

Abstract

Nuclear fusion is a promising source of energy. It has the potential to provide a clean and sustainable source of electricity for the future. One of the most common and promising approaches is the use of a donut-shaped vessel called the "tokamak". A favorable mode of operation in tokamaks is H-mode (or high confinement mode). However, the occurrence of harmful events called edge-localized modes (ELMs) during H-mode poses a significant challenge. Understanding and therefore detecting ELMs is a crucial part of the research and development of nuclear fusion. Simple ELM detection methods exist but do not suffice for more difficult types of ELMs.

The objective of this master's dissertation is to develop methods that can accurately and reliably detect multiple ELM types based on data from real tokamak devices. This requires identifying and extracting relevant features from the data.

Mathematical transformations and machine learning techniques are employed to identify and extract features from the data. Noise removal techniques are utilized to filter high-frequency variations and random fluctuations while preserving essential low-frequency information for ELM detection.

Multiple ELM detection methods are developed, each employing different approaches and having their advantages and disadvantages. The performance of these methods is evaluated and compared with existing methods using suitable metrics.

In conclusion, this master's dissertation addresses ELM detection challenges in tokamak reactors by developing robust methods. Through mathematical transformations and machine learning techniques applied to real JET data, it contributes to nuclear fusion research. These findings pave the way for efficient ELM research, advancing the goal of fusion energy realization.

Detection of plasma instabilities in tokamaks with machine learning

Casper Haems, Mateo Van Damme
Supervisor: prof. dr. Geert Verdoolaege
Counsellor: Jerome Alhage

Abstract—Nuclear fusion is a promising source of energy. It has the potential to provide a clean and sustainable source of electricity for the future. One of the most common and promising approaches is the use of a donut-shaped vessel called the "tokamak". A favorable mode of operation in tokamaks is H-mode (or high confinement mode). However, the occurrence of harmful events called edge-localized modes (ELMs) during H-mode poses a significant challenge. Understanding and therefore detecting ELMs is a crucial part of the research and development of nuclear fusion.

This master's dissertation focuses on developing a framework for detecting ELMs with machine learning. An evaluation of this framework shows a clear improvement over existing methods.

Index Terms—ELM detection, peak detection, data science, machine learning, fusion

I. INTRODUCTION

This paper describes the use of various machine learning techniques to detect edge-localized modes (ELMs) in fusion devices.

Nuclear fusion is an alternative to nuclear fission to generate massive amounts of energy to power our society. Unlike fission, fusion does not produce a lot of dangerous waste products. There is also no possibility of a runaway device meltdown since only small amounts of fuel are used and the fusion process will die out when not actively sustained.

To obtain a fusion reaction, hydrogen isotope gas must be heated to extreme temperatures under high pressure, becoming a plasma. This state of matter allows the hot atoms to be contained by powerful magnets, protecting the donut-shaped vessel called the "tokamak." In the 1980s a new mode of operation in tokamaks, H-mode, was found, with doubled performance and better confinement. However, as a side effect, it created periodic releases of energy (similar to a pressure cooker's valve releasing steam every now and then). The risk they pose is the wear and tear to the device walls, as materials are repeatedly exposed to short bursts of heat.

These phenomena are called edge-localized modes (ELMs), in relation to the source of these instabilities: magnetohydrodynamic instabilities occurring near the edge of the plasma, right by the vessel walls. ELMs are difficult to model based on fundamental physics principles. Therefore, studies also rely on empirical data from large international experiments, such as JET (the Joint European Torus), ASDEX-Upgrade, and DIII-D.

The Infusion group at Ghent University is specialized in data science and contributes to fusion research and Eurofusion

projects. While studies of these ELMs typically compare a few experiments at a time, the group is able to aggregate all the available data, by characterizing each experiment with statistical features extracted from the ELM signals.

A key step is therefore detecting when an ELM has occurred. This relies on detecting light emission which follows the release of energy from the plasma [1]. Simply put, whenever there is a peak in this light emissivity diagnostic, there should be an ELM. As the ELM also causes the release of energy, so the stored plasma energy diagnostic will drop. By tracking these characteristics an algorithm can be developed to detect ELMs.

Previously, the group developed a simple but robust algorithm to set an optimal threshold and filter out noise from the emissivity signal. Their method was successfully applied to data from a variety of fusion machines, and several papers were published using the results.

One recent area of interest concerns plasma configurations that create smaller, more manageable ELMs. The emissivity patterns which they create, however, are no longer as clear-cut, and so the detection algorithm needs to be improved. This leads to the goal of this master's dissertation: to evaluate the performance of the existing algorithm and to propose improvements using state-of-the-art tools, such as machine learning and artificial neural networks.

II. DETAILED DESCRIPTION OF THE DATA

A. Defining shots

Fusion reactions require extremely high temperatures and pressures. Sustaining these conditions requires an enormous amount of energy input. Because of this reason, tokamaks do not run continuously but in multiple pulses, which are called shots. The data used in this master's dissertation was collected from the Joint European Torus (JET). Only small sections of shots are used, which are called areas of interest.

B. Data used for detecting ELMs

For each shot a collection of signals is captured, two of which are used in this master's dissertation to detect ELMs. The first signal (EDG8/TBEO) measures the light emissivity from the plasma, which is the line-integrated emissivity of the Bell line looking at the outer divertor plate. The second signal (EHTR/WP) measures the energy stored in the plasma. In this paper and in the code, the emissivity signal (EDG8/TBEO) will sometimes be referred to as the light

signal. The EHTR/WP signal will be referred to as the energy or plasma energy signal. Even though the plasma energy signal contains valuable information, it is not always available, which is why it is important to also develop ELM detection methods that can function without it. The original data is sampled at 10 kHz but is slightly inconsistent in its sampling. Therefore, both signals have been re-sampled to have a constant sampling rate of 10 kHz.

C. Output format

Detected ELMs in this work are represented using simple intervals. Each interval consists of a beginning and an end timestamp, effectively capturing the temporal extent of an ELM event; This representation allows researchers to focus their analysis on the ELMs themselves, instead of locating them within the data.

D. Labels for training, evaluation and testing

Labeled ELM intervals were provided to act as the ground truth for training ELM detection methods. The labeling was done under the supervision of prof. dr. Geert Verdoolaege. The master's dissertation acknowledges the limitations of the labeling provided for the ELM detection study. The available labeling was limited in quantity and contained certain inaccuracies, which posed challenges in accurately assessing the performance of the proposed methods.

III. METRICS

To train and evaluate the ELM detection methods performance metrics are needed. Finding existing metrics posed a significant challenge because most existing metrics in the field of machine learning are meant to be applied to classification tasks and are not directly applicable to peaks represented as intervals. To address this, two new sets of metrics were developed specifically for the data structure used in this work. The IntervalMetrics employ the confusion matrix from statistical classification to evaluate peak detection performance. On the other hand, the GapMetrics, while similar to IntervalMetrics, aim to provide a more intuitive and human-readable assessment of the results.

IV. METHODS

This section gives a brief overview of the methods implemented in the framework. A detailed description can be found in the scripton.

A. Robust Optimal Threshold

The Robust Optimal Threshold (ROT) is a method internally developed and used by the infusion research unit at Ghent University [2]. The main goal of the master's dissertation is to achieve one or more methods that are better than ROT.

B. MACD

The MACD method uses the moving average and convergence-divergence (MACD) filter [3] to obtain a new signal from the emission signal with less noise and distinct features. This signal is then passed through an algorithm to obtain peaks/ELMs.

C. Rolling Z-score

The Rolling Z-score method calculates the Z-score [4] over a sliding window. An algorithm is then used to find peaks/ELMs in this filtered emissivity signal.

D. LOG

The LoG method for ELM detection involves filtering the emissivity signal using a Laplacian of Gaussian kernel. Peaks/ELMs are obtained by thresholding the filtered signal. [5]

E. LoGDoG

LoGDoG builds upon the LoG method. In this method, the signal is attenuated based on the derivative of the plasma energy signal. The derivative is obtained by filtering the plasma energy signal with a Derivative of Gaussian kernel. [6] As a result, peaks are detected only when there is a drop in the plasma energy.

F. CustomKernel

The CustomKernel method detects peaks by filtering the signal with a customized kernel. This kernel is determined through training the method on the input signal.

G. RNN

The RNN method uses a combination of filters on the emissivity and plasma energy signals, makes sequences, and passes these through a recurrent neural network (RNN). The predictions are then thresholded to obtain the ELMs.

H. CNN

The CNN works the same way as the RNN method but uses a convolutional neural network (CNN) that is based on an existing CNN [7].

I. Deconvolution

The Deconvolution method is based on the paper published in 2023 [8]. It employs a deconvolution process to decompose the signal into a superposition of individual pulses.

V. RESULTS

A. Overview

Table I shows a comparison between the results obtained from the different methods. The results presented in the table were obtained by calculating the metrics individually for each shot and then taking the mean of each metric. Therefore, the reported F_β -scores are the means of the F_β -scores calculated for each shot, and they are not derived from the mean values of true positives, false positives, and false negatives displayed in the table.

TABLE I

TABLE WITH RESULTS FOR ALL METHODS. THE INTERVAL METRICS ARE CALCULATED USING A THRESHOLD OF 0.75 FOR TP AND FN AND 0.5 FOR THE FP AND TN. THE F_{β} -SCORES ARE CALCULATED WITH A BETA VALUE OF 0.5.

Method	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)
ROT	34.41	65.59	17.53	82.47	52.48
MACD	87.17	12.83	21.63	78.37	82.25
Rolling Z-score	89.99	10.01	16.39	83.61	87.33
LoG	86.61	13.39	12.79	87.21	87.81
LoGDoG	80.78	19.22	10.91	89.09	87.42
CustomKernel	82.97	17.03	9.39	90.61	88.62
RNN (modified)	91.95	8.05	15.90	84.10	87.07
CNN (modified)	80.01	19.99	16.63	83.37	82.66
Deconvolution	78.14	21.86	70.30	29.70	56.28

B. Discussion of the results

The results show that all methods developed in this master's dissertation outperform the existing ROT method. It is also found that the deconvolution method is not suitable for the specific characteristics of ELMs encountered in this work.

Each method exhibits its own set of advantages and disadvantages. For instance, the RNN and CNN models are trained with specific parameters, which may require retraining when applied to datasets with different parameter requirements.

The CustomKernel method receives the highest F_{β} -score and exhibits the lowest number of false positives. However, it is surpassed by the MACD, Rolling Z-score, LoG, and RNN methods in terms of true positive detection.

The LoGDoG method demonstrates the potential to reduce the number of false positives compared to the LoG method. However, its robustness using the energy signal inadvertently leads to a decrease in the number of true positives.

Overall, these findings highlight the trade-offs and varying performance characteristics of each method, underscoring the importance of selecting an appropriate method based on specific application requirements and priorities.

VI. CONCLUSION

This master's dissertation addresses ELM detection challenges in tokamak reactors by developing robust methods. Through mathematical transformations and machine learning techniques applied to real JET data, the research contributes to nuclear fusion research. These findings pave the way for efficient ELM research, advancing the goal of fusion energy realization.

REFERENCES

- [1] H. Zohm, "Edge localized modes (elms)," *Plasma Physics and Controlled Fusion*, vol. 38, no. 2, p. 105, feb 1996. [Online]. Available: <https://dx.doi.org/10.1088/0741-3335/38/2/001>
- [2] Verdoolaeye, Geert and Frassinetti, L., "Statistical analysis of edge-localized mode timing in JET," in *Proceedings of the 45th EPS Conference on Plasma Physics*. European Physical Society, 2018, p. 4. [Online]. Available: <http://ocs.ciemat.es/EPS2018ABS/pdf/P2.1078.pdf>
- [3] G. Appel, *The moving average convergence-divergence trading method: advanced version*. Scientific Investment Systems, 1985.
- [4] C. Andrade, "Z scores, standard scores, and composite test scores explained," *Indian Journal of Psychological Medicine*, vol. 43, no. 6, pp. 555–557, 2021, pMID: 35210687. [Online]. Available: <https://doi.org/10.1177/02537176211046525>
- [5] R. Collins, "Laplacian of Gaussian Filter," in *CSE/EE486 Computer Vision I*, Penn State University, 2007. [Online]. Available: <https://www.cse.psu.edu/~rtc12/CSE486/lecture11.pdf>
- [6] N. Snavely, "Derivative of Gaussian" Cornell University, [Online]. Available: https://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02_filter.pdf Note: Slides 35-45
- [7] A. Kensert et al., "Convolutional neural network for automated peak detection in reversed-phase liquid chromatography," *Journal of Chromatography A*, vol. 1672, p. 463005, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021967322002035>
- [8] S. Ahmed, O. E. Garcia, and A. Theodorsen, "Reconstruction of intermittent data time series as a superposition of pulses," 2023. [Online]. Available: <https://arxiv.org/abs/1811.11033>

Detectie van plasma-instabiliteiten in tokamaks met machinaal leren

Casper Haems, Mateo Van Damme
Promotor: prof. dr. Geert Verdoolaege
Begeleider: Jerome Alhage

Abstract—Nucleaire fusie is een veelbelovende energiebron. Het heeft het potentieel om een schone en duurzame bron van elektriciteit te leveren voor de toekomst. Een van de meest voorkomende en veelbelovende benaderingen is het gebruik van een donutvormige constructie genaamd de "tokamak". Een gunstige bedrijfsmodus in tokamaks is H-mode (of hoge confinementsmodus). Echter, het optreden van schadelijke gebeurtenissen genaamd edge-localized modes (ELMs) tijdens H-mode vormt een aanzienlijke uitdaging. Het begrijpen en dus detecteren van ELMs is een cruciaal onderdeel van het onderzoek en de ontwikkeling van nucleaire fusie.

Deze masterproef richt zich op het ontwikkelen van een kader voor het detecteren van ELMs met behulp van machine learning. Een evaluatie van dit framework toont een duidelijke verbetering ten opzichte van bestaande methoden.

Index Terms—ELM detectie, piek detectie, data science, machine learning, fusie

I. INLEIDING

Dit artikel beschrijft het gebruik van verschillende machine learning technieken om edge-localized modes (ELMs) te detecteren in fusieapparaten.

Nucleaire fusie is een alternatief voor nucleaire splitsing om enorme hoeveelheden energie te genereren voor onze samenleving. In tegenstelling tot splijting produceert fusie niet veel gevaarlijke afvalproducten. Bovendien is er geen mogelijkheid van een onbeheersbare kernsmelting (meltdown) van het apparaat, aangezien er slechts kleine hoeveelheden brandstof worden gebruikt en het fusieproces vanzelf zal uitdoven wanneer het niet actief in stand wordt gehouden.

Om een fusiereactie te verkrijgen, moet waterstofisotoopgas tot extreme temperaturen worden verhit onder hoge druk, waardoor het een plasma wordt. Deze toestand van materie maakt het mogelijk om de hete atomen te bevatten met krachtige magneten, ter bescherming van de donutvormige constructie genaamd de "tokamak". In de jaren 80 werd een nieuwe bedrijfsmodus in tokamaks ontdekt, genaamd H-mode, met dubbele prestaties en een betere confinementsmodus. Echter, als bijwerking creëerde het periodieke energie-uitbarstingen (vergelijkbaar met het vrijkomen van stoom via een ventiel van een snelkookpan). Het risico dat ze vormen, is de slijtage van de wanden van het apparaat, aangezien materialen herhaaldelijk worden blootgesteld aan korte uitbarstingen van warmte.

Deze verschijnselen worden edge-localized modes (ELMs) genoemd, vanwege de bron van deze instabiliteiten: magnetohydrodynamische instabiliteiten die optreden nabij de rand

van het plasma, vlak bij de wanden van de constructie. ELMs zijn moeilijk te modelleren op basis van fundamentele natuurkundige principes. Daarom vertrouwen studies ook op empirische gegevens uit grootschalige internationale experimenten, zoals JET (Joint European Torus), ASDEX-Upgrade en DIII-D.

De Infusion-groep aan de Universiteit Gent is gespecialiseerd in data science en draagt bij aan fusieonderzoek en Eurofusion-projecten. Terwijl studies van deze ELMs meestal een paar experimenten tegelijk vergelijken, is de groep in staat om alle beschikbare gegevens te verzamelen door elke proef te karakteriseren met statistische kenmerken die zijn geëxtraheerd uit de ELM-signalen.

Een belangrijke stap is daarom het detecteren wanneer een ELM heeft plaatsgevonden. Dit berust op het detecteren van lichtemissiviteit die volgt op de energieafgifte van het plasma [1]. Simpel gezegd, telkens wanneer er een piek is in deze lichtemissiviteit, zou er een ELM moeten zijn. Omdat de ELM ook zorgt voor de energieafgifte, zal de opgeslagen plasma-energie afnemen. Door deze kenmerken te volgen, kan een algoritme worden ontwikkeld om ELMs te detecteren.

Eerder heeft de groep een eenvoudig maar robuust algoritme ontwikkeld om een optimale drempelwaarde in te stellen en ruis uit het emissiviteitssignaal te filteren [2]. Hun methode is met succes toegepast op gegevens van verschillende fusieapparaten, en er zijn verschillende papers gepubliceerd met behulp van de resultaten.

Een recent interessegebied betreft plasmaconfiguraties die kleinere, beter beheersbare ELMs creëren. De emissiviteitspatronen die ze genereren zijn echter niet langer zo duidelijk gedefinieerd, waardoor het detectie-algoritme verbeterd moet worden. Dit leidt tot het doel van deze masterproef: het evalueren van de prestaties van het bestaande algoritme en het voorstellen van verbeteringen met behulp van state-of-the-art tools, zoals machine learning en kunstmatige neurale netwerken.

II. GEDETAILLEERDE BESCHRIJVING VAN DE GEGEVENS

A. Definiëren van shots

Fusiereacties vereisen extreem hoge temperaturen en drukken. Het handhaven van deze omstandigheden vereist een enorme hoeveelheid energie-invoer. Daarom werken tokamaks niet continu, maar in meerdere pulsen, die shots worden genoemd. De gegevens die in deze masterproef worden gebruikt, zijn verzameld vanuit de Joint European Torus (JET). Alleen

kleine secties van shots worden gebruikt, die gebieden van interesse worden genoemd.

B. Gegevens gebruikt voor het detecteren van ELMs

Voor elke shot wordt een verzameling signalen vastgelegd, waarvan er twee in deze masterproef worden gebruikt om ELMs te detecteren. Het eerste signaal (EDG8/TBEO) meet de lichtemissiviteit van het plasma, wat de lijn-geïntegreerde emissiviteit is van de Bell-lijn die naar de buitenste divertorplaat kijkt. Het tweede signaal (EHTR/WP) meet de energie die is opgeslagen in het plasma. In dit artikel en in de code wordt het emissiviteitssignaal (EDG8/TBEO) soms aangeduid als het lichtsignaal. Het EHTR/WP-signaal wordt aangeduid als het energie- of plasma-energiesignaal. Hoewel het plasma-energiesignaal waardevolle informatie bevat, is het niet altijd beschikbaar, daarom is het belangrijk om ook ELM-detectiemethoden te ontwikkelen die zonder dit signaal kunnen functioneren. De oorspronkelijke gegevens zijn bemonsterd met een snelheid van 10 kHz, maar zijn enigszins inconsistent in hun bemonstering. Daarom zijn beide signalen opnieuw bemonsterd om een constante bemonsteringssnelheid van 10 kHz te hebben.

C. Uitvoerformaat

Gedetecteerde ELMs in dit werk worden weergegeven met behulp van eenvoudige intervallen. Elk interval bestaat uit een begin- en eindtijdstempel, waarbij effectief de tijdsduur van een ELM-evenement wordt vastgelegd. Deze representatie stelt onderzoekers in staat zich te concentreren op de ELMs zelf, in plaats van ze binnen de gegevens te moeten lokaliseren.

D. Labels voor training, evaluatie en testen

Gelabelde ELM-intervallen zijn verstrekt als de grondwaarheid voor het trainen van ELM-detectiemethoden. Het labelen werd uitgevoerd onder toezicht van prof. dr. Geert Verdoolaage. Het proefschrift erkent de beperkingen van de beschikbare labeling voor de ELM-detectiestudie. De beschikbare labeling was beperkt in hoeveelheid en bevatte bepaalde onnauwkeurigheden, wat uitdagingen met zich meebracht bij het nauwkeurig beoordelen van de prestaties van de voorgestelde methoden.

III. METRIEKEN

Voor het trainen en evalueren van de prestaties van de ELM-detectiemethoden zijn metrieken nodig. Het vinden van bestaande metrieken vormde een aanzienlijke uitdaging omdat de meeste bestaande metrieken op het gebied van machine learning bedoeld zijn voor classificatietaken en niet direct toepasbaar zijn op pieken die worden weergegeven als intervallen. Om dit aan te pakken, zijn er specifiek voor de datastructuur die in dit werk wordt gebruikt twee nieuwe sets metrieken ontwikkeld. De IntervalMetrics maken gebruik van de verwarringstabel uit statistische classificatie om de prestaties van de piekdetectie te evalueren. Aan de andere kant hebben de GapMetrics, hoewel vergelijkbaar met IntervalMetrics, tot doel een meer intuïtieve en leesbare beoordeling van de resultaten te geven die begrijpelijk is voor mensen.

IV. METHODEN

Deze sectie geeft een beknopt overzicht van de methoden die geïmplementeerd zijn in het framework. Een gedetailleerde beschrijving is te vinden in de scriptie.

A. Robust Optimal Threshold

De Robust Optimal Threshold (ROT) is een methode die intern ontwikkeld is en gebruikt wordt door de onderzoekseenheid Infusion aan de Universiteit Gent [2]. Het belangrijkste doel van de masterproef is om één of meerdere methoden te ontwikkelen die beter presteren dan ROT.

B. MACD

De MACD-methode maakt gebruik van het moving average convergence/divergence (MACD) filter [3] om een nieuw signaal te verkrijgen uit het emissiviteitssignaal met minder ruis en onderscheidende kenmerken. Dit signaal wordt vervolgens door een algoritme geleid om pieken/ELMs te verkrijgen.

C. Rolling Z-score

De Rolling Z-score-methode berekent de Z-score [4] over een schuivend venster. Vervolgens wordt een algoritme gebruikt om pieken/ELMs te vinden in dit gefilterde emissiviteitssignaal.

D. LOG

De LoG-methode voor ELM-detectie houdt in dat het emissiviteitssignaal gefilterd wordt met behulp van een Laplacian of Gaussian-kernel. Pieken/ELMs worden verkregen door het gefilterde signaal te thresholden. [5]

E. LoGDoG

LoGDoG bouwt voort op de LoG-methode. In deze methode wordt het signaal afgezwakt op basis van de afgeleide van het plasma-energiesignaal. De afgeleide wordt verkregen door het plasma-energiesignaal te filteren met een afgeleide van Gaussische kernel [6]. Hierdoor worden pieken alleen gedetecteerd wanneer er een daling in de plasma-energie optreedt.

F. CustomKernel

De CustomKernel-methode detecteert pieken door het signaal te filteren met een aangepaste kernel. Deze kernel wordt bepaald door de methode te trainen op het invoersignaal.

G. RNN

De RNN-methode maakt gebruik van een combinatie van filters op het emissiviteitssignaal en het plasma-energiesignaal, maakt sequenties en leidt deze door een recurrent neuraal netwerk (RNN). De voorspellingen worden vervolgens gethreshold om de ELMs te verkrijgen.

H. CNN

De CNN werkt op dezelfde manier als de RNN-methode, maar maakt gebruik van een convolutioneel neuraal netwerk (CNN) dat gebaseerd is op een bestaand CNN [7].

I. Deconvolution

De Deconvolution-methode is gebaseerd op het in 2023 gepubliceerde artikel [8]. Hierbij wordt een deconvolutieproces toegepast om het signaal te ontbinden in een superpositie van individuele pulsen.

V. RESULTATEN

A. Overzicht

Tabel I toont een vergelijking tussen de resultaten die zijn verkregen met de verschillende methoden. De resultaten in de tabel zijn verkregen door de metrieken individueel te berekenen voor elke shot en vervolgens het gemiddelde van elke metriek te nemen. Daarom zijn de gerapporteerde F_{β} -scores het gemiddelde van de F_{β} -scores die zijn berekend voor elke shot, en ze zijn niet afgeleid van de gemiddelde waarden van ware positieven, valse positieven en valse negatieven die in de tabel worden weergegeven.

TABEL I

TABLE MET RESULTATEN VOOR ALLE METHODES MET GEBRUIK VAN INTERVALMETRICS. DE METRIEKEN ZIJN BEREKEND MET EEN THRESHOLD VAN 0.75 VAN TP EN FN, EN 0.5 VOOR DE FP EN TN. DE F_{β} -SCORES ZIJN BEREKEND MET EEN BETA WAARDE VAN 0.5..

Method	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)
ROT	34.41	65.59	17.53	82.47	52.48
MACD	87.17	12.83	21.63	78.37	82.25
Rolling Z-score	89.99	10.01	16.39	83.61	87.33
LoG	86.61	13.39	12.79	87.21	87.81
LoGDoG	80.78	19.22	10.91	89.09	87.42
CustomKernel	82.97	17.03	9.39	90.61	88.62
RNN (modified)	91.95	8.05	15.90	84.10	87.07
CNN (modified)	80.01	19.99	16.63	83.37	82.66
Deconvolution	78.14	21.86	70.30	29.70	56.28

B. Bespreking van de resultaten

De resultaten tonen aan dat alle methoden die in deze masterproef zijn ontwikkeld beter presteren dan de bestaande ROT-methode. Ook blijkt dat de deconvolutiemethode niet geschikt is voor de specifieke kenmerken van de ELMs die in dit onderzoek zijn aangetroffen.

Elke methode heeft zijn eigen set van voordelen en nadelen. Zo zijn de RNN- en CNN-modellen getraind met specifieke parameters, die mogelijk opnieuw getraind moeten worden bij toepassing op datasets met andere parametervereisten.

De CustomKernel-methode behaalt de hoogste F_{β} -score en vertoont het laagste aantal valse positieven. Echter, het wordt overtroffen door de MACD-, Rolling Z-score-, LoG- en RNN-methoden wat betreft de detectie van ware positieven.

De LoGDoG-methode laat potentie zien om het aantal valse positieven te verminderen in vergelijking met de LoG-methode. Echter, de robuustheid van de methode met behulp van de energie-sigitaal leidt onbedoeld tot een afname van het aantal ware positieven.

Over het algemeen benadrukken deze bevindingen de afwegingen en variërende prestatiekenmerken van elke methode, waarbij het belang van het selecteren van een geschikte

methode op basis van specifieke toepassingseisen en prioriteiten wordt benadrukt.

VI. CONCLUSIE

Deze masterproef adresseert de uitdagingen bij het detecteren van ELMs in tokamak-reactoren door het ontwikkelen van robuuste methoden. Door wiskundige transformaties en machine learning technieken toe te passen op echte JET-data, draagt het onderzoek bij aan onderzoek naar kernfusie. Deze bevindingen banen de weg voor efficiënt ELM-onderzoek en dragen bij aan het realiseren van het doel van fusie-energie.

REFERENTIES

- [1] H. Zohm, "Edge localized modes (elms)," Plasma Physics and Controlled Fusion, vol. 38, no. 2, p. 105, feb 1996. [Online]. Available: <https://dx.doi.org/10.1088/0741-3335/38/2/001>
- [2] Verdoolaage, Geert and Frassinetti, L., "Statistical analysis of edge-localized mode timing in JET," in Proceedings of the 45th EPS Conference on Plasma Physics. European Physical Society, 2018, p. 4. [Online]. Available: <http://ocs.ciemat.es/EPS2018ABS/pdf/P2.1078.pdf>
- [3] G. Appel, The moving average convergence-divergence trading method: advanced version. Scientific Investment Systems, 1985.
- [4] C. Andrade, "Z scores, standard scores, and composite test scores explained," Indian Journal of Psychological Medicine, vol. 43, no. 6, pp. 555-557, 2021, pMID: 35210687. [Online]. Available: <https://doi.org/10.1177/02537176211046525>
- [5] R. Collins, "Laplacian of Gaussian Filter," in CSE/EE486 Computer Vision I, Penn State University, 2007. [Online]. Available: <https://www.cse.psu.edu/~rtc12/CSE486/lecture11.pdf>
- [6] N. Snavely, "Derivative of Gaussian" Cornell University, [Online]. Available: https://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02_filter.pdf Note: Slides 35-45
- [7] A. Kensert et al., "Convolutional neural network for automated peak detection in reversed-phase liquid chromatography," Journal of Chromatography A, vol. 1672, p. 463005, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021967322002035>
- [8] S. Ahmed, O. E. Garcia, and A. Theodorsen, "Reconstruction of intermittent data time series as a superposition of pulses," 2023. [Online]. Available: <https://arxiv.org/abs/1811.11033>

Contents

Abstract	v
List of Figures	xvi
List of Tables	xix
List of Acronyms	xx
List of Code Fragments	xxii
1 Introduction	1
2 Description of the problem	3
2.1 Background information on the problem	3
2.2 Detailed description of the data	3
2.3 Challenges and limitations in ELM detection	4
2.4 Other work related to ELM detection	5
2.5 Research questions and objectives	5
2.6 Comprehensive description of ELM characteristics required for detection	6
3 Technology review	8
3.1 Introduction to machine learning: algorithms, models and training	8
3.2 Convolution operation and its properties	8
3.2.1 Correlation operation	9
3.3 Filters in signal processing and machine learning	10
3.4 Introduction to neural networks	10
3.4.1 Pattern recognition using convolutional neural networks	10
3.4.2 Capturing dependencies across variable-length sequences using recurrent neural networks	10
3.5 Hyperparameter tuning for machine learning algorithms using gridsearch	11
3.6 Keypoint detection and description in computer vision using the SIFT algorithm	11
3.6.1 Keypoint detection	11
3.6.2 Simplified explanation of SIFT keypoint detection	12
3.7 Reusing code in Python through multiple inheritance with mixins	14

4	System architecture	15
4.1	Project Repositories: elms-detection, peaks, and dbtools	15
4.1.1	Distinction between ELMs-detection and peak detection	15
4.1.2	Modular peak detection and evaluation methods in the peaks repository	15
4.1.3	Modular ELM detection methods in the elms-detection repository	16
4.1.4	Utility functions for data loading and resampling in the dbtools repository	17
4.2	Continuous integration and continuous delivery	17
5	Metrics For Evaluating Peak Detection Methods	19
5.1	Imbalanced datasets	19
5.2	Interval Metrics	19
5.2.1	True Positive	20
5.2.2	False negative	20
5.2.3	False positive	20
5.2.4	True Negative	21
5.3	Gap Metrics	22
5.3.1	False Positive	22
5.3.2	True Negative	23
5.3.3	Conclusion	23
5.4	Precision	24
5.5	Recall	24
5.6	F-score	24
5.6.1	Other metrics	25
6	Signal Processing Filters	26
6.1	Noise removal using the moving average	26
6.2	Noise removal using the exponential moving average	27
6.3	Noise removal using the Wiener filter	28
6.4	Noise removal using the Hann window	29
6.5	Detect changes and anomalies using the difference filter	30
6.6	Detect changes and anomalies using the gradient filter	31
6.7	Noise removal, change and anomaly detection using the Savitzky-Golay filter	31
6.8	Detect changes and anomalies using the rolling Z-score	33
6.9	Noise removal, change and anomaly detection using the MACD filter	34
6.10	Custom Kernel filter for executing convolutions	35
6.11	Gaussian filter for noise removal	36
6.11.1	Mixin usage	37
6.12	Derivative of Gaussian for obtaining a robust derivative of the signal	37
6.13	Laplacian of Gaussian for detecting peaks in the signal	39

7	ELM detection methods	42
7.1	Method overview	42
7.2	Peak detection invariances	43
8	Filter-based ELM detection methods	45
8.1	Basic thresholding methods	45
8.1.1	Simple threshold	45
8.1.2	Average threshold method	45
8.1.3	Robust optimal threshold method	46
8.2	MACD method	47
8.2.1	MACD peak detection	49
8.2.2	MACD peak detection extension	49
8.2.3	MACD ELM detection	51
8.3	Rolling Z-score method	53
8.3.1	Rolling Z-score peak detection extension	54
8.3.2	Rolling Z-score ELM detection	55
8.4	LoG method	55
8.4.1	LoG resolution invariance problem	56
8.4.2	LoG resolution invariance solution	57
8.4.3	LoG peak detection	58
8.4.4	LoG ELM detection	59
8.4.5	Conclusion	60
8.5	LoGDoG method	60
8.5.1	LoGDoG ELM detection	61
8.5.2	Conclusion	63
8.6	CustomKernel method	63
8.6.1	Kernel upsampling	64
8.6.2	Determining the optimal kernel	65
8.6.3	CustomKernel peak detection	66
8.6.4	CustomKernel ELM detection	67
8.6.5	Conclusion	67
9	Neural network ELM detection methods	68
9.1	Preprocessing data for neural networks	68
9.1.1	Statistical feature extraction for usage in the input data	68
9.1.2	Sequencing the data to be compatible with fixed-length neural network models	69
9.2	Defining the output of the model	70
9.3	Preprocessing step on the labels used during training	70
9.4	Extra steps required for making the custom metrics compatible with the models during training	71
9.5	First important remark on training: potential problem with sequences	71

9.6	Second important remark on training: hardware and software limitations	71
9.7	Postprocessing the output of a neural network to be compatible with the other ELM detection methods	72
9.8	An overview of the recurrent neural network method	72
9.9	An overview of the convolutional neural network method	74
10	Deconvolution ELM detection method	78
10.1	Prototype ELM shape	79
10.2	Implementation	80
10.3	Conclusion	82
11	Results	83
11.1	Overview	83
11.1.1	Discussion about labeling quality	84
11.2	Performance evaluation of the CNN method	84
11.3	Performance evaluation of the LoG method	85
11.4	Performance evaluation of the CustomKernel method	85
11.5	Performance evaluation of the LoGDoG method	85
11.6	Performance evaluation of the MACD method	86
11.7	Performance evaluation of the rolling Z-score method	86
11.8	Performance evaluation of the RNN method	87
11.9	Performance evaluation of the Deconvolution method	87
11.10	Discussion about false positives and the plasma energy signal	88
11.11	Comparison to existing methods	88
11.11.1	ROT and Deconvolution	88
11.11.2	CNN	88
11.12	Peak detection on other datasets	89
11.13	ELM detection on other data	91
11.14	Conclusion	92
	Conclusion	93
	Ethical and social reflection	94
	References	95
	Appendices	98
	Appendix A: Usage of AI	99
	Appendix B: Applying ELM detection methods to shots	104
	Appendix C: Comparing results between the two labels	116

List of Figures

2.1	High-frequency noise and random variations in the emissivity and energy signals.	4
2.2	Shot 94321-0 with manually labeled ELMs marked in green.	6
3.1	5 types of SIFT invariances. Keypoints are marked with a red cross	12
3.2	Keypoint detection steps, using an image of a sunflower field [1] as an example	13
3.3	LoG kernels for image filtering in Figure 3.2.	14
4.1	UML diagram illustrating the utilization of mixins in the peaks repository. Some attributes and classes are omitted to maintain clarity	16
5.1	Custom Interval Metric - Examples	22
5.2	LogDoG ELM detection method trained with Gap Metrics applied to shot 94776-0. Note: This figure is based on the old labels	23
6.1	Removing noise from a signal using the moving average filter.	27
6.2	Comparing the moving average against the exponential moving average.	28
6.3	Applying multiple Wiener filters to the same input. Each filter has a different size parameter.	29
6.4	Comparing the moving average filter against the Hann filter.	30
6.5	Applying the difference and gradient filters to a signal.	31
6.6	Removing noise from data using the Sovitzky-Golay filter compared to the moving average filter.	32
6.7	Removing noise from the first derivative of a signal using the Sovitzky-Golay filter prevents the amplification of small fluctuations caused by high-frequency noise in the input data.	33
6.8	Applying multiple rolling Z-score filters to the same input. Each filter has a different size parameter.	34
6.9	MACD with EMA 12 and EMA 26, calculated from the input. The signal line is an EMA 9 applied on the MACD.	35
6.10	Example of the Gaussian filter	36
6.11	Gaussian PDF function	36
6.12	Normalized Gaussian kernel used in Figure 6.10	37
6.13	Example of the DoG filter.	38
6.14	DoG formula obtained by deriving the Gaussian PDF	38
6.15	Normalized derivative of Gaussian kernel	39
6.16	Example of the LoG filter	39
6.17	Laplacian of the Gaussian formula	40
6.18	LoG kernel used in figure 6.16	41

7.1	Different invariances that are applicable to peak detection in a time series signal	43
8.1	Detecting everything above the average as an ELM works for a very basic signal for a simple signal.	46
8.2	Detecting everything above the average as an ELM results in many false positives for a more difficult signal.	46
8.3	Detecting everything above the ROT as an ELM.	46
8.4	Detection is not always working in complex cases.	47
8.5	Basic peak detection using MACD on the emissivity signal.	49
8.6	Detection without MACD extension.	50
8.7	Cause of the peaks detected with the MACD method ending too early.	50
8.8	Detection with MACD extension.	51
8.9	Adding padding to MACD ELMs.	51
8.10	A complex signal with many false positives from the MACD method.	52
8.11	Merging and dropping ELMs using Savgol Gap. Not all problems are solved with Savgol.	52
8.12	Dropping ELMs using Savgol Gap. Not all problems are solved with Savgol Gap. Merging with Savgol Gap is disabled.	53
8.13	ELM detection using the MACD method.	53
8.14	Rolling Z-score ELM detection.	55
8.15	Time series signal subsampled by a factor of 1, 2 and 4 with a LoG kernel (interval_Log = 31 ms, resolution = 1ms)	56
8.16	Time series signal subsampled by factor 1, 2 and 4.	57
8.17	Light signal of shot 94423-0	58
8.18	Convolution step of the LoG peak detection method	58
8.19	Thresholding step of the LoG peak detection method	58
8.20	Padding step of the LoG ELM detection method	59
8.21	Combining step of the LoG ELM detection method	59
8.22	Result of the LoG method	60
8.23	Light and energy signal of shot 94776-0	61
8.24	Light signal of shot 94776-0 in blue and a version that is filtered with a LoG kernel	62
8.25	Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel	62
8.26	Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel	62
8.27	Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel	63
8.28	The light and energy signal of shot 94776-0 together with the detected peaks in red and correct peaks in green	63
8.29	Kernel upsampling demonstrated on the default kernel used in the CustomKernel method.	65
8.30	Kernel training process from beginning to completion	66
8.31	The input signal is correlated using the upsampled kernel	67
8.32	The light and energy signal of shot 94423-0 together with the detected peaks in red and correct peaks in green	67
9.1	Labels represented as a binary signal.	71
9.2	The RNN method applied on shot 94423-0.	74
9.3	The CNN method applied on shot 94423-0.	77

10.1	Representative ELM profiles	79
10.2	Prototype ELM shape to be used as a pulse shape for the deconvolution process	80
10.3	Deconvolution step	81
10.4	Padding of the peaks detected using <code>three_point_maxima</code>	81
10.5	Result of the Deconvolution ELM detection method	82
11.1	The light and energy signal of shot 94776-0 together with the detected peaks in red and correct peaks in green	85
11.2	The processed DoG signal using the old LoGDoG method	86
11.3	MACD peak detection on an illuminance signal.	89
11.4	CustomKernel peak detection on an illuminance signal.	90
11.5	MACD peak detection on a flipped illuminance signal.	90
11.6	CustomKernel peak detection on a temperature signal.	90
11.7	LoG peak detection on a temperature signal.	90
11.8	LoG ELM detection on an emissivity signal from DIII-D.	91
11.9	RNN (modified) ELM detection on an emissivity signal from DIII-D.	91
11.10	CNN (modified) ELM detection on an emissivity signal from DIII-D.	92
11.11	CustomKernel ELM detection on an emissivity signal from DIII-D.	92

List of Tables

5.1	Comparison between the results of the Gap Metric and Interval Metric for shot 94776-0	24
7.1	ELM and Peak Detection methods overview. The different types of invariances are: A - Amplitude invariant, T - Translation invariant, R - Resolution invariant. The execution time is measured by using the method to extract ELMs for each shot in the data set.	42
8.1	Hyperparameters and their descriptions for the ROT ELM detection method.	47
8.2	Hyperparameters and their descriptions for the MACD ELM and peak detection method.	48
8.3	Hyperparameters and their descriptions for the Rolling Z-score ELM and peak detection method.	54
8.4	Hyperparameters and their descriptions for the LoG ELM and peak detection method.	55
8.5	Properties of different kernels created to cover the same 31ms at different sampling resolutions.	57
8.6	Hyperparameters and their descriptions for the LoG ELM and peak detection method.	61
8.7	Hyperparameters and their descriptions for the LoG ELM and peak detection method.	64
8.8	Hyperparameters and their descriptions for the LoG ELM and peak detection method.	65
9.1	Hyperparameters and their descriptions for the RNN ELM detection method.	73
9.2	Hyperparameters and their descriptions for the CNN ELM detection method.	76
10.1	Hyperparameters and their descriptions for the LoG ELM and peak detection method.	78
11.1	Table with results for all methods. The GapMetrics are calculated using a threshold of 0.75 for TP and FN and 0.1 for the FP and TN. The IntervalMetrics are calculated using a threshold of 0.75 for TP and FN and 0.5 for the FP and TN. The F_{β} -scores are calculated for both metrics separately with a beta value of 0.5.	83

List of Acronyms

AI Artificial intelligence.

CNN Convolutional neural network.

CPU Central Processing Unit.

DoG Derivative of Gaussian.

ELM Edge-localized mode.

EMA Exponential moving average.

FN False Negative.

FP False Positive.

GRU Gated recurrent unit.

JET Joint European Torus.

LoG Laplacian of Gaussian.

LSTM long short-term memory.

ML Machine learning.

MSE Mean-squared error.

NN Neural network.

PDF Probability Distribution Function.

PSD Power spectral density.

RAM Random Access Memory.

RNN Recurrent neural network.

ROT Robust optimal threshold.

Savgol Savitzky-Golay.

SDG Sustainable Development Goal.

SIFT Scale-invariant Feature Transform.

TN True Negative.

TP True Positive.

List of Code Fragments

6.1	Moving average using SciPy	26
6.2	Improved moving average using SciPy and Numpy	27
9.1	Example input with extracted features. Each row represents one sample in the time series. Note that in practice, the values are numbers.	69
9.2	Sequencing applied to Fragment . With sequence length 3 and stride 1.	70
9.3	Merging multiple partially overlapping prediction sequences.	72
9.4	Creation of the CNN model with custom hyperparameters. The output shape here is not for the last layer because a new layer was added at the end of the model.	77

1

Introduction

This paper describes the use of various machine learning techniques to detect edge-localized modes (ELMs) in fusion devices.

Nuclear fusion is an alternative to nuclear fission to generate massive amounts of energy to power our society. Unlike fission, fusion does not produce a lot of dangerous waste products and the fusion device can't be used to make nuclear bomb fuel. There is also no possibility of a runaway device meltdown since only small amounts of fuel are used and the fusion process will die out when not actively sustained.

To obtain a fusion reaction, hydrogen isotope gas must be heated to extreme temperatures under high pressure, becoming a plasma. This state of matter allows the hot atoms to be contained by powerful magnets, protecting the donut-shaped vessel called the "tokamak." In the 1980s a new mode of operation in tokamaks, H-mode, was found, with doubled performance and better confinement. However, as a side effect, it created periodic releases of energy (similar to a pressure cooker's valve releasing steam every now and then). The risk they pose is the wear and tear to the device walls, as materials are repeatedly exposed to short bursts of heat.

These phenomena are called edge-localized modes (ELMs), in relation to the source of these instabilities: magnetohydrodynamic instabilities occurring near the edge of the plasma, right by the vessel walls. ELMs are difficult to model based on fundamental physics principles. Therefore, studies also rely on empirical data from large international experiments, such as JET (the Joint European Torus), ASDEX-Upgrade, and DIII-D.

The Infusion group at Ghent University is specialized in data science and contributes to fusion research and Eurofusion projects. While studies of these ELMs typically compare a few experiments at a time, the group is able to aggregate all the available data, by characterizing each experiment with statistical features extracted from the ELM signals.

A key step is therefore detecting when an ELM has occurred. This relies on detecting light emission which follows the release of energy from the plasma [2]. Simply put, whenever there is a peak in this light emissivity diagnostic, there should be an ELM. As the ELM also causes the release of energy, so the stored plasma energy diagnostic will drop. By tracking these characteristics an algorithm can be developed to detect ELMs.

Previously, the group developed a simple but robust algorithm to set an optimal threshold and filter out noise from the emissivity signal [3]. Their method was successfully applied to data from a variety of fusion machines, and several papers were published using the results.

1 Introduction

One recent area of interest concerns plasma configurations that create smaller, more manageable ELMs. The emissivity patterns which they create, however, are no longer as clear-cut, and so the detection algorithm needs to be improved. This leads to the goal of this master's dissertation: to evaluate the performance of the existing algorithm and to propose improvements using state-of-the-art tools, such as machine learning and artificial neural networks.

In addition to building on existing ELM detection methods, this master's dissertation also proposes some novel approaches.

The structure of the scripture is as follows. In Chapter 2, a comprehensive description of the problem is provided, highlighting the main challenges, limitations, and objectives. The subsequent chapter, Chapter 3, presents a detailed review of the technologies and algorithms employed throughout the dissertation.

Chapter 4 explains the design of the code, describing the modular and functional development of each component. Chapter 5 describes the evaluation metrics employed for training and optimizing the proposed methods. Additionally, Chapter 6 discusses the filters investigated and regularly utilized in the algorithms of the proposed methods, encompassing their functionality, goals, advantages, and disadvantages.

Moving forward, Chapter 7 offers a concise overview of the different methods employed. Subsequently, Chapters 8, 9, and 10 respectively delve into the implementation details of each method. The former chapter specifically focuses on methods relying solely on filters, the middle chapter discusses methods utilizing neural networks, and the latter chapter describes a method based on deconvolution.

Finally, Chapter 11 compares the results obtained from the aforementioned methods.

2

Description of the problem

2.1 Background information on the problem

Nuclear fusion is a promising source of energy. It has the potential to provide a clean and sustainable source of electricity for the future. One of the most common and promising approaches is the use of a donut-shaped vessel called the "tokamak". A favorable mode of operation in tokamaks is H-mode (or high confinement mode). However, the occurrence of harmful ELMs during H-mode poses a significant challenge. Understanding and therefore detecting ELMs is a crucial part of the research and development of nuclear fusion.

The objective of this master's dissertation is to develop methods that can accurately and reliably detect ELMs based on data of real tokamak devices. This requires identifying and extracting relevant features from the data. There are multiple ways of achieving this. Features can be extracted by applying some mathematical transformations to the data and/or using machine learning.

Machine learning is a subfield of artificial intelligence where algorithms make predictions on data. A machine learning algorithm learns the correct predictions by training on a training data set. The goal of machine learning is to develop models that learn patterns and relations in data and use these patterns to make accurate predictions or classifications on unseen data.

2.2 Detailed description of the data

Fusion reactions require extremely high temperatures and pressures. Sustaining these conditions requires an enormous amount of energy input. Because of this reason, tokamaks do not run continuously but in multiple pulses, which are called shots. The data used in this master's dissertation was collected from the Joint European Torus (JET).

At the start of each pulse, there is a startup phase in which the plasma conditions are not stationary. Experimentalists design pulses under operating conditions to search for regions of high performance under stationary plasma conditions. Experimentalists marked these as areas of interest, which will be used throughout this master's dissertation. However, this is not to say that there are no challenging parts in the data set used. When a particular area of interest is mentioned in this work it will often be described in the following format: 'Shot 94776-0'. This means 'area of interest 0 of the pulse with an assigned number of 94776'.

2 Description of the problem

For each shot a collection of signals is captured, two of which are used in this master's dissertation to detect ELMs. The first signal (EDG8/TBEO) measures the light emissivity from the plasma, which is the line-integrated emissivity of the Bell line looking at the outer divertor plate. [4] The second signal (EHTR/WP) measures the energy stored in the plasma. In this script and in the code, the emissivity signal (EDG8/TBEO) will sometimes be referred to as the light signal. The EHTR/WP signal will be referred to as the energy or plasma energy signal. Even though the plasma energy signal contains valuable information, it is not always available, which is why it is important to also develop ELM detection methods that can function without it. The original data is sampled at 10 kHz but is slightly inconsistent in its sampling. Therefore, both signals have been re-sampled to have a constant sampling rate of 10 kHz.

Detected ELMs in this work are represented using simple intervals. Each interval consists of a beginning and an end timestamp, effectively capturing the temporal extent of an ELM event. This representation allows researchers to focus their analysis on the ELMs themselves, instead of locating them within the data.

Labeled ELM intervals were provided to act as the ground truth for training ELM detection methods. The labeling was done under the supervision of prof. dr. Geert Verdoolaege. The master's dissertation acknowledges the limitations of the labeling provided for the ELM detection study. The available labeling was limited in quantity and contained certain inaccuracies, which posed challenges in accurately assessing the performance of the proposed methods. Also, note that the labeling of these ELMs was slightly subjective because the person that labeled the data had access to both the emissivity and the plasma energy signals and did the labeling in order to pursue a scientific goal. These two circumstances lead to some bias in the labels.

2.3 Challenges and limitations in ELM detection

One significant challenge is the noise in the data. As shown in Figure 2.1, a lot of high-frequency noise and random variations occur in the data. In chapter 6, a few filters will be discussed to remove the noise and retain the low-frequency information that is useful for ELM detection.

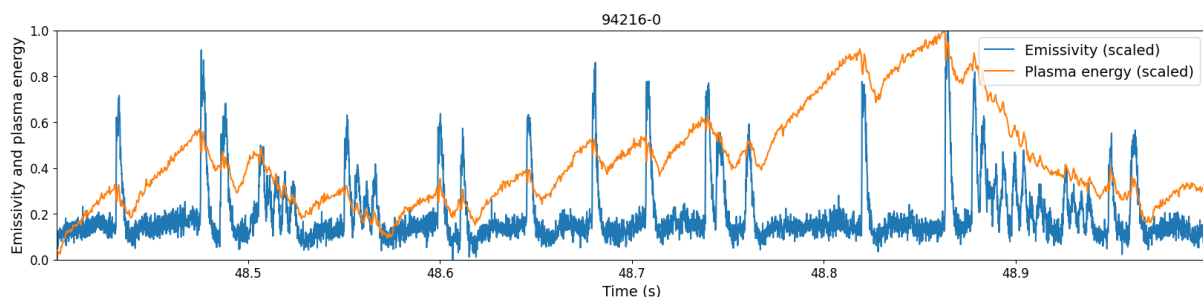


Figure 2.1: High-frequency noise and random variations in the emissivity and energy signals.

Another challenge is the diversity of ELM types, which can make it challenging to capture the full range of ELMs. This can potentially limit the effectiveness of the detection methods. For example, some ELMs stand on their own, but other ELMs are merged into what is called a compound ELM [2]. Both types of ELMs will be discussed in Section 2.6.

2 Description of the problem

Furthermore, there are also limitations to the data itself. Some types of ELMs can only be differentiated from each other using other signals, which are not available for this master's dissertation. This provides only a limited view of the plasma behavior and prevents capturing the full complexity of ELM events.

Finally, the ELM detection methods described in this paper assume that the time series data has a constant sampling frequency in a single signal. In reality, this is not always the case but this assumption was made to reduce the complexity of the ELM detection methods.

2.4 Other work related to ELM detection

Previously, the research unit Nuclear Fusion (infusion) at Ghent University developed a method for ELM detection called Robust Optimal Threshold (ROT) [3]. This method is able to detect Type I ELMs but can struggle with detecting ELMs of other types. In Section 8.1.3 the implementation is discussed and in Chapter 11 the method's output is compared with the output of methods proposed by this master's dissertation.

2.5 Research questions and objectives

The primary aim of this master's dissertation is to develop reliable and accurate methods for detecting ELMs in tokamaks, using light emissivity and energy time series data. The research questions that guide this study are as follows:

- 1. Research question:** What technology and algorithms are suitable for ELM detection in tokamaks?
Objective: Investigate and evaluate various technologies and algorithms that can be employed throughout the dissertation for efficient and effective ELM detection. Consider both traditional signal processing techniques, such as filters, and explore other potential methods for feature extraction and detection.
- 2. Research question:** How can relevant features be extracted from the data to facilitate ELM detection?
Objective: Explore different approaches to extracting meaningful features from the light emissivity and energy time series data. Explore mathematical and machine learning techniques that can effectively extract and represent the distinguishing characteristics of ELMs in the data.
- 3. Research question:** What methods can be employed to detect ELMs using the extracted features?
Objective: Develop and compare different algorithms and methodologies for detecting ELMs based on the extracted features. Evaluate the performance of traditional algorithms, such as threshold-based approaches, in detecting ELMs. Additionally, explore the use of machine learning algorithms, to detect ELMs and compare their effectiveness with the traditional approaches.
- 4. Research question:** How to evaluate ELM detection methods?
Objective: Develop evaluation metrics specifically designed for assessing the performance of peak and ELM detection methods that output detected peaks as intervals.

2.6 Comprehensive description of ELM characteristics required for detection

Determining what qualifies as an ELM and what does is unfortunately not a completely objective task. Different experts might disagree and there are many edge cases. After consulting with industry experts and consulting existing papers [2] [5], the following guidelines were established to guide the development of detection methods.

- An ELM is always paired with a spike in the emissivity signal. However, the amplitude of the spike can vary greatly. Generally, the start of the spike is a sudden increase and the tail end of the spike is a more gradual decrease. In Figure 10.2 a typical ELM shape is illustrated.
- An ELM should be in general accompanied by a drop in the plasma energy. The amplitude of this drop, however, can be minuscule. Conversely, no ELMs can occur if the plasma energy is increasing.

There are additional guidelines that apply specifically to the detection of compound ELMs.

- If detected ELMs are close together and the energy signal is dropping continuously throughout this region (meaning no intermittent increases in energy) it is an argument for classifying them as a single compound ELM.
- If the emissivity signal does not fall back to the baseline between two ELMs but instead stays at an elevated level it is an argument to group the peaks together in a compound ELM.

It is important to note here that the second compound rule was only introduced towards the end of the master's dissertation, together with a new set of labels. As a result, not all methods respond equally well to this rule.

Figure 2.2 shows a manually labeled shot. Interesting areas are marked with a letter and are explained below.

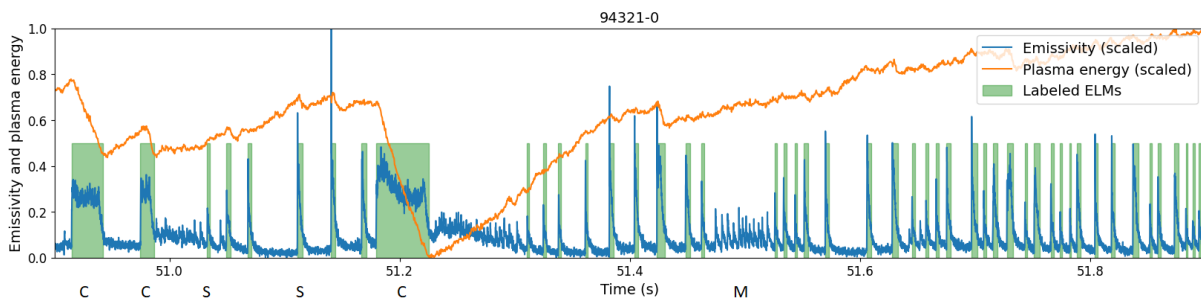


Figure 2.2: Shot 94321-0 with manually labeled ELMs marked in green.

- **Simple:** These are examples of normal ELMs. They correspond with a peak in amplitude of the emissivity signal and a drop in the plasma energy signal. Note that the drop in the plasma energy signal can be minuscule, as is the case in the first ELM marked with a **S**.
- **Compound:** Areas marked with this letter are examples of compound ELMs. They are classified as such because the emissivity signal remains at an elevated level during the entire interval.
- **Minor:** This region contains many small peaks that are not considered ELMs. This is because there is no corresponding drop in plasma energy.

2 Description of the problem

Finally, it is important to note that there are many edge cases. Examples of edge cases are the many small peaks in the **M** region and the first peak marked with an **S**. The amplitudes are small, and it is difficult to assess if there is a sufficient energy drop present. Experts in the field will have differing opinions on which of these edge cases should be classified as ELMs. Luckily, these edge cases are less important to the physicists, more important ELMs will often have a more pronounced signature.

3

Technology review

In this chapter, a few essential pieces of technology are explained, which are used throughout the master's dissertation.

3.1 Introduction to machine learning: algorithms, models and training

An important subfield of artificial intelligence (AI) is machine learning (ML) [6]. It focuses on developing algorithms and methods that let machines learn to execute tasks based on data. The machine learning field has many applications, like SPAM filtering for emails, object recognition in computer vision, and in the case of this master's dissertation, detecting ELMs. The main advantage of machine learning is that it can have the ability to learn directly from data. This allows the discovery of complex patterns, which could be hard for humans to manually define in a program.

Often, this is done by developing a model. Which is a mathematical representation of a system, able to provide the desired results from a set of input data. Take for example a SPAM filter: the model takes the header and body of an email and returns whether the mail server should flag it as SPAM or not. Going back to the master's dissertation, the models here are able to detect ELMs from light and energy data, captured by sensors in a tokamak device.

A model first has to learn what the desired results are. This is done by a process called training. Here two data sets are supplied. A training data set and a validation data set. The process works by iteratively adjusting the model's parameter based on the training data set and evaluating its performance using the validation data set. During training, the model analyzes the input, makes a prediction, and compares it with the desired output. The model gradually improves itself by iteratively minimizing errors it makes through the training iterations. Typically, this involves adjusting the model's internal weights. The quality of the data and the labels used during training determines the performance of the resulting model.[7]

3.2 Convolution operation and its properties

Convolution is a fundamental mathematical operation used in various domains, including machine learning and computer vision. It involves applying a filter or kernel to an input signal, resulting in a convolved signal. This operation plays a pivotal role in tasks such as feature extraction, image processing, and signal analysis, enabling the transformation and manipulation of data. Multiple ELM detection methods described in this master's dissertation rely on the convolution operation and its properties.

3 Technology review

The behavior of the convolution operation under this specific condition has proven especially useful: propose that a signal is constant for an unspecified amount of units of n that is larger than the length of the convolution kernel. The convolution operation reduces to the following:

$$(f * h)[n] = \sum_{m=-\infty}^{\infty} h[m] \cdot f[n - m]$$

If the kernel has a finite length 's' this function can be rewritten as:

$$(f * h)[n] = \sum_{m=0}^{s-1} h[m] \cdot f[n - m]$$

If the input signal doesn't change and has a constant amplitude 'c' during a period $>s$ units, the function is equal to the following only in that region:

$$(f * h)[n] = \sum_{m=0}^{s-1} h[m] \cdot c$$

Consequently, the constant can be brought outside the summarization and the output of the function is equal to the integral of the kernel multiplied by c . Again this only applies in the region where the input signal is constant:

$$(f * h)[n] = c \cdot \sum_{m=0}^{s-1} h[m]$$

This has two interesting consequences:

- If the integral of the kernel is equal to 1, the input and output functions will be equal. This means that filtering with a Gaussian PDF function, which has an integral equal to 1, does not change the amplitude of the input signal.
- If the kernel has an integral equal to 0, the output signal will be 0 as long as the input signal is zero. This means that if all values in the input signal are shifted by a large number, it will have no impact on regions where the input signal is constant.

3.2.1 Correlation operation

The correlation operation is intricately linked to the convolution operation, with a single distinction: the flipping of the kernel. Unlike convolution, where the kernel is flipped, correlation retains the original orientation of the kernel. Consequently, correlation and convolution yield identical results when the kernel is symmetrical.

In the context of this work, when an asymmetrical kernel is used, it will be explicitly specified whether the convolution or correlation operation is employed.

3.3 Filters in signal processing and machine learning

In the context of signal processing and machine learning, filters are mathematical operations or algorithms applied to data. In chapter 6, multiple filters will be discussed.

Traditional signal processing filters aim to modify the input data by applying specific mathematical operations. With the goal to extract specific features, reduce noise and enhance certain characteristics, such as smoothing neighboring values to reduce noise, convolving with a kernel and more.

On the other hand, there are also filters that are designed specifically for machine learning tasks. They aim to transform the raw input data in a way that improves the performance and convergence of the machine learning model. Typically, these filters help with normalizing, scaling, and encoding the input data to ensure compatibility with the requirements of the machine learning model.

3.4 Introduction to neural networks

A neural network (NN) [6] is a type of machine learning model that is inspired by the human brain. It is composed of nodes called artificial neurons or "units", organized in layers. The connections between the units of connecting layers have weights, which require to be trained.

3.4.1 Pattern recognition using convolutional neural networks

Convolutional neural networks (CNNs) are built around the concept of convolution layers. These layers apply a convolution operation to the input data and produce a convoluted output. In the context of CNNs, this operation involves applying a set of learnable filters, also known as kernels, to the input data. Each filter is a small matrix of weights that are adjusted during the training process. The purpose of these filters is to detect localized patterns and features within the input

The initial convolution layers in a CNN typically capture low-level features, while deeper layers learn to combine these low-level features to form higher-level representations. This hierarchical layering of multiple convolution layers allows CNNs to effectively extract and represent complex patterns. To enhance the robustness of the network, CNN architectures often incorporate additional layers, such as pooling layers, to reduce spatial dimensions and extract the most salient information.

3.4.2 Capturing dependencies across variable-length sequences using recurrent neural networks

Recurrent neural networks (RNNs) are specifically designed to handle sequential data, making them well-suited for tasks involving time series analysis in this master's dissertation. RNNs differ from other neural network architectures by introducing a feedback mechanism that enables information to persist and be shared across different time steps.

3 Technology review

The distinctive feature of RNNs is their ability to maintain an internal state, often referred to as memory, which is implemented as a layer within the network. This memory layer takes into account the current input as well as the output from the previous time step to update its internal state. The updated state is then passed to the next time step, allowing the network to incorporate information from previous steps when making predictions.

Multiple types of memory layers exist, such as long short-term memory (LSTM) and gated recurrent unit (GRU). These variants introduce specialized memory cells and gating mechanisms, enabling RNNs to capture and retain long-term dependencies more effectively.

The main benefit of RNNs is the possibility to model and capture dependencies across variable-length sequences. This makes them particularly suitable for tasks that involve analyzing and generating sequential data, where context and temporal relationships play a crucial role.

3.5 Hyperparameter tuning for machine learning algorithms using gridsearch

Gridsearch is a technique used to tune the hyperparameters of algorithms. This involves systematically going through a predefined set of hyperparameters, to identify the combination with the best results. Just like with the training of a machine learning model, a training set is required to find which hyperparameters yield results that are the closest to the desired output. Again, the quality of the training data is very important for the resulting performance of the algorithm.

Gridsearch will be used throughout the chapter 8.

3.6 Keypoint detection and description in computer vision using the SIFT algorithm

SIFT is a computer vision algorithm for keypoint detection and keypoint description. The aim of keypoint detection is to find recognizable points in an image of an object that can be found in other images of the same object. If for example two pictures are taken of the same object but from a different angle the same keypoint should be found in both images. The goal of keypoint description is to describe the keypoints in such a way that keypoints in one image can be mapped to keypoints in another image if the images are of the same object. Some of the applications for SIFT are object detection, object recognition, image stitching and many others. [8]

3.6.1 Keypoint detection

Because SIFT is able to detect the same keypoints regardless of varying levels of illumination it is said that the SIFT keypoint detector is "invariant" to changes in illumination. A good keypoint detector is strongly invariant to as many properties as possible which means that as long as two pictures are of the same object it will find the same keypoints. It is equally important that if a keypoint is not detected in a certain region of the original image, the corresponding region in the transformed image is also not detected as a keypoint. In practice being completely invariant is impossible, for example, if two pictures are taken of the same object, but one picture is taken from the opposite side it's impossible to find the same keypoints because the points on the object are occluded by the object itself. The figure below illustrates the different SIFT invariances using a mock-up of detected keypoints:

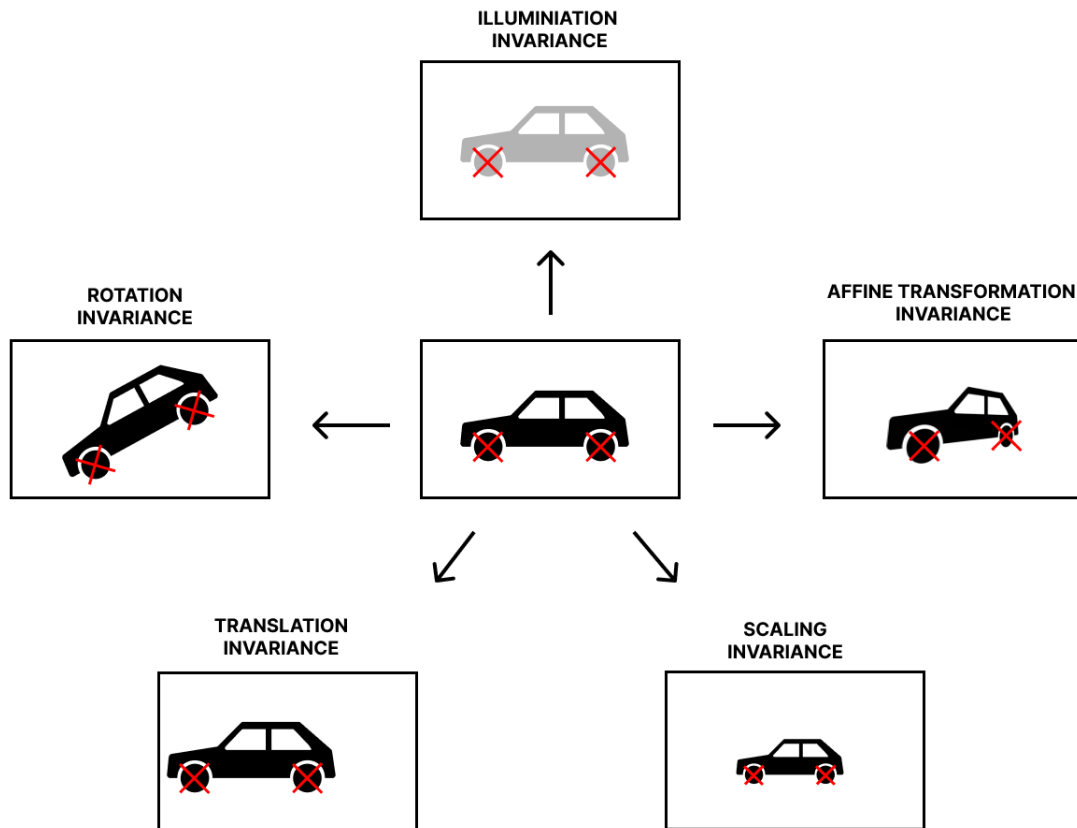


Figure 3.1: 5 types of SIFT invariances. Keypoints are marked with a red cross

The central picture represents the original image with a car as the subject. The SIFT keypoint detection algorithm was able to find two keypoints corresponding with the two visible wheels of the car. The five other pictures contain the same object but with some transformation applied to it. In all five examples SIFT should ideally find the same keypoints.

3.6.2 Simplified explanation of SIFT keypoint detection

In this section, a simplified version of the SIFT keypoint detection algorithm is presented. The purpose of this simplified algorithm is to provide a basic demonstration of the underlying principles of the SIFT algorithm. The original SIFT algorithm involves filtering the image using Gaussian kernels at various scales. By subtracting the filtered images from each other, distinct bright and dark blobs are extracted. From these blobs, robust keypoints are then generated.

In the simplified keypoint algorithm presented here, certain modifications have been made. Instead of employing the Difference of Gaussian technique, a Laplacian of Gaussian (LoG) kernel is utilized. Although computationally slower, this approach offers enhanced accuracy, as supported by the documentation of the SciKit-image image processing framework, which offers both solutions for blob detection [9]. Additionally, only dark blobs are detected by the simplified algorithm. In Figure 3.2

below, the different steps of the simplified algorithm are illustrated:

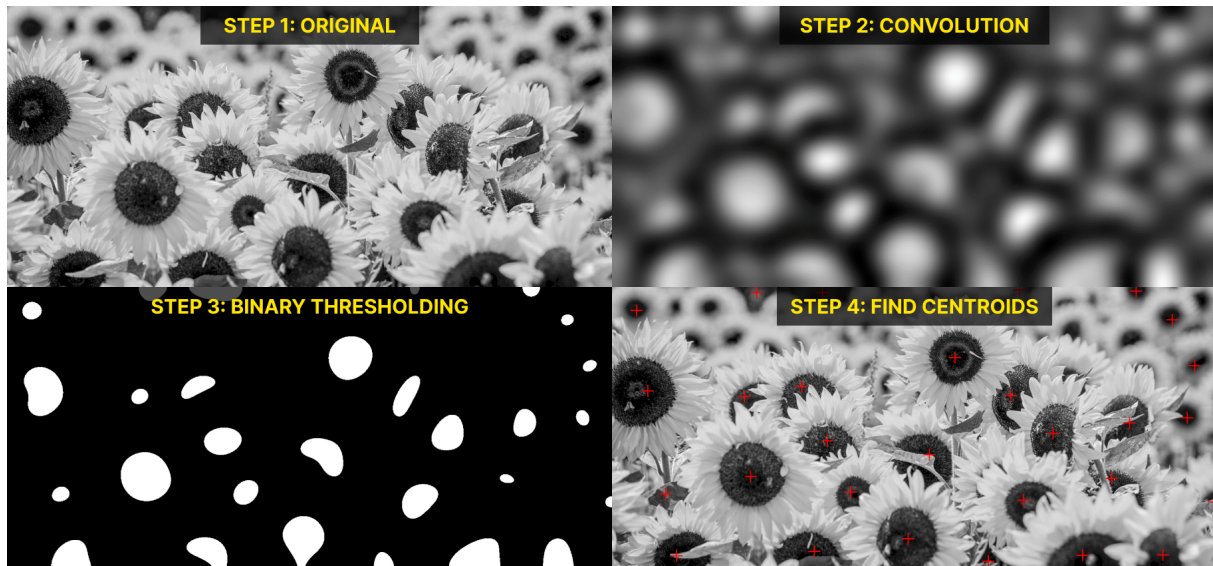


Figure 3.2: Keypoint detection steps, using an image of a sunflower field [1] as an example

Each panel contains a separate step in the custom algorithm:

1. First the image is loaded in memory and converted to grayscale.
2. A convolution is applied with a two-dimensional Laplacian of Gaussian (LoG) kernel. The dark hearts of the sunflowers now show up as white blobs. The resulting image is normalized
3. Binary thresholding is applied. The pixels with a value above a constant value are set to white and the others become black.
4. Finally, the centroids of the white blobs are calculated. These are the keypoints and are displayed on top of the original image as red crosses.

In Figure 3.3 below, two kernels are visualized by mapping the values to grayscale values. The left-hand image is the kernel used in the example from Figure 3.2 to detect the dark hearts of the sunflowers. It features a dark center with a bright ring around it. This kernel detects blobs that are darker than their immediate surroundings. The right-hand kernel is not used in Figure 3.2 but could be used to detect bright blobs on a darker background. It is created by multiplying every value in the kernel that detects dark blobs by -1.

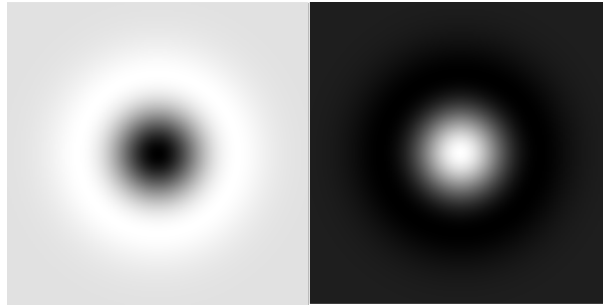


Figure 3.3: LoG kernels for image filtering in Figure 3.2.

The convolution step is responsible for complete illumination, translation and rotation invariance:

- Translation invariance is inherent to convolution.
- Rotation invariance: Because the kernel is symmetric in all directions complete rotational invariance is achieved.
- Illumination invariance: Because the kernel has an integral equal to zero the background brightness is normalized to zero in the convolved image and only the blobs remain. (Whether these are dark or bright blobs depends on the kernel).

SIFT is moderately invariant to affine transformations because blobs in images are often still recognizable from an angle. Finally, SIFT accomplishes scale invariance by running the blob detection multiple times at different scales. In this simplified algorithm, keypoint detection was only performed at one scale and as a result, some flowers at the top of Figure 3.2 are not detected as blobs because the kernel is too big.

3.7 Reusing code in Python through multiple inheritance with mixins

In Python, a mixin [10] is a technique for reusing code in multiple classes. A mixin is implemented as a separate class that provides a specific set of methods or attributes. By inheriting from a mixin, a class can incorporate the mixin's functionality into its own implementation. Unlike regular inheritance, inheriting from a mixin does not imply an "is-a" relationship. A mixin is not intended to be used as a standalone class. Instead, it is meant to be combined with other classes that provide the main functionality. Through multiple inheritance, a class can inherit from both the mixin and other relevant classes, incorporating the mixin's functionality while maintaining a flexible class hierarchy.

This approach avoids creating a deep class hierarchy and provides a flexible way to extend the functionality of different classes. The usage of mixins allows organizing and sharing common code in a modular manner, enhancing code reuse and maintainability. It promotes the separation of concerns and allows for greater flexibility in class design.

An example of mixin usage in this master's dissertation is demonstrated in Figure 4.1.

4

System architecture

4.1 Project Repositories: elms-detection, peaks, and dbtools

The project consists of three repositories. The main one is called elms-detection. The other two repositories, peaks and dbtools, are loaded within the main repository as submodules.

The elms-detection repository and its submodules can be found at: <https://github.ugent.be/infusion/elms-detection>. Information about how to run each method can be found in the README.md file of this repository.

Up until the embargo date of this master's dissertation, access is restricted on all repositories. After the embargo date, only the dbtools repository will have restricted access. To request access, contact J. Alhage by e-mail using jerome.alhage@ugent.be.

4.1.1 Distinction between ELMs-detection and peak detection

For broader applicability of this master's dissertation, a distinction is made between ELM detection methods and peak detection methods. The former are implemented in the main elms-detection repository under the 'elms-detection/methods' directory, while the latter are implemented in the peaks repository under the directory '/elms-detection/peaks/methods'. ELM detection methods often rely on peak detection techniques for their operation, which will be explained further in Section 4.1.3. However, it is possible to use all peak detection algorithms separately on other non-ELM-related data, albeit with different hyperparameters.

4.1.2 Modular peak detection and evaluation methods in the peaks repository

The peaks repository is a significant component of the elms-detection project, dedicated to detecting peaks from data. It encompasses a modular architecture that enables the implementation of multiple peak detection and evaluation methods. The repository consists of several key directories and files, each serving a specific purpose in the overall functionality of the system.

A prominent directory within peaks is the filters module. This directory houses a collection of filters implemented as mixins, see Section 3.7 and Chapter 6 for more information. They are easily integrated into other classes, enabling the seamless incorporation of various filtering operations within the peak detection process. By using mixins, the filters can be shared across different classes, promoting code reuse and maintainability.

4 System architecture

The most crucial directory within the peaks repository is the methods module. This directory contains a collection of peak detection methods, each implemented as a separate class. These methods are implemented using carefully designed algorithms, with each having its own advantages and limitations. All peak detection methods derive from a common abstract base class, which is found in the file "PeakDetection.py", within the same directory. This base class provides a standardized interface for accessing and utilizing the detection functionality. This allows straightforward interchangeability between the different peak detection methods and facilitates comparative analysis. Parameters used to tune a method are supplied with the constructor but can be updated after construction.

Figure 4.1 presented below illustrates various peak detection methods and their utilization of mixins.

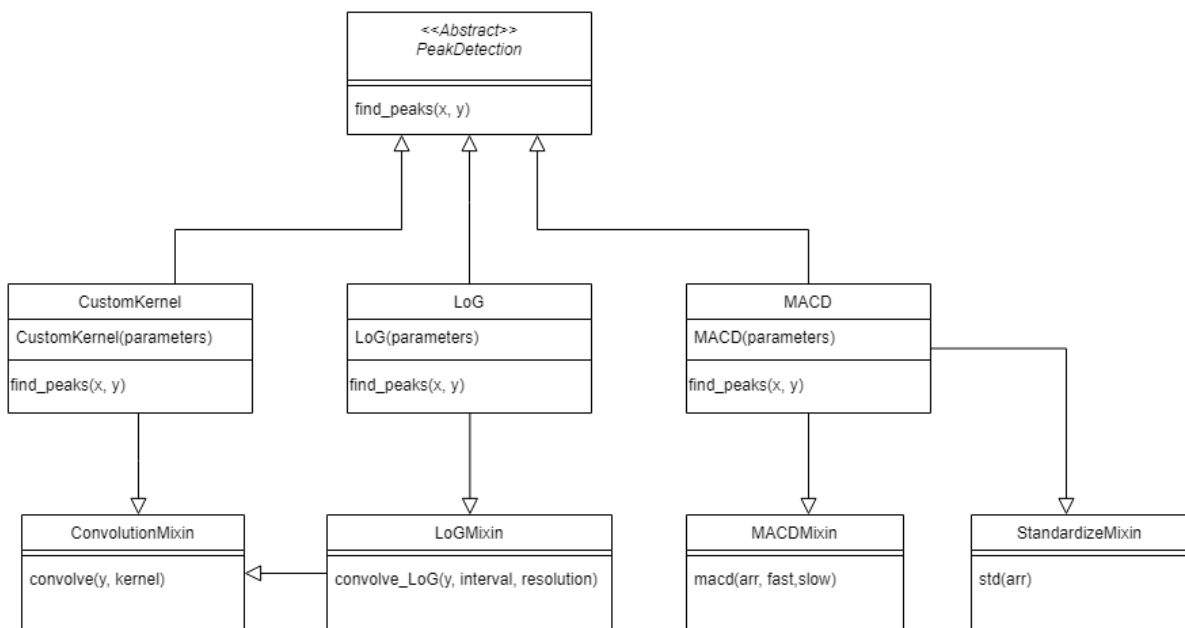


Figure 4.1: UML diagram illustrating the utilization of mixins in the peaks repository. Some attributes and classes are omitted to maintain clarity

The repository also contains an evaluation module, which plays a vital role in training and assessing the performance of the detection methods. It contains the necessary code to evaluate and compare the results obtained using the different methods. This provides valuable insights into the strengths and weaknesses of each detecting method, aiding in the selection and fine-tuning of the most suitable approach for specific applications. Multiple evaluation metrics have been developed, which are discussed in chapter 5.

4.1.3 Modular ELM detection methods in the elms-detection repository

The elms-detection repository is the main repository used for this master's dissertation. It contains the utilities and notebooks used for the dissertation, but also the code made specifically for detecting ELMs from time series data. It is designed to provide a comprehensive framework for ELM detection, incorporating various methods and techniques for accurate and

4 System architecture

reliable identification of these phenomena. Like the peaks repository, elms-detection follows a modular architecture that facilitates flexibility and extensibility.

The repository contains a directory called `compound`, which included methods dedicated to detecting compound ELMs. The methods within the `compound` module allow for the combination or removal of specific ELMs from a given list, enabling more precise results. As with the filters in the peaks repository, the `compound` methods are implemented as mixins, see Section 3.7.

Another notable directory in the `elms-detection` repository is the `methods` folder. Similar to the `methods` module in the peaks repository, this module includes an abstract base class called `ElmDetection` and derived classes representing different ELM detection methods, which implement the methods of the abstract base class. These classes leverage the peak detection results obtained from the peak detection methods and treat the detected peaks as potential ELMs.

ELM detection method may internally call peak detection methods if desired. Within these ELM detection methods, there are cases where the input data undergoes transformations using filters from the peaks repository or is passed through a neural network before being processed by the peak detection method. It should be noted that this preprocessing step is not always necessary, as in some cases, one of the signals may be passed directly to the peak detection method without any further transformation.

After receiving the output from the peak detection method, the ELM detection method can then implement additional logic and algorithms to refine and correct the detected ELMs, such as by using the `compound` mixins from the `"compound"` directory. This modular design allows for easy creation, implementation, and integration of different ELM detection methods and promotes flexibility in selecting the most suitable approach for a given data set.

4.1.4 Utility functions for data loading and resampling in the `dbtools` repository

This repository is separate from the rest of the codebase. Its purpose is to provide functionality for tasks such as loading and resampling the input data specific to JET. Therefore `"dbtools"` is only used in utility files and notebooks. It is not integrated with the inner workings of the other repositories. This allows the ELM detection methods to work with other sources too.

4.2 Continuous integration and continuous delivery

During the development, unit tests were created to ensure the correct functioning of the code. Additionally, code was developed to apply methods to a predefined set of shots, assess their performance, and store snapshots of the results on disk. This enables tracking the performance of the methods over time to ensure that no changes are made that unexpectedly lower their performance.

Initially, an attempt was made to automate this process using a CI/CD pipeline. However, this approach proved to be impossible due to data safety restrictions and Ghent University policy. Firstly, it is not allowed to upload any signal data from the JET repository on servers not hosted by the university, even on a private repository. Secondly, the CI/CD feature is not supported on the GitHub Enterprise server of Ghent University for cost-saving measures.

4 System architecture

Due to these limitations, compromises had to be made. The user can manually run the code that stores snapshots of the performance over time to prevent degraded performance. Additionally, a script was created to run all the unit tests in the repository automatically. These two approaches automate some of the most tedious tasks without relying on CI/CD pipelines.

5

Metrics For Evaluating Peak Detection Methods

To train and evaluate the ELM detection methods performance metrics are needed. Finding existing metrics posed a significant challenge because most existing metrics in the field of machine learning are meant to be applied to classification tasks and are not directly applicable to peaks represented as intervals.

For this reason, new metrics were created that are compatible with the data structure for peaks used in this work. The new metrics should be grounded on existing metrics in the ML field, mathematically sound, and easy to understand intuitively.

For brevity the manually labeled peaks that are regarded as the ground truth are called "real peaks" from now on. ELMs that are automatically detected by one of the ELM detection methods are called "automatic peaks".

5.1 Imbalanced datasets

Due to the nature of ELM signals, it is more likely that an ELM event is not occurring at any given time. In the dataset utilized for this master's dissertation, approximately one-fifth of the total labeled time corresponds to ELM events, while the remaining four-fifths represent non-ELM periods. As a result, the majority of the signal consists of negatives or gaps between peaks, while positives occupy only a small portion. However, when considering peaks and gaps as individual units, the situation changes. There can be, at most, one more gap than the number of peaks. Therefore, in signals with multiple ELMs, the ratio is typically close to 50%.

Another challenge arises from the varying lengths of peaks. Compound ELMs, discussed in Section 2.6, can be considerably longer than their normal counterparts. It is crucial that metrics do not allow compound ELMs to disproportionately affect the overall score.

To address these challenges, metrics should focus on capturing the fraction of overlap between peaks and gaps, rather than solely considering the actual time of overlap.

5.2 Interval Metrics

These metrics apply the confusion matrix from statistical classification to peak detection. To do this they look at the interval of both the real and automatic peaks.

5 Metrics For Evaluating Peak Detection Methods

5.2.1 True Positive

The true positive metric examines how many of the real peaks are also detected by the automatic method. The result is presented as a fraction e.g. "5/10 real peaks were detected". This metric is represented symbolically as:

$$TP(\text{automatic peaks, real peaks})$$

To calculate this the method first iterates over the intervals of all real peaks. It calculates what fraction of the interval of the real peak is also detected by one or more automatic peaks. A configurable threshold is used to create a pass/fail grade. After doing this for all real peaks the metric outputs what fraction of real peaks passed.

5.2.2 False negative

The false negative metrics look at how many real peaks were missed in the list of automatic peaks. Because of this, it can be calculated by taking the complement of the fraction returned by the previous metric, e.g. if the true positive metric was equal to "4/10 real peaks detected" it means that the false negative is "6/10 real peaks missed". The false negative is represented symbolically as:

$$FN(\text{automatic peaks, real peaks}) = 1 - TP(\text{automatic peaks, real peaks})$$

5.2.3 False positive

The false positive metric describes what fraction of the automatic peaks do not exist, meaning that they have no corresponding real peaks. This metric is calculated by iterating over all automatic peaks instead of the real peaks as the true positive metric did. Per automatic peak, it is calculated what fraction of its interval is not present in any real peaks. If this value is greater than a configurable threshold the automatic peak is counted as a false positive. After iterating over all automatic peaks the metric returns the end result as a fraction e.g. "3/10 automatic peaks are false positives".

This metric can also be calculated using the existing true positive metric by swapping the arguments:

$$FP(\text{automatic peaks, real peaks}) = 1 - TP(\text{real peaks, automatic peaks})$$

By applying the true positive metric with the automatic peaks as the first argument and the real peaks as the second argument, it calculates the fraction of automatic peaks detected in the list of real peaks, instead of the usual fraction of real peaks detected in the list of automatic peaks. Complementing this fraction gives the ratio of automatic peaks that are not detected in the list of real peaks, representing the false positive metric.

5 Metrics For Evaluating Peak Detection Methods

5.2.4 True Negative

In statistical classification, the true negative metric explains how many non-peaks are correctly identified as non-peaks by an ELM detection algorithm. At first glance, it seems like this method requires looking at the intervals between peaks. However, when adhering to the definition in statistical classification the true negatives are the complement of the false positives, e.g. when 3/10 actual negatives were incorrectly classified as false positives 7/10 actual negatives must be true negatives.

$$\text{TN}(\text{automatic peaks, real peaks}) = 1 - \text{FP}(\text{automatic peaks, real peaks})$$

When substituting the definition of false positives obtained in section 5.2.3

$$\text{FP}(\text{automatic peaks, real peaks}) = 1 - \text{TP}(\text{real peaks, automatic peaks})$$

the final definition of true negatives becomes the following:

$$\text{TN}(\text{automatic peaks, real peaks}) = \text{TP}(\text{real peaks, automatic peaks})$$

This intuitively makes little sense which is why this metric is only listed for the sake of completeness. For a more intuitive definition of true negative that does take into account the space between intervals see the next section 5.3.

Figure 5.1 shows a series of examples where the real peak intervals are marked in green and the automatic peaks in red. At the top of each example the true positive, false negative, false positive, and true negative scores are displayed assuming a threshold of 0.5.

5 Metrics For Evaluating Peak Detection Methods

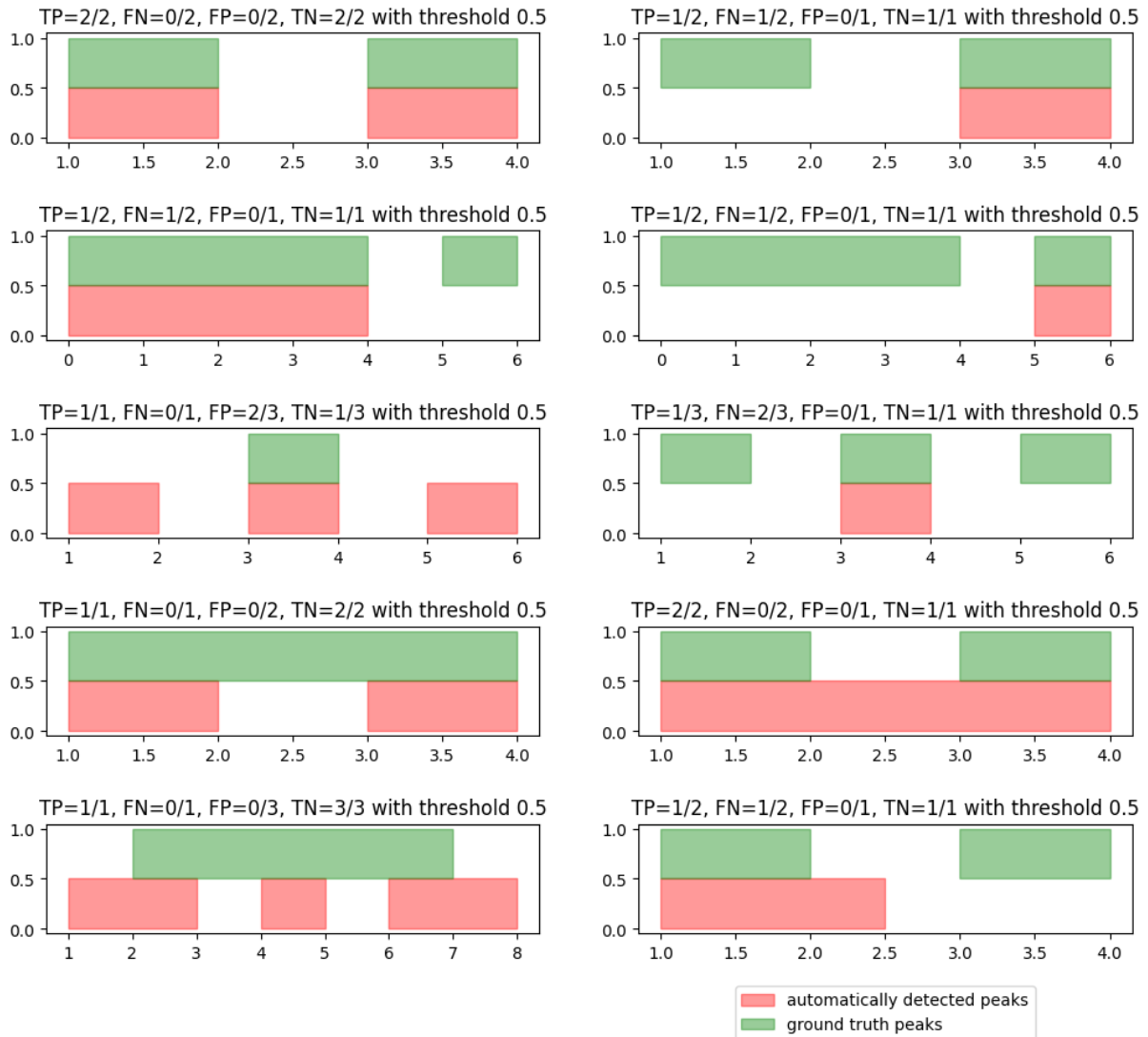


Figure 5.1: Custom Interval Metric - Examples

5.3 Gap Metrics

Gap Metrics are similar to the Interval Metrics discussed previously. The implementation for the true positive and false negative is entirely the same but the false positive and true negative are calculated differently as explained in the sections below.

5.3.1 False Positive

The false positive now gives the fraction of gaps between real peaks that are falsely detected as real peaks. The metric first converts the list of ground truth peaks to a list of gaps between and around the peaks. The gaps are considered as negatives compared to the peaks which are the positives. Then the metric uses the true positive implementation from 5.2.1 to find what

5 Metrics For Evaluating Peak Detection Methods

fraction of gaps or negatives are falsely detected by automatic peaks. Again a configurable threshold is taken into account to adjust how sensitive the metric is. This method can be represented symbolically as:

$$FP(\text{automatic peaks, real peaks}) = TP(\text{automatic peaks, real gaps})$$

Note that even though there could be more automatic peaks than gaps the false positive can at most be equal to one because the true positive method iterates only over the second argument which are the gaps in this case.

5.3.2 True Negative

The true negative calculates what fraction of gaps between real peaks is correctly not detected as peaks or positives. The true negative is equal to the complement of the false positive fraction.

5.3.3 Conclusion

Defining the false positive to look at the gaps has two advantages. The first is that a negative is clearly defined as a gap between peaks. The positives and negatives intervals add up to the total time in the shot. The second advantage is that the definition of false positives and especially true negatives becomes more intuitive to understand.

In spite of these advantages, this metric should not be used for training for the current dataset. As discussed in Section 5.1, the gaps between peaks last much longer than the peaks themselves. This means that an automatic peak placed in the middle of a gap is very unlikely to result in a false positive.

To illustrate this problem a grid search was performed on the LogDoG ELM detection method (Section 8.5) using an F_{β} -score ($b=0.5$) calculated from the true positive and false positive metrics provided by Gap Metrics with a threshold of 0.75 or 75%. This means that 75% of a real peak needs to be detected by automatic peaks in order to count. Similarly, 75% of a gap between real peaks needs to be falsely detected in order to count as a false positive. In Figure 5.2 the trained method is applied to a shot with large gaps. Multiple automatic peaks are falsely detected in the gaps between the real peaks.

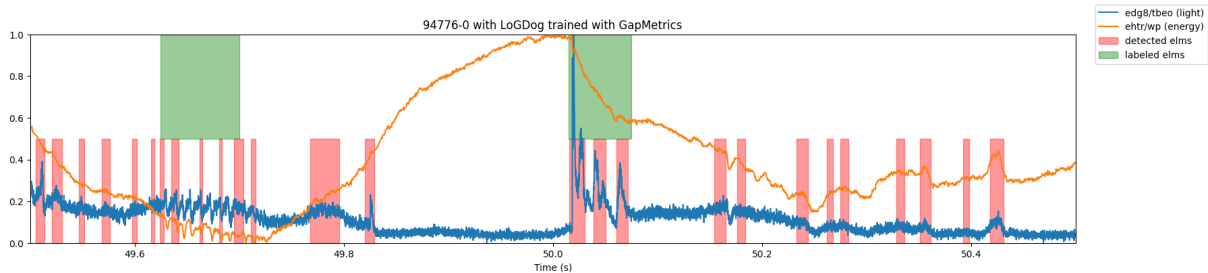


Figure 5.2: LogDoG ELM detection method trained with Gap Metrics applied to shot 94776-0. Note: This figure is based on the old labels

This figure clearly demonstrates that the method has not learned to avoid false positives. The reason why is explained using the following table which compares the Gap and Interval Metrics. The gap metrics find three gaps between and around the

5 Metrics For Evaluating Peak Detection Methods

Table 5.1: Comparison between the results of the Gap Metric and Interval Metric for shot 94776-0

	TP	FN	FP	TN
Gap	0/2	2/2	0/3	3/3
Interval	0/2	2/2	18/26	8/26

two real peaks. None of which are more than 50% detected by real peaks, so no false positives are detected in the entire shot. In contrast, interval metrics do detect 18 false positives.

Adjusting the threshold for the false positives to a very low number (10%) would remedy this problem, but it would lead to many false positives when the peaks are close together. In that scenario covering 10% of a small gap between peaks can happen easily even with a good ELM detection method. In other words, this would lead to too many false positives in the case of small gaps between peaks and not enough false positives when the gaps are bigger.

Because of these shortcomings, it is not recommended to use GapMetrics for training, instead, it is opted to use IntervalMetrics. However, both GapMetrics and IntervalMetrics are used for evaluating the methods in the results chapter 11.

5.4 Precision

Precision, also known as the positive predictive rate, is a metric that measures the accuracy of positive predictions and is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

5.5 Recall

Recall, also known as the true positive rate or sensitivity, measures the rate of TP predicted out of all positive instances.

$$Recall = \frac{TP}{TP + FN}$$

5.6 F-score

The F-score is a measure of accuracy, that combines precision and recall. The F_1 -score, traditionally used, is calculated using the harmonic mean, defined in the following formula:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

However, a more general F-score, known as the F_β -score, can be employed. The F_β -score allows for different weights between precision and recall. It is calculated as follows:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times precision + recall} = \frac{(1 + \beta^2) \times TP}{(1 + \beta^2) \times TP + \beta^2 \times FN + FP}$$

5 Metrics For Evaluating Peak Detection Methods

In essence, β controls the relative importance of precision and recall. A higher value of β makes recall more important, while lower values place more emphasis on precision. Notice that if β is equal to one, the F_β -score can be reduced to the F_1 -score.

The ability to configure the metric to prioritize between precision and recall offers a configurable preference between false negatives (FN) and false positives (FP). By adjusting the metric its priority between precision and recall, the prioritization of ELM detection methods can be indirectly influenced. Because of this flexibility, the F_β -score metric is used in Section 3.5 and the machine learning methods described in Chapter 9. A β value of 0.5 was chosen to make the methods more precise.

5.6.1 Other metrics

Furthermore, a host of other metrics from statistical classification can be calculated based on the true positive, false positive, false negative, true negative obtained either with IntervalMetrics or GapMetrics. Examples of implemented metrics are:

- False Negative Rate
- False Positive Rate
- True Negative Rate
- Positive Likelihood Ratio
- Negative Likelihood Ratio
- Diagnostic Odds Ratio
- Threat Score
- Positive Predictive Value
- False Omission Rate
- False Discovery Rate
- Negative Predictive Value
- Balanced Accuracy
- Fowlkes-Mallows Index
- Markedness
- Matthews Correlation Coefficient

6

Signal Processing Filters

Filters are fundamental tools used in signal processing and machine learning to preprocess, transform and enhance data. They enable the extraction of specific features, reduction of noise, enhancement of certain characteristics, and the training of machine learning models. Ultimately this leads to more accurate analysis, predictions, and insights.

The peak detection modules for this master's dissertation are written in a modular way so components can be shared between methods. This is in part accomplished through the use of filters implemented as Mixins in Python. See system architecture 4.1.2 and the technology review 3.7 for more information about this.

A filter takes a signal as an input and applies a certain operation to it. Examples of operations are applying a moving average, noise removal with Gaussian Blur, or normalizing the signal.

Some applications of filters are discussed. Firstly, the baseline of the signal is not always centered around zero, this can be remedied with a filter that detrends the signal. Secondly, the data is characterized by significant noise, which can be mitigated by filters that effectively reduce noise levels while preserving relevant peaks. Finally, the employment of filters facilitates trend analysis of the data.

6.1 Noise removal using the moving average

The moving average [11] is a method used to smooth data by calculating the average value within a sliding window. The window represents a fixed number of consecutive data points. It is shifted along the time axis, creating overlapping sections. Each section is then averaged, resulting in a smoothed data series. For example, with a window size of 5, the moving average is calculated by averaging the first 5 values, then shifting to the next section, and so on. This process reduces noise while preserving important trends in the data. The implementation using SciPy [12] of the moving average is straightforward and can be achieved in a single line of code as seen in Code Fragment 6.1.

```
scipy.signal.convolve(data, np.ones(span), "same") / span
```

Code Fragment 6.1: Moving average using SciPy

To apply convolution as a kernel on the input data, an array of ones with a length of 'span' is created. The convolution operation involves sliding this kernel along the input array and computing the sum of the element-wise products within the

6 Signal Processing Filters

span. The resulting sum array is then divided by the span to obtain the average value.

However, a limitation of this approach is that it produces an output with a shorter length than the input data. This occurs because the convolution operation is performed only when the kernel is fully contained within the input data, leading to data loss at the edges.

To address this issue, padding is applied using NumPy [13]. Which is shown in Code Fragment 6.2. In this case, the array is extended at the front with 'span' times the first value and at the back with 'span' times the last value. This padding, known as 'same' padding, ensures that the original input length is maintained after the convolution. The padding is subsequently removed, resulting in the desired output length.

```
data = np.insert(data, 0, np.repeat(data[0], span))
data = np.append(data, np.repeat(data[-1], span))
scipy.signal.convolve(data, np.ones(span), "same") / span [span:-span]
```

Code Fragment 6.2: Improved moving average using SciPy and Numpy

In Figure 6.1 below, the input data is plotted together with the output of the previous code fragments. Two different spans are applied to clarify the effect of the parameter. If the span is set to big, too much information in the signal is removed. On the other hand, if the span is too small, not enough noise will be removed.

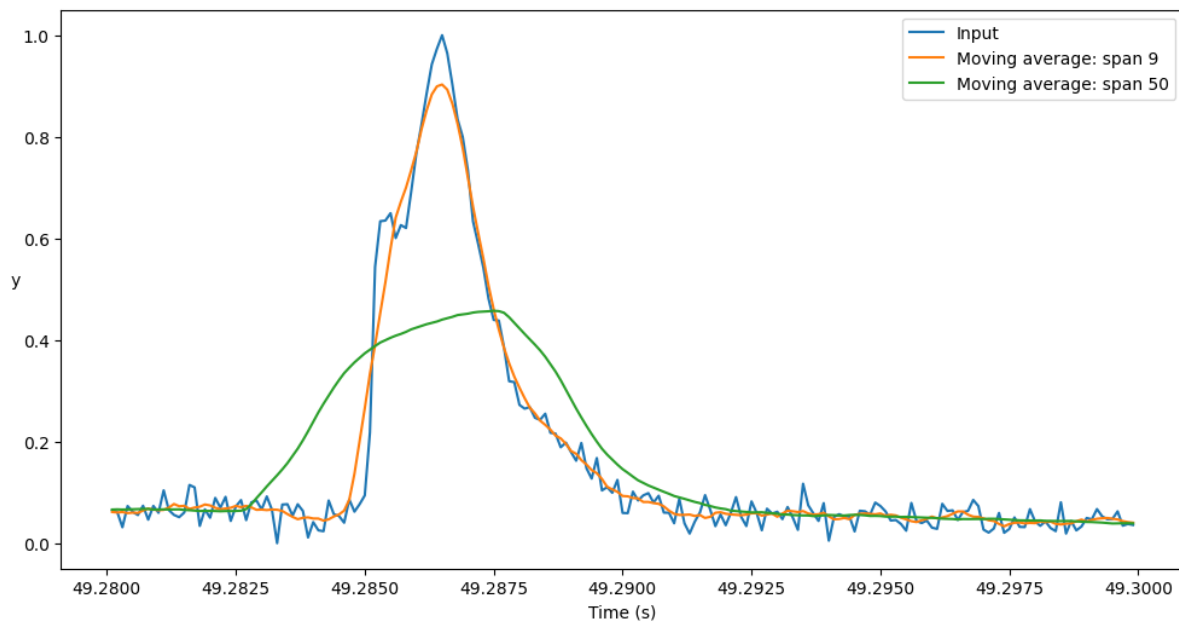


Figure 6.1: Removing noise from a signal using the moving average filter.

6.2 Noise removal using the exponential moving average

The exponential moving average (EMA) [14], as seen in Figure 6.2, is a variant of the moving average. Here, more weight is given to the values closer to the end of the subset. A decay α is calculated based on the span: $\alpha = 2 / (\text{span} + 1)$. This

6 Signal Processing Filters

has the advantage of still keeping some important features of the input that otherwise would have been smoothed away.

Depending on the span, more or fewer features and noise are removed. Careful consideration is needed when selecting the span width.

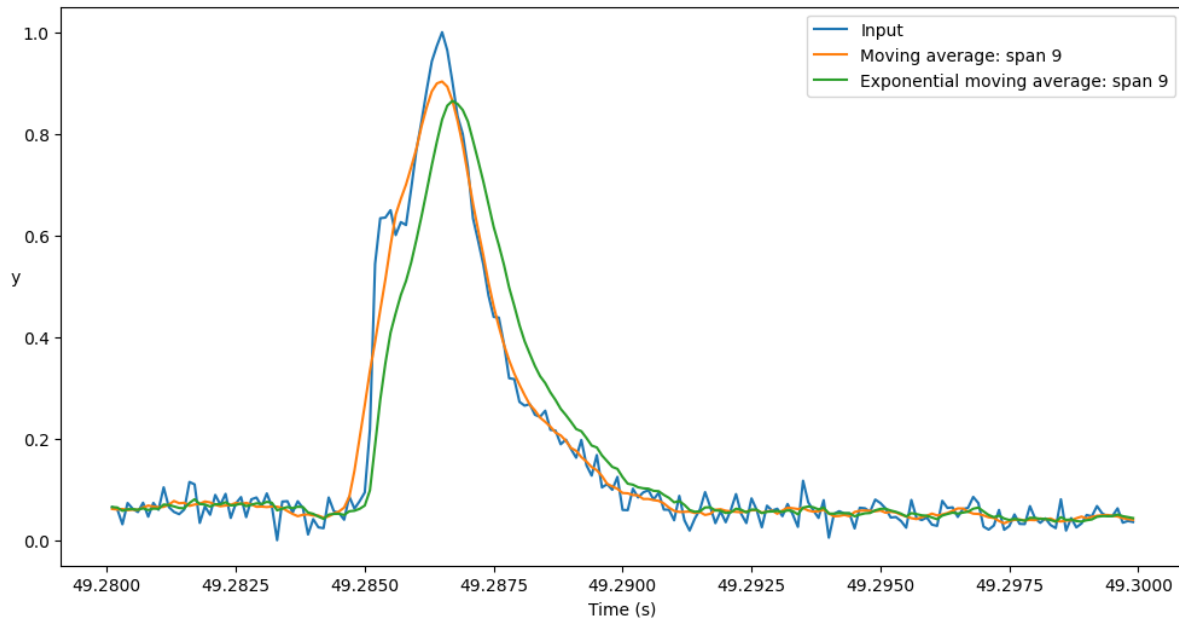


Figure 6.2: Comparing the moving average against the exponential moving average.

6.3 Noise removal using the Wiener filter

The Wiener filter [15], plotted in Figure 6.3, is yet another method for removing noise from data. The filter tries to determine how the signal and the noise are related by analyzing both. This works by estimating the power spectral density (PSD) of the data and the noise. Then it derives a filter that minimizes the mean-squared error (MSE) between the original signal and the filtered signal. This filter is specifically designed to maximize the signal-to-noise ratio of the output signal.

The minimization of the MSE makes it an effective technique for enhancing the quality of noisy signals.

6 Signal Processing Filters

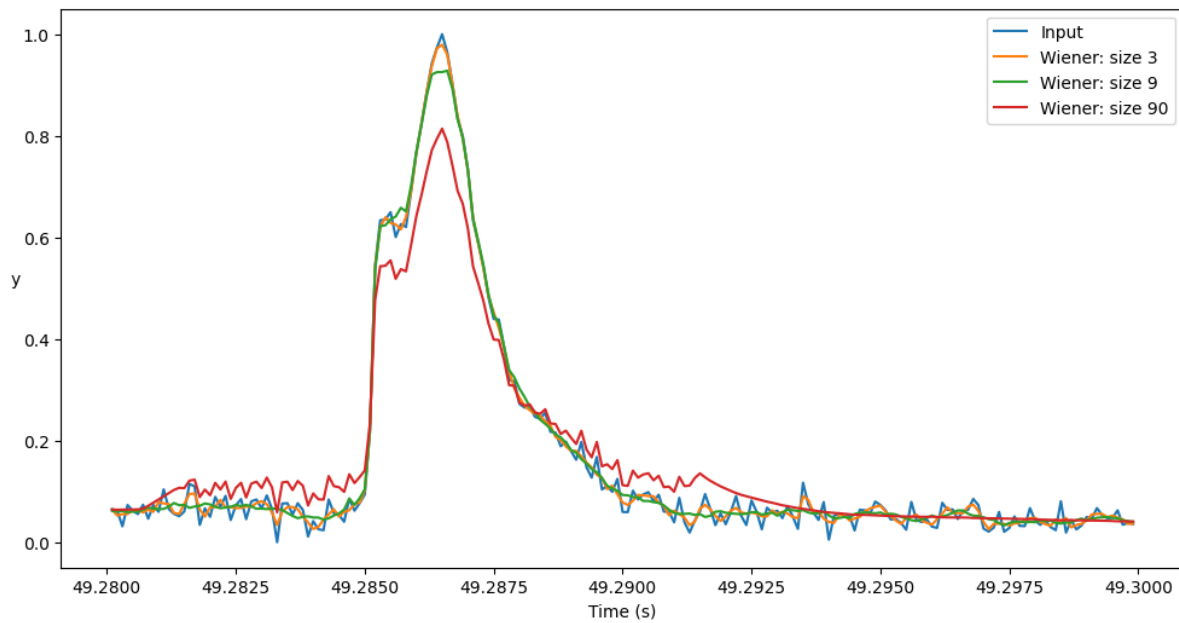


Figure 6.3: Applying multiple Wiener filters to the same input. Each filter has a different size parameter.

6.4 Noise removal using the Hann window

The Hann window [16] is used to reduce spectral leakage compared to the previously discussed noise removal filters. It does this by tapering the edges of the signal to zero. This means that it gradually reduces the amplitude of the signal window towards its edges or endpoints. This reduces abrupt discontinuities at these edges, which could cause spectral leakage and result in a much smoother signal. The difference with the moving average filter is shown in Figure 6.4.

6 Signal Processing Filters

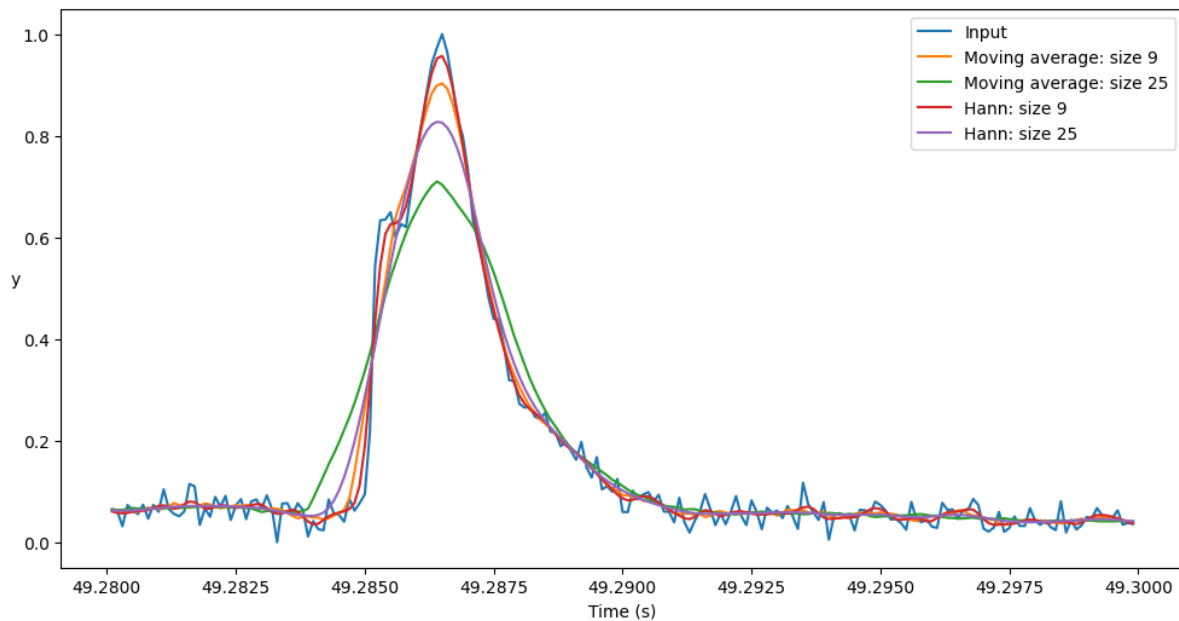


Figure 6.4: Comparing the moving average filter against the Hann filter.

6.5 Detect changes and anomalies using the difference filter

The difference filter [16] [17] can be used to detect changes and anomalies in a signal. This allows applications like edge detection in image processing, feature extraction in audio processing, and trend analysis in time series.

The difference filter calculates the discrete first-order derivative of a signal. This technique involves calculating the difference between two adjacent values. By doing this for every value in the given input array, a new signal is achieved. The main weakness of the difference filter is that high-frequency noise and small fluctuations are amplified in the output. Figure 6.5 shows the difference filter applied on a signal.

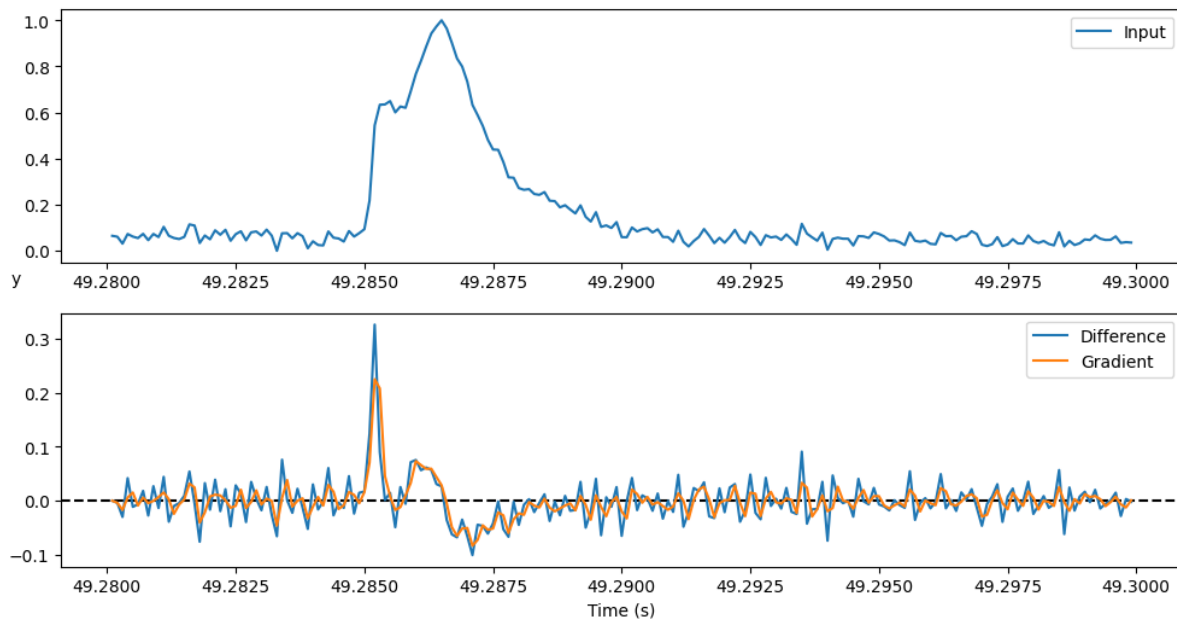


Figure 6.5: Applying the difference and gradient filters to a signal.

6.6 Detect changes and anomalies using the gradient filter

The gradient filter [18] describes the rate of change of an input array using NumPy's gradient function. In the context of time series data, the gradient filter provides an estimate of the slope or, as mentioned, the rate of change of the input array. This continuous estimate provides a more nuanced output compared to the difference filter. Figure 6.5 show the small smoothing effect that the gradient has on the 1-dimensional input array compared to the difference.

6.7 Noise removal, change and anomaly detection using the Savitzky-Golay filter

The Savitzky-Golay (Savgol) filter [19] is a mathematical method for removing noise from data. The filter works by fitting a polynomial function to a window of the input data. This polynomial function is then used to estimate the value for the center of this window. By moving the window along the data, and continually refitting the function, a new set of values is calculated.

Compared with a moving average, it does not give an average within a window but an approximation computed from a polynomial function. This unique property can provide a more accurate and precise approximation to the original input but without noise. The difference between these two approaches only becomes clear when greater windows are used, as seen in 6.6.

6 Signal Processing Filters

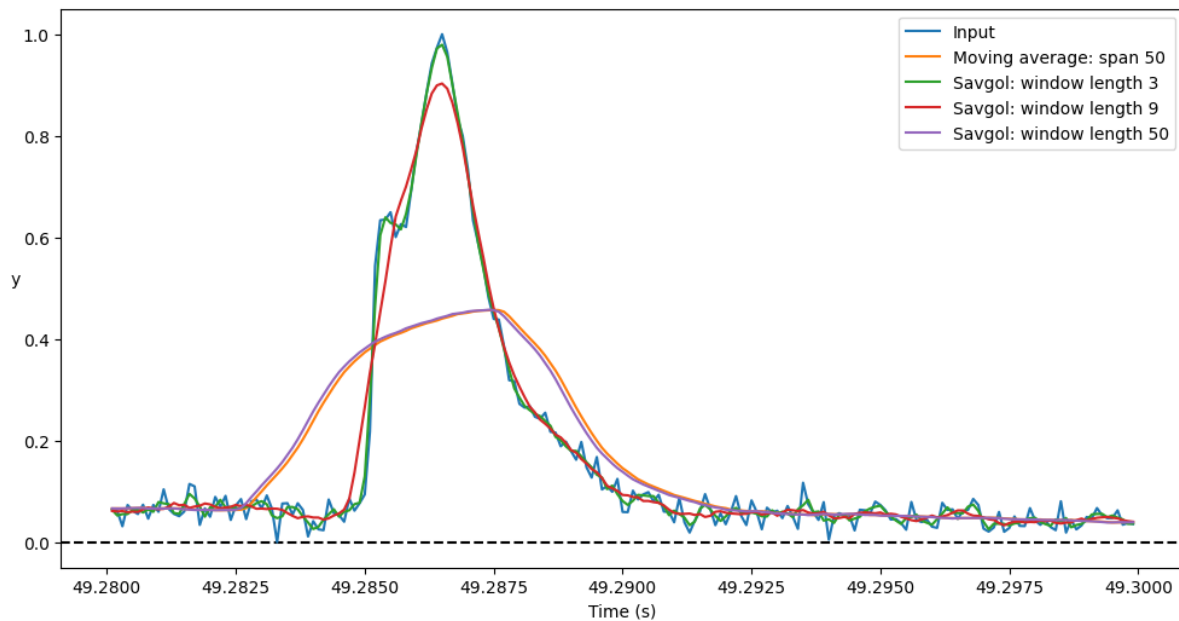


Figure 6.6: Removing noise from data using the Savitzky-Golay filter compared to the moving average filter.

The SciPy implementation of the Savitzky-Golay filter offers an additional feature: support for derivatives. By utilizing the filter with a derivative order of one, an output similar to the difference and gradient filters will be achieved. Additionally, depending on the window size, the filter leverages its polynomial fitting approach to effectively remove noise from the signal. This negates the effect where a difference filter would amplify small fluctuations from high-frequency noise. Figure 6.7 compares the Savitzky-Golay filter with the difference filter and shows the noise removal property.

6 Signal Processing Filters

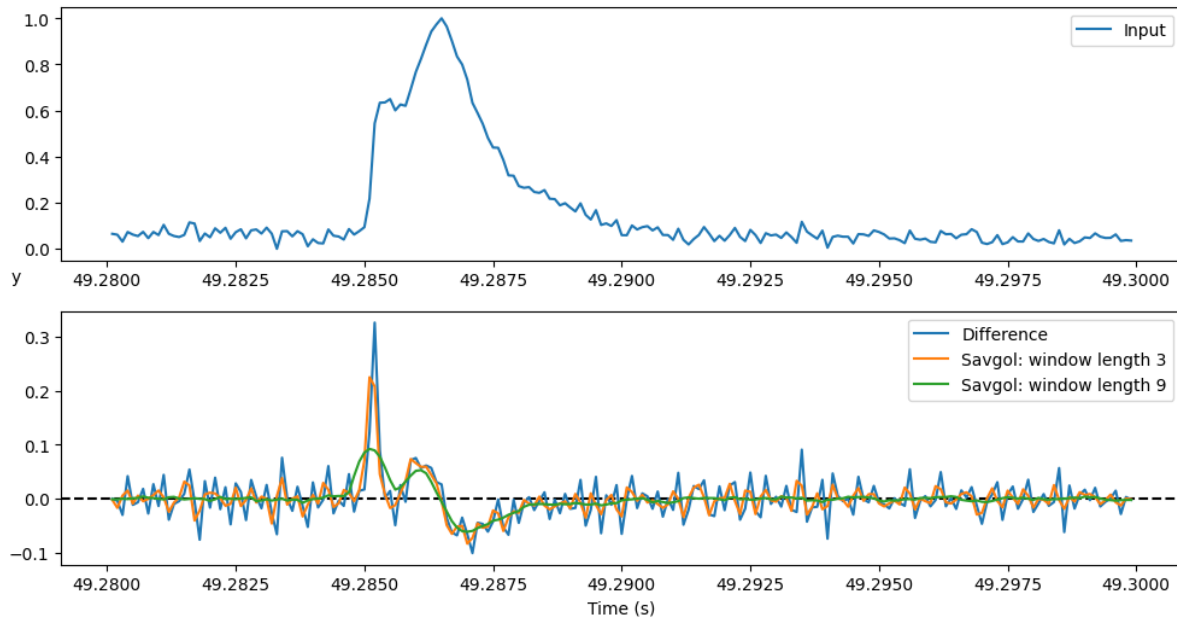


Figure 6.7: Removing noise from the first derivative of a signal using the Savitzky-Golay filter prevents the amplification of small fluctuations caused by high-frequency noise in the input data.

6.8 Detect changes and anomalies using the rolling Z-score

A standard score or Z-score [20] is a statistical measure that quantifies how many standard deviations a value is from the mean of the data set. The rolling Z-score is frequently used in the financial market to detect outliers or anomalies.

The Z-score is calculated using the following formula $Z = (X - \mu) / \sigma$. In this formula, X is the data set. μ is the mean of the data set. σ is the standard deviation of the data set.

A rolling Z-score is a variation on the Z-score, where a rolling window is used over the data set to get an output set of Z-scores. The effect of the Z-score on a signal can be seen in Figure 6.8. The sudden rise in the input results in a sharp peak when passed through the rolling Z-score filter. Note that the rolling Z-score also amplifies small peaks caused by high-frequency noise.

6 Signal Processing Filters

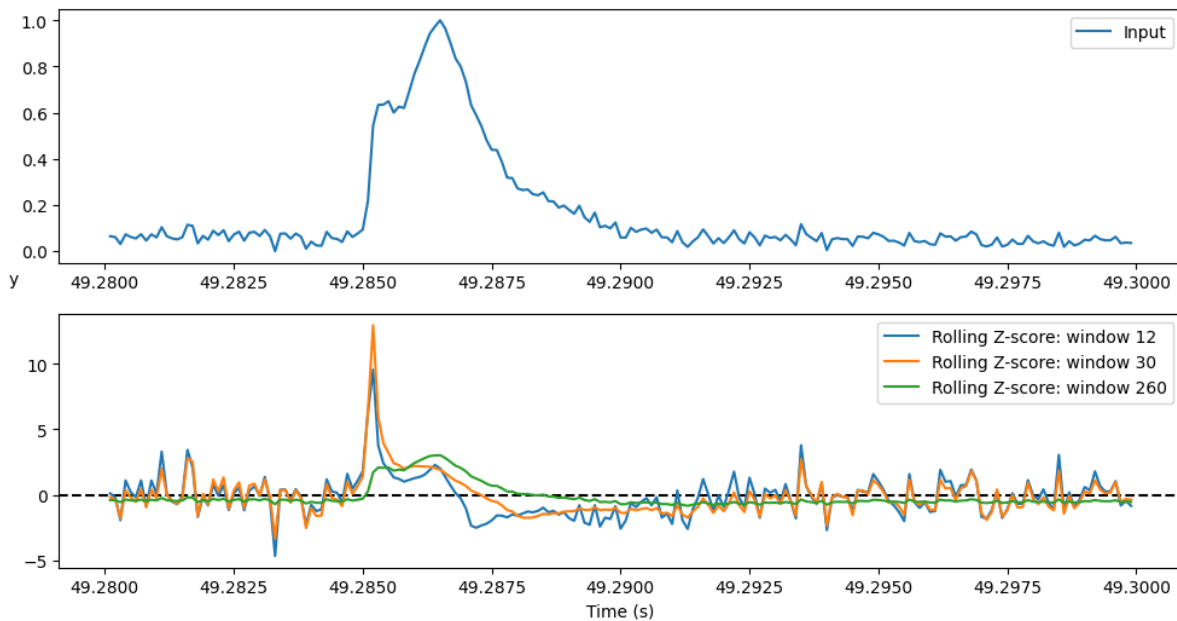


Figure 6.8: Applying multiple rolling Z-score filters to the same input. Each filter has a different size parameter.

6.9 Noise removal, change and anomaly detection using the MACD filter

The Moving average convergence/divergence (MACD) [21] is a popular indicator for technical analysis of the stock market. By comparing exponential moving averages with different span widths, changes in stock trends can be indicated. It works by calculating the difference between two EMAs with different spans. Two commonly selected span widths are 12 and 26 in the financial sector. The one with the greater span is subtracted from the EMA with the smaller span. The result of this operation is a signal that is centered around zero, as seen in Figure 6.9.

An exponential moving average of the MACD can also be calculated. This is called the signal line. Typically, a span width of 9 is used for this EMA in the financial sector. When subtracting the signal line from the MACD, a new zero-centered signal is achieved. Which is called the divergence line or just divergence. A convergence is happening when the divergence line is moving closer to zero. A divergence is happening when the line is moving away from zero. Whenever the divergence is positive, the input signal is increasing. Whenever the divergence is negative, the input signal is decreasing.

6 Signal Processing Filters

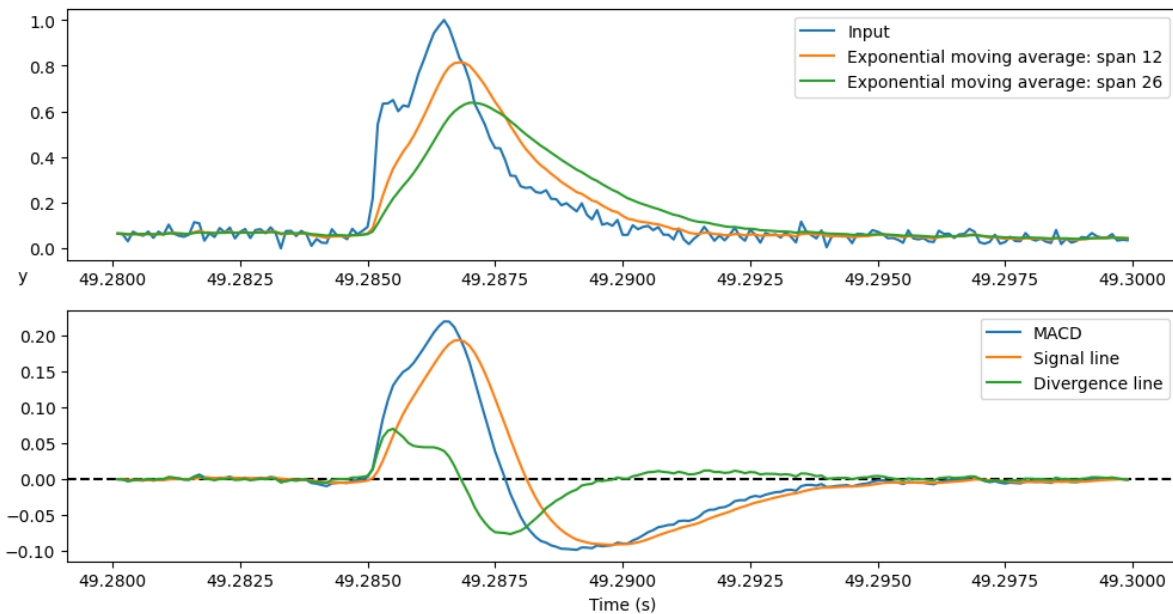


Figure 6.9: MACD with EMA 12 and EMA 26, calculated from the input. The signal line is an EMA 9 applied on the MACD.

It is important to note that MACD can be interpreted as a low-pass filter. The high-frequency noise is removed and only the important features are retained. Additionally, the signal is zero-centered, which makes it easy for detecting peaks using a simple threshold mechanism.

Finally, note that the MACD indicator is a lagging indicator. This means that the output signals always lag behind the current trend. Peak detection methods should take this into account.

Compared with the difference filter, where input noise is amplified in the output, the MACD can provide more nuanced information about the trend and momentum of the data. However, implementation is a bit more complex to comprehend.

6.10 Custom Kernel filter for executing convolutions

This filter convolves the input signal with a user-supplied kernel. Different convolution filters that will be discussed below all use this filter to execute the convolution. The filter allows for the use of different convolution implementations from OpenCV [22], SciPy [12], and NumPy [13]. Before the convolution is applied the amount of items in the kernel is checked to be uneven. If this is not the case the filter returns an error. This is done to ensure consistent behavior because kernels with an even amount of items are treated differently in some implementations.

Convolutions can be executed extremely quickly in parallel on a GPU but most libraries require the kernel and input arrays to have a certain data type. Examples of frameworks that allow GPU-accelerated convolutions are OpenCV [22] and CuPy

(NumPy and SciPy for GPU) [23].

6.11 Gaussian filter for noise removal

The Gaussian filter is a convolution with a kernel which contains a Gaussian probability density function or (Gaussian PDF). The operation is commonly used in image processing to remove white noise where each pixel has been changed by a random deviation from its original value. An example of this can be found in the 'GaussianBlur' function of OpenCV. [22]

When applied to a time series signal the convolution replaces each value by a weighted average of the nearest values where closer values have a larger weight and impact on the result. [24] In the figure below the Gaussian filter is applied to an artificial signal and the result is normalized, so the amplitude is equal to 1:

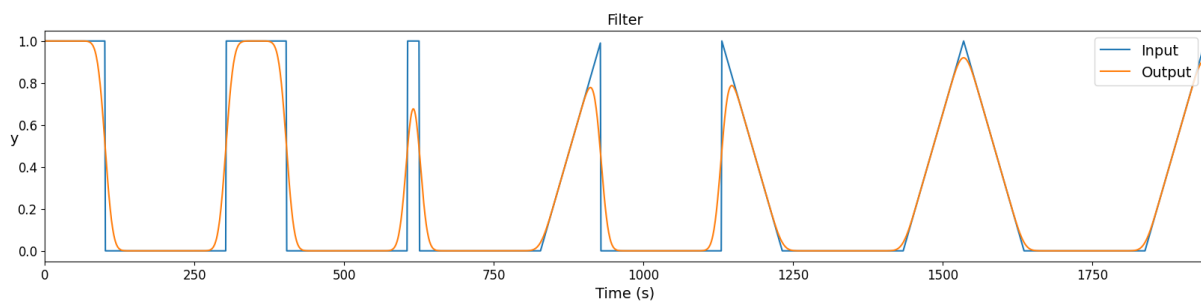


Figure 6.10: Example of the Gaussian filter

To execute the convolution a Gaussian kernel is required. This kernel consists of a NumPy array containing the Gaussian PDF as defined by Formula 6.1 below.

$$\frac{e^{-\frac{1}{2}\left(\frac{x}{s}\right)^2}}{s \cdot \sqrt{2\pi}} \quad (6.1)$$

The Gaussian PDF function is displayed in Figure 6.11 below.

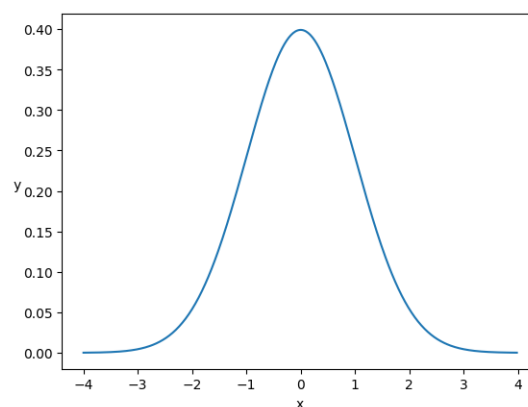


Figure 6.11: Gaussian PDF function

6 Signal Processing Filters

When the standard deviation s becomes larger the maximum amplitude of the PDF decreases. This is because the integral of a PDF function is always equal to 1. This means that for large values of s the values in the kernel become very small. As an example when s is equal to 5 the maximum amplitude has already decreased to 0.08, when s is equal to 100 the maximum is only 0.004. This makes plotting the kernels and debugging difficult. To rectify this the formula is normalized, so the peak of the function is always equal to one regardless of the standard deviation s . This has the additional benefit of simplifying the formula. The modified Formula 6.2 is presented below:

$$e^{-\frac{1}{2}\left(\frac{x}{s}\right)^2} \quad (6.2)$$

One negative consequence of normalizing the kernel is that the convolved signal has to be scaled afterwards to match the amplitude of the original. This is because the integral of the new kernel isn't equal to one anymore. If the kernel was created with the original PDF scaling wouldn't be necessary. See Section 3.2. In practice this isn't an issue because the result of convolutions are always normalized in the codebase.

6.11.1 Mixin usage

The user of the filter specifies the interval over which the kernel is active and the resolution of the signal. A kernel is created automatically based on these parameters. The amount of items in the array is proportional to the supplied resolution and inversely proportional to the resolution. See Section 8.4.2. The figure below shows the kernel that was used to create the convolved signal seen in Figure 6.10.

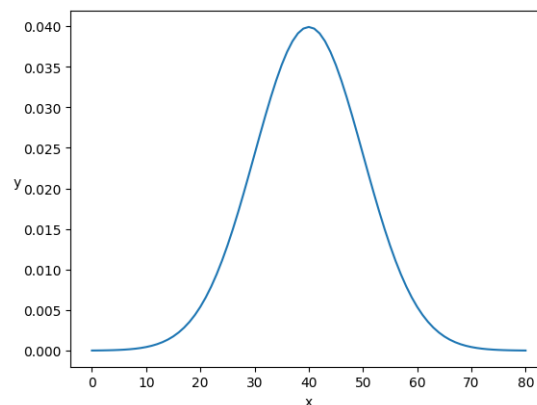


Figure 6.12: Normalized Gaussian kernel used in Figure 6.10

6.12 Derivative of Gaussian for obtaining a robust derivative of the signal

This filter is a convolution with a Derivative of Gaussian (DoG) kernel. This filter derives the input signal and smooths it with a Gaussian blur in one convolution step. The smoothing aspect makes it less susceptible to noise compared to only deriving the signal. [25]

6 Signal Processing Filters

It is worth noting that the acronym "DoG" is also used in image processing context to describe the Difference of Gaussians, this is an approximation of the Laplacian of Gaussian used to speed up computation, see [9]. Unless explicitly mentioned, DoG will always refer to Derivative of Gaussian in this work.

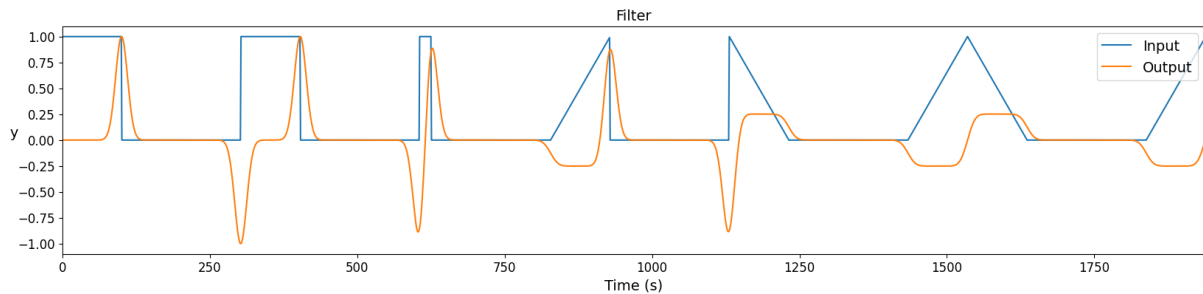


Figure 6.13: Example of the DoG filter.

The filtered signal is similar to a normal derivative, but sudden changes in amplitude don't cause an excessive spike in the convolved signal. The Derivative of Gaussian (DoG) kernel used for the filter is created using Formula 6.3. The formula is obtained by deriving the Gaussian PDF Formula 6.1 used for the Gaussian filter.

$$\frac{-xe^{-\frac{1}{2}\left(\frac{x}{s}\right)^2}}{s^3 \cdot \sqrt{2\pi}} \quad (6.3)$$

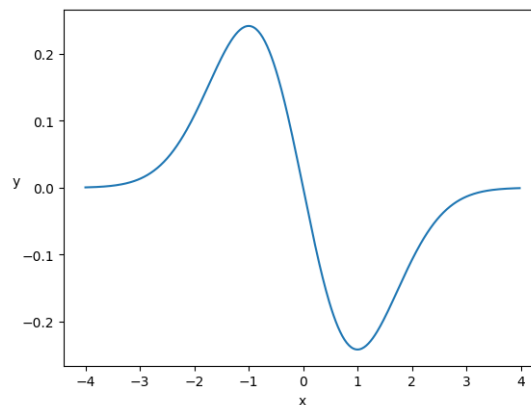


Figure 6.14: DoG formula obtained by deriving the Gaussian PDF

The function is normalized for the reasons explained in section 6.11 so the maximum amplitude is 1 regardless of the standard deviation. This results in Formula 6.4:

$$-\frac{\sqrt{e}}{s} x e^{-\frac{1}{2}\left(\frac{x}{s}\right)^2} \quad (6.4)$$

6 Signal Processing Filters

It is important to note that the integral of the function is still equal to zero. This is important for reasons outlined in Section 3.2. The figure below contains the normalized kernel used for the filtering operation from Figure 6.13:

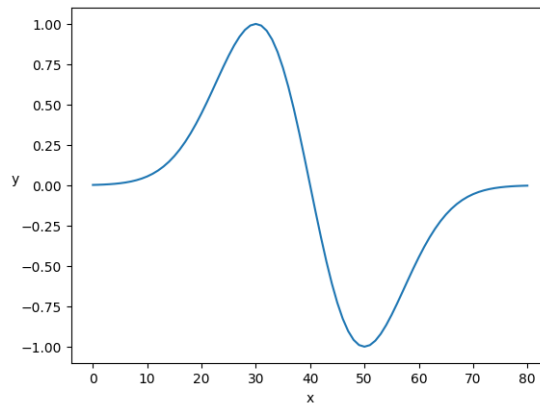


Figure 6.15: Normalized derivative of Gaussian kernel

6.13 Laplacian of Gaussian for detecting peaks in the signal

This filter is a convolution with a Laplacian of Gaussian (LoG) kernel. Filtering with a LoG kernel is commonly done in image processing for blob detection. The most known use case of this is the SIFT computer vision algorithm described in Section 3.6. When applied to a one dimensional signal it is a robust way to detect peaks that is invariant to changes in amplitude.

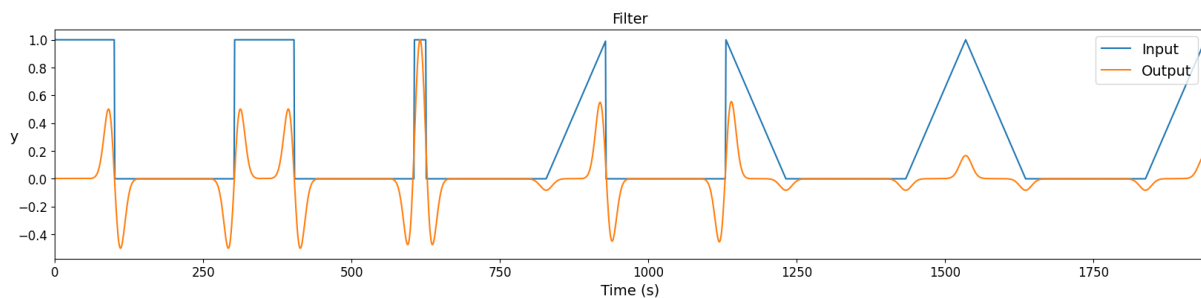


Figure 6.16: Example of the LoG filter

The Laplacian is a mathematical operator that gives the divergence of the gradient of a scalar function in Euclidean space. A scalar function requires one or more input variables and returns one output variable. The gradient of a scalar function in Euclidean space is a vector that represents the rate of change in all directions. It is calculated by deriving the scalar function in all directions. Divergence is a concept that measures the rate at which a vector field "spreads out" or "converges" at a given point. It is calculated by taking the partial derivatives for all variables and adding them together.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

6 Signal Processing Filters

When applied to a univariate function the Laplacian is equal to the second-order derivative of that function:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2}$$

The Laplacian of Gaussian function is the Laplacian operator applied to the Gaussian distribution:

$$\frac{(x^2 - s^2) e^{-\frac{1}{2}(\frac{x}{s})^2}}{s^5 \cdot \sqrt{2\pi}} \quad (6.5)$$

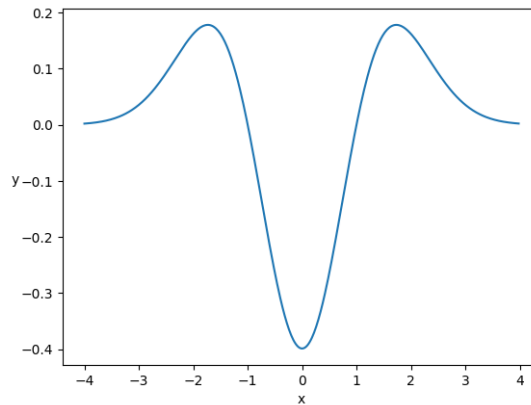


Figure 6.17: Laplacian of the Gaussian formula

The total integral of this new function is again equal to zero which is important for reasons outlined in Section 3.2. If a convolution was performed with a kernel created using Formula 6.5 the output signal would exhibit a spike when there is a dip in the input signal. This is because the second order derivative is high when the first order derivative is changing at the top of the peak. Given that ELMs often correspond with a spiking light emissivity signal, it is desirable for the filter's output signal to exhibit spikes when there are peaks in the input signal. To achieve this, the formula is vertically inverted. This effect is also mentioned in 3.6.2, albeit in a two-dimensional context rather than one-dimensional. Finally, the formula is scaled so that the maximum value of the inverted function becomes one.

$$\frac{(s^2 - x^2) e^{-\frac{1}{2}(\frac{x}{s})^2}}{s^2} \quad (6.6)$$

Figure 6.18 below shows the kernel used for the convolution shown in Figure 6.16

6 Signal Processing Filters

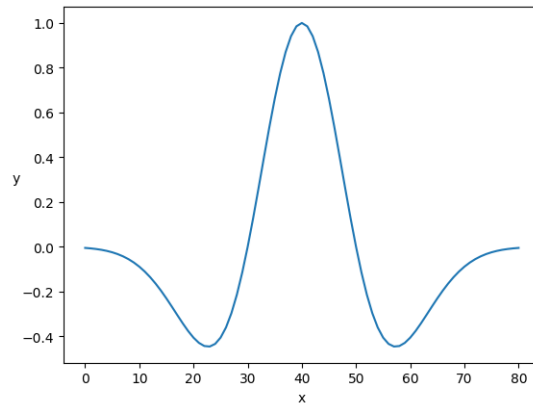


Figure 6.18: LoG kernel used in figure 6.16

7

ELM detection methods

This chapter provides a general overview of the ELM detection methods present in this work, as well as some desirable properties that should be present in ELM detection methods. Each ELM detection method uses a set of hyperparameters of which the optimal values are determined by a gridsearch. For each method, the used hyperparameters along with their optimal value will be illustrated in a table along with their explanation. The introduction of Chapter 4 gives a short explanation of where to find the code related to these methods.

7.1 Method overview

Table 7.1 provides an overview of ELM detection and peak detection methods, which are discussed in Chapters 8, 9 and 10.

Name	Purpose	Principle	Invariance	Execution Time per 100 shots	Input
Simple threshold	Peak	Thresholding	R	10 s	Light
ROT	ELM	Thresholding	A/T/R	10 s	Light
MACD	Peak + ELM	Filter	A/T	10 s	Light + optionally energy
Rolling Z-score	Peak + ELM	Filter	A/T	10 s	Light + optionally energy
LoG	Peak + ELM	Filter	A/T/R	10 s	Light + optionally energy
LoGDoG	ELM	Filter	A/T/R	10 s	Light + energy
CustomKernel	Peak + ELM	Filter	A/T/R	10 s	Light
Deconvolution	Peak + ELM	Deconvolution	A/T	1 min	Light
RNN	ELM	NN	A/T	15 min	Light + energy
CNN	ELM	NN	A/T	15 min	Light + energy

Table 7.1: ELM and Peak Detection methods overview. The different types of invariances are: A - Amplitude invariant, T - Translation invariant, R - Resolution invariant. The execution time is measured by using the method to extract ELMs for each shot in the data set.

The 'purpose' column highlights whether the method is applicable to peaks, ELMs or both. The 'principle' column mentions

7 ELM detection methods

the underlying working principle of the method. The 'execution time' column is meant to give an indication of what the method could be used for. Real-time applications are for example constrained to methods with a lower execution time. Finally, the 'input' signal explains which input signals the method need to work. Some methods use only the emissivity signal, but others also require the energy to work. Both categories are needed because the plasma energy signal is often not available as explained in Section 2.2.

7.2 Peak detection invariances

To ensure robust peak detection methods that are widely applicable inspiration is taken from keypoint detection in image processing. Although SIFT is used for 2D images some of the SIFT invariances are very desirable in peak detection as well. The invariances that are applicable to peak detection are demonstrated using the figure below. Keep in mind that a blob is now a 2D peak and a region in an image is now a time interval in the time series signal.

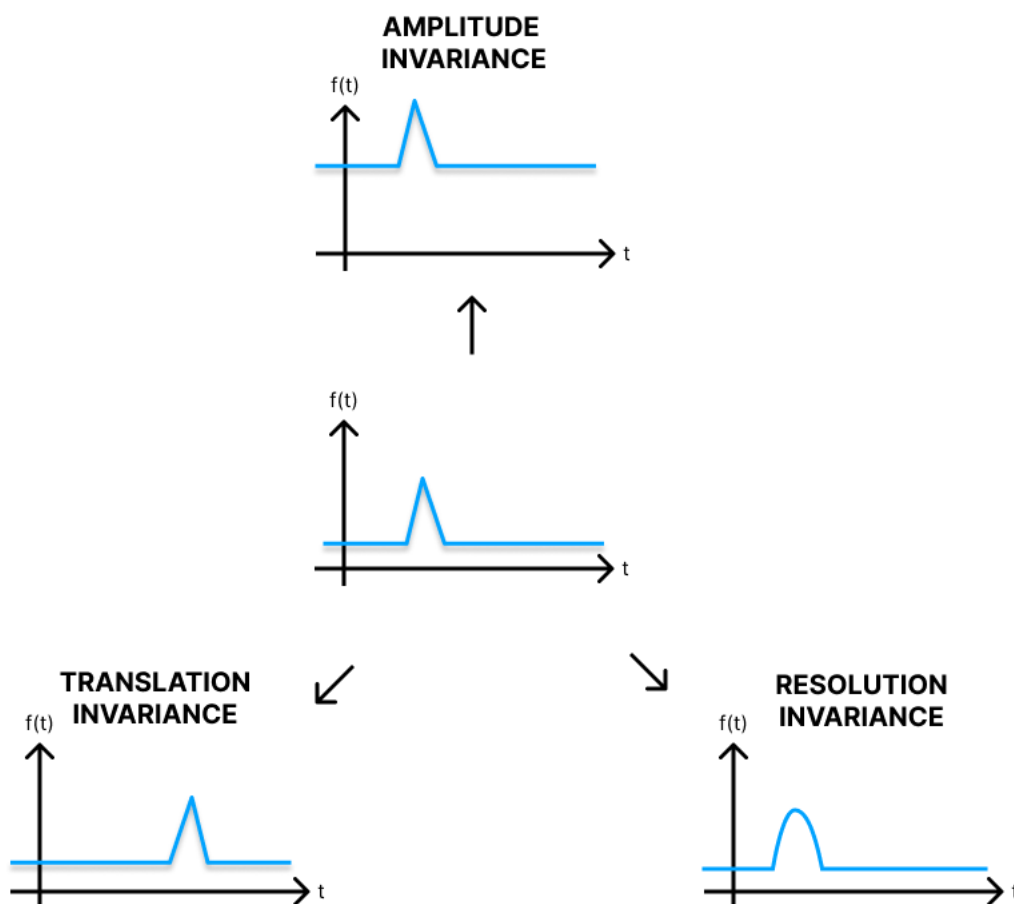


Figure 7.1: Different invariances that are applicable to peak detection in a time series signal

7 ELM detection methods

- Translation invariance: If a peak is found at a certain timestamp in a signal the same peak shape should also be detected if it occurred at a different timestamp in the same signal or with a different baseline. Similarly, if no peak is found in a certain interval no peak will be detected if the same shape occurred later or earlier in the signal. The equivalent in SIFT is that the same keypoint should be detected regardless of the position on the image.
- Amplitude invariance: At any point in time a time series signal has a certain amplitude. The average amplitude of the signal around the peak should have no effect on if the peak is detected or not. Only the amplitude of the peak itself compared to the surrounding signal is relevant. The counterpart in images is illumination: A blob should only be brighter or darker relative to its surroundings. The overall light level is irrelevant.
- Resolution invariance: The sampling resolution of a signal should have no effect on which peaks are detected. In SIFT the size of the blob should not influence how likely it is to be detected.

The other SIFT invariances (rotation invariance and affine transformation invariance) have no meaning in 1D time series signals and are thus not required for peak detection.

Although almost all ELM detection methods discussed in this master's dissertation are strongly translation and amplitude invariant some are not resolution invariant meaning that the sampling rate has to be 10kHz. In resolution invariant methods like LoG 8.4 and LoGDoG peaks can be detected at any resolution as long as the resolution is specified beforehand. Additionally, lower sampling resolutions (higher sampling frequency) will allow more accurate peak start and end times. Finally, if the resolution is increased dramatically peaks might disappear from the data in which case they can not be detected even by a resolution invariant method. However, if an ELM detection method is able to perform at all sampling rates without loss in performance it means the underlying peak detection method is more widely applicable.

8

Filter-based ELM detection methods

8.1 Basic thresholding methods

To establish a baseline performance and facilitate method development, it is valuable to employ some ELM detection methods that are easy to implement and comprehend. These base methods serve as reference points for evaluating the effectiveness of more advanced techniques, while also providing building blocks for the creation of more complex methods described later in this chapter.

8.1.1 Simple threshold

This method uses binary thresholding to find peaks. Everything above a configurable threshold is considered a peak and everything below the threshold is not a peak. The same logic can be applied to ELM detection by looking at the light signal. This method is used within the algorithm of many other methods, often after some data preprocessing such as the Average, LoG, LoGDoG, CustomKernel, RNN and CNN methods.

8.1.2 Average threshold method

To illustrate the Simple threshold method, the average method has been developed. It uses the average of the signal as the threshold in the simple threshold method. This method can be applied to both peak detection and ELM detection. In figure 8.1, the results of this method can be seen when used on a simple case, where the light emissivity is used as the signal. Figure 8.2 show the most obvious flaw with this method. It is clear that marking ELMS everywhere the input goes above the average, results in false predictions. The method is also not capable of differentiating between single and compound ELMS. Therefore, one should not use this method. As a result, it is not included in the overview in Section 7.1.

8 Filter-based ELM detection methods

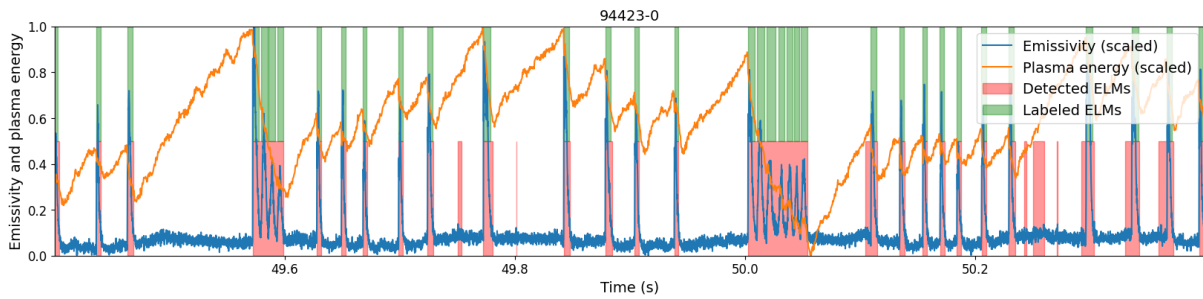


Figure 8.1: Detecting everything above the average as an ELM works for a very basic signal for a simple signal.

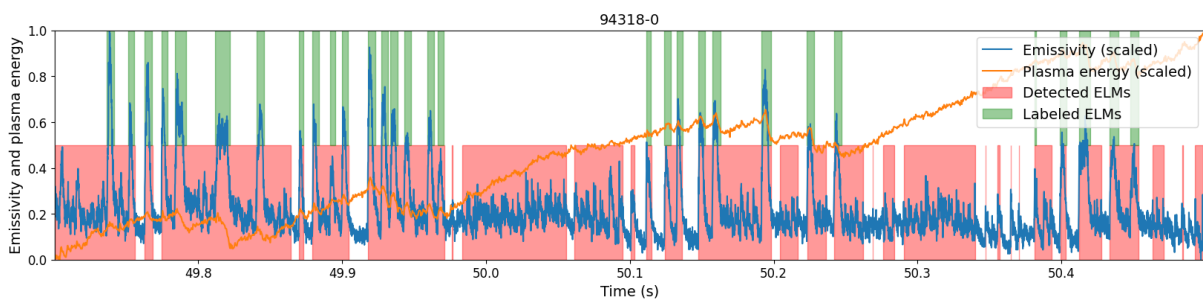


Figure 8.2: Detecting everything above the average as an ELM results in many false positives for a more difficult signal.

8.1.3 Robust optimal threshold method

The robust optimal threshold (ROT) algorithm was developed with the goal of improving the ELM detection performance. In the simplest terms, it is an algorithm that calculates a smart threshold. The method combines statistical analysis and thresholding techniques to robustly detect ELM events.

The threshold is computed using a function that considers the statistical characteristics of the data to determine an appropriate threshold level. This threshold is then used in combination with the find peaks method from SciPy. In figure 8.3, one can notice that the false positives from the Average method are gone. However, the ROT method still fails to properly detect more complex ELMs. One can find in figure 8.4, that an important but difficult set of ELMs was not detected.

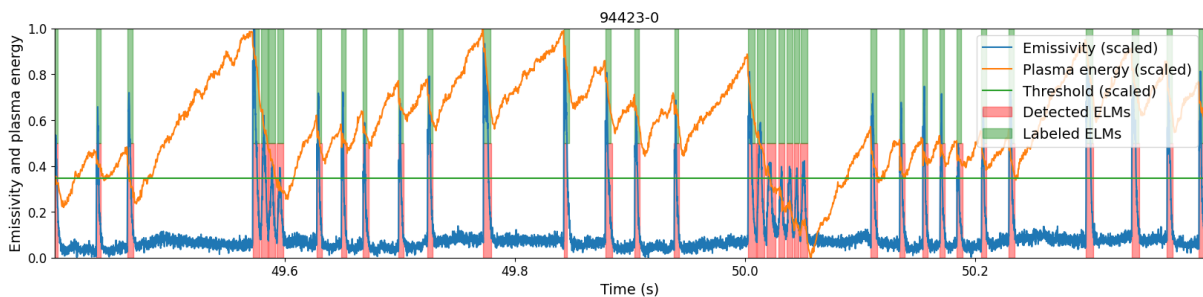


Figure 8.3: Detecting everything above the ROT as an ELM.

8 Filter-based ELM detection methods

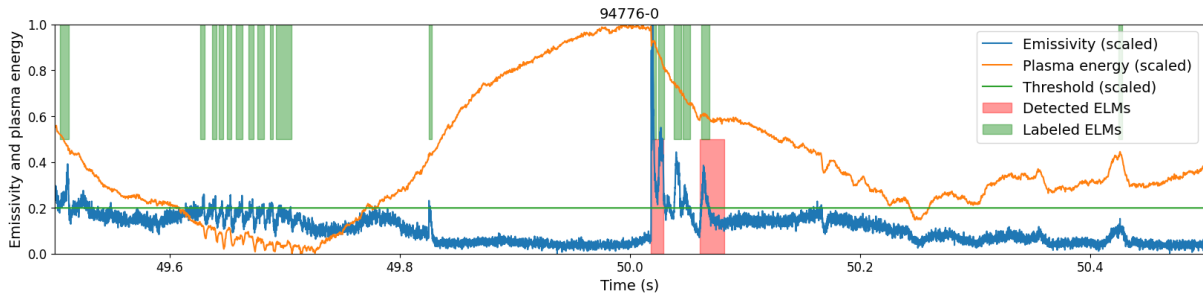


Figure 8.4: Detection is not always working in complex cases.

The ROT method is extended with a padding functionality and a merging functionality, with the latter being on the distance between two ELMs. These are discussed in greater detail in Section 8.2. Table 8.1 shows all the parameters available in the ROT method.

Hyperparameter	Explanation	Change Impact	Optimal Value
quantile	The initial guess for ELM threshold.	.	0.95
window	The number of samples for moving average filter.	Higher values remove lower frequency noise.	3
prominence	The minimum prominence for ELMs.	.	0.7
distance	The minimum distance between ELMs (ms, milliseconds).	.	1.0
fraction	The fraction of signal corresponding to startstop times.	.	0.1
max_compound_gap	Maximum gap allowed between peaks in a compound ELM. Two peaks with a gap less than this parameter are combined.	Larger values will lead to more ELMs being combined into compound ELMs.	0
padding_start	Padding added to the start of each peak.	Larger values will lead to more padding at the front of a peak.	0
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0

Table 8.1: Hyperparameters and their descriptions for the ROT ELM detection method.

8.2 MACD method

This method is mainly implemented as a peak detection algorithm and operates by analyzing the MACD (see 6.9) of the emissivity signal. The output of the peak detection algorithm is then passed through some other improvement steps. As a

8 Filter-based ELM detection methods

result, the implementation is quite complex and uses many configurable hyperparameters. These are listed in Table 8.2.

Hyperparameter	Explanation	Change Impact	Optimal Value
<code>slow</code>	The greater span width used for calculating the MACD.	Higher values remove lower frequency noise.	12
<code>fast</code>	The smaller span width used for calculating the MACD. Has to be less than <code>slow</code> .	Higher values remove lower frequency noise.	9
<code>smooth</code>	The span width used for calculating the signal line if used for the end detection.	Higher values remove lower frequency noise for the end detection.	6
<code>threshold_start</code>	Greater values than this will signify the start of a peak.	A smaller threshold triggers more peaks.	0.5
<code>threshold_end</code>	Greater values than this will signify the end of a peak.	A smaller threshold triggers more peak ends. A negative value that is too low, might prevent the end of a peak from being detected.	-0.1
<code>threshold_extender</code>	Lower values than this may cause peak end extension.	The lower, the less effect noise has but makes it less likely that the peak will be extended.	-1
<code>max_extender_gap</code>	The max amount after the current end of a peak that the extension may happen.	The greater value, the greater the gap allowed to bridge for one peak.	0.01
<code>max_compound_gap</code>	Maximum gap allowed between peaks in a compound ELM. Two peaks with a gap less than this parameter are combined.	Larger values will lead to more ELMs being combined into compound ELMs.	0
<code>padding_start</code>	Padding added to the start of each peak.	Larger values will lead to more padding at the front of a peak.	0.001
<code>padding_end</code>	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	-0.002
<code>merge_using_savgol</code>	Whether or not to merge ELMs into a compound ELM based on the energy signal.	If true, merges peaks when the energy goes down between the time that both peaks occur.	False
<code>drop_using_savgol</code>	Whether or not to remove detected ELMs from the output based on the energy signal.	If true, Removes a peak if the energy is going up.	True
<code>savgol_window</code>	The window used for the Savitzky-Golay filter.	Higher values merge and remove more ELMs from the detection output.	30
<code>end_detection</code>	The filter to use for the end detection. Either <code>macd</code> or <code>signal_line</code> .	When the end detection uses <code>macd</code> , more noise is present.	<code>macd</code>

Table 8.2: Hyperparameters and their descriptions for the MACD ELM and peak detection method.

8 Filter-based ELM detection methods

8.2.1 MACD peak detection

The process involves identifying peaks by monitoring the MACD values. When the MACD crosses above a predefined threshold, it signifies the start of a peak. Subsequently, the algorithm continues scanning the time series, awaiting a dip below a secondary, lower threshold. Once the MACD crosses this second threshold again for a second time, it indicates the end of the peak.

To illustrate this approach, consider Figure 6.9. In this example, a first threshold of 0.25 can be employed, and a second threshold of -0.25 can prompt the algorithm to set the end marker immediately after the peak gradually declines back to the baseline. Note, that the second threshold has to be passed twice after the start in order to trigger an end.

It's important to note that in certain cases, an alternative approach using the signal line (EMA of the MACD) for end detection might yield better results. When integrating this method into a project, it is advisable to experiment with both options, as the outcomes may vary depending on the specific characteristics of the data set being analyzed. Keep in mind that most noise will have been removed by the MACD but the signal line removes extra noise from the MACD, which might be desirable.

Figure 8.5 shows the same shot from before, but with MACD as the detection method.

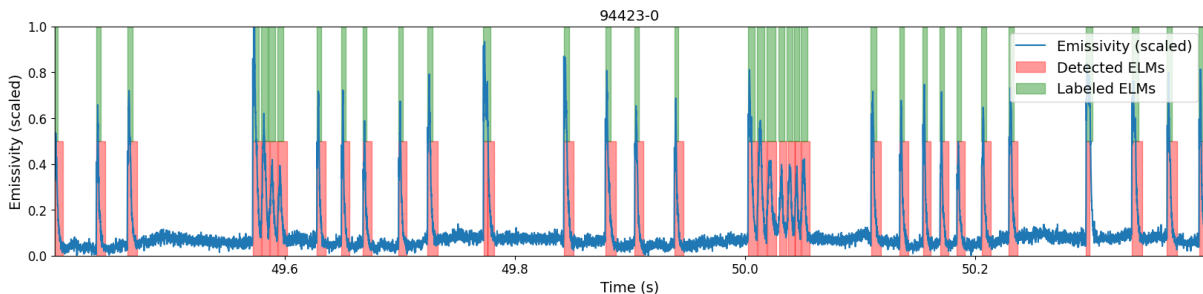


Figure 8.5: Basic peak detection using MACD on the emissivity signal.

8.2.2 MACD peak detection extension

A certain characteristic that ELMs have, is that their light emissivity rises fast, and drops slowly. Some ELMs also tend to have their emissivity stay high for a bit before the gradual decrease starts. Other ELMs are a variant of this, where the light signal decreases gradually, then stays semi-high and then decreases again. This can result in problems for the MACD method, as can be seen in Figure 8.6.

8 Filter-based ELM detection methods

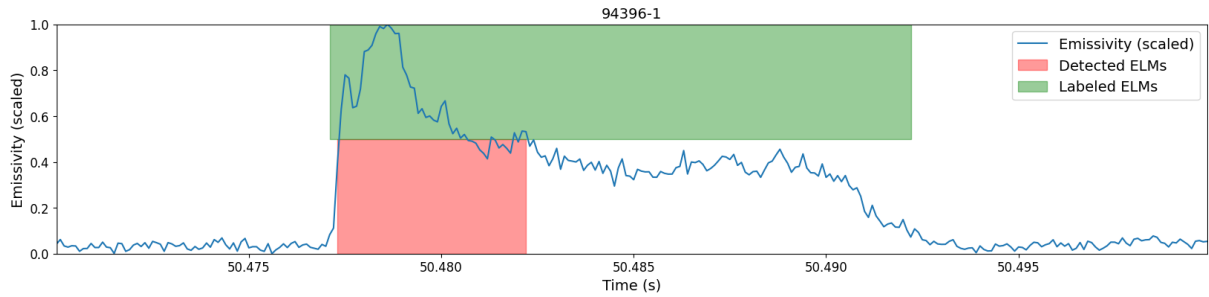


Figure 8.6: Detection without MACD extension.

The flaw in the detection algorithm becomes clear with Figure 8.7. When the input becomes flat before the ELM/peak has ended, the MACD and signal line both move back to zero. However, from the moment the input starts to drop again, the both filters drop below zero. By detecting this event, the signal can be extended. Two new parameters are therefore introduced. A threshold that prevents simple noise from triggering the extension algorithm. This means that the filter has to drop below the threshold for the extension to trigger. Secondly, the extension may only happen within a certain span after the end of the ELM/peak. This span or gap that may be bridged, is the second parameter for the extension algorithm. Figure 8.8 show that the issue is resolved with this extra algorithm.

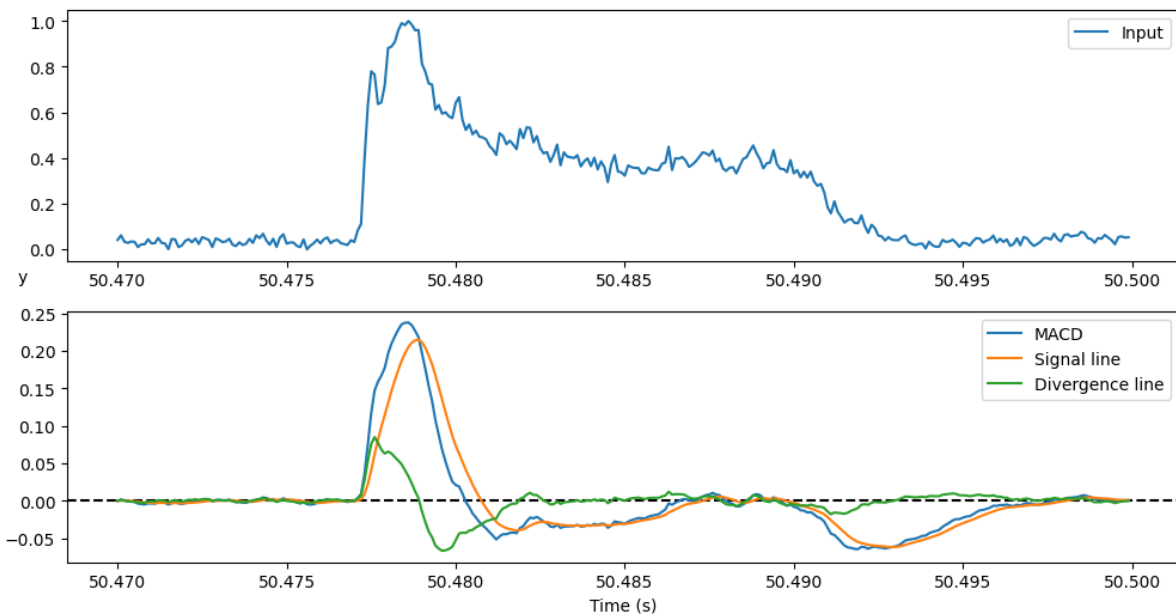


Figure 8.7: Cause of the peaks detected with the MACD method ending too early.

8 Filter-based ELM detection methods

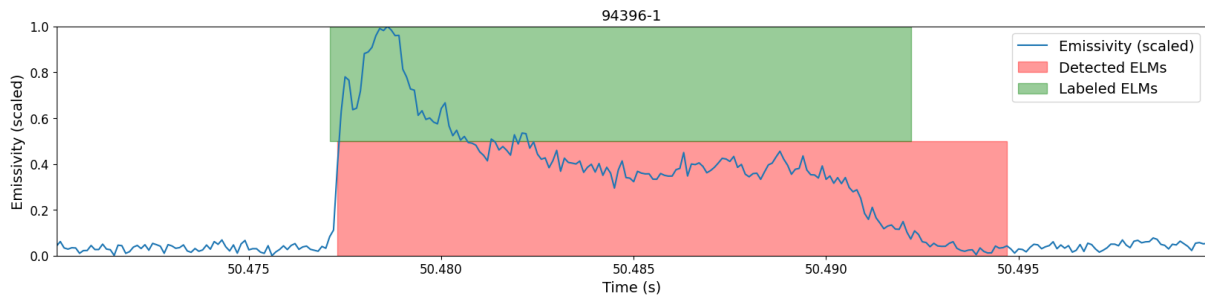


Figure 8.8: Detection with MACD extension.

8.2.3 MACD ELM detection

A few operations are still needed for converting the detected peaks to ELMs.

Padding post-processor

One may notice that the detected peaks in Figure 8.8 do not align with the labels as they should. This can be fixed by adding padding around each peak. This padding can even be made negative if needed. The result can be seen in Figure 8.9.

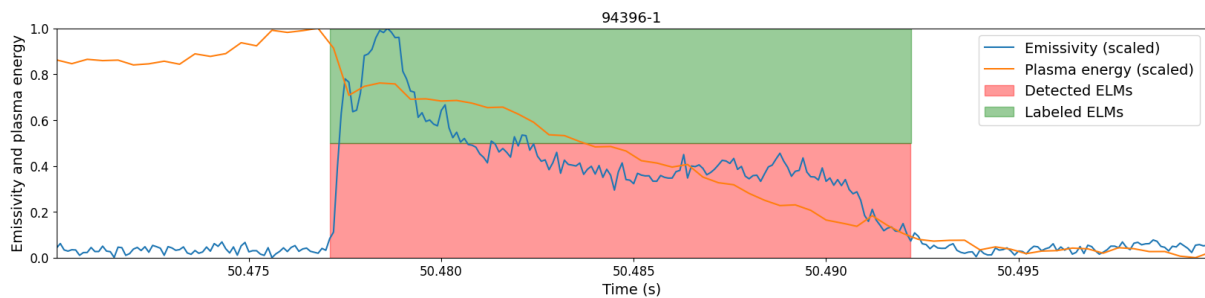


Figure 8.9: Adding padding to MACD ELMs.

Savgol Gap post-processor

Another issue that arises is that ELMs sometimes should be merged into compound ELMs. Section 2.6 explains that compound ELMs can be seen as multiple ELMs within a continuous drop of energy. Section 2.6 also explains that ELMs do not occur when the energy signal is rising. So two actions should be implemented. One that merges ELMs when the energy signal is continuously dropping and one that removes ELMs from the output when the energy signal is effectively increasing.

For this, a new compound detection algorithm has been designed called Savgol Gap. It applies the Savitzky-Golay filter from 6.7 to the negated energy signal. The Savitzky-Golay filter was chosen above the difference filter to remove noise while also getting an approximation of the first derivative of the signal. This output is then passed through the Simple threshold method with threshold zero, which outputs every region where the energy is dropping as a peak interval.

8 Filter-based ELM detection methods

Now merging ELMs into compound ELMs when they fall under the same falling energy interval and removing ELMs when they are not within a falling energy interval, results in fewer individual peaks and fewer false positives. Applying Savgol Gap to Figure 8.10 results in Figure 8.11, which now contains detected compound ELMs and less false positives.

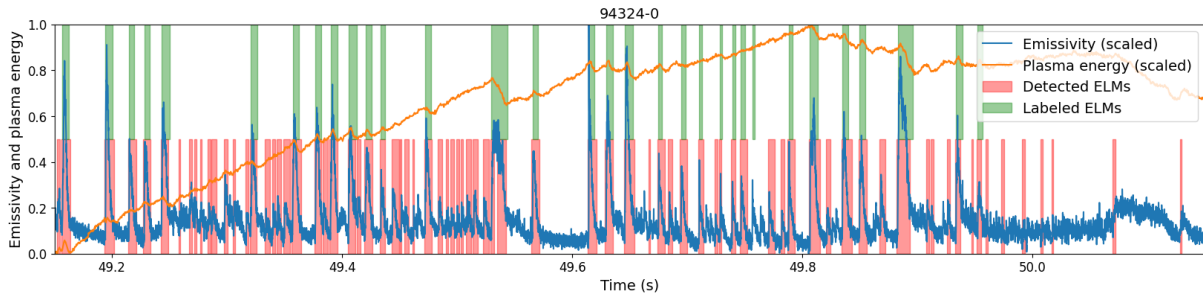


Figure 8.10: A complex signal with many false positives from the MACD method.

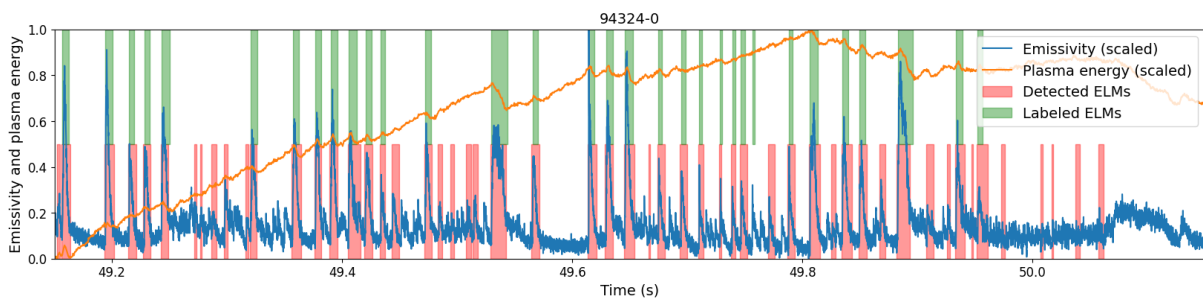


Figure 8.11: Merging and dropping ELMs using Savgol Gap. Not all problems are solved with Savgol.

Note however that Section 2.6 implicitly mentions that if the emissivity signal drops back to the baseline, two ELMs should not be merged, even though the energy is continuously dropping. This extra rule was only given late in development, so it was not implemented in the Savgol Gap merging algorithm. As a side effect, single ELMs are regularly merged into compound ELMs, when they shouldn't. Therefore, the merging mechanism should only be activated when absolutely necessary. Future work could incorporate the extra rule to make the merging more robust.

Figure 8.12 disables the merging algorithm. Compared to 8.11, one can notice that the compound ELM between 49.5 and 49.6 seconds is not merged. This shows that even though enabling the merging option in Savgol Gap is generally not recommended, it does hold an advantage in some cases. Figure 8.12 also shows that disabling the merging option in Savgol Gap reduces false positives, as seen at the end of the shot. This comparison between these two figures shows that careful consideration is needed when enabling the merging option.

8 Filter-based ELM detection methods

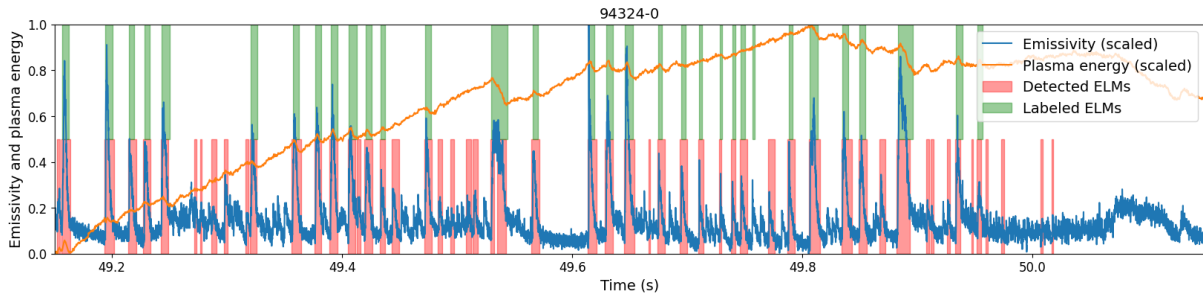


Figure 8.12: Dropping ELMs using Savgol Gap. Not all problems are solved with Savgol Gap. Merging with Savgol Gap is disabled.

Static Gap post-processor

Another compound detection algorithm was developed to merge individual ELMs into compound ELMs when required. This algorithm is rather simple and merges two ELMs when they are closer than a given distance from each other. This works rather well but does not follow the energy rules that are characteristic of ELMs. By default, the algorithm only merges the ELMs when the first ELM its end time is equal to the start time of the second ELM.

Figure 8.13 shows the same figure as Figure 8.5 before, but now with all the small correction algorithms. For a basic shot like this, these corrections do not have a great effect. They do yield improvements for difficult shots, as proven by the results in Chapter 11.

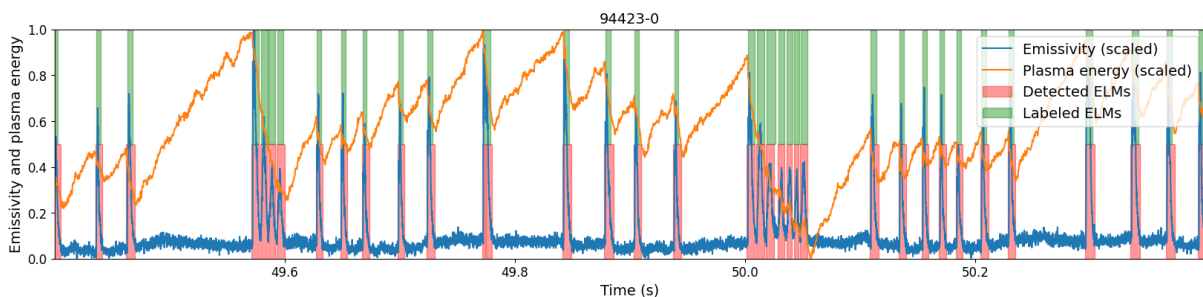


Figure 8.13: ELM detection using the MACD method.

8.3 Rolling Z-score method

The Rolling Z-score algorithm is rather simple with only a few hyperparameters that can be tuned. These are listed below in Table 8.3

8 Filter-based ELM detection methods

Hyperparameter	Explanation	Change Impact	Optimal Value
window	The greater span width used for calculating the rolling Z-score. Max length if set to -1.	Lower spans remove less noise.	-1
threshold_start	Greater values than this will signify the start of a peak.	A smaller threshold triggers more peaks.	0.5
threshold_end	Smaller values than this will signify the end of a peak.	A smaller threshold triggers more peak ends. A negative threshold that is too low, might prevent the end of a peak from being detected.	0
max_compound_gap	Maximum gap allowed between peaks in a compound ELM. Two peaks with a gap less than this parameter are combined.	Larger values will lead to more ELMs being combined into compound ELMs.	0
padding_start	Padding added to the start of each peak.	Larger values will lead to more padding at the front of a peak.	0.001
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0
merge_using_savgol	Whether or not to merge ELMs into a compound ELM based on the energy signal.	If true, merges peaks when the energy goes down between the time that both peaks occur.	False
drop_using_savgol	Whether or not to remove detected ELMs from the output based on the energy signal.	If true, Removes a peak if the energy is going up.	True
savgol_window	The window used for the Savitzky-Golay filter.	Higher value merge and remove more ELMs from the detection output	30

Table 8.3: Hyperparameters and their descriptions for the Rolling Z-score ELM and peak detection method.

8.3.1 Rolling Z-score peak detection extension

The rolling Z-score method is another peak detection algorithm developed for the master's dissertation. It differs from the MACD-based approach by utilizing the rolling Z-score of the light emissivity signal for analysis. The method follows a similar pattern of peak identification by monitoring the values of the rolling Z-score. When the rolling Z-score surpasses a predefined threshold, it signifies the start of a peak. As the algorithm progresses through the time series, it awaits a subsequent drop below a secondary, lower threshold, which serves as the indicator for the peak's end.

There are two key distinctions between the rolling Z-score method and the MACD method. Firstly, instead of examining the

8 Filter-based ELM detection methods

MACD, the rolling Z-score is utilized as the basis for peak detection. Secondly, the ending point of the peak is determined by the first instance the rolling Z-score crosses the second threshold, in contrast to the MACD method where the second threshold crossing is only used as the endpoint when it crosses that threshold for the second time. These differences contribute to the unique characteristics and behavior of the rolling Z-score method compared to the MACD approach.

8.3.2 Rolling Z-score ELM detection

To treat the Rolling Z-score peak detection as ELM detection, three extra steps were added. Firstly, the padding, as discussed in section 8.2.3 and secondly, the Savgol Gap from section 8.2.3 and finally the Static Gap from section 8.2.3. Figure 8.14 shows the detected ELMs on a simple shot.

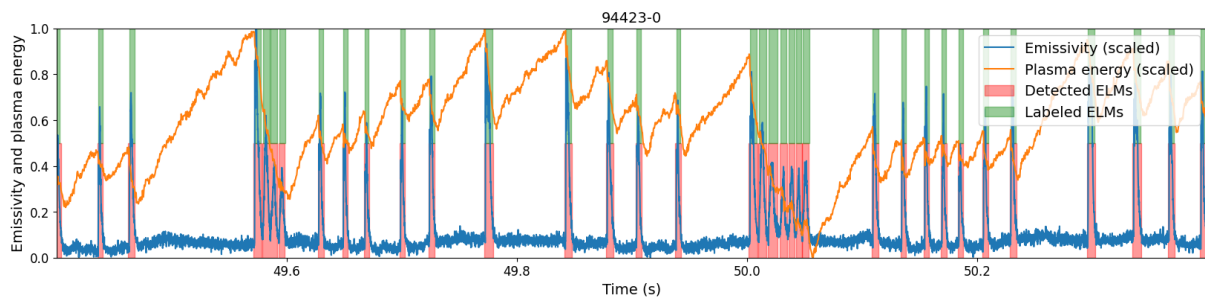


Figure 8.14: Rolling Z-score ELM detection.

8.4 LoG method

This method only looks at the emissivity signal, and uses the LoG filter to obtain a smoothed second order derivative of the signal. The method is completely translation, amplitude and resolution invariant.

Hyperparameter	Explanation	Change Impact	Optimal Value
interval_Log	The interval over which the LoG kernel is active	Longer intervals detect longer peaks and vice versa	0.012s
threshold	Value for SimpleThreshold step of peak detection	Larger values decrease the sensitivity.	0.06
padding_start	Padding added to the start of each peak	Larger values will lead to more padding at the front.	0.001s
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.001s
max_gap	Maximum gap allowed between peaks in a compound ELM	Larger values will lead to more compound ELMs	0s

Table 8.4: Hyperparameters and their descriptions for the LoG ELM and peak detection method.

8 Filter-based ELM detection methods

Even though the method can not run without knowing the resolution of the signal it is not included in the table of hyperparameters. This because the method is resolution invariant meaning that it should perform the same at any resolution. This peak detection method is based on the LoG filter discussed in Section 6.13 that convolves an input signal with a Laplacian of Gaussian kernel.

SIFT is completely invariant to changes in translation and illumination because of the convolution step. Because this method also utilizes convolution the equivalent invariances in peak detection: translation and amplitude invariance are also achieved. The last invariance needed for peak detection is resolution invariance. SIFT accomplishes scale invariance by searching for the same blobs at different sizes. It does by filtering the image multiple times with LoG kernels of different lengths. For elm detection searching for peaks at multiple scales is not required because a single ELM has roughly the same duration each time. Compound ELMs can be detected by grouping multiple singular ELMs together afterwards.

8.4.1 LoG resolution invariance problem

Still the kernel needs to be adjusted to match the sampling frequency of the signal. To illustrate the need for this it is useful to examine the negative effects if the same kernel is used for different resolutions. Propose that the training has concluded that the best performance is achieved when the LoG kernel is applied over an interval of 31ms ($\text{interval_LoG} = 0.031\text{s}$). In the figure below a time series signal is displayed and then subsampled with a factor of 2 and 4. The same LoG kernel that spans 31ms at a resolution of 1ms with 31 elements is placed over each variation of the signal:

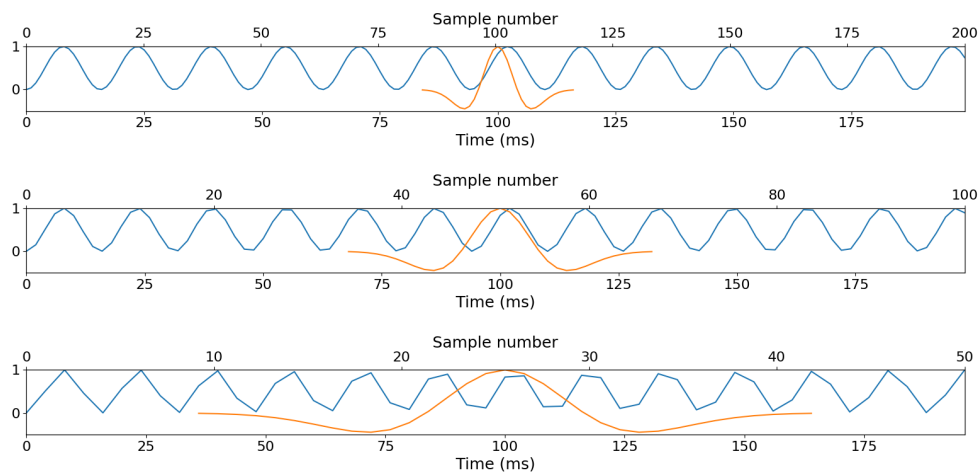


Figure 8.15: Time series signal subsampled by a factor of 1, 2 and 4 with a LoG kernel ($\text{interval_LoG} = 31\text{ms}$, $\text{resolution} = 1\text{ms}$)

At higher resolutions the same kernel covers a bigger interval because the time between each measurement is no longer 1ms. This means that peaks of the desired duration can not be detected anymore. By scaling the kernel proportionally to the resolution this negative effect can be completely mitigated.

8 Filter-based ELM detection methods

8.4.2 LoG resolution invariance solution

Using the following formula it is possible to determine how many items are needed in a kernel for a given resolution and interval so the kernel spans the correct amount of time when superimposed on a time series signal.

$$size = \lceil \frac{interval}{resolution} \rceil_{\text{odd}} \quad (8.1)$$

Subsequently, the right standard deviation is determined for this kernel size. This principle is illustrated in the table below.

Sampling resolution (ms)	Elements	Standard deviation	Interval covered (ms)
1	31	4	31
2	17	2	34
4	9	1	36

Table 8.5: Properties of different kernels created to cover the same 31ms at different sampling resolutions.

In the figure below the kernel now spans roughly the same time interval at each sampling rate so peaks of the correct length are targeted.

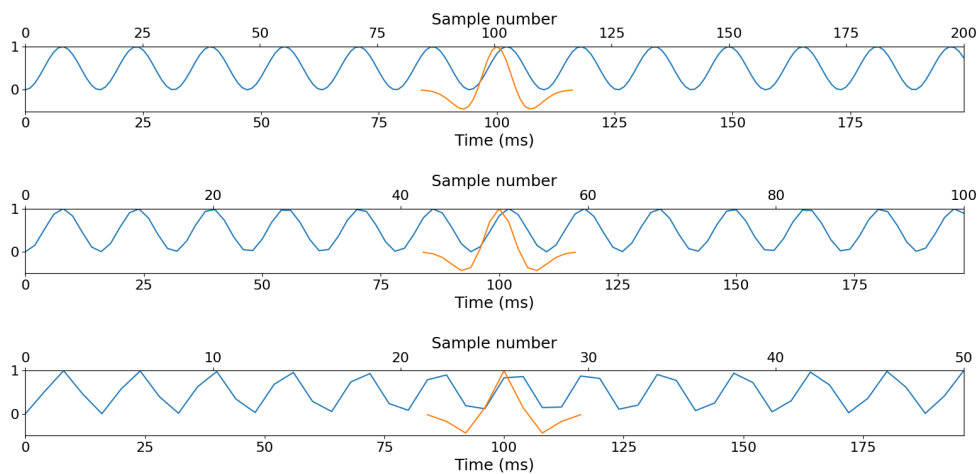


Figure 8.16: Time series signal subsampled by factor 1, 2 and 4.

In practice, the interval in seconds over which the kernel is active is determined by the hyperparameter 'interval_Log' of which the optimal value was found with a gridsearch during training at a resolution of 0.0001 seconds. When the ELM method is executed the user needs to supply the sampling resolution of the ELM light signal. A new kernel of the correct length and with the correct standard deviation is automatically created.

With amplitude, translation, and resolution invariance covered the steps of the LoG peak and elm detection method itself can be explained.

8 Filter-based ELM detection methods

8.4.3 LoG peak detection

To facilitate the explanation shot number 94423-0 is used. Only the emissivity signal is plotted because this method doesn't use the energy signal.

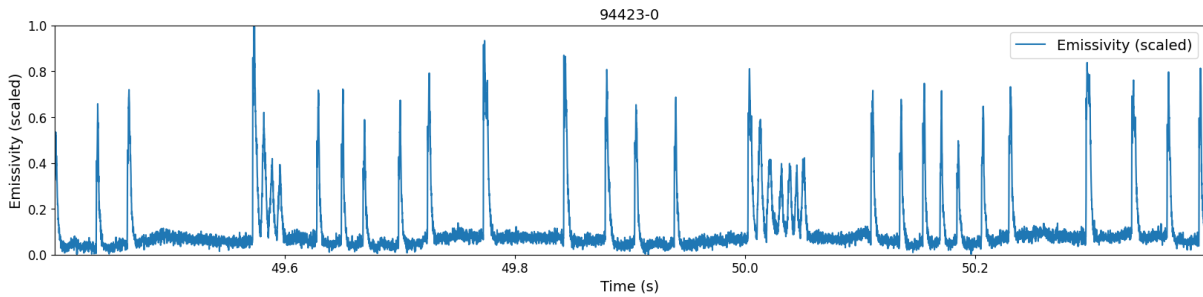


Figure 8.17: Light signal of shot 94423-0

The first step is convolving the signal with a LoG kernel. The best performing interval_{LoG} was determined using a grid search.

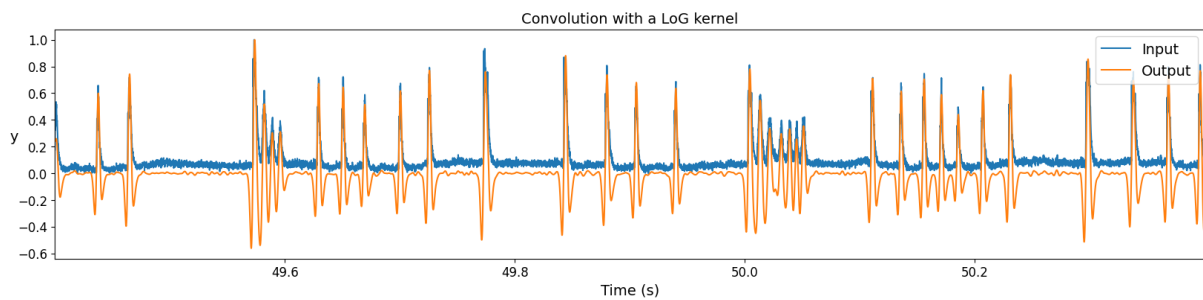


Figure 8.18: Convolution step of the LoG peak detection method

Subsequently, peaks are extracted out of the signal using the SimpleThreshold method, discussed in Section 8.1.1. The threshold used is determined by the 'threshold' hyperparameter.

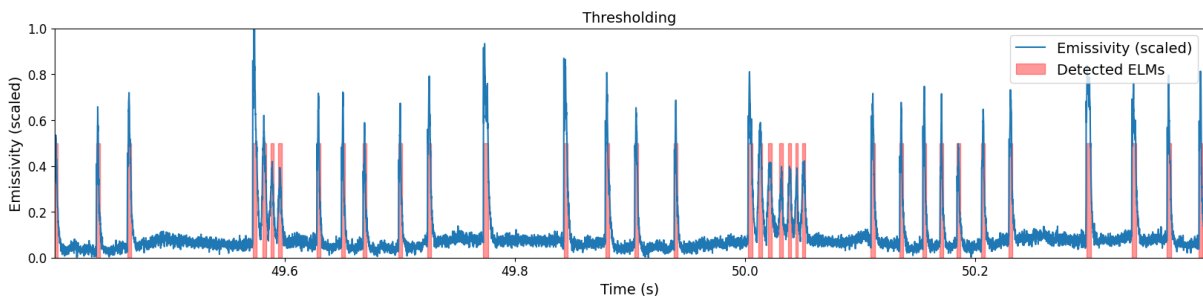


Figure 8.19: Thresholding step of the LoG peak detection method

This concludes the peak detection portion of the LoG method.

8 Filter-based ELM detection methods

8.4.4 LoG ELM detection

The LoG method characteristically only returns the spikes itself. As mentioned in Section 2.6, ELMs have a short buildup and a gradual decline in the emissivity signal which also needs to be included. The detected peaks are padded on the front and tail end by hyperparameters 'padding_start' and 'padding_end'.

In this example, 1 ms is added at the front and 3 at the back of each detected ELM.

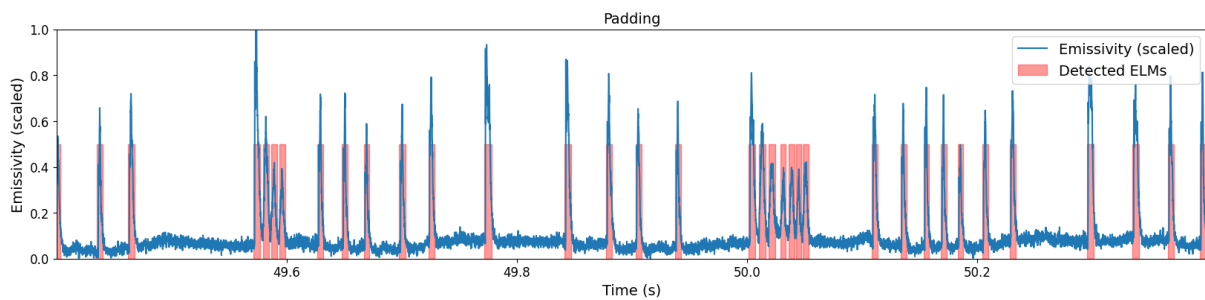


Figure 8.20: Padding step of the LoG ELM detection method

After padding, it is possible that peaks overlap. In the final step ELMs that are closer to each other than a static time interval are combined using the StaticGap method from Section 8.2.3 with hyperparameter `max_gap`. These are most likely compound elms.

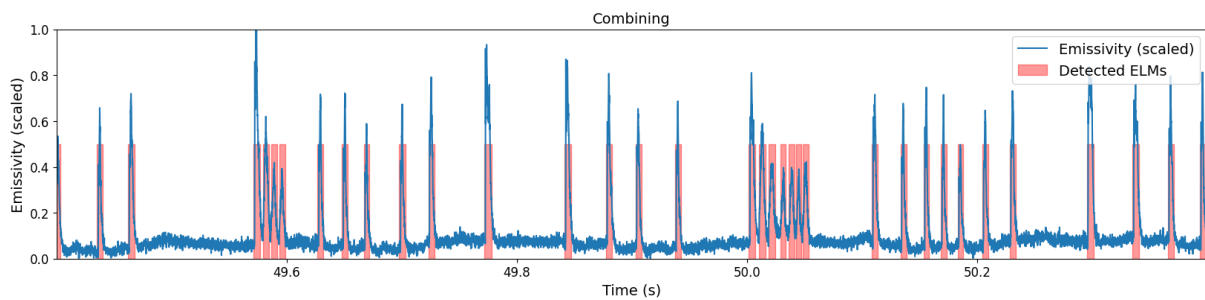


Figure 8.21: Combining step of the LoG ELM detection method

Finally, in Figure 8.22 the result is displayed alongside the ground truth labels.

8 Filter-based ELM detection methods

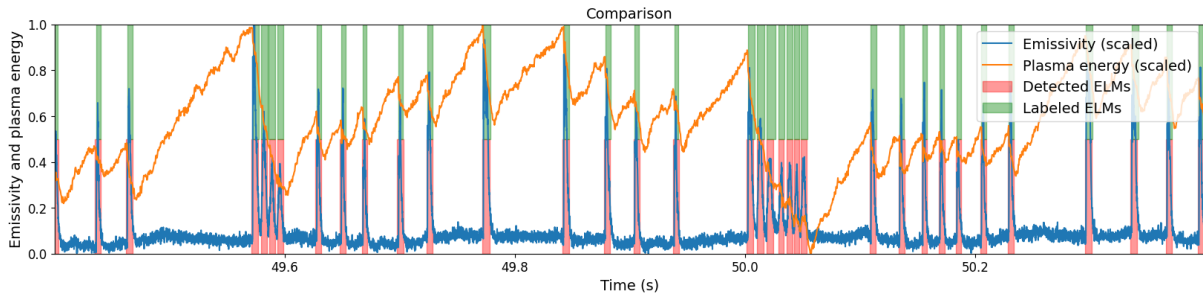


Figure 8.22: Result of the LoG method

8.4.5 Conclusion

Chapter 11 shows that the LoG method is comparatively one of the better methods in this master's dissertation, but it is constrained by the fact that it only takes the light signal into account. As mentioned in section 2.6, there can be no ELM if the energy signal is rising. This ignorance of the energy signal causes the LoG method to be prone to false positives.

8.5 LoGDoG method

This ELM detection method builds upon the LoG method by attempting to decrease the false positive rate. It is also invariant to translation, resolution and amplitude.

As mentioned in section 2.6, every ELM is accompanied by a drop in plasma energy, however small it may be. This method starts by convolving the light signal with a LoG kernel similarly to the LoG method. Next the method attenuates the filtered signal based on the derivative of the energy signal. The aim is to remove spikes in the LoG signal that occur when the energy signal is constant or rising. This also allows the method to be more sensitive to peaks in the light signal since false positives are less of a concern.

This method serves some resemblance to the Savgol drop step described in section 8.2.3 which also attempts to remove false positives by looking at the energy signal but this method is completely resolution invariant. This method can not work without a plasma energy signal.

8 Filter-based ELM detection methods

Hyperparameter	Explanation	Change Impact	Optimal Value
interval_Log	The interval over which the LoG kernel is active	Longer intervals detect longer peaks and vice versa	0.012s
interval_DoG	The interval over which the DoG kernel is active	Longer intervals take into account more time around the current peak to determine if a peak in the light signal is an ELM	0.0015s
gain	Multiplier for the tanh activation function	Larger values are less likely to disqualify ELMs because the drop in energy is not big enough.	34
threshold	Value for SimpleThreshold step of peak detection	Larger values decrease the sensitivity.	0.05
padding_start	Padding added to the start of each peak	Larger values will lead to more padding at the front.	0.0009s
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.0009s
max_gap	Maximum gap allowed between peaks in a compound ELM	Larger values will lead to more compound ELMs	0.001s

Table 8.6: Hyperparameters and their descriptions for the LoG ELM and peak detection method.

8.5.1 LoGDoG ELM detection

To explain the LoGDoG method shot 94776-0 is used. This shot is chosen because it has a very noticeable peak in the light signal at around 49.8 seconds that is not an ELM but looks like it at first sight. The peak in question can not be an ELM because the plasma energy signal is rising. The energy and light are visible in the figure below:

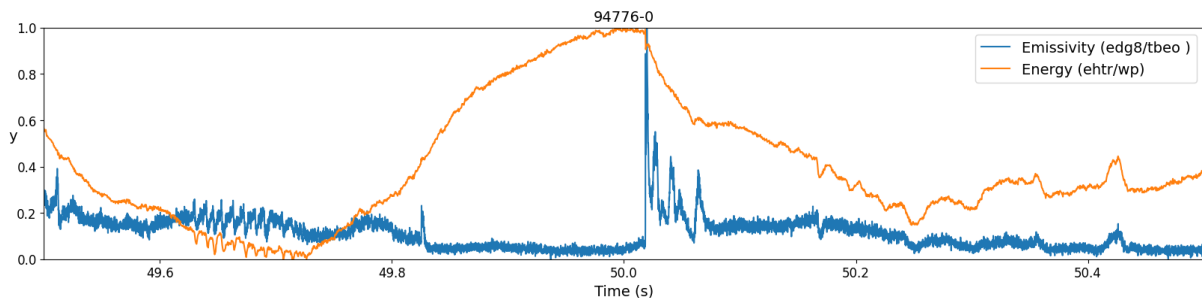


Figure 8.23: Light and energy signal of shot 94776-0

The initial convolution of the energy step is identical to LoG ELM detection. The light signal is convolved with a LoG kernel that acts on an interval of length 'interval_Log' seconds. The result of this convolution is visible in the figure below.

8 Filter-based ELM detection methods

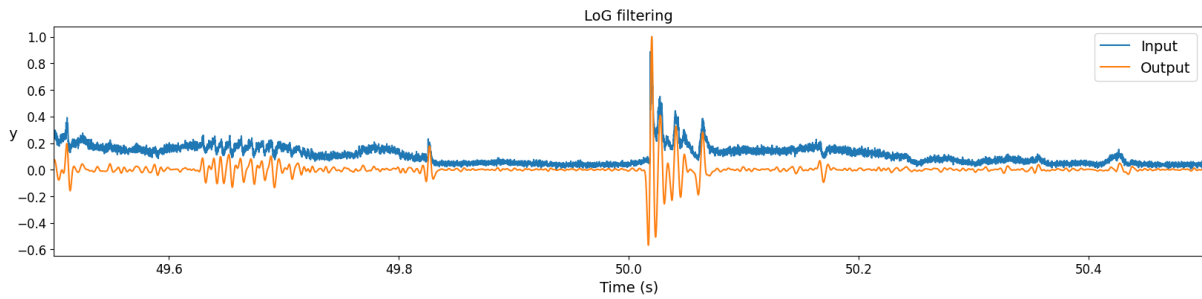


Figure 8.24: Light signal of shot 94776-0 in blue and a version that is filtered with a LoG kernel

The next step is convolving the energy signal with a DoG kernel. The interval over which this kernel is active is determined by parameter. 'interval_DoG'. The result is shown in the figure below. When the plasma energy is drops the convolved signal rises.

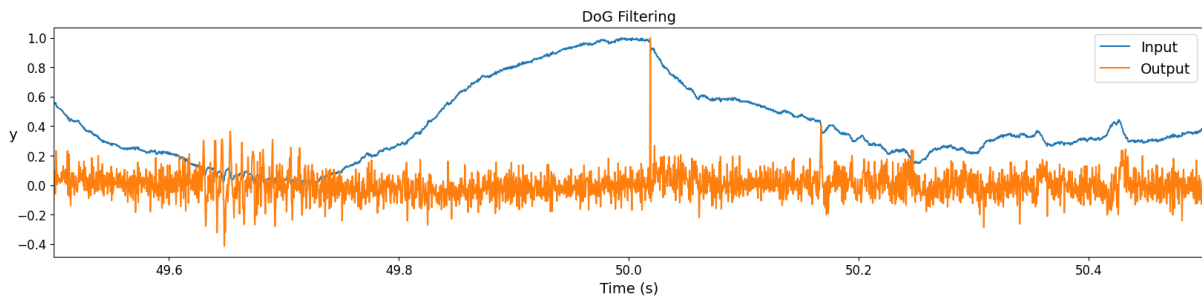


Figure 8.25: Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel

Next the signal obtained by the DoG convolution is processed. First the signal is multiplied by a hyperparameter called 'gain'. Then the signal is then given to a hyperbolic tangent function that saturates the values between -1 and +1. Finally, the values below 0 are set to 0. The resulting signal shown in the figure below acts as a 'gate' with values between 0 and 1.

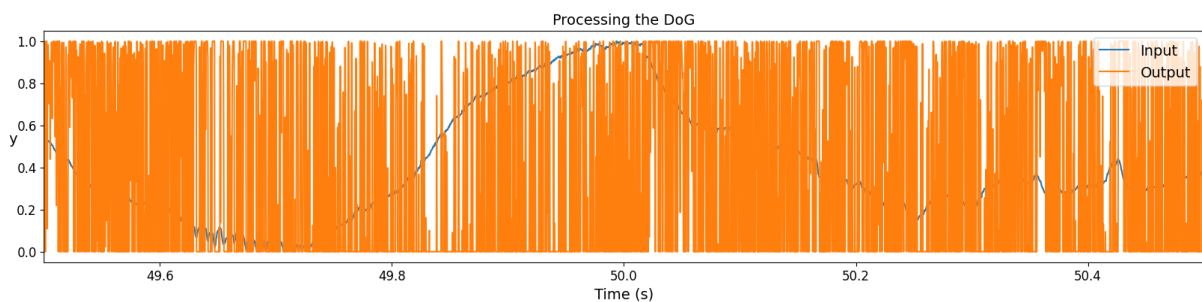


Figure 8.26: Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel

In this step the light signal convolved with a LoG kernel show in Figure 8.24 is multiplied by the processed energy from Figure 8.26 above. The resulting final signal is shown below and has no peaks anymore in regiois where the energy signal is rising. The peak at 49.8 seconds is now completely gone.

8 Filter-based ELM detection methods

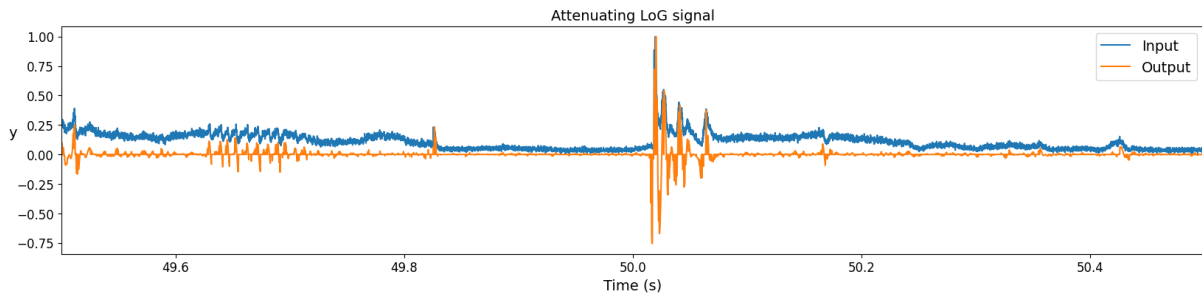


Figure 8.27: Energy signal of shot 94776-0 in blue and a version that is filtered with a DoG kernel

The final three steps are equivalent to the ELM detection portion of the LoG method. First the signal is thresholded with `SimpleThreshold` to extract the peaks, the peaks are padded on the left and right, and the peaks are combined using the `StaticCombiner`. The hyperparameters are explained in Table 8.6.

The final result is shown in the figure below:

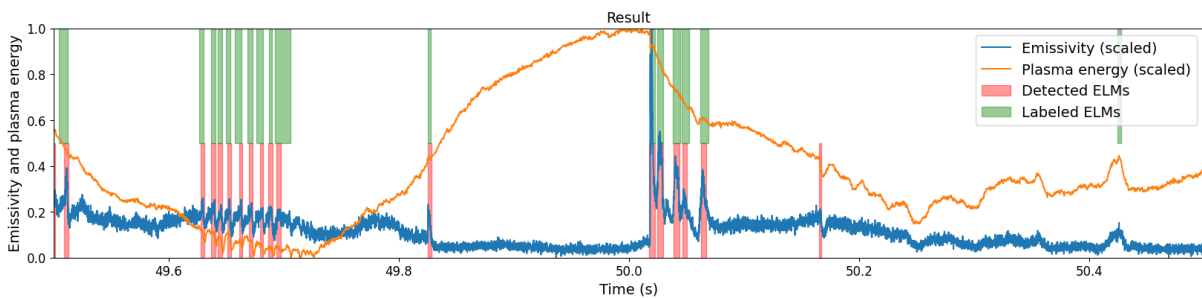


Figure 8.28: The light and energy signal of shot 94776-0 together with the detected peaks in red and correct peaks in green

8.5.2 Conclusion

This method improves upon the LoG method by taking into account the energy signal. As a result the false positives have decreased. See Chapter 11 for more information.

8.6 CustomKernel method

This final filter based method is applicable to both peaks and ELMs. It is similar to the LoG method in the sense that it relies on a single filtering operation to find peaks. It is also completely translation, amplitude and resolution invariant. Unlike LoG it uses a custom kernel of which the optimal values have been determined by a custom gridsearch algorithm. Another difference is that the CustomKernel method uses a correlation instead of a convolution. This makes the filtering operation more intuitive because the kernel isn't reversed in a correlation, see Section 3.2.1. The LoG kernel is perfectly symmetrical, but in section 2.6 it is mentioned that a spike is characterized by a sudden spike in the emissivity signal that gradually goes away again. Incorporating this information into the kernel could lead to better performance. The hyperparameters are presented in the table below:

8 Filter-based ELM detection methods

Hyperparameter	Explanation	Change Impact	Optimal Value
interval	Length of the interval over which the custom kernel is applied.	Longer intervals detect longer peaks and vice versa	0.014s
custom_kernel	custom kernel to detect peaks	/	See Figure8.29
threshold	Value for SimpleThreshold step of peak detection	Larger values decrease the sensitivity.	0.08
edges_0	Option to pad both sides of the kernel with zero.	Set to True when the kernel was trained with padded zero's on both ends.	False
max_gap	Maximum gap allowed between peaks in a compound ELM	Larger values will lead to more compound ELMs	0.001s
padding_start	Padding added to the start of each peak	Larger values will lead to more padding at the front.	0.001s
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.001s

Table 8.7: Hyperparameters and their descriptions for the LoG ELM and peak detection method.

Because determining the optimum kernel is computationally very expensive a small kernel is used. When the method is executed this kernel is upsampled as required using a process explained in Section 8.6.1. When experimenting the largest kernel contained only 13 elements. Based on experimentation increasing the amount of items in the kernel does not improve performance.

8.6.1 Kernel upsampling

As mentioned in section 8.4.2 of the LoG method, the amount of items in the kernel is calculated by the resolution and the interval over which the kernel is applied. In the LoG and LoGDoG methods a new kernel is generated to fill the kernel by using mathematical functions like the Gaussian PDF.

This is not possible here because the custom kernel is not determined by a formula, it is only defined by values in an array with a (small) fixed size. This means that the small kernel needs to be upsampled to the right amount of elements. This is done by a function that interpolates the kernel using a cubic spline function. This function also shifts the kernel, so the integral is equal to zero. this is important for reasons explained in Section 3.2. The figure below contains the default kernel that is used by the CustomKernel method, and an upsampled variant that is ready to be applied to an emissivity signal.

8 Filter-based ELM detection methods

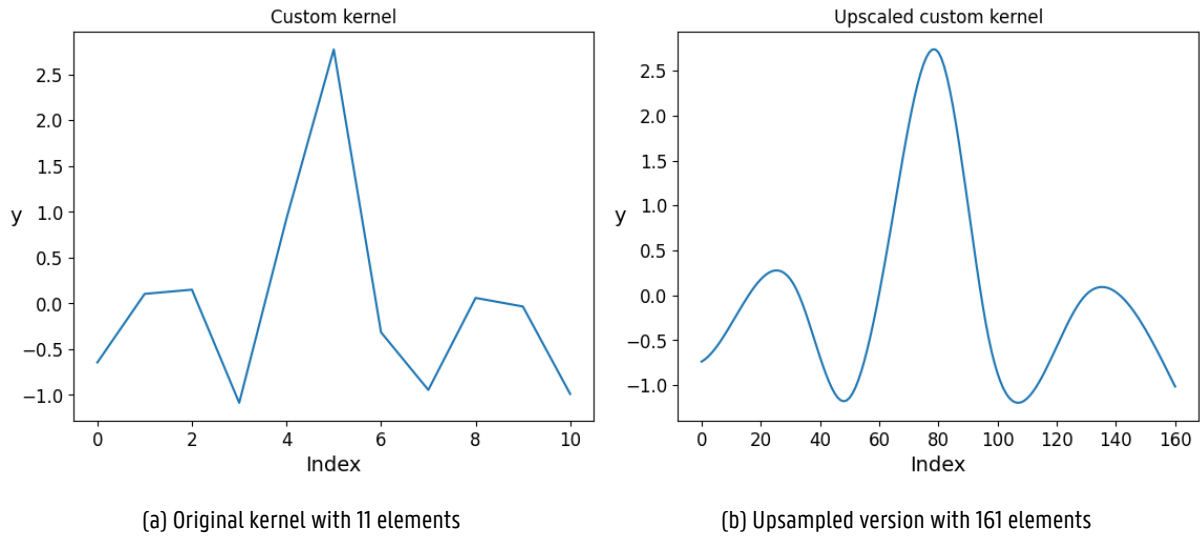


Figure 8.29: Kernel upsampling demonstrated on the default kernel used in the CustomKernel method.

It is to be noted that the same method can also be applied for downsampling an existing kernel.

8.6.2 Determining the optimal kernel

The 'optimal' kernel is determined using a modified grid search that brute forces different combinations. As a time saving measure the amount of items in the kernel should be limited to a small number (<15). If this method is expanded upon in future research it is recommendable to implement this with gradient descent to speed up the process. This process is implemented in a notebook called notebook nbKernelExperimentation. The training process uses hyperparameters, which are explained in Table8.8 below:

Hyperparameter	Explanation	Change Impact	Default Value
kernel_length	Amount of items in the custom kernel	More items allow for a more complex kernel shape	11
step	Amount of variation possible in the first iteration	/	0.5
shrink_factor	Multiplier to decrease the step after each iteration $\in]0, 1[$	Larger values will converge slower but are less likely to get stuck in local minima	0.9
minimum	Below this threshold the step stops shrinking	Larger values will stop training faster but will lead to less optimal values for the kernel	0.1

Table 8.8: Hyperparameters and their descriptions for the LoG ELM and peak detection method.

First the kernel is initialized by random numbers $\in]-step, step[$. Subsequently, all possible variations ($2^{\text{kernel_length}}$ options)

8 Filter-based ELM detection methods

of this random kernel are generated where an element is either increased or decreased by offset 'step'. All possible variations are evaluated and the best option is chosen. Next the step is decreased by multiplying it by 'shrink_factor' and the same process is repeated. The process only stops when two conditions are met: The step variable is below a limit defined by hyperparameter 'minimum' and the kernel performed the same as the kernel in the previous iteration. The figure below shows an example of a kernel converging on a potential solution after 39 iterations.

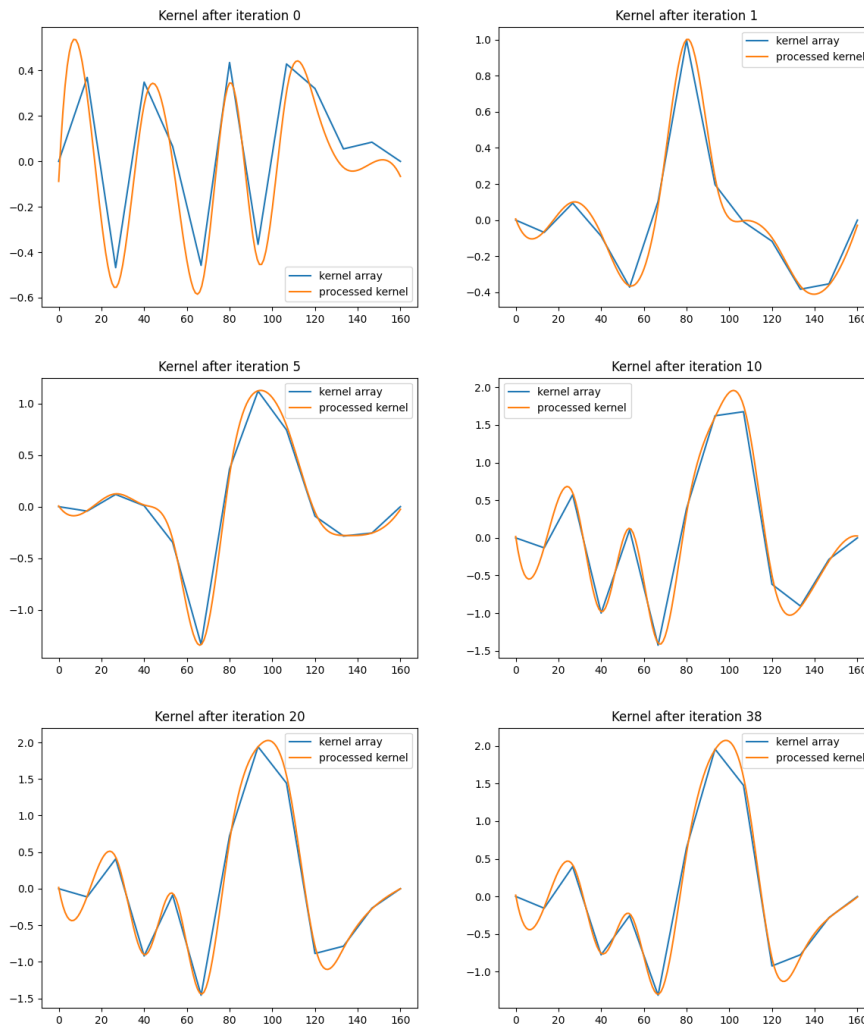


Figure 8.30: Kernel training process from beginning to completion

8.6.3 CustomKernel peak detection

To facilitate the explanation, shot 94423-0 is used which is illustrated in Figure 8.17. The first step is upsampling the kernel to the correct size. This is explained in Section 8.6.1. The next step is correlating the input signal with the upsampled kernel. The result of this correlation is illustrated in Figure 8.31 below.

8 Filter-based ELM detection methods

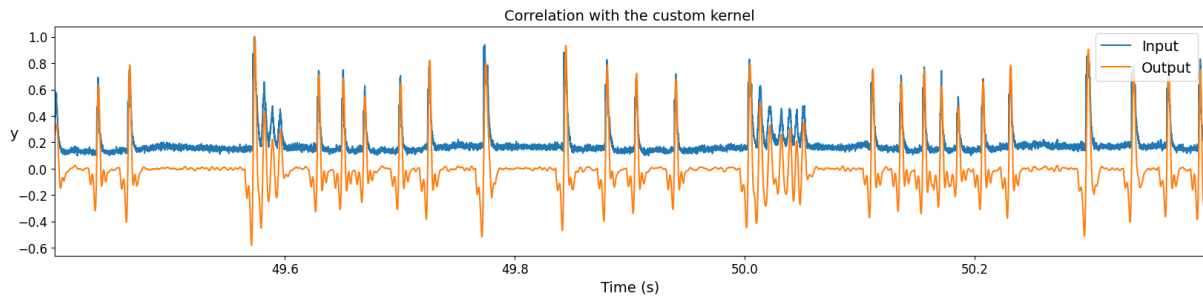


Figure 8.31: The input signal is correlated using the upsampled kernel

Peaks are obtained using the SimpleThreshold method using the 'threshold' hyperparameter.

8.6.4 CustomKernel ELM detection

The final steps are completely analogous to the LoG method, see Section 8.4. Subsequently, these peaks are padded and combined. The parameters for these operations are available in Table 8.6. The final result is displayed below in Figure 8.32.

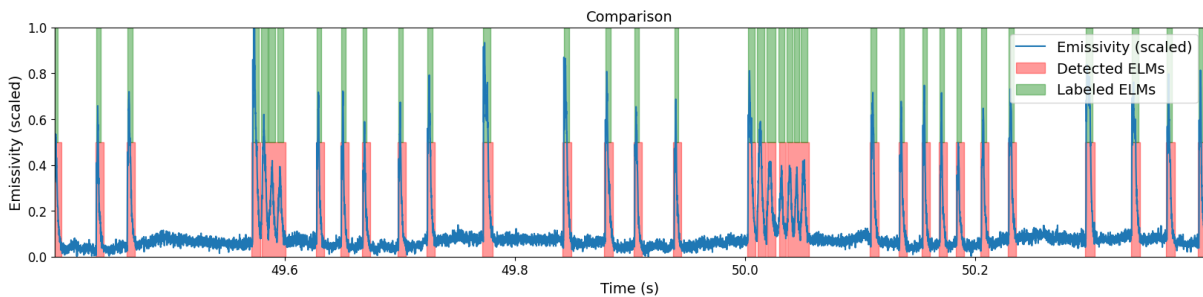


Figure 8.32: The light and energy signal of shot 94423-0 together with the detected peaks in red and correct peaks in green

8.6.5 Conclusion

The main disadvantage of this method is that a custom kernel is required. The kernel in this work was trained on the JET database. How well this applies to tokamaks from other institutions is unfortunately unknown, but as long as the peak shape isn't drastically different the method has the potential to perform well. See Chapter 11 for a more in-depth analysis.

9

Neural network ELM detection methods

This chapter focuses on the application of neural network-based methods, developed in TensorFlow [26], for ELM detection. Leveraging their ability to learn complex patterns and relationships from large amounts of data, neural networks offer a promising approach to tackle the challenges associated with ELM detection.

Two distinct neural network methods for ELM detection will be described in this chapter, each with its approach to building a neural network model. These methods aim to enhance the accuracy, efficiency, and robustness of ELM detection.

In the first few sections, an overview is provided of the common preprocessing and postprocessing and training techniques employed in both methods. This includes data normalization, feature extraction, handling the output of the neural network model and training. Understanding the preprocessing and postprocessing steps is essential as they form the foundation for the subsequent neural network architectures.

The last sections delve into the individual methods: one using a recurrent neural network (RNN) and the second one using a convolutional neural network (CNN). For each method, a detailed description is presented of the network architecture, which highlights the purpose and characteristics of each layer.

9.1 Preprocessing data for neural networks

As with the filter-based methods, both neural network methods get energy and light emissivity signals as input. In the neural network-based methods, the input signals undergo a preprocessing step. This step is composed of two main substeps: statistical feature extraction and sequencing of the input.

9.1.1 Statistical feature extraction for usage in the input data

The first substep involves extracting statistical features from the energy and light signals using multiple filters. The light signal is processed through two filters, namely the Moving Average Convergence Divergence (MACD) filter and the Laplacian of Gaussian (LoG) filter. Similarly, the energy signal is filtered using the Difference of Gaussians (DoG) filter and the Savitzky-Golay filter. These filters generate new signals that capture different aspects of the original signals, enhancing their discriminative properties.

9 Neural network ELM detection methods

To ensure consistent scaling and prevent the influence of input scaling on the results, all the generated signals are normalized.

Overall, this substep yields six signals:

- Light signal (A)
- Energy signal (B)
- Light signal filtered by LoG (C)
- Energy signal filtered by DoG (D)
- Light signal filtered by MACD (E)
- Energy signal filtered by Savitzky-Golay (F)

Each signal is denoted by a corresponding letter, which will be referenced in the subsequent section 9.1.2.

When researching the effect of each signal on the model, it was found that each signal is essential for providing good predictions but the LoG and DoG signals are the most influential for the output of the model.

Future work could incorporate more signals or try to make the whole set of input signals resolution invariant.

9.1.2 Sequencing the data to be compatible with fixed-length neural network models

In the sequencing substep, the six signals obtained from the statistical feature extraction are organized into sequences of equal length. This step is necessary as both neural network models require fixed-length input. Take for example the input in Fragment 9.1

```
[  
  [A1, B1, C1, D1, E1, F1],  
  [A2, B2, C2, D2, E2, F2],  
  [A3, B3, C3, D3, E3, F3],  
  [A4, B4, C4, D4, E4, F4],  
  [A5, B5, C5, D5, E5, F5],  
]
```

Code Fragment 9.1: Example input with extracted features. Each row represents one sample in the time series. Note that in practice, the values are numbers.

Sequences are generated by sliding a window of defined length over the signals with a stride of 1. For instance, if the sequence length is set to 3, the resulting partially overlapping sequences from Fragment 9.1 would be as shown in Fragment 9.2.

9 Neural network ELM detection methods

```
[
  [
    [A1, B1, C1, D1, E1, F1],
    [A2, B2, C2, D2, E2, F2],
    [A3, B3, C3, D3, E3, F3],
  ],
  [
    [A2, B2, C2, D2, E2, F2],
    [A3, B3, C3, D3, E3, F3],
    [A4, B4, C4, D4, E4, F4],
  ],
  [
    [A3, B3, C3, D3, E3, F3],
    [A4, B4, C4, D4, E4, F4],
    [A5, B5, C5, D5, E5, F5],
  ]
]
```

Code Fragment 9.2: Sequencing applied to Fragment . With sequence length 3 and stride 1.

9.2 Defining the output of the model

The output of each neural network model is a set of sequences constrained to a one-dimensional array of values between zero and one. The length of each array is the same as the input sequence length. So in the example above, this would be 3. The number of sequences in the output corresponds to the number of sequences in the input. This output format facilitates implementation and training, with each value in the array indicating the likelihood of an ELM's presence at a specific moment in time. A value closer to zero signifies the absence of an ELM, while a value closer to one signifies its presence.

9.3 Preprocessing step on the labels used during training

In order to train the neural networks, labeled sequences need to be created because the framework requires the labels to be in the same format as the model's output.

To accomplish this, the labels indicating the presence of an ELM are converted into a binary signal, where zero represents the absence of an ELM and one represents the presence of an ELM. Figure 9.1 provides a visual representation of this binary signal. Once the binary signal is generated, it undergoes the same sequencing process as described in Section 9.1.2.

9 Neural network ELM detection methods

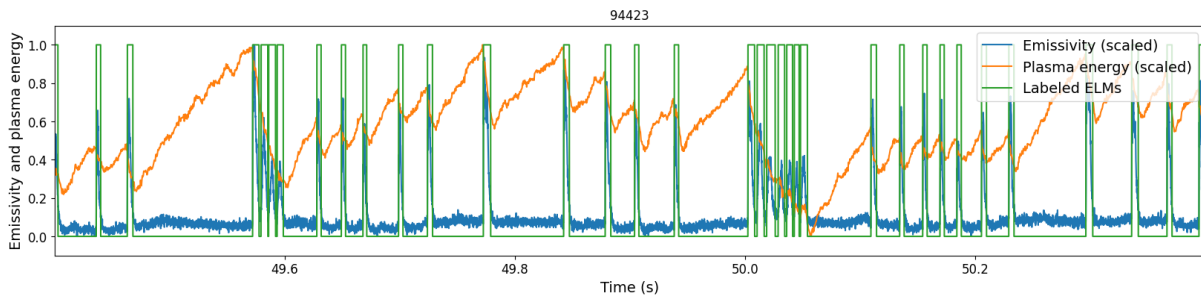


Figure 9.1: Labels represented as a binary signal.

9.4 Extra steps required for making the custom metrics compatible with the models during training

To use the metrics defined in Chapter 5, it is necessary to convert the model outputs and label sequences from a binary representation to an interval representation.

As mentioned previously, both the model output and label sequences are represented as one-dimensional arrays. To convert them into intervals, the Simple Threshold method discussed in Section 8.1.1 is applied with a threshold of 50%. This method assigns values above the threshold to the ELM intervals, while values below the threshold are ignored.

9.5 First important remark on training: potential problem with sequences

The training process is conducted using multiple shots to enhance the model's ability to learn the complexities of different ELM types. To achieve this, the sequences from all the used shots are concatenated together. In other words, the sequences from one shot are appended at the end of the array of sequences from another shot. This results in a single large array of sequences for the labels and a single large array of sequences for the preprocessed input data.

While each sequence from one shot is kept separate and not mixed with data from another shot, this approach can pose challenges for recurrent neural networks (RNNs) due to their ability to remember temporal dependencies from past sequences. However, it does not pose a problem for other neural networks like the CNN model described in Section 9.9.

9.6 Second important remark on training: hardware and software limitations

It should be noted that, during the master's dissertation, the training of these methods was constrained to an Intel® Core™ i7-8750H CPU, which boasts 6 cores with hyperthreading and a maximum turbo frequency of 4.10 GHz and 16 GB RAM. This is because due to software problems, CUDA [27] could not be used to speed up the training process. As a result, the number of epochs was kept low at two epochs. The batch size is also set to 32 to have a balance between memory usage, training speed, and accuracy in the estimation of the gradient. Future work could try increasing the number of epochs and setting

another batch size.

9.7 Postprocessing the output of a neural network to be compatible with the other ELM detection methods

In the postprocessing stage, merging multiple partially overlapping prediction sequences is required. The predictions made on multiple sequences can be merged by calculating the mean value for each moment from the sequences, resulting in a new prediction array of the same length as the original input arrays. Similar to the training phase, the output is initially in a binary-like representation, but it needs to be transformed into intervals. This is accomplished by applying the simple threshold method with a threshold of 50% to the prediction array. In other words, means greater than 50% are considered part of an ELM, and means below 50% are not considered as an ELM. Code fragment 9.3 shows how this is implemented.

```
# Generate predictions using the model
model_output = model.predict(np.array(X_sequences))

# Combine partially overlapping predictions
length = len(model_output)
temp = [[] for _ in range(length + self.sequence_length - 1)]
for index, pred in enumerate(model_output):
    for n in range(self.sequence_length):
        temp[index + n].append(pred[n])
predictions = [sum(values) / len(values) for values in temp]

# Detect peaks using a simple threshold
peaks = SimpleThreshold(threshold=0.5).find_peaks(time, predictions)
```

Code Fragment 9.3: Merging multiple partially overlapping prediction sequences.

During the testing phase, it was observed that the detected ELM intervals were excessively narrow, drastically lowering the true positive rate. This occurrence could potentially be attributed to inconsistencies in the labeling process. To mitigate this, the padding options and the dropping mechanism using the Savgol Gap algorithm from Section 8.2.3 were added as extra postprocessing steps. As with the other methods, these are optional and can be disabled.

9.8 An overview of the recurrent neural network method

The RNN ELM detection method utilizes several hyperparameters, listed in Table 9.1. To accommodate time and computing limitations, mentioned in Section 9.6, default parameters were selected, drawing inspiration from the corresponding filter-based detection methods. Future endeavors could focus on enhancing these parameters. Nonetheless, the results, as explored in Chapter 11, exhibit significant promise.

9 Neural network ELM detection methods

Hyperparameter	Explanation	Change Impact	Optimal Value
interval_LoG	The interval over which the LoG kernel is active	Longer intervals detect longer peaks and vice versa	0.007 s
interval_DoG	The interval over which the DoG kernel is active	Longer intervals take into account more time around the current peak to determine if a peak in the light signal is an ELM	0.001 s
gain_DoG	Multiplier for the tanh activation function	Larger values are less likely to disqualify ELMs because the drop in energy is not big enough.	21
slow_MACD	The greater span width used for calculating the MACD.	Higher values remove lower frequency noise.	12
fast_MACD	The smaller span width used for calculating the MACD. Has to be less than slow.	Higher values remove lower frequency noise.	9
span_Savgol	The window used for the Savitzky-Golay filter.	Higher values remove lower frequency noise.	30
resolution	Time between two samples. Depends on the sampling rate.	Method needs to be retrained after change.	0.0001
sequence_length	Sets the length of the input and output sequences used by the model.	Longer sequences may yield better results but have a higher risk of overfitting.	16
batch_size	The number of training examples that are processed together in a single forward and backward pass during the training phase.	Increasing this causes more memory usage during training but may lower the risk of overfitting.	32
model_path (not a hyperparameter, just a normal parameter)	Where the model's weights are saved.	.	modelsRNN
padding_start	Padding added to the start of each peak.	Larger values will lead to more padding at the front of a peak.	0.0005
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.0005
drop_using_savgol	Whether or not to remove detected ELMs from the output based on the energy signal.	If true, Removes a peak if the energy is going up.	True

Table 9.1: Hyperparameters and their descriptions for the RNN ELM detection method.

9 Neural network ELM detection methods

The RNN model processes input data in a sequential manner, taking into account the temporal dependencies present in the signals.

The architecture of the RNN model consists of multiple Gated Recurrent Unit (GRU) layers [28], followed by dropout layers for regularization [29], and a final dense layer with sigmoid activation output [30].

The GRU layers in the model are responsible for capturing and encoding the temporal information in the input sequences. They have memory cells that allow them to maintain an internal state. This enables them to remember information from past sequences while processing a newer sequence. This memory mechanism can help in capturing the dynamics and patterns of ELMs in the input signals.

The model developed for this master's dissertation begins with a GRU layer with 64 units, which returns sequences and takes an input of the shape (sequence length, number of features). While the sequence length is a parameter set in the constructor of the RNN method, the number of features is equal to six, as defined in section 9.1.2.

To prevent overfitting during training and to improve the model's generalization ability, dropout regularization is applied directly after the first GRU layer. The dropout rate, which indicates the percentage of units to randomly drop during each training step is set to 20% during training and 0% when not in training.

Following the first GRU layer, another GRU layer is added with 32 units, which again returns sequences. This layer further refines the learned representations and captures more complex patterns in the input sequences.

A second dropout layer is then applied to regularize the outputs of the second GRU layer.

Finally, the model ends with a third GRU layer with 16 units, followed by a dense layer. The dense layer uses the sequence length for the number of units and uses the sigmoid activation function to produce the model's output. As mentioned in section 9.2, this output now represents the likelihood of an ELM occurring at that specific moment in time.

Figure 9.2 shows the final result, obtained when applying the method to a shot.

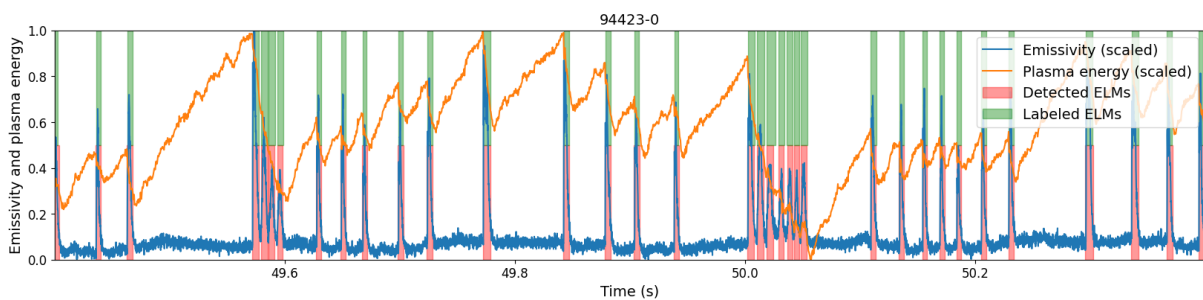


Figure 9.2: The RNN method applied on shot 94423-0.

9.9 An overview of the convolutional neural network method

To compare the results with another neural network, an existing peak detection CNN [31] model was implemented. However, some modifications were made to make it compatible with the neural network base code that was written for this master's

9 Neural network ELM detection methods

dissertation and defined in the previous sections of this chapter. The hyperparameters are listed in Table 9.2. As with the RNN, to accommodate time and computing limitations, default parameters were selected, drawing inspiration from the corresponding filter-based detection methods.

9 Neural network ELM detection methods

Hyperparameter	Explanation	Change Impact	Optimal Value
interval_LoG	The interval over which the LoG kernel is active	Longer intervals detect longer peaks and vice versa	0.007 s
interval_DoG	The interval over which the DoG kernel is active	Longer intervals take into account more time around the current peak to determine if a peak in the light signal is an ELM	0.001 s
gain_DoG	Multiplier for the tanh activation function	Larger values are less likely to disqualify ELMs because the drop in energy is not big enough.	21
slow_MACD	The greater span width used for calculating the MACD.	Higher values remove lower frequency noise.	12
fast_MACD	The smaller span width used for calculating the MACD. Has to be less than slow.	Higher values remove lower frequency noise.	9
span_Savgol	The window used for the Savitzky-Golay filter.	Higher values remove lower frequency noise.	30
resolution	Time between two samples. Depends on the sampling rate.	Method needs to be retrained after change.	0.0001
sequence_length	Sets the length of the input and output sequences used by the model.	Longer sequences may yield better results but have a higher risk of overfitting.	8
batch_size	The number of training examples that are processed together in a single forward and backward pass during the training phase.	Increasing this causes more memory usage during training but may lower the risk of overfitting.	32
model_path (not a hyperparameter, just a normal parameter)	Where the model's weights are saved.	.	modelsCNN
padding_start	Padding added to the start of each peak.	Larger values will lead to more padding at the front of a peak.	0
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.0015
drop_using_savgol	Whether or not to remove detected ELMs from the output based on the energy signal.	If true, Removes a peak if the energy is going up.	False

Table 9.2: Hyperparameters and their descriptions for the CNN ELM detection method.

9 Neural network ELM detection methods

The modifications made are the following. One dense layer was added as a new output layer, with one unit and sigmoid as the activation function. As for the hyperparameters used to construct the model, these are defined in Fragment 9.4.

```
dropout_rate = 0.2 if training else 0.0
model = ConvNet(
    dropout=dropout_rate,
    pool_sizes=[1, 1, 1],
    input_shape=(sequence_length, num_features),
    output_shape=(sequence_length, 3),
)
```

Code Fragment 9.4: Creation of the CNN model with custom hyperparameters. The output shape here is not for the last layer because a new layer was added at the end of the model.

While the architecture and hyperparameters of the CNN model were adjusted for ELM detection, the fundamental principles of convolutional operations, pooling, and non-linear activations remain intact. The CNN model consists of several convolutional layers, activation functions, batch normalizations, dropout, and addition operations to extract features and learn representations from the input data. These operations are designed to capture relevant patterns and facilitate the prediction of peaks, and thus ELMs accurately.

Figure 9.3 shows the final result, obtained when applying the method to a shot.

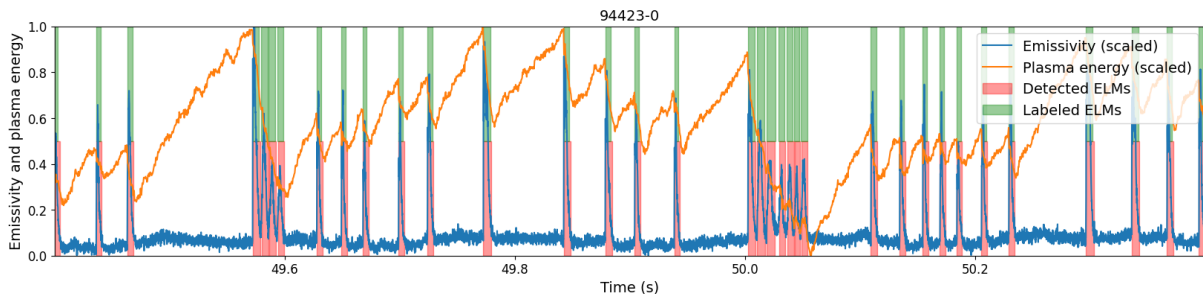


Figure 9.3: The CNN method applied on shot 94423-0.

10

Deconvolution ELM detection method

This ELM detection method deviates from the other methods discussed in this work because it is not based on filtering the input signal or processing the input signal with neural networks. Instead, it is based on a paper titled "Fluctuations in a vast range of physical systems can be described as a superposition of uncorrelated pulses with a fixed shape" by S. Ahmad. [32] In this paper, the authors propose a novel deconvolution technique that can be used to estimate arrival times and amplitudes of pulses in time series data. The method is designed to handle processes that can be described as a superposition of uncorrelated pulses with a fixed shape, resembling the characteristics of some ELM signals. The deconvolution method outlined in the paper has been implemented and adapted for ELM detection in the current work.

Hyperparameter	Explanation	Change Impact	Optimal Value
prominence	The required prominence of the peaks to be detected	Larger values require the peaks to stick out more from the surrounding baseline.	0.3
interval_peak	Interval of the peak_shape.	Longer intervals detect longer peaks and vice versa	0.006s
iterations	How many iterations are preformed in the deconvolution algorithm.	Increasing this value detected more peaks.	400
padding_start	Padding added to the start of each peak	Larger values will lead to more padding at the front.	-0.001s
padding_end	Padding added to the end of each peak	Larger values will increase the padding at the end of the peak.	0.006s
peak_shape	Peak shape meant to represent an ELM, used for deconvolution process	Unknown	See Figure 10.2

Table 10.1: Hyperparameters and their descriptions for the LoG ELM and peak detection method.

10.1 Prototype ELM shape

The deconvolution process requires a pulse shape that is to be used as a deconvolution kernel. This pulse shape should be representative of all peaks in the signal. To create a suitable pulse shape, ELMs are manually selected and combined into a singular "Prototype ELM shape". The prototype ELM should have a low noise level and contain the features present in most ELMs as outlined in Section 2.6. First representative ELM profiles are chosen out of the data. Representative ELM profiles should have low noise and a regular shape. Subsequently, the intervals of the representative ELM profiles are manually adjusted to contain the entire ELM and nothing more. Figure 10.1 below contains twenty-two representative ELM profiles.

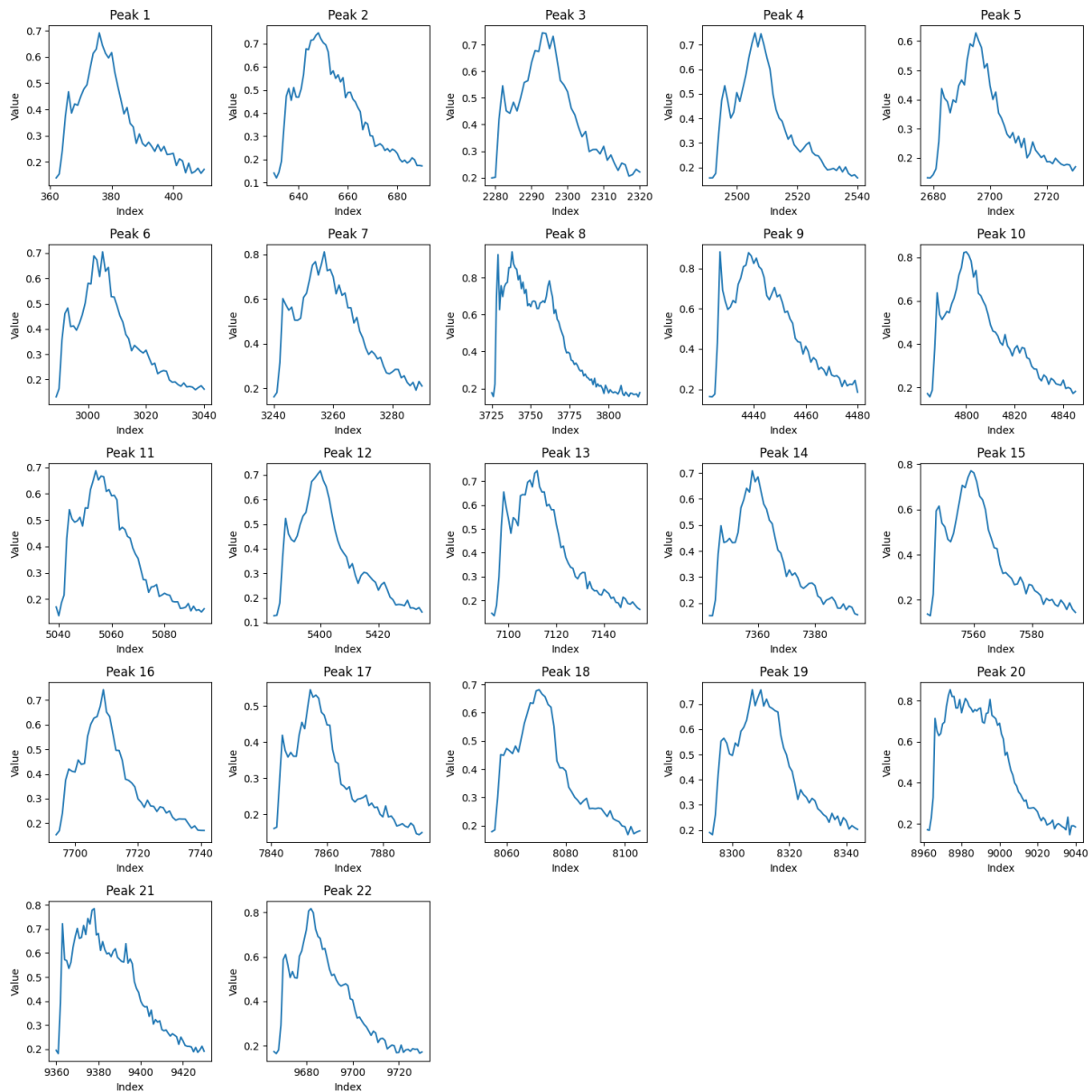


Figure 10.1: Representative ELM profiles

10 Deconvolution ELM detection method

In the next step the representative ELM profiles are resampled to have the same amount of elements. This done using the method described in Section 8.6.1. Finally, the different candidates are averaged to create the prototype ELM shape. The result is presented in Figure 10.2 below.

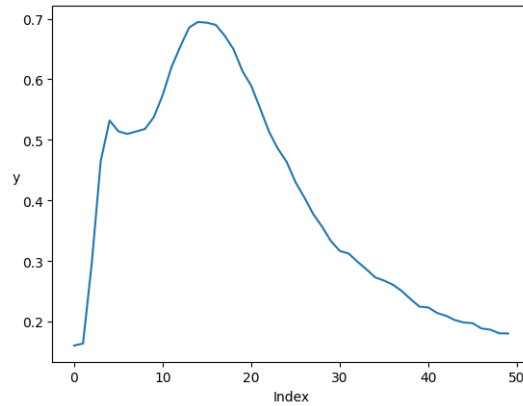


Figure 10.2: Prototype ELM shape to be used as a pulse shape for the deconvolution process

This prototype ELM is upsampled and downsampled depending on the interval hyperparameter and the resolution of the input signal. The counselor of this master's dissertation mentioned that he arrived at a similar approximate shape in independent research using a functional data clustering approach.

10.2 Implementation

To facilitate the explanation, shot number 94423-0 is used. The signal is illustrated in Figure 8.17 where the same signal was used to explain the implementation of the LoG method.

The first step is resampling the prototype ELM so it contains the correct amount of items for a given resolution and interval_peak hyperparameter. This is done using the logic detailed in 8.6.1.

Subsequently, the light signal is deconvolved using the method `RL_gauss_deconvolve` provided by the original implementation. [33] The prototype ELM is supplied as the deconvolution kernel. The hyperparameter iterations determines how many iterations are preformed in the deconvolution process. The original signal and deconvolved signal are displayed in the Figure 10.3 below.

10 Deconvolution ELM detection method

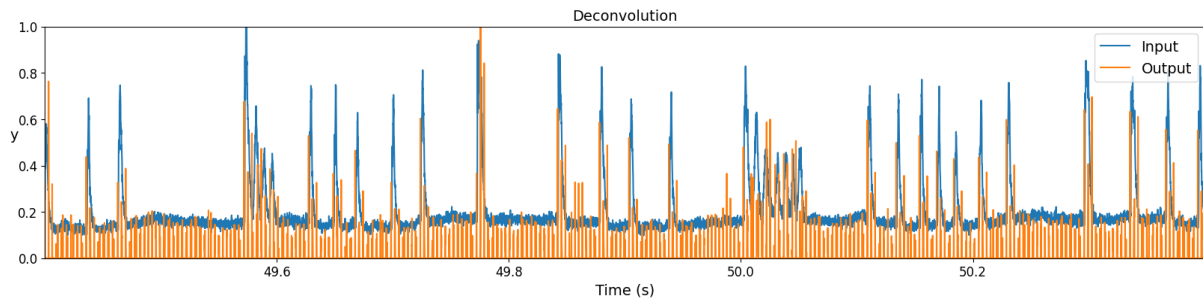


Figure 10.3: Deconvolution step

The signal is now represented as a superposition of individual pulses. To extract the pulses the method `three_point_maxima` from the original implementation is used. It is a wrapper for the method `find_peaks` from SciPy. Because of the significant amount of noise present in the signal the 'prominence' and 'wlen' parameters are used, as suggested in the original implementation. [33] The former determines how much a peak needs to rise above its surroundings to be detected, while the latter determines the size of the window in which the prominence is to be calculated. [34].

The output of the `three_point_maxima` method needs to be processed further because the method only extracts the timestamp at which the maximum amplitude occurs, while this work attempts to extract the start and end times of each peak. Each timestamp is converted to an interval by using the padding hyperparameters in the following manner: $peak = [timestamp - padding_start, timestamp + padding_end]$. It is notable that the padding parameter is negative. This is because the deconvolution technique has a tendency to detect peaks slightly before the peak occurs in the input signal. The result of this padding is illustrated in Figure 10.4 below.

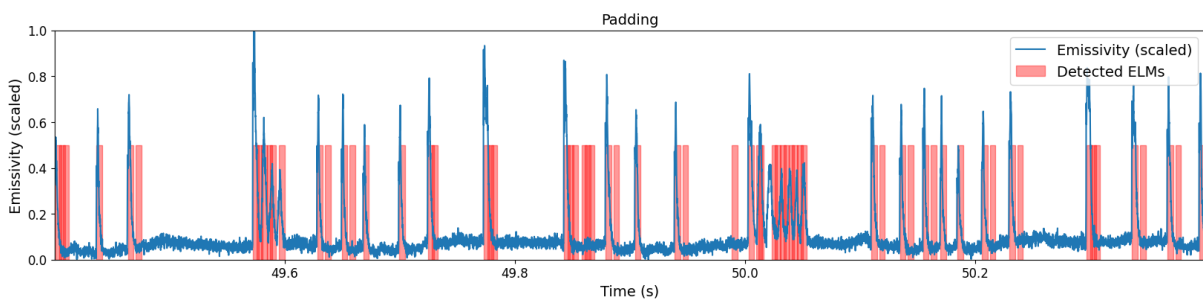


Figure 10.4: Padding of the peaks detected using `three_point_maxima`

After padding a lot of peaks overlap. This is solved by applying the `StaticGap` mixin with a `max_gap` of zero. The final result is illustrated in Figure X below.

10 Deconvolution ELM detection method

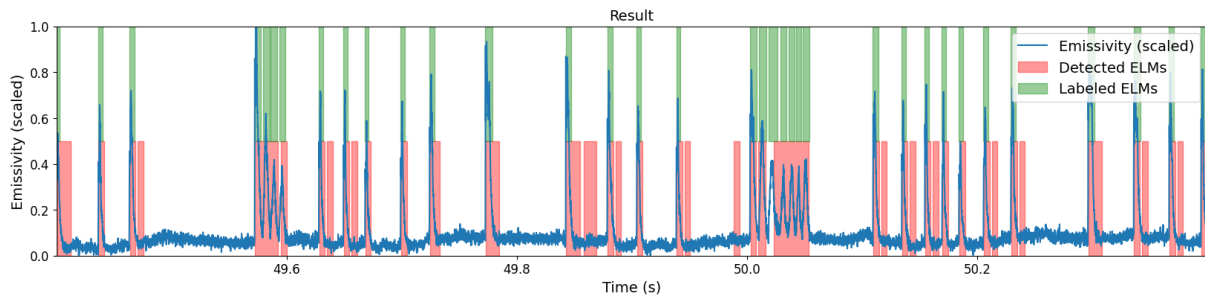


Figure 10.5: Result of the Deconvolution ELM detection method

During the implementation it was discovered that this method is strongly dependent on resolution. Each time the resolution changes the 'threshold' and 'iterations' hyperparameters need to be reconfigured by performing a computationally intensive grid search. Because the result was unsatisfactory on relatively easy shots, further attempts at experimentation, such as executing several deconvolutions with different pulse shapes and combining the results, were halted due to time constraints.

10.3 Conclusion

The performance of the deconvolution-based method in accurately identifying and characterizing ELMs is not as robust as initially anticipated. More information regarding its shortcomings and results can be found in Chapter 11.

11

Results

The following chapter presents a comprehensive analysis of the performance of various ELM detection methods employed in this study. A brief overview of the method's properties and execution time was given in Table 7.1. The evaluation encompasses both novel approaches developed specifically for this master's dissertation and existing methods. The objective is to assess the effectiveness and limitations of these methods in accurately identifying ELM events.

11.1 Overview

Table 11.1 below contains the results achieved by the ELM detection methods. A visual comparison between the results can be found in Appendix B. Among the evaluated methods, the rolling Z-score, LoG, LoGDoG, CustomKernel, and RNN (modified) demonstrate the most promising results.

	gap					interval				
	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)
ROT	34.41	65.59	25.62	74.38	44.69	34.41	65.59	17.53	82.47	52.48
MACD	87.17	12.83	78.13	21.87	57.25	87.17	12.83	21.63	78.37	82.25
Rolling Z-score	89.99	10.01	60.82	39.18	64.16	89.99	10.01	16.39	83.61	87.33
LoG	86.61	13.39	57.86	42.14	64.46	86.61	13.39	12.79	87.21	87.81
LoGDoG	80.78	19.22	45.15	54.85	67.59	80.78	19.22	10.91	89.09	87.42
CustomKernel	82.97	17.03	46.01	53.99	68.24	82.97	17.03	9.39	90.61	88.62
RNN	83.16	16.84	20.01	79.99	81.53	83.16	16.84	14.60	85.40	84.65
RNN (modified)	91.95	8.05	59.52	40.48	65.46	91.95	8.05	15.90	84.10	87.07
CNN	53.35	46.65	12.50	87.50	69.35	53.35	46.65	10.89	89.11	70.61
CNN (modified)	80.01	19.99	54.00	46.00	63.27	80.01	19.99	16.63	83.37	82.66
Deconvolution	78.14	21.86	84.39	15.61	51.90	78.14	21.86	70.30	29.70	56.28

Table 11.1: Table with results for all methods. The GapMetrics are calculated using a threshold of 0.75 for TP and FN and 0.1 for the FP and TN. The IntervalMetrics are calculated using a threshold of 0.75 for TP and FN and 0.5 for the FP and TN. The F_{β} -scores are calculated for both metrics separately with a beta value of 0.5.

Table 11.1 presents the results of the evaluation of various ELM detection methods using two sets of metrics: Gap Metrics and Interval Metrics. While both metrics provide valuable insights, the Interval Metrics offer a more comprehensive overview of the actual performance of the methods. Therefore, unless explicitly specified, the following discussions regarding the evaluation results should be assumed to be based on the Interval Metrics, which offer a better perspective of the methods and their effectiveness in ELM detection.

The results presented in the table were obtained by calculating the metrics individually for each shot and then taking the mean of each metric of a shot per method. Therefore, the reported F_{β} -scores are the means of the F_{β} -scores calculated for each shot, and they are not derived from the mean values of true positives, false positives, and false negatives displayed in the table.

11.1.1 Discussion about labeling quality

At the inception of this master's dissertation, a set of manually labeled intervals was provided to facilitate the evaluation of the ELM detection methods. However, it is noteworthy that only half of the shots were labeled, and these labeled shots exhibited a considerable number of inaccuracies in the form of false positives and false negatives. Nonetheless, the methods were trained successfully using the available labeled shots. The results obtained using these labels can be found in Appendix C.

Six days prior to the completion of the master's dissertation, a new set of labels was introduced. This updated labeling encompassed all shots and effectively resolved numerous inconsistencies that were prevalent in the previous labeling scheme.

The transition to the new set of labels posed a substantial challenge. In addition to rectifying the inconsistencies present in the previous labeling, the new labels introduced a distinct paradigm. Specifically, the detection of compound ELMs now entailed the consideration of a new rule, as elucidated in Section 2.6. Furthermore, the average number of detected ELMs experienced a marked increase. This can be attributed to the reclassification of numerous small-scale ELMs, which, despite lacking a clear energy drop, were now classified as ELMs. Both of these modifications exerted substantial and consequential effects on the performance of the ELM detection methods.

Furthermore, it is important to acknowledge that a closer examination of individual shots and a comparison between the labels provided by the ELM detection methods and the ground truth regularly revealed instances where the method's outputs were more accurate than the assigned labels. This observation suggests that the performance results presented in Table 11.1 should be interpreted with caution.

11.2 Performance evaluation of the CNN method

A consequence of these inconsistencies can be observed with the CNN method. The presence of inconsistencies causes the CNN model to exhibit a tendency to output very thin ELM intervals, which subsequently leads to a reduced detection rate in the true positives (TP rate). As discussed in Section 9.7, padding was introduced as a solution to increase the number of true positives, resulting in the modified CNN variant (CNN (modified)). However, an undesired side effect of this modification is

an increase in the number of false positives being detected.

11.3 Performance evaluation of the LoG method

The LoG method, developed immediately after the commencement of the master's dissertation, is characterized by its simplicity and robust performance. Its straightforward implementation and symmetric LoG kernel contribute to its wide applicability. Despite its simplicity, the method remains one of the top-performing approaches. It demonstrates fast execution, easy retraining, and successful performance with both the old and new labeling sets. Furthermore, the LoG method exhibits the advantageous quality of achieving satisfactory results even with manually estimated hyperparameters. This favorable characteristic will be leveraged in Section 11.12, where the method is applied to a novel dataset with promising outcomes.

11.4 Performance evaluation of the CustomKernel method

The CustomKernel method demonstrates excellent performance, with a slightly lower true positive score (3.64% lower) compared to the LoG method, but also a reduced false positive score (3.4% lower). This method achieves the lowest false positive score among all the evaluated methods. These findings suggest that when excluding ELMs based on the absence of significant energy drops becomes ineffective, relying on the shape of the ELM becomes a viable alternative. As mentioned in Section 3.2, this method relies on a custom kernel for its operation. Therefore, when applying the CustomKernel method to other datasets with significantly different peak shapes, determining a new optimal custom kernel may be necessary. Further details can be found in Section 11.12.

11.5 Performance evaluation of the LoGDoG method

LoGDoG is designed to achieve a high detection rate comparable to that of LoG, while mitigating the occurrence of false positives. Under the old labeling scheme, LoGDoG consistently exhibited effective filtering of false positives by leveraging the energy signal, occasionally even outperforming the ground truth labels. Moreover, it demonstrated greater consistency in correctly identifying small peaks as ELMs by analyzing the energy drop. Figure 11.1 illustrates how the previous version of LoGDoG adeptly disregards the peak around 49.8 seconds due to the negligible energy drop.

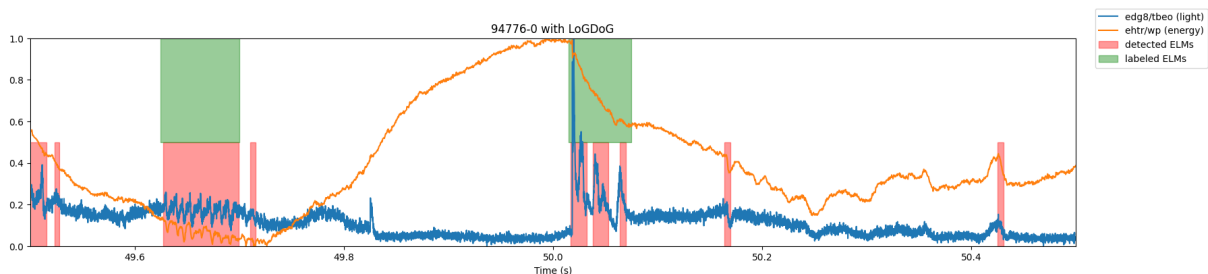


Figure 11.1: The light and energy signal of shot 94776-0 together with the detected peaks in red and correct peaks in green

The introduction of the new labels resulted in the retraining of the LoG method due to numerous small labeling changes,

such as the reclassification of the peak at 49.8 seconds as an ELM. As a consequence, the underlying principle of the LoGDoG method, which relied on excluding peaks based on a lack of energy drop, became less effective. This is evident in Figure 11.2, which displays the processed DoG signal based on the old hyperparameters. The signal remains close to zero for significant portions, indicating the absence of detectable ELMs. Consequently, when applying the method without retraining, the true positives would be substantially reduced. For a direct comparison, Figure 8.26 and Figure 8.28 depict the same processing step and final result for the same shot with the new hyperparameters following retraining on the revised labeling dataset.

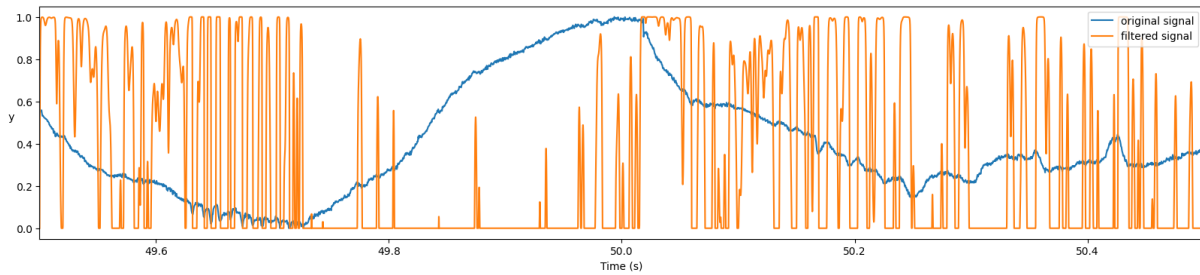


Figure 11.2: The processed DoG signal using the old LoGDoG method

The transition to the new labels and the associated modifications in the labeling paradigm has influenced the performance of the LoGDoG method. The method exhibits a decrease in the true positives by 5.83% and a marginal decrease in the false positives by 1.88%. These changes highlight the impact of the new labeling set on the effectiveness of the LoGDoG method. However, it is worth noting that the method still holds promise when applied to different tokamaks or within alternative labeling paradigms.

11.6 Performance evaluation of the MACD method

The MACD method is a strong contender in achieving high true positives but at a cost of a higher number of false positives compared to most other methods. Despite the MACD's origin, as indicator for the stock market, it is still able to achieve a high F_{β} -score. With both the old and new labels, the MACD was able to accurately predict ELMs. The number of parameters make it highly tuneable and it also has a fast execution time.

11.7 Performance evaluation of the rolling Z-score method

The rolling Z-score method is able to provide the second-highest number of true positives. This is again at the cost of a higher number of false positives, but not as bad as with the MACD. This method is very easy to implement and understand, which makes it a strong choice for ELM detection. However, the performance of the method was a lot worse when trained

and evaluated on the old labels, which may indicate that the method is not suitable for each application.

11.8 Performance evaluation of the RNN method

The RNN method uses multiple feature-extracting filters in its preprocessing step. Combined with the simplicity of the model, and only 125 kB of weights, this method is very accurate in detecting ELMs. One should note though that it is slow to run, which makes it not suitable for real-time applications.

Though not as bad as the unmodified CNN method, the RNN method does have predicted ELMs that are too small to be counted by the metrics. By adding some padding, the number of true positives increased from 83.16 % to 91.95 %, which is the highest of all methods. This padding does increase the number of false positives from 14.60 % to 15.90 %. Note that RNN (modified), which is the padded version, also enables the Savgol Gap drop mechanism, which helps reduce the impact of the padding on the number of false positives.

As an end result, the RNN (modified) is one of the best-proposed methods to use for detecting ELMs. It is able to detect small intricacies in the data which other methods are not able to do and also uses the energy signal to make it extra robust.

11.9 Performance evaluation of the Deconvolution method

The deconvolution-based method, implemented based on a 2023 paper by S. Ahmad. [32], falls short of the initial expectations in terms of accurately identifying and characterizing Edge Localized Modes (ELMs). The method exhibits the highest false positive score and the second lowest F_{β} -score among the evaluated ELM detection methods.

One major factor contributing to the suboptimal performance is the high levels of noise present in the data. The deconvolution process relies on the assumption of uncorrelated pulses with fixed shapes, which may not hold well in the presence of significant noise.

Another factor that impacts the effectiveness of the deconvolution method is the irregularity and variability of the peak shapes exhibited by ELMs. The method assumes a constant pulse shape, which may not fully capture the diverse range of ELM characteristics encountered in our specific study.

Finally, the variability in the gaps between ELMs, or the waiting times, further complicates the deconvolution process.

In light of these challenges, it is important to acknowledge that the deconvolution-based approach, while conceptually intriguing, may not be well-suited for the specific characteristics of the ELMs encountered in this study. The unique nature of ELM events encountered, including high noise levels, irregular peak shapes, and variability in the waiting times, presents significant obstacles to the accurate and reliable application of the deconvolution process. Further research is needed to

explore its potential in different ELM scenarios with reduced challenges.

11.10 Discussion about false positives and the plasma energy signal

One noticeable trend in the data is that methods that do not take into account the plasma energy signal (LoG and Deconvolution) generally tend to exhibit a higher number of false positives. CustomKernel is an exception to this. False positives can often be identified and ruled out by considering the plasma energy information. To cope with these false positives, the Savgol Gap drop mechanism was added to some methods (MACD, Rolling Z-score, and RNN (modified), CNN (modified)), which was able to reduce the false positives from 16.63 % to 16.15 % in the CNN (modified) method. The effect of this change is equally small in the other methods as well and can have a counterproductive result. For example, it reduced the F_β score from 82.66 % to 82.55% in CNN (modified) because it also slightly reduces the true positives. As the effect of the Savgol Gap drop mechanism is rather minor, enabling this should be chosen on a per-method and per-dataset basis. The effect could however be increased, for better or worse, by increasing the Savgol window parameter available in these methods.

An alternative approach to mitigate false positives is provided by the LoGDoG method. This method tries to reduce the number of false positives by attenuating the result of a LoG filter by looking at drops in the energy signal. As discussed in Section 11.5, this approach works reasonably well but the new labeling set decreases the efficacy of this approach.

11.11 Comparison to existing methods

11.11.1 ROT and Deconvolution

In addition to the developed methods, existing approaches were also evaluated: ROT, developed by the research group In-fusion at UGent, and Deconvolution, a peak detection method based on a paper by the Complex Systems Modelling research group at UiT. While Deconvolution was originally designed for peak detection, the UiT research group suggested its potential suitability for ELM detection. The evaluation revealed that the new methods consistently outperformed both ROT and Deconvolution. It is important to acknowledge that the performance of methods can vary depending on the specific task and dataset. Therefore, the findings highlight the effectiveness of the approaches in the context of ELM detection.

11.11.2 CNN

As with the RNN, this method is rather slow and thus not suitable for real-time use. One other fact to give is that the model is a lot more complex than the RNN counterpart and its weights are 3263 kB in size.

Still, based on the evaluation results presented in Table 11.1, the CNN (modified) method demonstrates great performance in accurately identifying ELM events. It achieves high true positives, showing its effectiveness in correctly detecting ELM events. While it also maintains low false positives, indicating its ability to minimize incorrect detections. This is especially impressive, given the fact that the model used in the CNN method was not designed for the specific needs of this master's dissertation. Although it was designed as a peak detection method, the preprocessed data passed to the input layer is much more extensive and proves to be a great help in making this method also work for ELM detection.

One note, however, compared to the RNN is that the CNN is much more complex. The complexity of this model could result in overfitting. The lower results than some other methods, make it clear that this isn't the case.

11.12 Peak detection on other datasets

In order to assess the performance of the peak detection framework, two datasets were captured to obtain relevant time series information. The first dataset recorded the illuminance levels in Ghent over an eleven-day period. Correspondingly, the second dataset documented the outside temperature during the same timeframe. These datasets were chosen due to their distinct trends and served as valuable inputs for evaluating the peak detection methods within the framework. As both signals contain distinct trends, different parameters needed to be set for each signal within the framework. It is important to note that these signals were unlabeled and were solely used for visual demonstration purposes, showcasing the performance of the peak detection methods.

The advantages and disadvantages of the different ELM detection methods are also conveyed here for their peak detection counterparts.

Figure 11.3 illustrates how the MACD peak detection method is applied to the illuminance signal. Smaller peaks, such as the one between $x = 750$ and $x = 1000$, are grouped together.

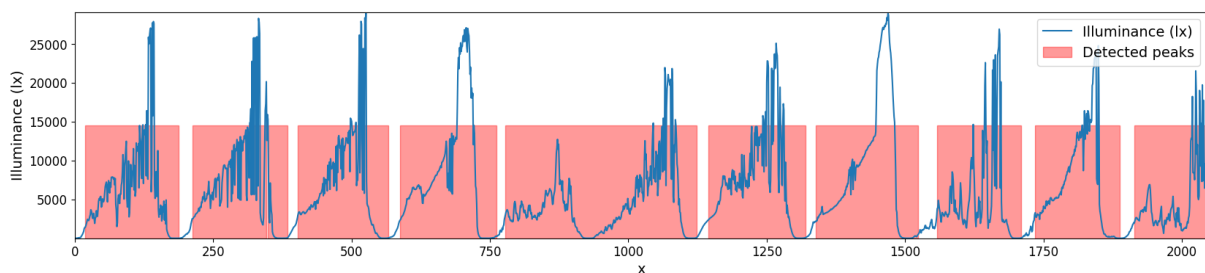


Figure 11.3: MACD peak detection on an illuminance signal.

On the other hand, Figure 11.4 demonstrates the application of the CustomKernel peak detection method to the illuminance signal, which yields superior results. This outcome aligns with expectations since the MACD method is highly sensitive to trends, as demonstrated by its ability to detect peaks in the reverse illuminance signal, as shown in Figure 11.5. Now each peak is detected individually, but the end detection is quite irregular.

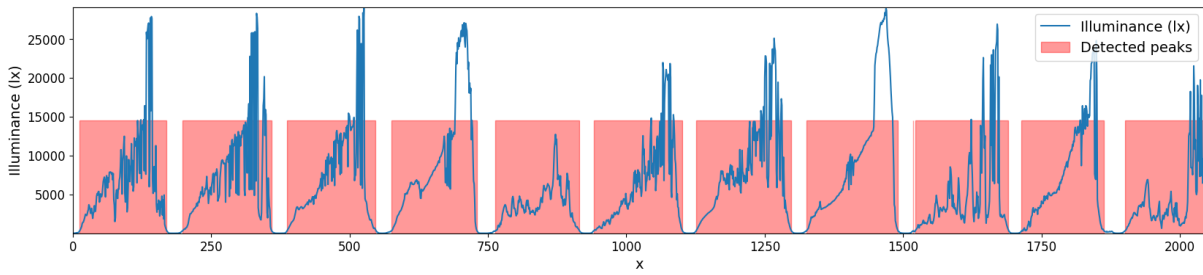


Figure 11.4: CustomKernel peak detection on an illuminance signal.

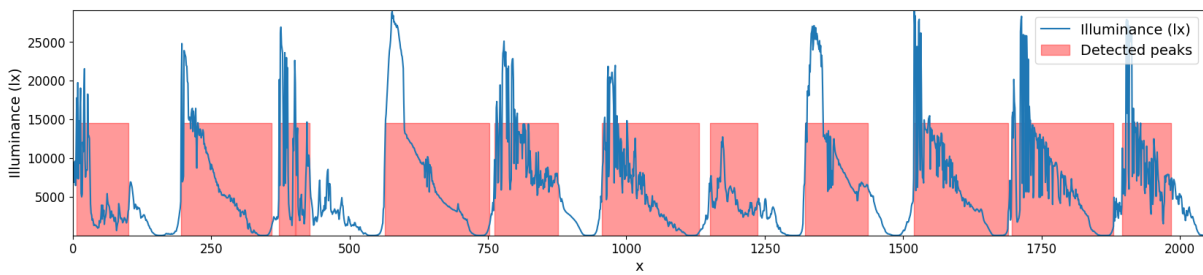


Figure 11.5: MACD peak detection on a flipped illuminance signal.

Figures 11.6 and 11.7 illustrate the results obtained by applying the CustomKernel and LoG peak detection methods to the temperature data. In the context of this particular signal, both methods achieve satisfactory results.

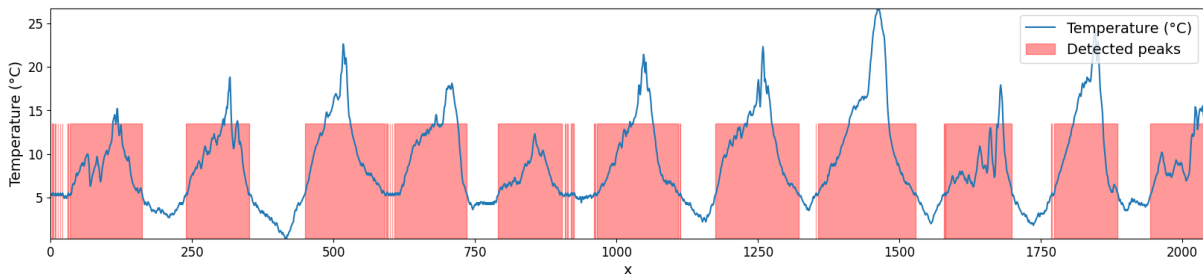


Figure 11.6: CustomKernel peak detection on a temperature signal.

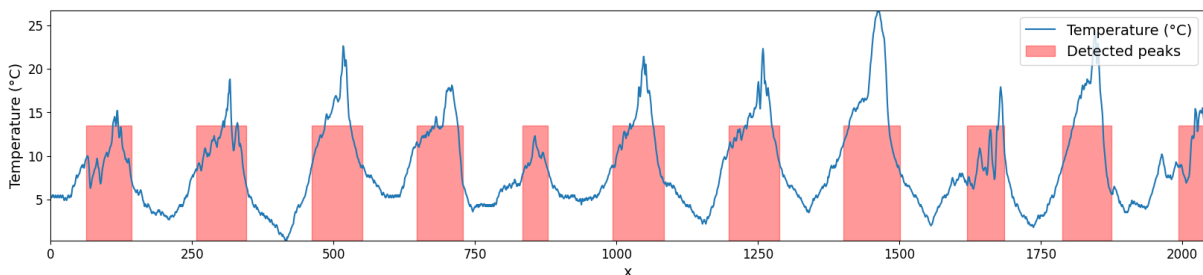


Figure 11.7: LoG peak detection on a temperature signal.

11.13 ELM detection on other data

The ELM detection methods were also tested on data from another fusion device, namely DIII-D [35]. However, only unlabeled light emission signals were available for this master's dissertation, and no energy signals were provided. To address this limitation, a fabricated energy signal was created, which linearly transitions from 1 to 0.

It should be noted that the presence of this artificial energy signal can adversely affect the performance of certain ELM detection methods, namely the LoGDoG, RNN, and CNN methods. Conversely, methods that employ the Savgol Gap drop algorithm from Section 8.2.3 are not significantly impacted, as they only exclude ELMs when the energy is increasing. However, if these methods were to enable the Savgol Gap merge algorithm, which is disabled by default, they would also be influenced by the fabricated energy signal.

To assess the robustness of the ELM detection methods, Figures 11.8, 11.9, 11.10, and 11.11 depict the ELM detection outcomes using the LoG, RNN (modified), CNN (modified), and CustomKernel methods, respectively. These figures provide a visual representation of the performance of each method with the default hyperparameters.

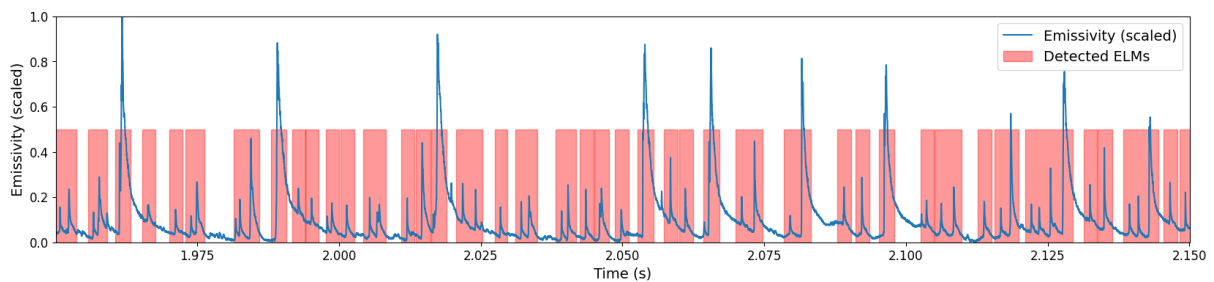


Figure 11.8: LoG ELM detection on an emissivity signal from DIII-D.

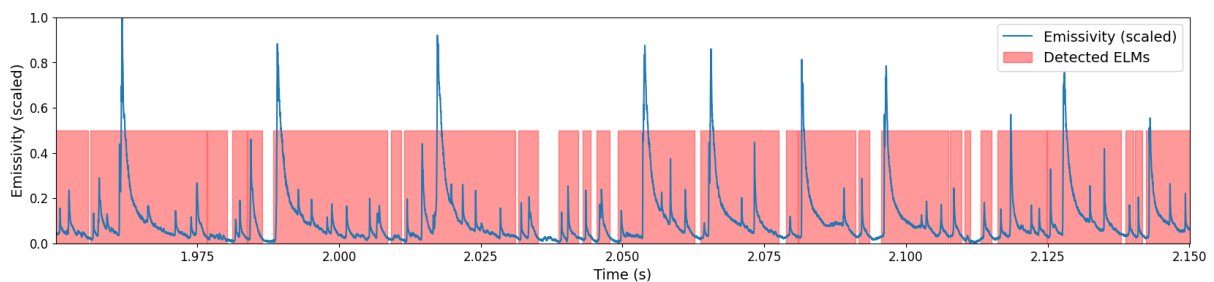


Figure 11.9: RNN (modified) ELM detection on an emissivity signal from DIII-D.

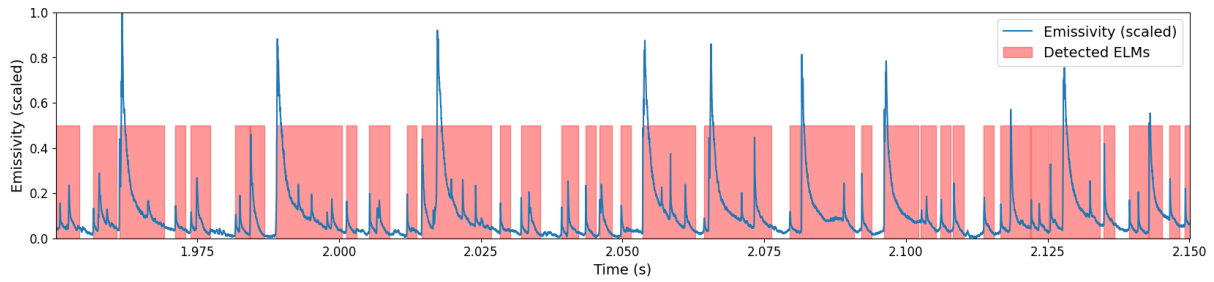


Figure 11.10: CNN (modified) ELM detection on an emissivity signal from DIII-D.

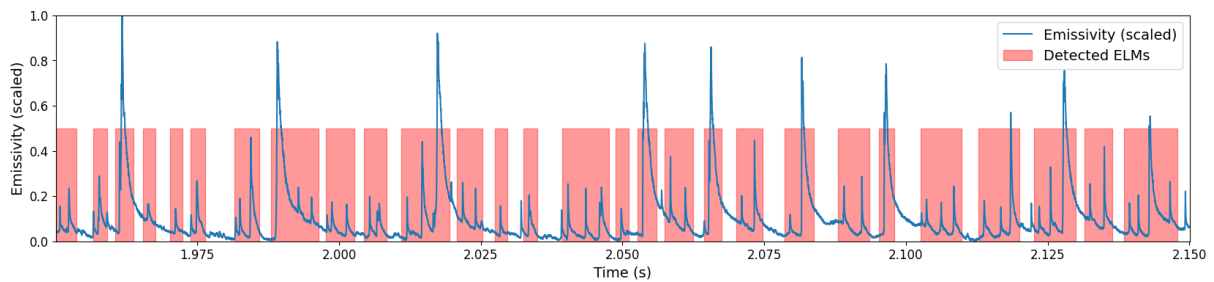


Figure 11.11: CustomKernel ELM detection on an emissivity signal from DIII-D.

11.14 Conclusion

The results show that all methods developed in this master's dissertation outperform the existing ROT method. It is also found that the deconvolution method is not suitable for the specific characteristics of ELMs encountered in this work.

Each method exhibits its own set of advantages and disadvantages. For instance, the RNN and CNN models are trained with specific parameters, which may require retraining when applied to datasets with different parameter requirements. The CustomKernel method receives the highest F_{β} -score and exhibits the lowest number of false positives. However, it is surpassed by the MACD, Rolling Z-score, LoG, and RNN methods in terms of true positive detection. The LoGDoG method demonstrates the potential to reduce the number of false positives compared to the LoG method. However, its robustness using the energy signal inadvertently leads to a decrease in the number of true positives. Overall, these findings highlight the trade-offs and varying performance characteristics of each method, underscoring the importance of selecting an appropriate method based on specific application requirements and priorities.

The results in Table 11.1 show that the rolling Z-score, LoG, LoGDoG, CustomKernel, and RNN (modified) methods emerge as the clear winners in terms of ELM detection performance. These methods exhibit high true positives, indicating their ability to accurately identify ELMs while maintaining low false positives, which suggests a minimal tendency to incorrectly label non-ELM events. The combination of these favorable characteristics positions these methods as strong contenders for effective ELM detection in the given context.

Conclusion

The primary aim of this master's dissertation was to develop reliable and accurate methods for detecting ELMs in tokamaks with machine learning. Throughout the research, each objective from Section 2.5 was successfully achieved, leading to significant advancements in ELM detection techniques;

To begin with, the study thoroughly explored a range of technologies and algorithms suitable for ELM detection, encompassing both traditional signal processing techniques and machine learning approaches. By considering diverse methodologies, the research aimed to identify the most effective strategies for detecting ELMs in tokamaks.

Secondly, a range of performance metrics were specifically developed to assess the effectiveness of ELM detection methods. These metrics facilitated comprehensive training and evaluation processes, tailored specifically to the task of detecting ELMs and their associated peaks as intervals.

Next, through meticulous experimentation and analysis, it was found that a combination of traditional signal processing, a smart custom implementation of metrics and machine learning, proved to be highly performant. Multiple approaches were explored, including the filters, such as LoG and DoG. These filters enabled the extraction of meaningful features from the light emissivity and energy data, capturing the unique characteristics of ELMs. These extracted features were then utilized in both thresholding algorithms and machine learning models to accurately detect ELMs.

Multiple ELM detection methods were developed, each employing different approaches, each with its own advantages and disadvantages. The performance of these methods was thoroughly evaluated and compared to existing approaches, allowing for a comprehensive assessment of their effectiveness.

In addition to the accomplishments mentioned earlier, it is important to highlight that for this master's dissertation a comprehensive framework has also been developed. This framework not only encompasses the detection of ELMs but also includes a peak detection library. Its modular design enables effortless integration and expansion, providing a flexible and adaptable platform for future research and advancements in the field.

The work plan outlined at the beginning of the master's dissertation included the implementation of existing methods for comparison with the ELM detection methods. However, certain planned options were replaced with alternative ELM detection methods due to the unavailability or lack of open-source access to the original methods.

Adapting to a new set of labels, which were provided six days before the final deadline of the master's dissertation, posed a significant challenge. These labels serve as the ground truth for the ELM detection methods. The new labels were based on different criteria for ELM detection, necessitating the retraining of the methods. Furthermore, due to the substantial dissimilarity in the labels, the performance of the methods underwent significant changes, resulting in a prolonged delay in completing the results section.

This master's dissertation addresses ELM detection challenges in tokamak reactors by developing robust methods. Through mathematical transformations and machine learning techniques applied to real JET data, the research contributes to nuclear

fusion research. These findings pave the way for efficient ELM research, advancing the goal of fusion energy realization.

Ethical and social reflection

ELM detection contributes to scientific and technological advancements and holds significant ethical and social implications. The most apparent implication is the contribution to the seventh Sustainable Development Goal (SDG).

The research aligns directly with SDG 7, which aims to ensure access to affordable and clean energy. ELMs can impact the performance and stability of fusion reactors. By developing effective ELM detection methods, fusion researchers can focus on analyzing the ELM events themselves instead of locating the ELMs in the data. This makes their research more efficient, thereby contributing to the goal of sustainable energy access.

Additionally, the master's dissertation also supports SDG 9, which focuses on promoting sustainable industries, fostering innovation, and developing an efficient infrastructure. ELM detection research helps technological innovation, enabling the optimization of fusion reactors and thus the development of reliable and efficient infrastructure.

SDG 12, which stands for responsible consumption and production, is also positively impacted by ELM detection, because ELMs can cause damage to tokamak walls, resulting in material erosion and more waste. A good ELM detection algorithm enables researchers to mitigate the negative impact of ELMs and thus minimizes the waste of expensive and rare materials used in the fusion device walls.

The calls for urgent action to combat climate change, which is SDG 13, is also partially addressed by the fusion research that ELM detection contributes to. Fusion research tries to develop climate-friendly energy solutions and works toward a sustainable low-carbon future.

Finally, the master's dissertation emphasizes the importance of collaboration and partnerships, which aligns with SDG 17. Progress in fusion research relies on exchanging knowledge, resources, and expertise among researchers and institutions worldwide. Fostering this accelerates the road to achieving the other SDGs. The methods proposed by this master's dissertation are trained on JET data. JET research is important for the functioning of ITER since they use similar plasma configurations, which makes these methods applicable to ITER. ITER is a huge multinational project, where exchanging knowledge and resources is essential.

In conclusion, the master's dissertation not only contributes to scientific and technological advancements but also holds positive ethical and social implications. ELM detection helps to achieve multiple SDGs, particularly those related to the development of affordable and clean energy

References

- [1] Sunflowers. Licensed under CC0 1.0 Universal (CC0 1.0). [Online]. Available: <https://cleanpublicdomain.com/downloads/sunflowers-2/>
- [2] H. Zohm, "Edge localized modes (elms)," *Plasma Physics and Controlled Fusion*, vol. 38, no. 2, p. 105, feb 1996. [Online]. Available: <https://dx.doi.org/10.1088/0741-3335/38/2/001>
- [3] Verdoolaege, Geert and Frassinetti, L., "Statistical analysis of edge-localized mode timing in JET," in *Proceedings of the 45th EPS Conference on Plasma Physics*. European Physical Society, 2018, p. 4. [Online]. Available: <http://ocs.ciemat.es/EPS2018ABS/pdf/P2.1078.pdf>
- [4] H. Guo, G. Matthews, I. Coffey, S. Erents, M. Groth, P. Harbour, M. Von Hellermann, D. Hillis, J. Hogan, L. Horton, L. Ingesson, K. Lawson, J. Lingertat, C. Maggi, G. McCracken, R. Monk, P. Morgan, M. Stamp, P. Stangeby, and G. Vlases, "Effects of divertor geometry and chemical sputtering on impurity behaviour and plasma performance in jet," *Nuclear Fusion*, vol. 40, p. 379, 05 2002.
- [5] L. Wang, G. Xu, H. Guo, H. Wang, S. Liu, K. Gan, X. Gong, Y. Liang, N. Yan, L. Chen, J. Liu, W. Zhang, R. Chen, L. Shao, H. Xiong, J. Qian, B. Shen, G. Liu, R. Ding, X. Zhang, C. Qin, S. Ding, L. Xiang, G. Hu, Z. Wu, G. Luo, J. Chen, L. Hu, X. Gao, B. Wan, J. Li, and the EAST Team, "Characterizations of power loads on divertor targets for type-i, compound and small elms in the east superconducting tokamak," *Nuclear Fusion*, vol. 53, no. 7, p. 073028, jun 2013. [Online]. Available: <https://dx.doi.org/10.1088/0029-5515/53/7/073028>
- [6] J. Sen, Ed., *Machine Learning - Algorithms, Models and Applications*. IntechOpen, dec 2021. [Online]. Available: <https://doi.org/10.5772/intechopen.94615>
- [7] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance," 2022.
- [8] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.
- [9] "scikit-image blob detection," https://scikit-image.org/docs/0.21.x/auto_examples/features_detection/plot_blob.html, scikit-image, accessed on 2023-06-05.
- [10] M. Lind, "Deep dive into python mixins and multiple inheritance," 2019, accessed: 2023-06-04. [Online]. Available: <https://coderbook.com/@marcus/deep-dive-into-python-mixins-and-multiple-inheritance/>
- [11] R. Hyndman, *Moving Averages*. Springer Berlin, Heidelberg, 01 2010, pp. 866–869.
- [12] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

11 References

- [13] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: <https://doi.org/10.1038/s41586-020-2649-2>
- [14] F. Klinker, "Exponential moving average versus moving exponential average," *Mathematische Semesterberichte*, vol. 58, no. 1, pp. 97–107, dec 2010. [Online]. Available: <https://doi.org/10.1007/s00591-010-0080-8>
- [15] S. A. Kassam and T. L. Lim, "Robust wiener filters," *Journal of the Franklin Institute*, vol. 304, no. 4, pp. 171–185, 1977. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0016003277900114>
- [16] A. M. Kabe and B. H. Sako, "Chapter 5 - analysis of continuous and discrete time signals," in *Structural Dynamics Fundamentals and Advanced Applications*, A. M. Kabe and B. H. Sako, Eds. Academic Press, 2020, pp. 271–427. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128216156000058>
- [17] H. Hsu, *Schaum's Outline of Signals and Systems, Fourth Edition*. McGraw Hill LLC, 2019. [Online]. Available: <https://books.google.be/books?id=8wWSDwAAQBAJ>
- [18] E. W. Weisstein, "Gradient," <https://mathworld.wolfram.com/Gradient.html>, accessed on 2023-06-01.
- [19] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: <https://doi.org/10.1021/ac60214a047>
- [20] C. Andrade, "Z scores, standard scores, and composite test scores explained," *Indian Journal of Psychological Medicine*, vol. 43, no. 6, pp. 555–557, 2021, PMID: 35210687. [Online]. Available: <https://doi.org/10.1177/02537176211046525>
- [21] G. Appel, *The moving average convergence-divergence trading method: advanced version*. Scientific Investment Systems, 1985.
- [22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [23] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. [Online]. Available: http://learningsys.org/nips17/assets/papers/paper_16.pdf
- [24] "Opencv: Smoothing images," https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html, OpenCV, accessed on 2023-06-05.
- [25] N. Snavely. Cs6670: Computer vision. Cornell University. Slides 35-45. [Online]. Available: https://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02_filter.pdf
- [26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale

11 References

- machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [27] "Cuda toolkit documentation 12.1," <https://docs.nvidia.com/cuda/archive/12.1.0/>, NVIDIA, accessed on 2023-06-07.
- [28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.
- [29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>
- [30] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," 2022.
- [31] A. Kensert, E. Bosten, G. Collaerts, K. Efthymiadis, P. Van Broeck, G. Desmet, and D. Cabooter, "Convolutional neural network for automated peak detection in reversed-phase liquid chromatography," *Journal of Chromatography A*, vol. 1672, p. 463005, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021967322002035>
- [32] S. Ahmed, O. E. Garcia, and A. Theodorsen, "Reconstruction of intermittent data time series as a superposition of pulses," 2023.
- [33] —, "Deconvolution methods," https://github.com/uit-cosmo/fpp-analysis-tools/blob/fd2edaacf41df7d04ec2950e444ed0f0fb607f66/fppanalysis/deconvolution_methods.py, accessed on 2023-06-06.
- [34] "Scipy find peaks documentation," https://docs.scipy.org/doc/scipy-1.10.1/reference/generated/scipy.signal.find_peaks.html, SciPy, accessed on 2023-06-06.
- [35] "Diii-d national fusion facility | general atomics," <https://www.ga.com/magnetic-fusion/diii-d>, General Atomics, accessed on 2023-06-06.
- [36] A. Bahrini, M. Khamoshifar, H. Abbasimehr, R. J. Riggs, M. Esmaeili, R. M. Majdabadkohne, and M. Pasehvar, "Chatgpt: Applications, opportunities, and threats," 2023.
- [37] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, Z. Ming, and Jiang, "Github copilot ai pair programmer: Asset or liability?" 2023.

Appendices

Appendix A: Usage of AI

With the emergence of advanced language models like ChatGPT [36] and Github CoPilot [37], Artificial Intelligence (AI) has gained significant prominence in software development and data research. While AI can be a valuable tool, it is important to acknowledge its limitations, such as occasional hallucinations and inadequate source referencing. Students need to use AI responsibly and ethically. Consequently, the use of AI was permitted in this master's dissertation; however, to ensure transparency, a supplementary attachment is included. This attachment provides detailed explanations of the implementation of AI and offers sample prompts to illustrate its usage.

Research

Bing Chat was often used in the research process to find relevant papers. Prompts like the following were used:

"I'm doing research about the data quality for machine learning. Could you link me some relevant papers, with a summary for each of them?"

Bing Chat would respond with:

"I found a paper on arxiv.org titled "The Effects of Data Quality on Machine Learning Performance" by Lukas Budach et al. [1] The paper explores empirically the relationship between six of the traditional data quality dimensions and the performance of fifteen widely used machine learning (ML) algorithms covering the tasks of classification, regression, and clustering, with the goal of explaining their performance in terms of data quality. The paper concludes with an extensive discussion of their observations.

Another paper I found is "Data Quality Toolkit: Automatic assessment of data quality and remediation for machine learning" by Saurabh Gupta et al. [2] The authors attempt to re-look at the data quality issues in the context of building a machine learning pipeline and build a tool that can detect, explain and remediate issues in the data, and systematically and automatically capture all the changes applied to the data.

Lastly, I found a paper on SpringerLink titled "Machine Learning: Algorithms, Real-World Applications and Research Directions" by S. Suresh et al. [3] The paper provides a comprehensive view on various types of machine learning algorithms that can be applied to enhance the intelligence and the capabilities of an application. I hope this helps!

[1]: <https://arxiv.org/abs/2207.14529>

[2]: <https://arxiv.org/abs/2108.05935>

[3]: <https://link.springer.com/article/10.1007/s42979-021-00592-x> "

The advantage here above a normal search engine is that multiple summaries can be provided within the same response. As a result, the researcher does not have to read the abstract of every paper returned by a normal search engine, and finding a paper can take less time than before.

Code

To reduce the time spend reading documentation of libraries. Simple prompts like the following were used:

"I have a time series in python called 'arr'. How can I calculate the the exponential moving average?"

To which, ChatGPT responds with:

To calculate the exponential moving average (EMA) of a time series in Python, you can use the pandas library, which provides a convenient function called `ewm()`.

Here's an example of how you can calculate the exponential moving average using pandas:

```
import pandas as pd
# Assuming 'arr' is a list or a pandas Series containing your time series data
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# Convert the list to a pandas Series (if it's not already)
arr_series = pd.Series(arr)
# Calculate the exponential moving average with a smoothing factor of 0.5
ema = arr_series.ewm(alpha=0.5).mean()
# Print the exponential moving average
print(ema)
```

In this example, we use a smoothing factor (also known as the decay factor or alpha) of 0.5. You can adjust this value based on your requirements and the desired level of smoothness for the EMA.

The `ewm()` function applies the exponential weighting to the data, and the `mean()` function calculates the moving average. The resulting `ema` variable will contain the exponential moving average of the `arr` time series.

Note that you may need to install the pandas library if you haven't done so already. You can install it using `pip`:

```
pip install pandas
```

Of course, everything that ChatGPT returned was thoroughly tested, to make sure that the code it provided functioned correctly. The parts used from the response are also modified to fit into the code base of the master's dissertation. The code below shows how its response was implemented.

```

class ExponentialMovingAverageMixin(BorderMixin):
    def ema(self, arr, span: int) -> np.ndarray:
        # Apply border padding to input array
        data = self.border_same_value(arr, span)
        # Calculate EMA using pandas library
        return self.ema_pandas(data, span)[span:-span]

    def ema_pandas(self, arr, span: int) -> np.ndarray:
        # Calculate EMA using pandas library
        return pd.Series(arr).ewm(span=span, adjust=False).mean().values

    def ema_numpy(self, arr, span: int) -> np.ndarray:
        # Calculate alpha value using span parameter
        alpha = 2 / (span + 1)
        # Initialize output array with zeros
        s = np.zeros_like(arr)
        # Set first value of output array to first value of input array
        s[0] = arr[0]
        # Loop over input array and calculate EMA using previous value and alpha
        for i in range(1, len(arr)):
            s[i] = (alpha * arr[i]) + ((1 - alpha) * s[i - 1])
        return s

```

Documentation

To facilitate the writing of documentation, ChatGPT was used, but only to provide a rudimentary draft. For example, the code above shows a Python class that provides two implementations for calculating an exponential moving average using a span. An effort was made to explain the code with comments, but the documentation could be expanded further. To help with this, the following prompt was sent to ChatGPT:

"Can you add docstrings and comments for me to this code? If possible, add the typedefs in the docs. For this keep in mind what the libraries their functions return. If you see anything that you want to improve, add an extra codeblock with your proposed changes.

« original undocumented code here »"

The output documentation, supplied by ChatGPT was used as a draft, as it does not know the context of the whole code base and can therefore make wrong assumptions. This means that the output is used more as an inspiration. The final version of the code is now:

```

class ExponentialMovingAverageMixin(BorderMixin):
    """

```

Mixin class that provides functions to calculate the exponential moving average of a given array.

Attributes:

Inherits attributes from BorderMixin class.

"""

```
def ema(self, arr, span: int) -> np.ndarray:
```

"""

Calculates the exponential moving average of the input array.

Args:

arr (np.ndarray): Input array to calculate EMA from.

span (int): The span parameter for the EMA calculation.

Returns:

np.ndarray: The exponential moving average of the input array.

"""

Apply border padding to input array

```
data = self.border_same_value(arr, span)
```

Calculate EMA using pandas library

```
return self.ema_pandas(data, span)[span:-span]
```

```
def ema_pandas(self, arr, span: int) -> np.ndarray:
```

"""

Calculates the exponential moving average of the input array using pandas.

Args:

arr (np.ndarray): Input array to calculate EMA from.

span (int): The span parameter for the EMA calculation.

Returns:

np.ndarray: The exponential moving average of the input array.

"""

Calculate EMA using pandas library

```
return pd.Series(arr).ewm(span=span, adjust=False).mean().values
```

```
def ema_numpy(self, arr, span: int) -> np.ndarray:
```

"""

Calculates the exponential moving average of the input array using numpy.

Args:

arr (np.ndarray): Input array to calculate EMA from.

span (int): The span parameter for the EMA calculation.

Returns:

np.ndarray: The exponential moving average of the input array.

"""

Calculate alpha value using span parameter

alpha = 2 / (span + 1)

Initialize output array with zeros

s = np.zeros_like(arr)

Set first value of output array to first value of input array

s[0] = arr[0]

Loop over input array and calculate EMA using previous value and alpha

for i in range(1, len(arr)):

*s[i] = (alpha * arr[i]) + ((1 - alpha) * s[i - 1])*

return s

Scription

To help the writing process of the master's dissertation, ChatGPT was regularly asked things like the following:

"I'm writing a paper about ELM detection for my master's dissertation. In this paper, there is a chapter about filters. This is the introduction to that chapter:

« original chapter introduction here »

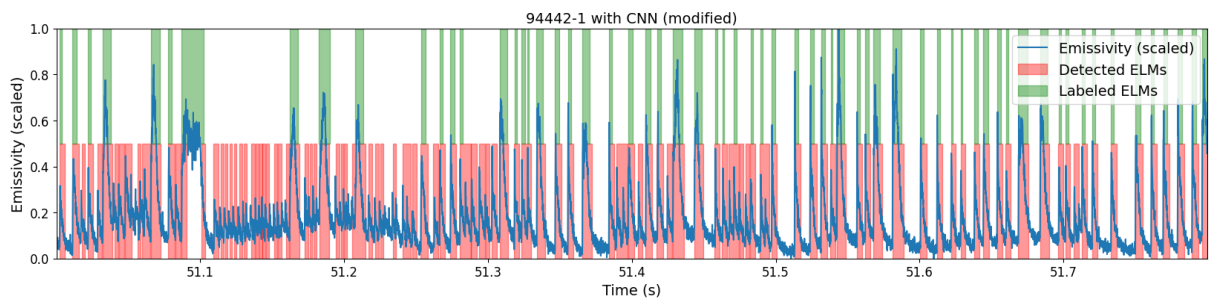
What would you improve about this introduction?"

A prompt like this would trigger ChatGPT to return an improved version. However, the response is never perfect, which is why it is only used as a guideline. Some improved sentences may be taken, but never the whole response.

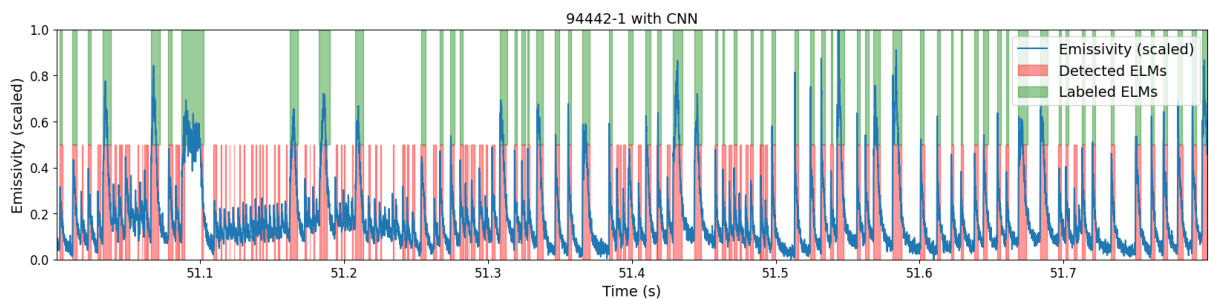
Appendix B: Applying ELM detection methods to shots

This section applies various ELM detection methods to a selection of shots to showcase their respective strengths and weaknesses.

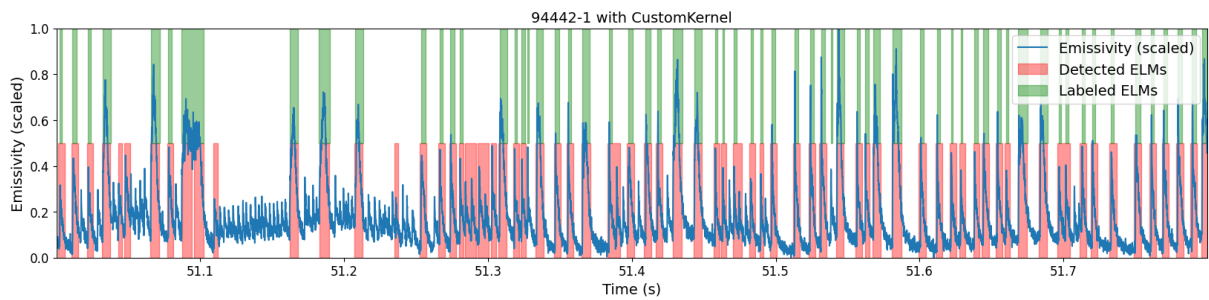
Shot 94442-1



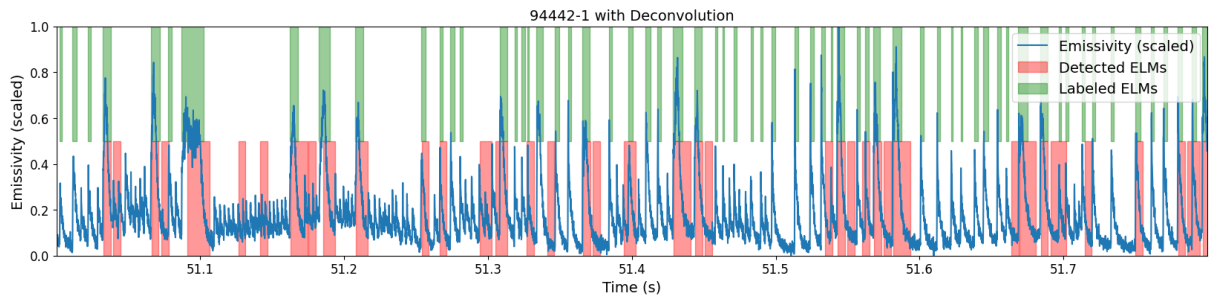
The CNN (modified) method applied on shot 94442-1.



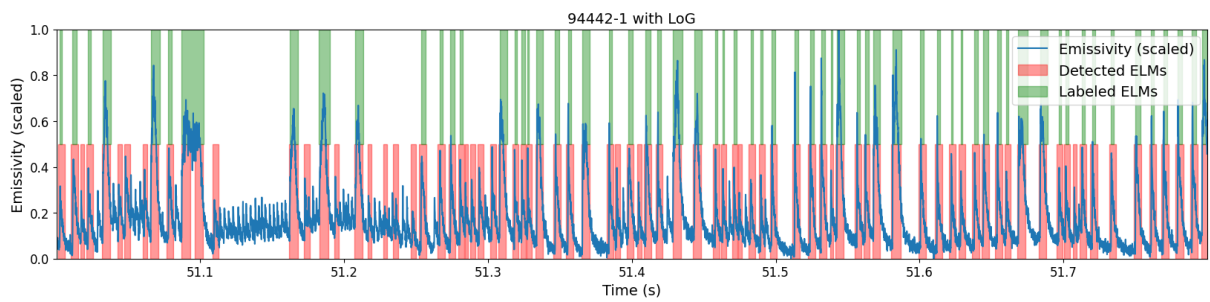
The CNN method applied on shot 94442-1.



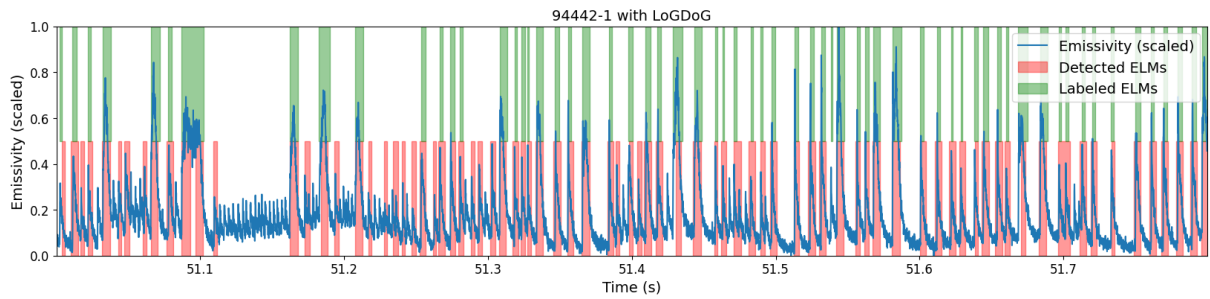
The CustomKernel method applied on shot 94442-1.



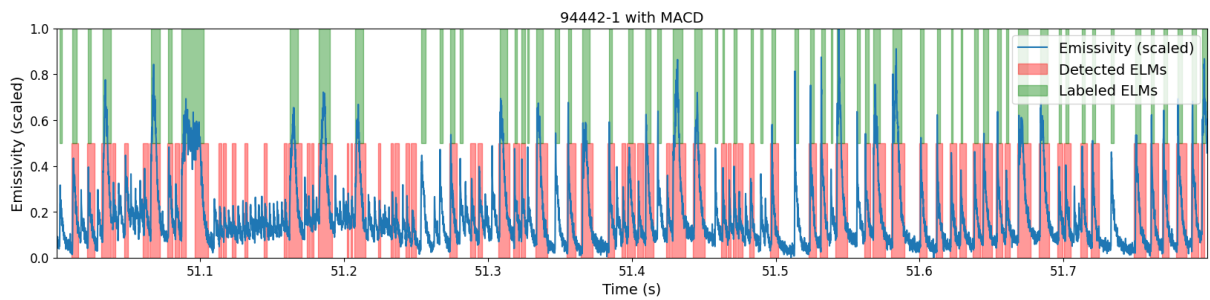
The Deconvolution method applied on shot 94442-1.



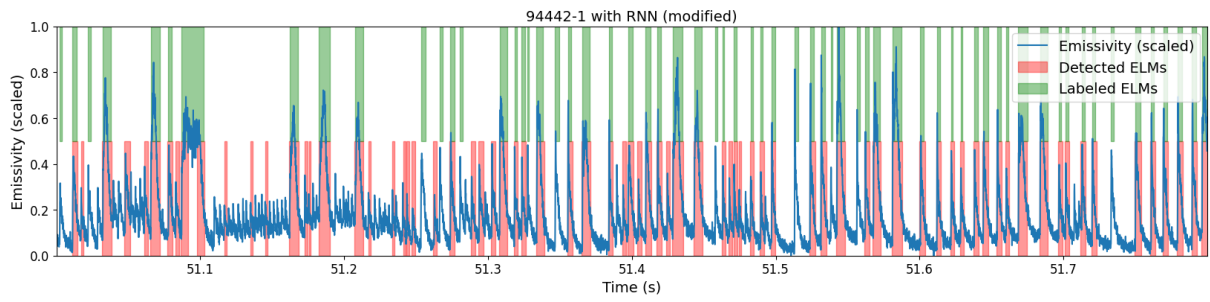
The LoG method applied on shot 94442-1.



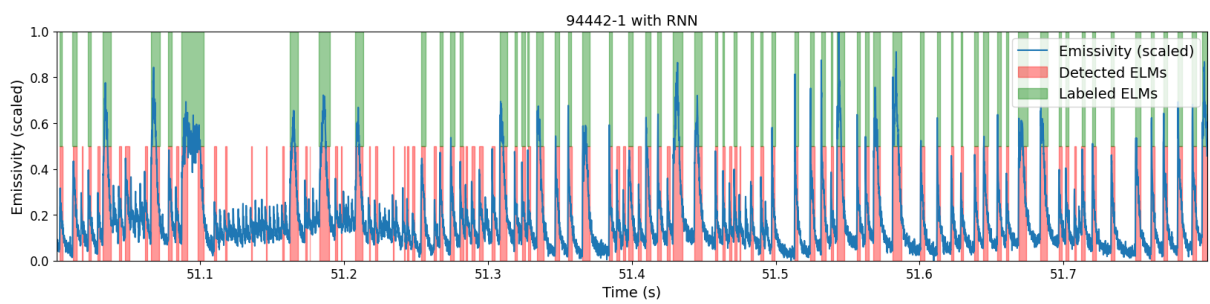
The LoGDoG method applied on shot 94442-1.



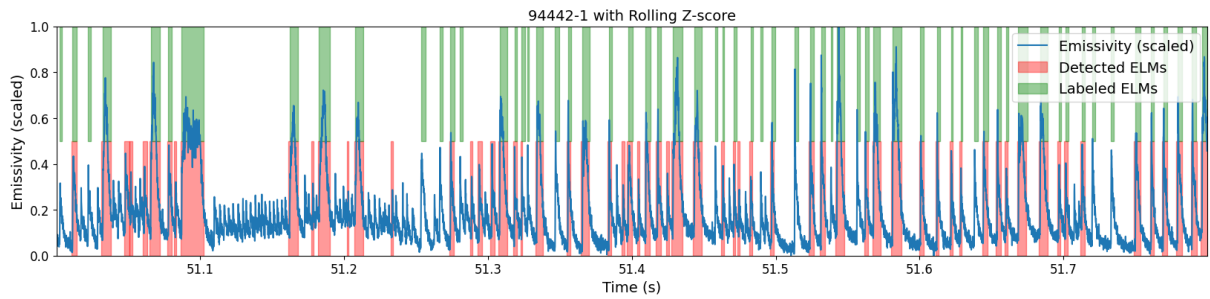
The MACD method applied on shot 94442-1.



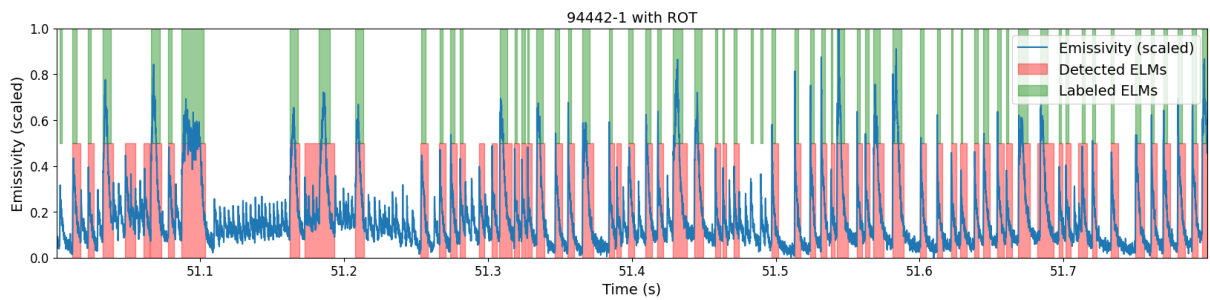
The RNN (modified) method applied on shot 94442-1.



The RNN method applied on shot 94442-1.

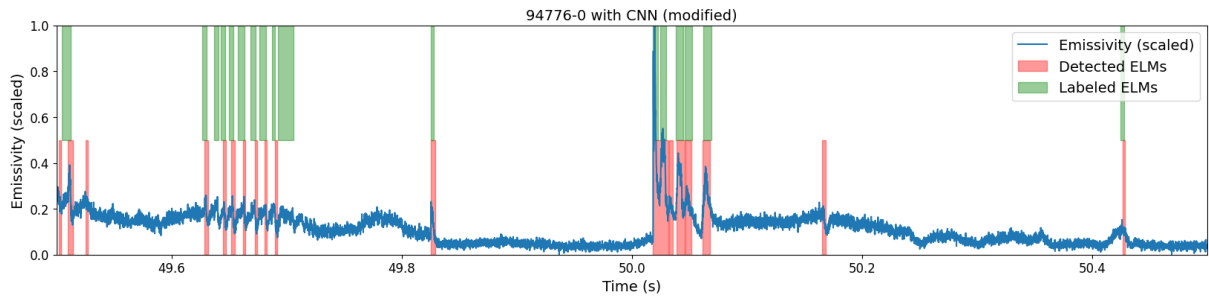


The Rolling Z-score method applied on shot 94442-1.

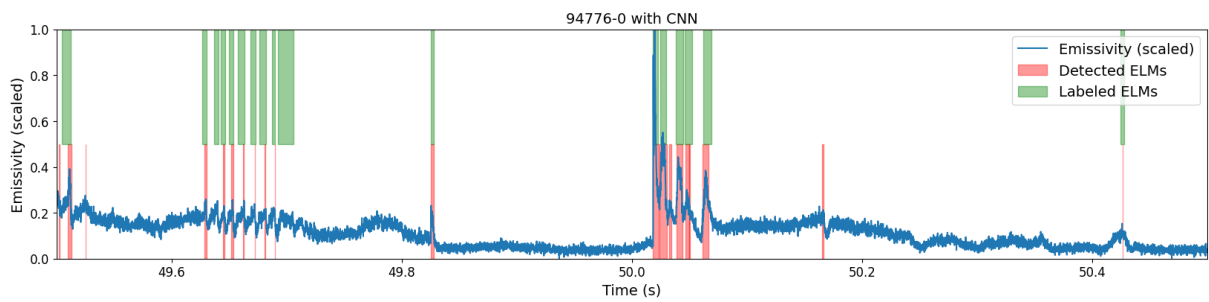


The ROT method applied on shot 94442-1.

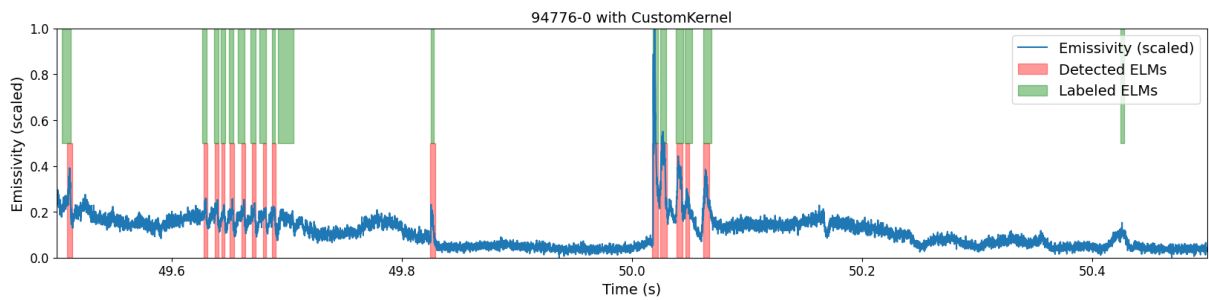
Shot 94776-0



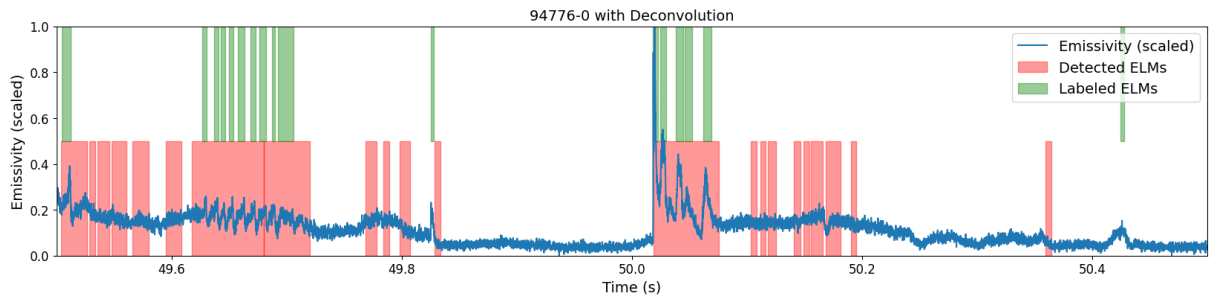
The CNN (modified) method applied on shot 94776-0.



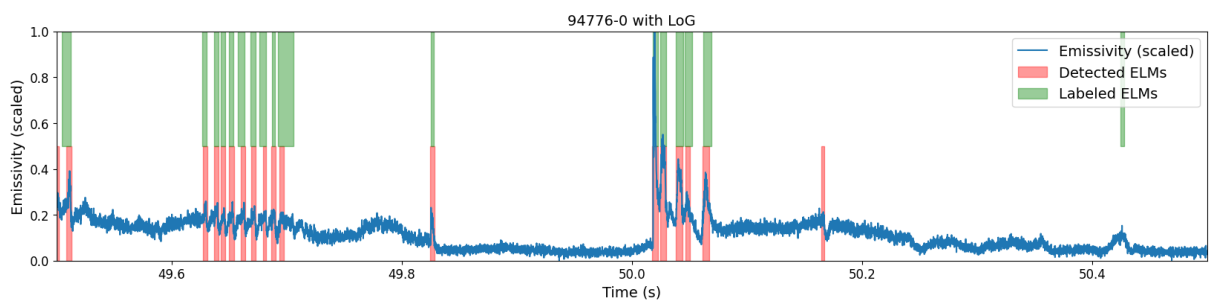
The CNN method applied on shot 94776-0.



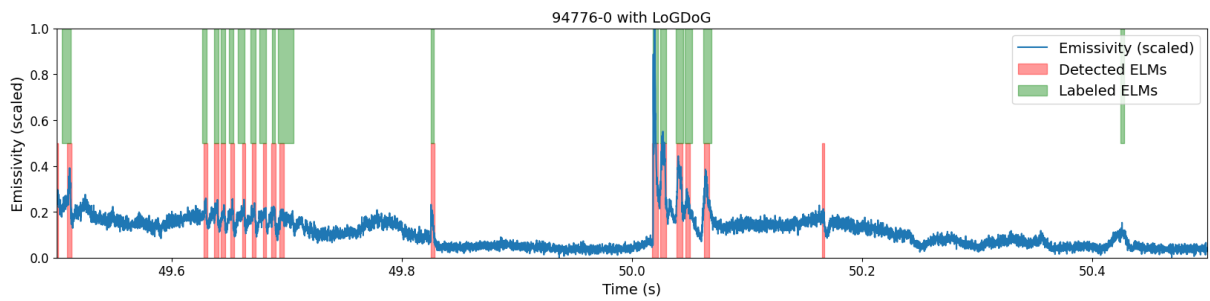
The CustomKernel method applied on shot 94776-0.



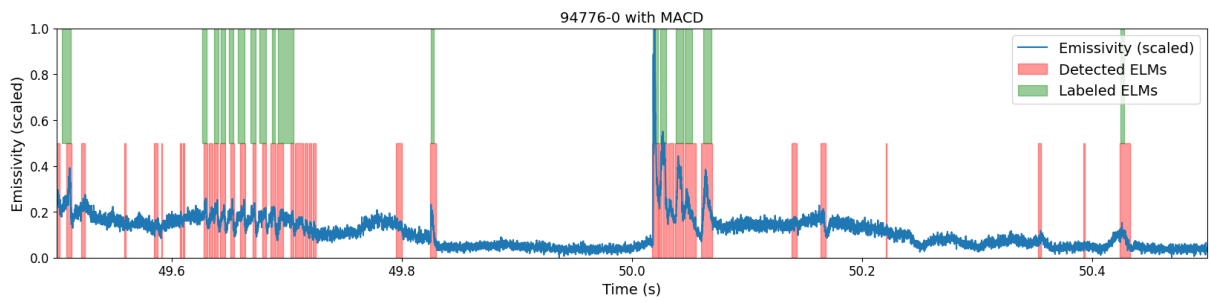
The Deconvolution method applied on shot 94776-0.



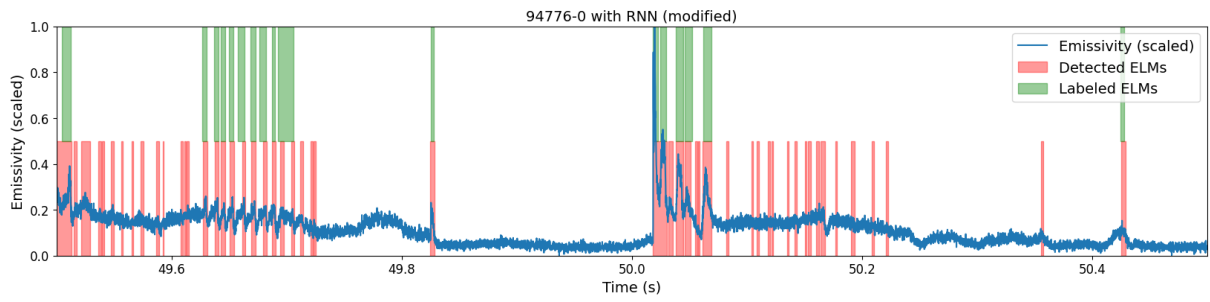
The LoG method applied on shot 94776-0.



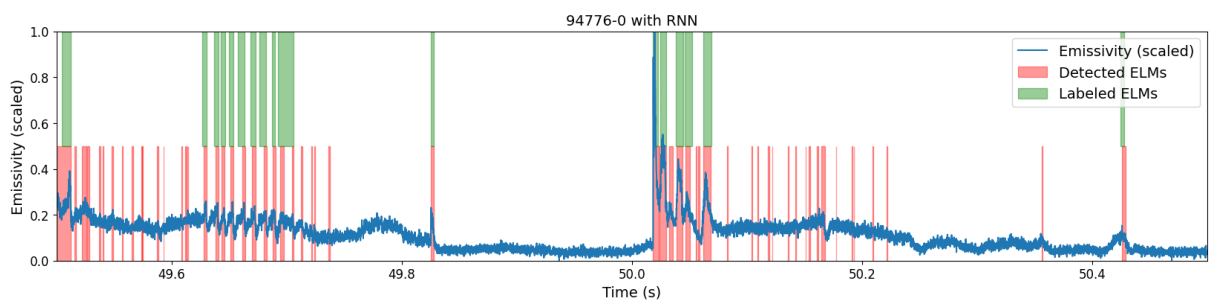
The LoGDoG method applied on shot 94776-0.



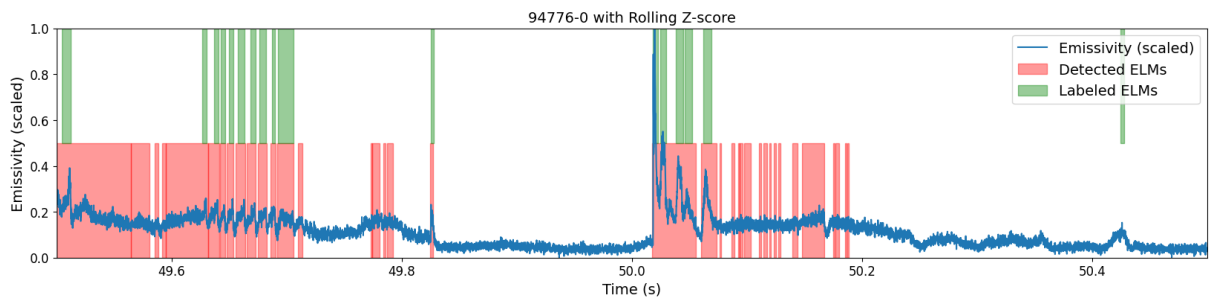
The MACD method applied on shot 94776-0.



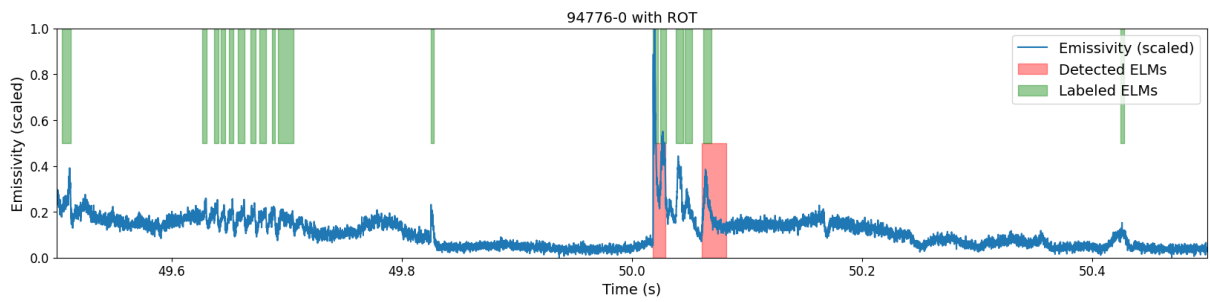
The RNN (modified) method applied on shot 94776-0.



The RNN method applied on shot 94776-0.

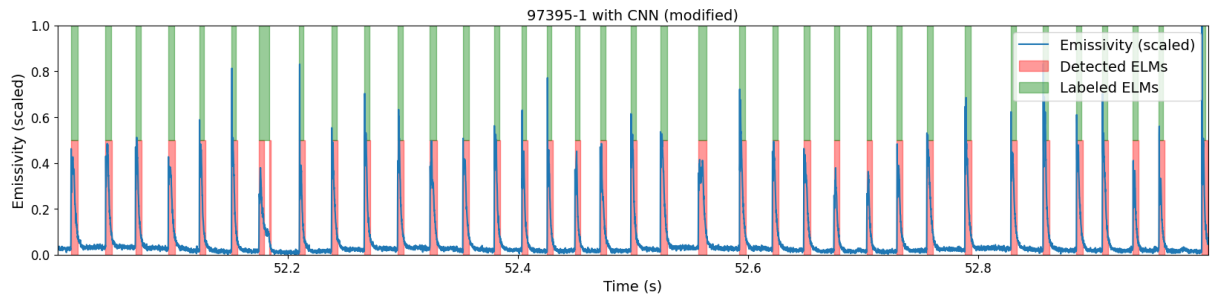


The Rolling Z-score method applied on shot 94776-0.

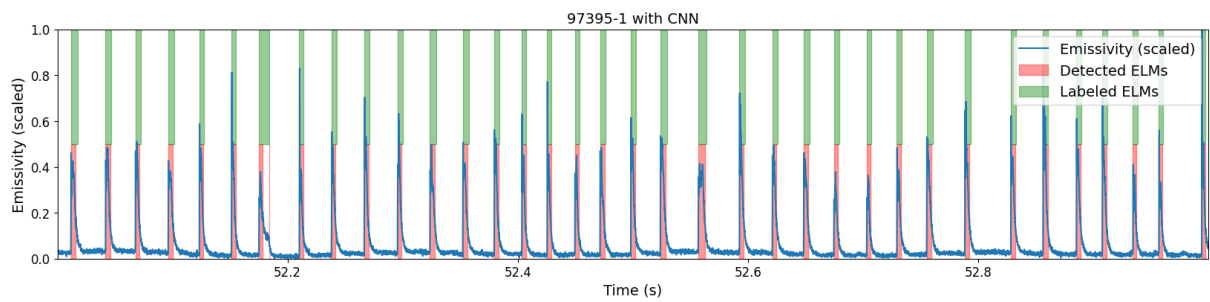


The ROT method applied on shot 94776-0.

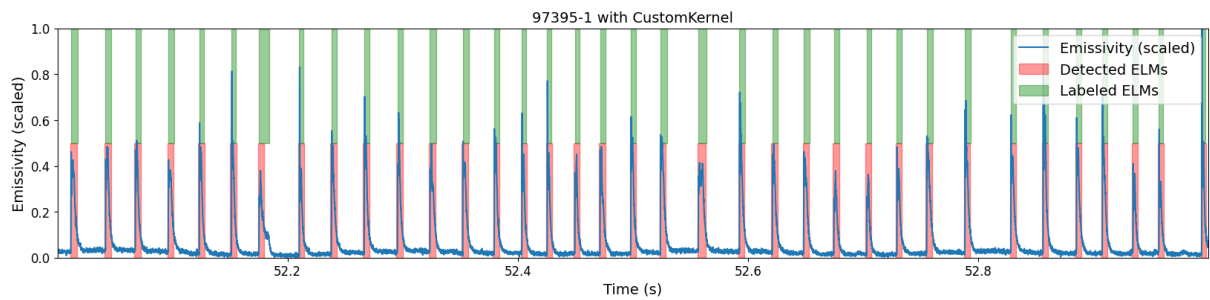
Shot 97395-1



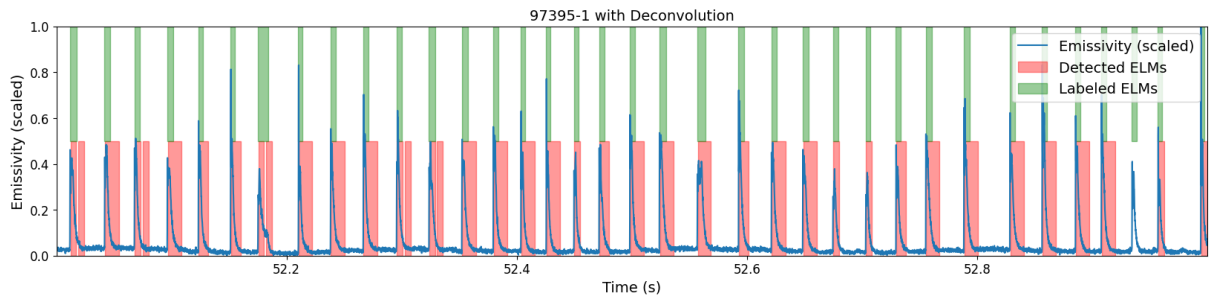
The CNN (modified) method applied on shot 97395-1.



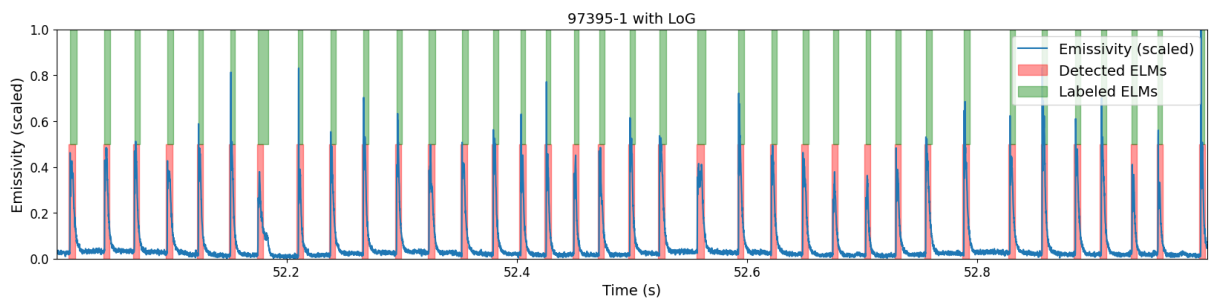
The CNN method applied on shot 97395-1.



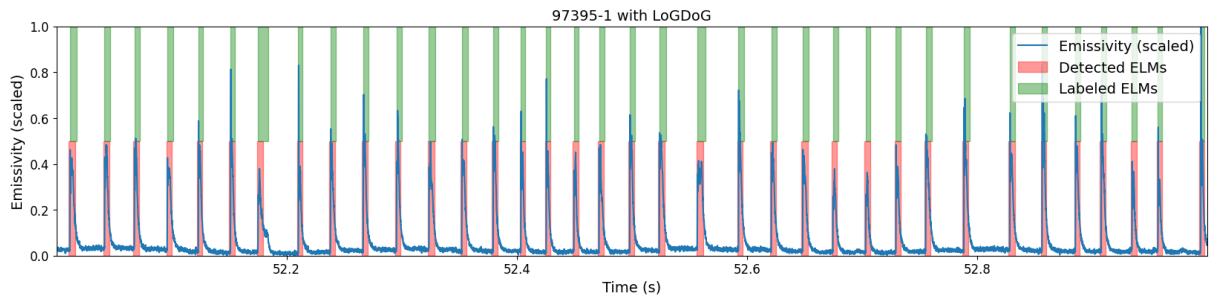
The CustomKernel method applied on shot 97395-1.



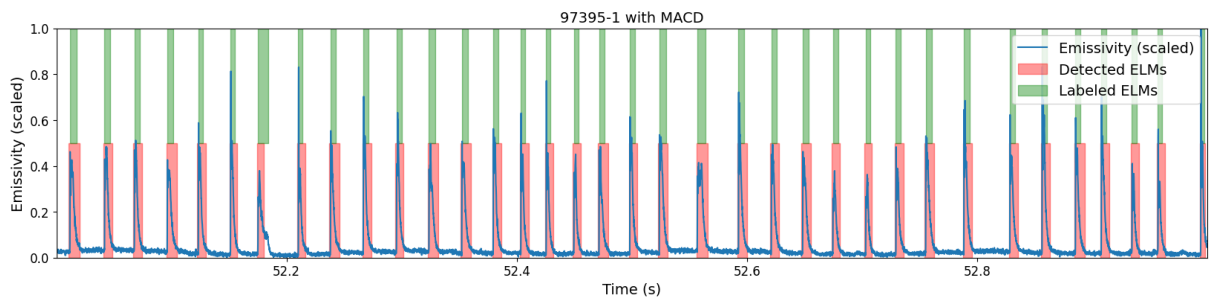
The Deconvolution method applied on shot 97395-1.



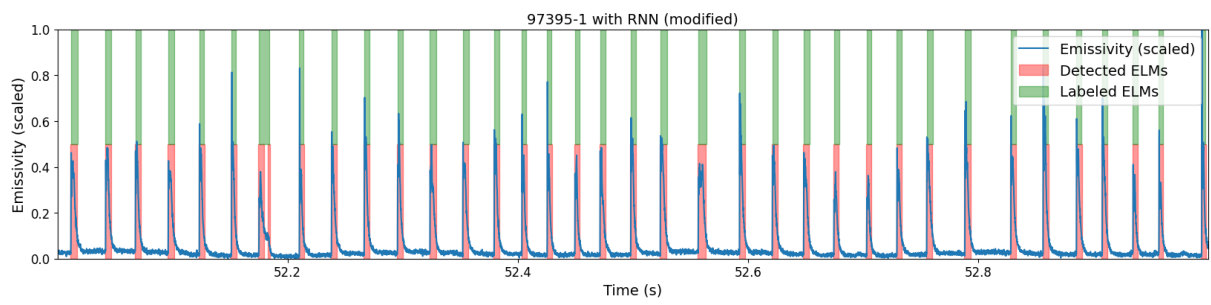
The LoG method applied on shot 97395-1.



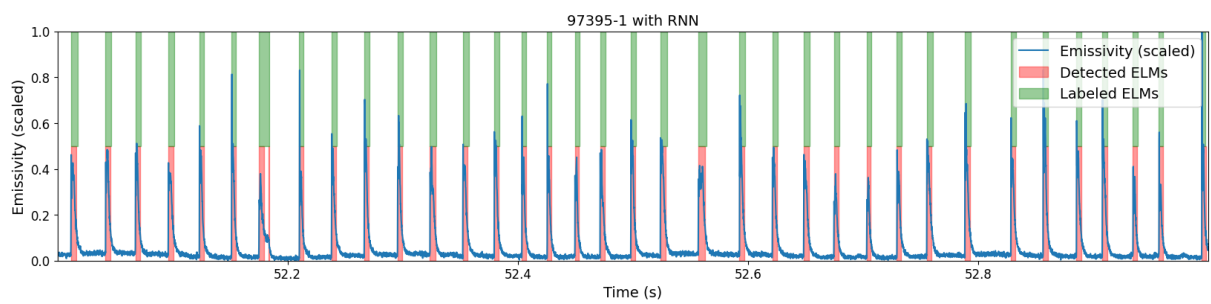
The LoGDoG method applied on shot 97395-1.



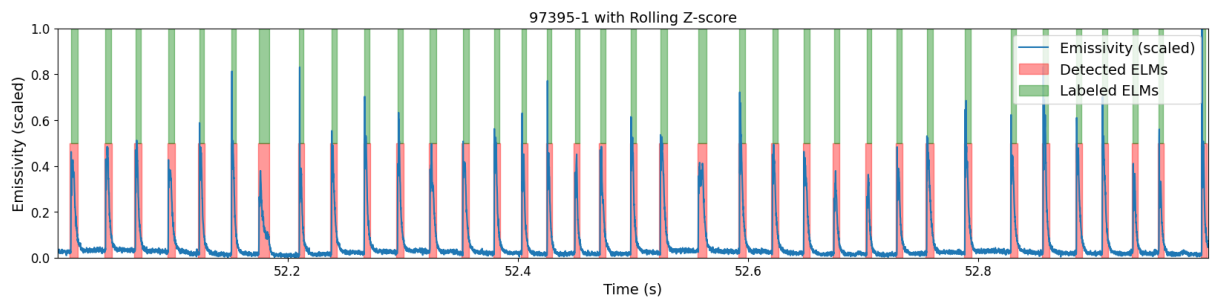
The MACD method applied on shot 97395-1.



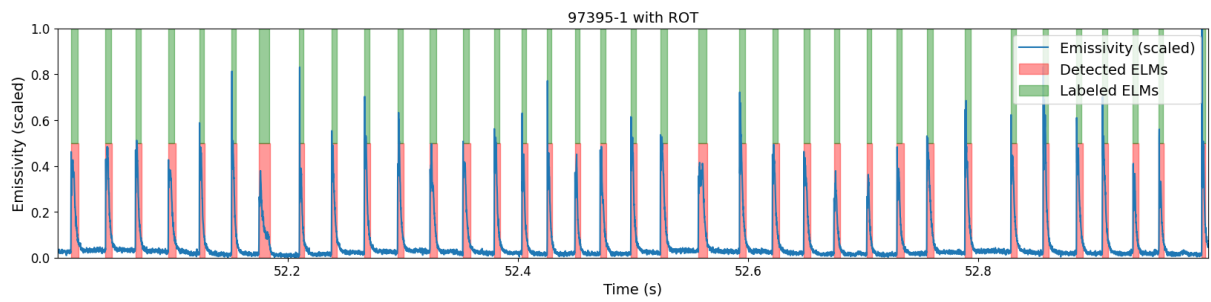
The RNN (modified) method applied on shot 97395-1.



The RNN method applied on shot 97395-1.

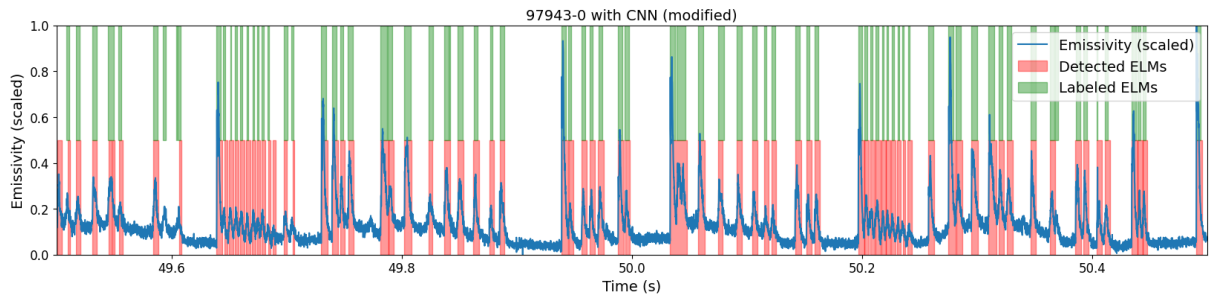


The Rolling Z-score method applied on shot 97395-1.

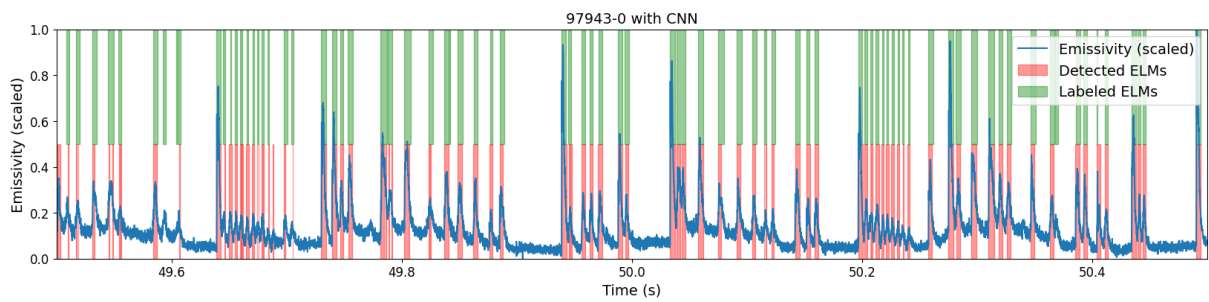


The ROT method applied on shot 97395-1.

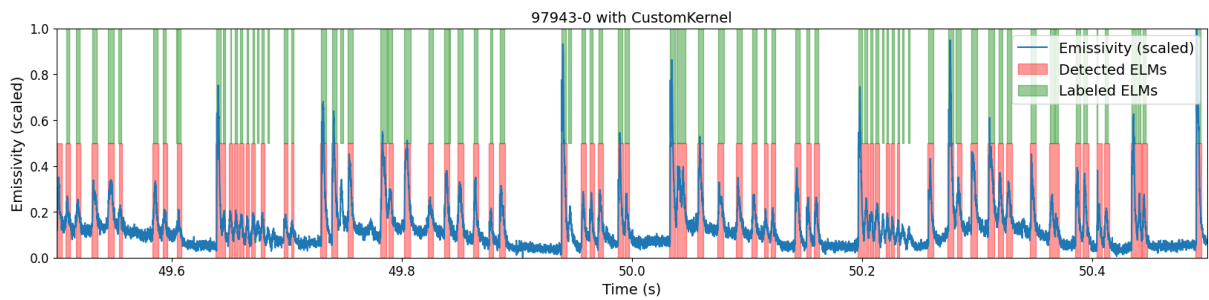
Shot 97343-0



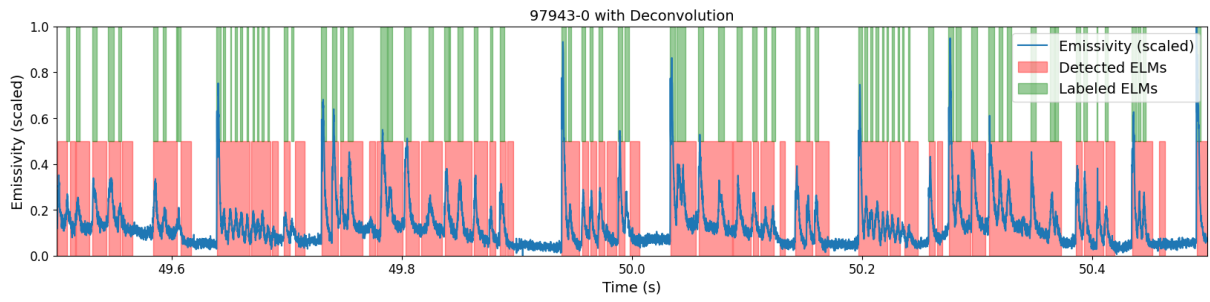
The CNN (modified) method applied on shot 97943-0.



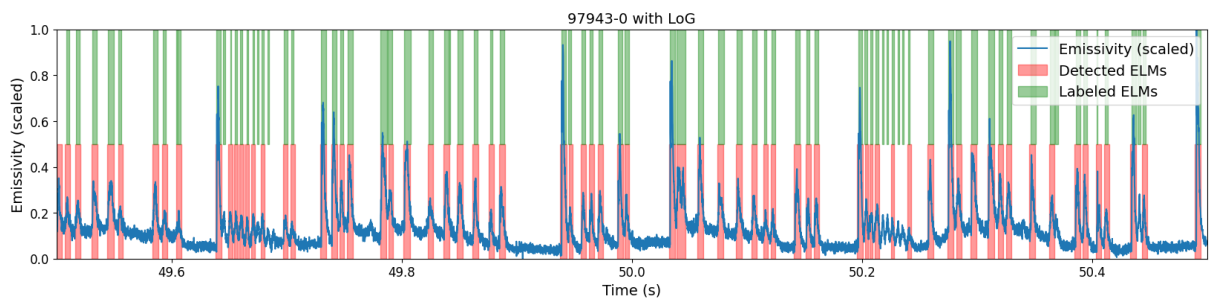
The CNN method applied on shot 97943-0.



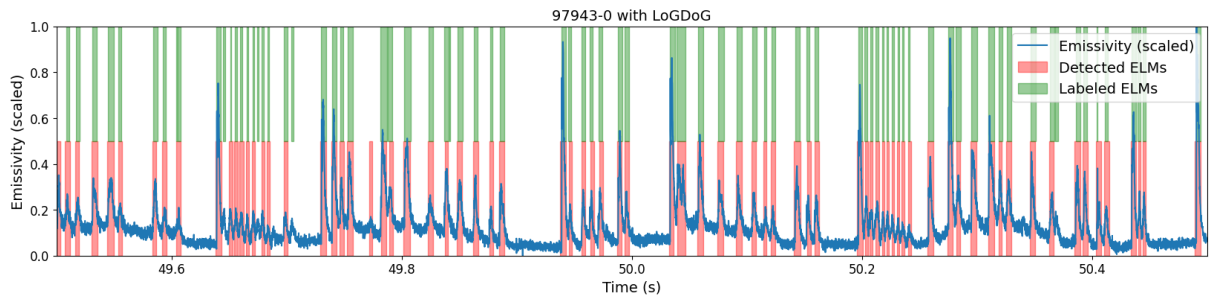
The CustomKernel method applied on shot 97943-0.



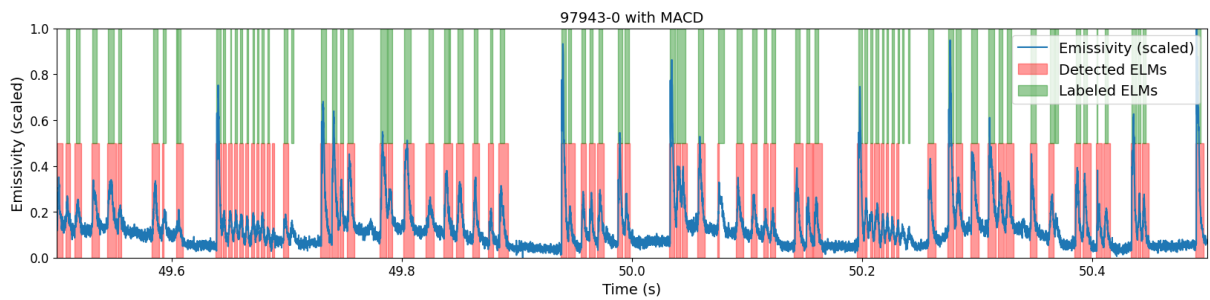
The Deconvolution method applied on shot 97943-0.



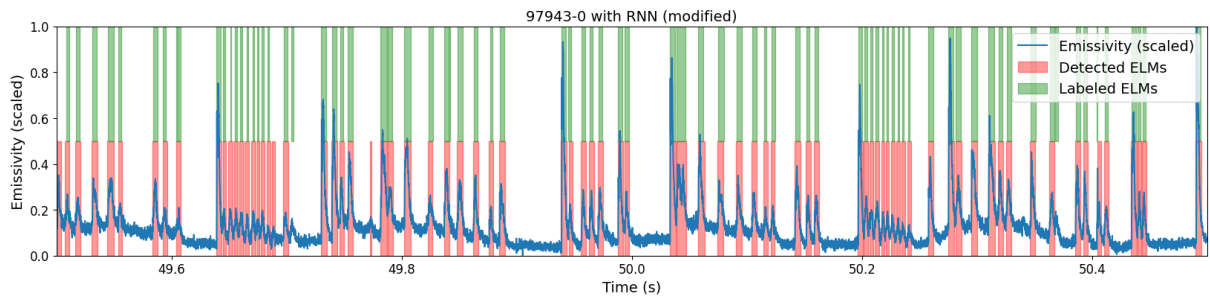
The LoG method applied on shot 97943-0.



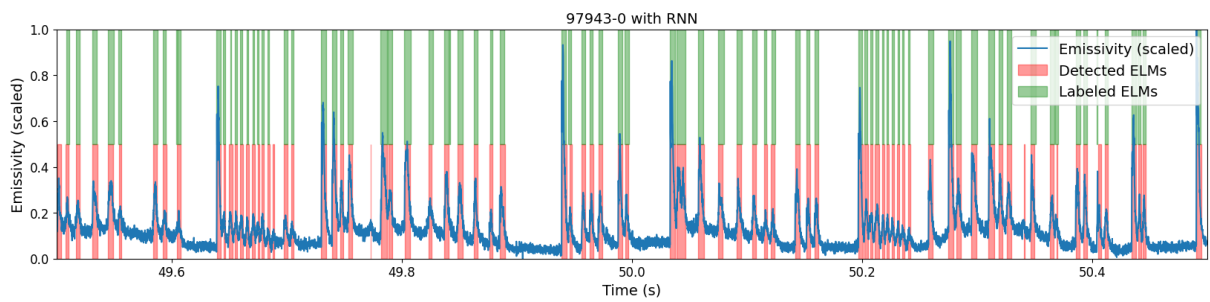
The LoGDoG method applied on shot 97943-0.



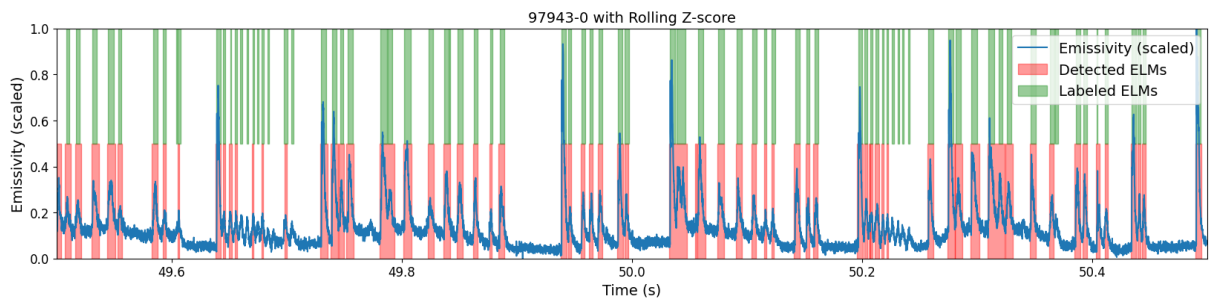
The MACD method applied on shot 97943-0.



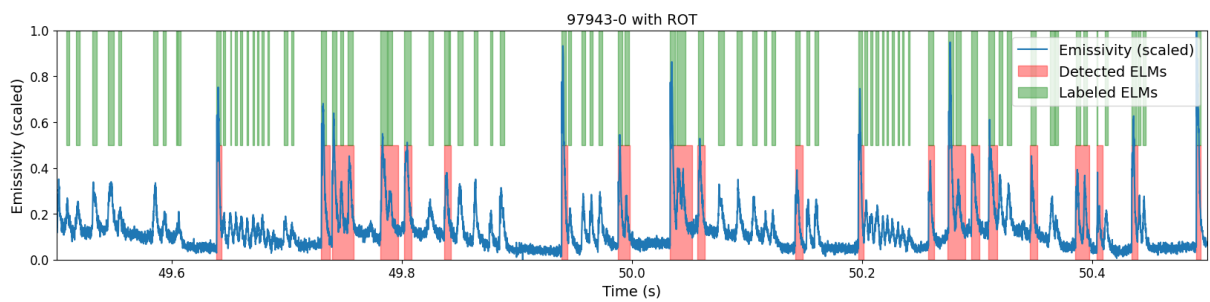
The RNN (modified) method applied on shot 97943-0.



The RNN method applied on shot 97943-0.



The Rolling Z-score method applied on shot 97943-0.



The ROT method applied on shot 97943-0.

Appendix C: Comparing results between the two labels sets

	gap					interval				
	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)
ROT	34.41	65.59	25.62	74.38	44.69	34.41	65.59	17.53	82.47	52.48
MACD	87.17	12.83	78.13	21.87	57.25	87.17	12.83	21.63	78.37	82.25
Rolling Z-score	89.99	10.01	60.82	39.18	64.16	89.99	10.01	16.39	83.61	87.33
LoG	86.61	13.39	57.86	42.14	64.46	86.61	13.39	12.79	87.21	87.81
LoGDoG	80.78	19.22	45.15	54.85	67.59	80.78	19.22	10.91	89.09	87.42
CustomKernel	82.97	17.03	46.01	53.99	68.24	82.97	17.03	9.39	90.61	88.62
RNN	83.16	16.84	20.01	79.99	81.53	83.16	16.84	14.60	85.40	84.65
RNN (modified)	91.95	8.05	59.52	40.48	65.46	91.95	8.05	15.90	84.10	87.07
CNN	53.35	46.65	12.50	87.50	69.35	53.35	46.65	10.89	89.11	70.61
CNN (modified)	80.01	19.99	54.00	46.00	63.27	80.01	19.99	16.63	83.37	82.66
Deconvolution	78.14	21.86	84.39	15.61	51.90	78.14	21.86	70.30	29.70	56.28

Table with results for all methods using the new labels.

	gap					interval				
	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)	TP (%)	FN (%)	FP (%)	TN (%)	F_{β} (%)
ROT	32.31	67.69	20.77	79.23	44.25	32.31	67.69	2.69	97.31	56.46
MACD	85.18	14.82	82.41	17.59	55.30	85.18	14.82	24.60	75.40	80.54
Rolling Z-score	26.92	73.08	24.99	75.01	40.33	26.92	73.08	17.66	82.34	44.76
LoG	87.27	12.73	51.61	48.39	67.64	87.27	12.73	8.00	92.00	91.05
LoGDoG	72.47	27.53	46.97	53.03	62.17	72.47	27.53	6.53	93.47	85.47
CustomKernel	79.59	20.41	45.21	54.79	66.21	79.59	20.41	5.19	94.81	89.53
RNN	82.62	17.38	30.53	69.47	74.88	82.62	17.38	7.39	92.61	89.45
CNN	86.63	13.37	44.46	55.54	70.28	86.63	13.37	11.65	88.35	88.16
Deconvolution	43.13	56.87	67.59	32.41	36.32	43.13	56.87	45.25	54.75	44.23

Table with results for all methods using the old labels.

A notable trend is an increase in both false positives and true positives when comparing the new labels to the old labels. This may be attributed to the prevalence of short-duration, low-amplitude ELMs that occur consecutively in the new labels, resembling noise. Consequently, the new labels pose greater difficulty for the ELM detection methods. In an attempt to capture an adequate number of ELMs, the methods also identify some noise as ELMs. In contrast, the old labels allowed for more effective noise reduction, resulting in a lower count of false positives.