TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM School of Computation, Information and Technology

Multi-Class and Cross-Tokamak Disruption Prediction using Shapelet-Based Neural Networks

Victor Maria Allan Artigues

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung der akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Die Dissertation wurde am 12.10.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 10.05.2023 angenommen.

TECHNISCHE UNIVERSITÄT MÜNCHEN

DEPARTMENT OF INFORMATICS

Doctoral Thesis in Informatics

Multi-Class and Cross-Tokamak Disruption Prediction using Shapelet-Based Neural Networks

Author:	Victor Maria Allan Artigues
Supervisor:	Hon.-Prof. Dr. Frank Jenko
Advisor:	Hon.-Prof. Dr. Frank Jenko
Submission Date:	28th September, 2022

I hereby declare that this doctoral thesis in informatics has been written by myself and that all sources used are documented.

Munich, 28th September, 2022 Victor Maria Allan Artigues

Hiermit erkläre ich dass die vorliegende Doktorarbeit in Informatik selbst verfasst habe und dass alle verwendeten Quellen belegt sind.

München, 28. September 2022 Victor Maria Allan Artigues

Publications

Parts of the results from Chapter 4 have been submitted to Nuclear Fusion.

Abstract

Since the 1960s, nuclear fusion has been a promising source of energy to produce electricity. After decades of planning and construction, ITER, the largest tokamak ever built, will start its experimental campaigns in 2025. This tokamak will provide answers to key questions on the path toward fusion power plants as well as demonstrate the feasibility of an energy output greater than the energy input.

Disruptions, the sudden loss of magnetic confinement in tokamaks, remains to this day an unsolved issue. These violent events produce substantial heat and mechanical loads that could jeopardize ITER's structural integrity.

Protecting tokamaks from disruptions includes many intricate elements such as the physical understanding of the phenomena at play, the development of mitigation tools, and disruption predictors. For disruption prediction, the lack of first-principle models calls for data-driven approaches. The development of data-driven disruption predictors for existing machines has been studied in the past decades and good results have been achieved. The next step in this field consists of building predictors for future devices. To avoid damages to a new tokamak the disruption prediction systems would ideally be available on day one. Cross-tokamaks predictions are being studied to answer this need, and our work is conducted in this optic.

This thesis focuses on event-based predictions and the study of machine learning tools for cross-tokamak disruption prediction. We investigate the use of shapelet-based neural networks as they provide interpretability which is a valuable feature in the field of physics.

Our work is divided into three sections. First, we propose a new formulation of the shapelet transform using spline interpolators to constrain their regularity. On the UCR 2018 benchmark, we show that this new formulation does not deteriorate the performance of the shapelet while allowing the user to define the spline basis to obtain different properties. Additionally, using a reduced basis lowers the training computational cost of the shapelets.

Next, we report on our study on JET data from the years 2000 to 2008 in which we compare our shapelet models to two architectures from the literature. The two architectures borrowed from the literature come from the APODIS model and an LSTM neural network. Both binary and multi-class tasks are used to evaluate the three models. After an ablation study, we attribute the good performance of our models to the shapelets, which show promising performance for cross-tokamak applications.

Finally, we performed a zero-shot transfer learning test using JET data ranging from the year 2012 to 2020 and a small dataset from ASDEX-Upgrade. This study targets specifically the impurity accumulation event which has been the main cause of disruption at JET since the installation of its metallic wall. We provide a thorough analysis of the different models implemented making use of the interpretability of the shapelets. We find that only the models using exclusively the normalized shapelet detected the accumulation of impurities at ASDEX-Upgrade.

Zusammenfassung

Die Kernfusion gilt als eine vielversprechende Energiequelle für die Stromerzeugung. Nach jahrzehntelanger Planung und Konstruktion wird ITER, der größte jemals gebaute Tokamak, in 2025 seine Versuchskampagnen aufnehmen. Dieser Tokamak wird Antworten auf wichtige Fragen auf dem Weg zu Fusionskraftwerken geben. Er wird die Machbarkeit eines Energieoutputs demonstrieren, der größer ist als der Energieinput.

Störungen wie der plötzliche Verlust des magnetischen Einschlusses in Tokamaks, stellen ein bis heute ungelöstes Problem dar. Diese heftigen Ereignisse erzeugen erhebliche Hitze und mechanische Belastungen, die die strukturelle Integrität von ITER gefährden könnten.

Der Schutz von Tokamaks vor solchen Störungen umfasst viele komplizierte Elemente wie das physikalische Verständnis der beteiligten Phänomene, die Entwicklung von Hilfsmitteln zur Schadensbegrenzung und Störungsvorhersagen. Da für die Störungsvorhersage keine First-Principles-Modelle zur Verfügung stehen, sind datengestützte Ansätze erforderlich. Die Entwicklung von datengesteuerten Störungsvorhersagen für bestehende Maschinen wurde in den letzten Jahrzehnten untersucht und es wurden gute Ergebnisse erzielt. Der nächste Schritt in diesem Bereich besteht in der Entwicklung von Vorhersagemodellen für zukünftige Experimente. Um Schäden an einem neuen Tokamak zu vermeiden, sollten die Störungsvorhersagesysteme idealerweise vom ersten Tag an verfügbar sein. Um diesem Bedürfnis gerecht zu werden, werden torkamakübergreifende Vorhersagen untersucht, und unsere Arbeit ist in diesem Sinne ausgerichtet.

Der Schwerpunkt dieser Arbeit liegt auf ereignisbasierten Vorhersagen und der Untersuchung von Werkzeugen des maschinellen Lernens für die Vorhersage von Störungen für zwei Tokamaks. Wir untersuchen die Verwendung von Shapelet-basierten neuronalen Netzen, da diese eine Interpretierbarkeit bieten, die im Bereich der Physik von großem Wert ist.

Unsere Arbeit ist in drei Abschnitte unterteilt. Zunächst schlagen wir eine neue Formulierung der Shapelet-Transformation vor, die Spline-Interpolatoren verwendet, um ihre Regelmäßigkeit zu beschränken. Anhand des UCR 2018-Benchmarks zeigen wir, dass diese neue Formulierung die Leistung der Shapelet-Transformation nicht verschlechtert und es dem Benutzer ermöglicht, die Spline-Basis zu definieren, um verschiedene Eigenschaften zu erhalten. Darüber hinaus senkt die Verwendung einer reduzierten Basis die Rechenkosten für das Training der Shapelets.

Als nächstes berichten wir über unsere Studie mit JET-Daten aus den Jahren 2000 bis 2008, in der wir unsere Shapelet-Modelle mit zwei Architekturen aus der Literatur vergleichen. Bei den beiden aus der Literatur entlehnten Architekturen handelt es sich um das APODIS-Modell und ein neuronales LSTM-Netz. Zur Bewertung der drei Modelle werden sowohl binäre als auch Mehrklassenaufgaben verwendet. Nach einer Ablationsstudie führten wir die gute Leistung unserer Shapelet-basierten Modelle auf die Shapelets zurück. Dies zeigte eine vielversprechende Leistung für Cross-Tokamak-Anwendungen.

Schließlich führten wir einen Zero-Shot-Transfer-Learning-Test mit JET-Daten aus den Jahren 2012 bis 2020 und einem kleinen Datensatz von ASDEX-Upgrade durch. Diese

Studie zielt speziell auf die Verunreinigungsakkumulation ab, die seit der Installation der Metallwand die Hauptursache für Störungen bei JET ist. Wir liefern eine gründliche Analyse der verschiedenen implementierten Modelle, wobei wir die Interpretierbarkeit der Shapelets nutzen. Von den verschiedenen implementierten Modellen haben nur die Modelle, die die normierten Shapelets und nicht den absoluten Wert der Eingangssignale verwenden, Verunreinigungsansammlungen in den ASDEX-Upgrade-Daten erkannt.

Contents

1 Introduction

1.1 General introduction

Energy has been at the center of the development of the human race. It revolutionized our way of life, from the first discovery of fire to the invention of the steam engine during the industrial revolution and today's use of fossil fuels. The control of different energy sources has allowed humans to replace manual labor with machines. It has transformed western societies from agricultural and industrial to service economies. Most of these transformations came from the use of fossil fuels, which the overwhelming majority of climate scientists agree is the main cause of climate change [1].

The CO2 released from burning fossil fuels reduces the amount of heat the Earth radiates out into space. Other gases have similar or worse effects, but CO2 is released in the largest quantities. More importantly, the CO2 concentration has been increasing at unprecedented rates since the 20th century, reaching its highest point today [2].

The various predictions [3] show an increase in CO2 emissions and effects on the climate if no action is taken. In order to reduce our impact on the climate, it is crucial to find alternative energy sources that do not emit as much greenhouse gas as fossil fuels. To this end, many different strategies have been proposed, including a variety of existing sustainable energy sources. It is in this context, that research is being conducted towards a fusion-based power plant.

Producing commercially viable electricity from fusion energy is unfortunately still decades away and cannot be the answer needed today to fight climate change. It is viewed as a possible solution on a longer timescale.

Nuclear fusion, the process powering the Sun, combines light elements into heavier ones, such as hydrogen into helium. The resulting helium nucleus is lighter than the combined mass of the original hydrogen. Through this process, a small fraction of the mass has been converted into energy, calculated using Einstein's formula

$$E = mc^2 \tag{1.1}$$

with m and c being the difference in mass and the speed of light, respectively.

The fusion of two nuclei requires extremely high temperatures, such that their kinetic energy is sufficient to overcome their Coulomb repulsion. In addition, the fusing particles need to be confined using strong force fields. In the Sun, this is achieved through its massive gravitational force. However reproducing those conditions on earth is not feasible, instead, scientists have developed machines that use strong magnetic fields. One of the devices designed around this principle is the tokamak. Tokamaks achieve fusion with a very small density but 10 times the temperature in the core of the Sun, i.e. 150 million degrees Celsius.

Contrary to fission, this process produces very little amount of dangerous byproducts. Plus, fusion cannot start a reaction chain making it a lot safer to operate.

Fusion reactors are still in a research phase and have not been able to produce any net electricity yet as no tokamak produced more energy than was put in. The ratio of energy output and input is described by the so-called Q-factor, given by

$$Q = \frac{E_{\text{out}}}{E_{\text{in}}} \tag{1.2}$$

So far, the highest Q achieved by a tokamak was 0.67 in 1997 at JET meaning that out of 100 units of heating energy, the machine returned 67. The first goal is now to achieve $Q > 1$ to show net energy can be made from fusion. This Q only includes the heating energy put in the machine and not the energy of the entire system. Once $Q > 1$ is reached, achieving an $Q_{\text{total}} > 1$ will allow for net electricity production.

The biggest experimental tokamak is currently being built in southern France to demonstrate the feasibility of fusion for electricity generation with an energy output greater than the heating input with a goal of $Q \geq 10$ [4].

This colossal project, named the ITER [4], is the result of a worldwide collaboration of over thirty countries: China, the European Union, India, Japan, Korea, Russia, and the United States of America. With an estimated cost of around 20 billion euros, ITER is one of the most expensive scientific research project worldwide. The first plasma is currently planned for the end of 2025 [5].

Given the investment in terms of money, time, and work, the best preparation possible is required. Researchers have been, and are currently trying to answer as many questions as possible to allow for a successful operational phase. The two biggest topics are turbulence [6] and disruptions [7]. Turbulence is one of the elements that kept tokamaks away from the crucial net energy gain as it dissipates the energy at the core of the plasma outward, preventing researchers to achieve high enough temperatures. It is being studied and modeled in order to predict the best plasma regime. The second element, disruptions, is the sudden loss of confinement of the plasma. This slows down the research and, although it does not cause risks to the scale of an incident at a nuclear fission reactor, disruptions can damage the machine itself. The damages can vary from small impacts, and partially melted components, to an inoperable machine.

Preventing disruptions is of the utmost importance for ITER and future fusion reactors and is being studied from multiple sides. There is a need to better understand disruptions from a physics perspective. Controlling the plasma, and avoiding instabilities is another domain. From a practical point of view, the development of predictors, or alarms is key to the safe operation of tokamaks. This thesis deals with the development of such predictors. With the use of machine learning, we study the prediction of disruptions and disruption-relevant events.

In the next sections of this introduction, we will discuss the properties of fusion and tokamaks in more detail, as well as the issue we focus on: disruptions. We then introduce the field of machine learning, the success it has achieved in recent years, and how it has been applied to physics. Lastly, we give an overview of the content of this thesis.

1.2 Nuclear fusion

Nuclear fusion was first proposed just over a century ago in 1920 [8] as the mechanism powering the stars. Around a decade later the first human-made fusion reactions were achieved using a particle accelerator [9]. The experiments consisted of a metal foil with the element of interest, shot at with deuterium nuclei. They introduce among others, the Deuterium-Deuterium (D-D) reaction

$$\mathrm{D}^2 + \mathrm{D}^2 \longrightarrow \mathrm{He}^3 + \mathrm{n} + 3.27\mathrm{MeV} \tag{1.3}$$

This work also introduces the Deuterium-Tritium reaction, which is the primary candidate for future fusion power plants.

It has been estimated that in order to have enough collisions, enough fusion, for the reaction to be self-sustained, the fuel would need to reach $1.5 \cdot 10^8$ Kelvin at a density of 10^{20} particles per cubic meter. Additionally, the time confinement time τ_E, i.e. the rate at which the stored energy escapes the confinement, has to be greater than 2 seconds. Under these conditions, a self-sustained thermonuclear fusion would be achieved [10].

At these conditions of density and temperature, the state of the fuel is a plasma, a state in which the electrons can move freely and are no longer bound to their atomic nuclei.

To enclose this plasma at such extreme conditions, multiple devices relying on magnetic confinement fusion (MCF) have been proposed. These devices trap the charged electrons and ions of the plasma in a magnetic field. The most successful devices to this day are the aforementioned tokamaks.

1.3 Tokamak

Tokamaks are thermonuclear devices that heat a plasma kept inside a toroidal magnetic field, see figure 1.1. As the particles forming the plasma are charged they react to electromagnetic fields and gyrate around the magnetic lines due to the Lorentz force

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \tag{1.4}$$

where q is the charge of the particle, $\mathbf{E}$ the electric field, $\mathbf{v}$ the particle velocity, and $\mathbf{B}$ the magnetic field.

Because of the geometry of the torus, the magnetic field is stronger on the inside than the outside, which results in a high field side and a low field side. This gradient in the magnetic field causes the particles to drift, with the positively charged ions drifting in one direction and the negatively charged electrons drifting in the opposite direction. This separation of charges is avoided by twisting the field lines such that particles alternates between the high-field side and low-field side, resulting on average in no significant drift. The twisting of the magnetic field lines is achieved through the use of a solenoid in the center of the torus. The current created in the plasma by the solenoid induces a poloidal magnetic field. The combination of the toroidal and poloidal magnetic fields see figure 1.1, results in the helix-shaped magnetic field lines, see figure 1.2.

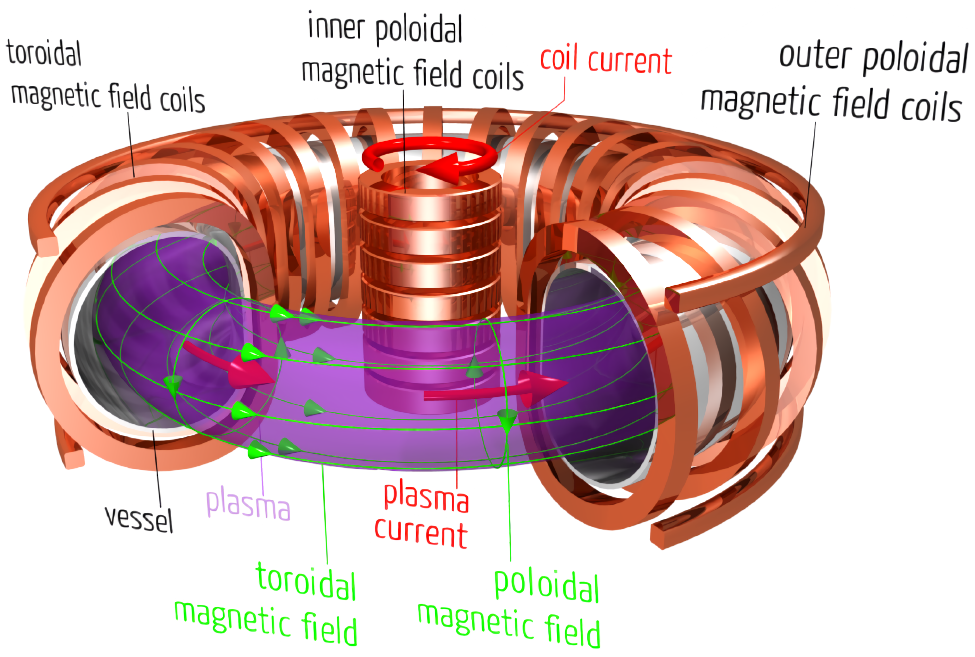


Figure 1.1: Schematic of the plasma geometry, toroidal and poloidal magnetic field coils, and the central solenoid. Arrows represent the individual magnetic fields that result in the helix-shaped magnetic field confining the plasma. Image (adapted) courtesy of IPP, Dr. Christian Brandt.

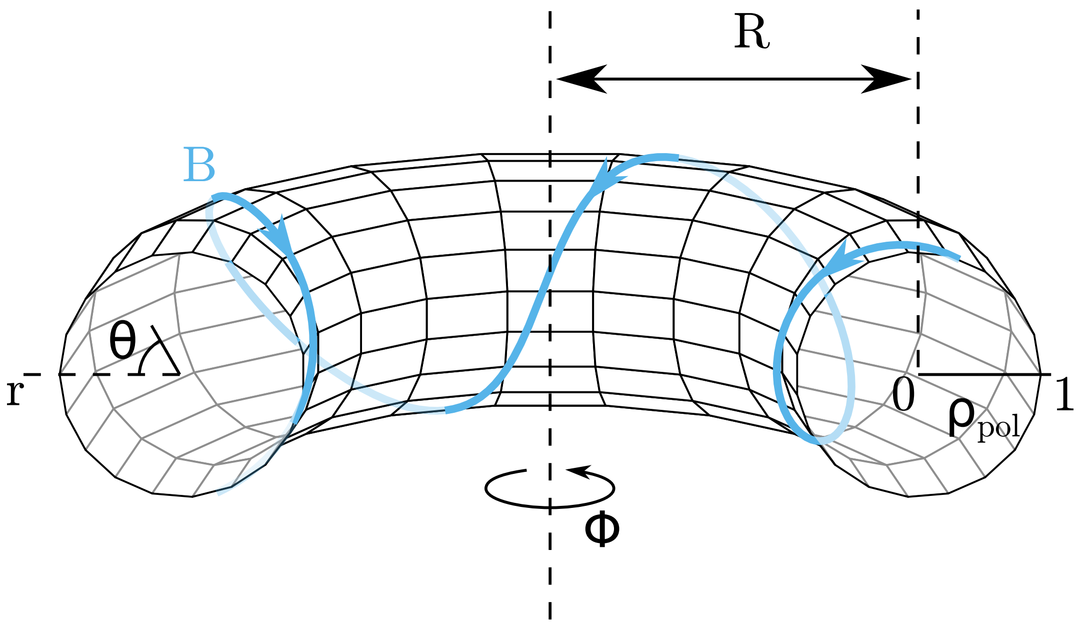


Figure 1.2: Annotated geometry of the plasma with the toroidal (Φ) and poloidal (θ) angles, minor radius (r), major radius (R), and ρ_{pol} axis. A magnetic field line is illustrated in blue. Adapted from [11].

As mentioned, the planned reaction for future tokamak reactors is the Deuterium-Tritium (D-T) fusion

$$^{2}\text{D}^{+} + {}^{3}\text{T}^{+} \longrightarrow {}^{4}\text{He}^{2+} + \text{n} + 17.6\text{MeV} \tag{1.5}$$

Tritium is one of the most expensive materials in the world today, with low estimates of around 30 million dollars per kilogram [12]. This very high cost comes from its half-life of 12.3 years. This short-lived element is therefore extremely hard to find in nature. Hence the only way to obtain a sufficient amount of tritium is to breed it.

The 17.6 MeV from the D-T reaction is split into the α-particle $^{4}\text{He}^{2+}$ carrying 3.3 MeV and the neutron with 14.3 MeV. The α-particles, being positively charged, stay trapped in the magnetic field while the neutrons exit the confinement. The D-T 14.3 MeV neutron can activate the surrounding material, i.e. making it radioactive. Therefore, facilities operating with D-T require much more expensive safety measures.

Most tokamaks today do not require tritium as they aim to study the plasma behavior rather than achieving fusion.

To assess the progress of research a metric is used which combines multiple factors characterizing a well-performing plasma into a single quantity $n \cdot T \cdot \tau_E$ called triple product [13].

The triple product is a relatively simple metric to evaluate progress in fusion research. It combines the density of the plasma, its temperature, and the confinement time. To achieve a self-sustaining plasma, meaning a plasma producing enough heat to sustain the fusion reaction without external heating, thresholds are calculated over the triple product.

Ref. [13] shows the following criterion to achieve ignition, i.e. a self-sustaining plasma

$$n \cdot T \cdot \tau_E > 3 \times 10^{21}\,\mathrm{m}^{-3}\mathrm{KeVs} \tag{1.6}$$

To put this number in context the Joint European Torus (JET) reached a triple product of $6.1 \cdot 10^{20}$ [14] and $4.7 \times 10^{20}\,\mathrm{m}^{-3}\mathrm{KeVs}$ [15], the Axially Symmetric Divertor Experiment - Upgrade (ASDEX - U) $2.2 \times 10^{19}\,\mathrm{m}^{-3}\mathrm{KeVs}$ in 2016 [16]. In Japan, JT-60U achieved a triple product of $5.6 \times 10^{20}\,\mathrm{m}^{-3}\mathrm{KeVs}$ [17]. Ref. [18] provides a review of the progress made not only by tokamaks but fusion devices in general.

The size of new tokamaks stopped increasing at the end of the 20th century as we understood that to increase the triple product further a prohibitively large. ITER was planned at this time to be the largest tokamak ever built, with a minor radius of 2 meters and an outer radius of 6.2 meters. The projections place its triple product at $7.4 \times 10^{21}\,\mathrm{m}^{-3}\mathrm{KeVs}$, therefore achieving ignition for the first time. With experiments, also called discharges, lasting multiple minutes ITER will demonstrate the feasibility of fusion reactors as power plants.

ITER is still an experimental project and will not be connected to the electrical grid, nor produce any electricity. The excess power will be discarded. The natural step after ITER would be a tokamak producing electricity. Currently, in the design phase, DEMO [19] will fulfill this role.

1.4 Key issues towards ITER

In the preparation of ITER, two main issues remain: turbulence and disruptions. We focus on disruptions as they are the topic of this thesis.

Disruptions [7] are a sudden loss of magnetic confinement. Those instabilities occur on very fast time scales and their predictability is limited. The many different physical events that may lead to a disruption, their different time scales, and possible combinations, make the study of disruption especially hard.

During a disruption, the massive amount of energy stored in the plasma is released through different processes. A fraction of the energy is transferred to the plasma-facing components as heat. The changes in the current and magnetic field create eddy and halo currents in the electrically conducting metal structures of the tokamak, resulting in mechanical loads that can cause significant damage to the machine. A third way of releasing the stored energy during a disruption is through the generation of highly energetic relativistic electrons, called runaway electrons. It was already observed on existing machines that beams of relativistic electrons can melt the metal tiles inside the tokamak.

The damage caused by a disruption is proportional to the energy stored in the plasma. As tokamaks are built bigger, the deleterious effects of disruptions increase. The plasma stored energy which is released during a disruption is given by equation 1.7.

$$E_m = \frac{1}{2} L_i I_p^2 \tag{1.7}$$

It grows linearly with the internal inductance L_i but quadratically with the plasma current I_p.

ITER, as the biggest tokamak to date with a plasma current of 15 MA is therefore at greater risk than present machines. JET can currently achieve the highest plasma current at a maximum of 4.8 MA. This two to threefold increase for ITER would result in a four to nine-time higher magnetic energy according to equation 1.7.

For these reasons, it is crucial to better understand, avoid, predict and mitigate disruptions. Although ITER runs a great risk at full power, the experimental campaigns will slowly bring the machine to its limit over 10 years. Only then D-T fuel will be introduced and full-power experiments will be run. If there is still a need for the best possible disruption avoidance, prediction, and mitigation systems on day one, there will also be a lot of opportunities to learn from the early years of operation.

Our work fits in the frame of disruption prediction. To ensure safe operation at ITER, we investigate the use of machine learning methods to build alarms for incoming disruptions or detect instabilities that might lead to disruptions.

1.5 Machine learning

Machine learning (ML) [20–22] is the field of study of algorithms that improve their modeling performance by learning from examples. It is an interdisciplinary field relying on, among others, statistics, optimization, and computer science. Machine learning can be divided into a few main approaches, namely supervised learning [23], unsupervised learning [24], and reinforcement learning [25].

1.5.1 Supervised learning

Supervised learning makes use of labeled data. The goal is to determine, given a set of data points x, and their corresponding labels y, the function f such that $f(x) = y$. One key element that the function f should satisfy is, as defined in the machine-learning field, *generalization*. Constructing f as defined previously is trivial. We define a function, not necessarily continuous, using only the points in our data. The difficult task is to have $f(\tilde{x}) = \tilde{y}$, for any new $\tilde{x}$ from the same distribution as x, and $\tilde{y}$ its label. The trivial function above might not even be defined at $f(\tilde{x})$. It does not have any generalization capability, although it perfectly reproduces the original, training, data points. The general task is therefore to learn the original distribution of our dataset, knowing only a finite number of realizations. Supervised learning has been successfully applied in many domains. In medicine, ref. [26] achieved on-par or better results in lung cancer detection than experts. In computational biology, the prediction of protein structure has been an extremely hard task. AlphaFold 2 [27] used a neural network with supervised training and significantly outperformed other methods on this task.

1.5.2 Unsupervised learning

Any method that makes use of data, without labels might be called unsupervised learning. The most well-known field of unsupervised learning is clustering [28]. Clustering refers to the task of partitioning data. The aim is to extract knowledge by identifying patterns directly from the data without prior information, i.e. labels. It can be used as a first step followed by further learning and analysis. Today's most advanced language processing model GPT-3 [29] first learns in an unsupervised manner from gigantic text collections of petabytes of text from the internet, before being fine-tuned to specific tasks.

1.5.3 Reinforcement learning

One last method from machine learning that produced impressive results in recent years is reinforcement learning. Here a reward function is defined over the state of the environment, e.g. a score in a game or a binary winning/losing result. The model can interact with the environment through a set of actions a. The learning of the actions a, given previous environment states and actions to optimize a reward function, is called reinforcement learning. In many tasks, like in chess, shogi and Go [30, 31] the impact of the very first action on the outcome of the game is only known much later. The difficulty comes from the delay between actions and rewards.

Reinforcement learning produced amazing results outside of the fusion community such as ref. [32]. These results were quickly followed by applications to tokamaks and plasma control [33–36].

1.5.4 Machine learning in fusion

The field of physics only recently adopted the use of machine learning methods. This slow adoption has multiple explanations. As physics searches for the understanding of the universe and the laws that govern it, the appearance of black-box models that can do accurate predictions but not explain their reasoning was not very interesting at first. Secondly, early machine learning methods could only use data and not the immense knowledge humanity has accumulated in Physics. Machine learning has gained attraction in the field of physics with the recent development of Physics Informed Machine learning [37]. Additionally, physicists have found the need for complex neural networks on specific tasks that our current understanding cannot solve.

Interpretabilty of complex models has also been studied by the machine learning community, allowing for better understanding and confidence in the models [38]. The question of interpretability also appears in healthcare [39, 40], as very deep ethical questions arise when we start putting lives into the hands of black-box algorithms.

For disruption prediction, a false alarm, i.e. wrongly predicting a disruption, would result in the interruption of the experiment. It slows down research which in turn costs money. A missed alarm can be much worse. In the case of ITER, an unmitigated disruption could render the machine unusable, wasting decades of research and investments. Given those risks, there is a natural need to understand the alarms and how they are predicted.

Disruptions are one of the phenomena that have proven very hard to fully understand, model, and predict. There are very few first-principle approaches allowing for the prediction of disruptions. With the risk of serious damage to the largest tokamaks, the need for disruption prediction methods steered a branch of the research towards machine learning. The first attempts focused on binary disruption predictions. Either a disruption is incoming, or the plasma is in s stable phase and it will not disrupt. Because disruptions combine many different events, physics boundaries, and technical limits, researchers tried to predict the cause or type of disruption instead of the binary stable or disruptive label. These methods trained and applied to existing machines face a major challenge, that is the transfer to a new machine such as ITER. As ITER will face the biggest risks with disruptions it needs disruption alarms before starting to run full-power experiments.

Our work focused first on binary and multi-class disruption prediction on JET data from the years 2000 to 2008, before the installation of the metallic wall. The second section of our work investigated zero-shot transfer learning for the main cause of disruption at JET in its metallic wall configuration, i.e. after the year 2011, and we also analyzed our results on a few discharges from another tokamak with metallic walls: ASDEX-Upgrade. Our models are based on neural networks and shapelets. We propose a new version of the shapelet transform making use of splines to control the regularity of the learned shapelets.

1.6 Thesis structure

As our work is based in a plasma physics context and uses machine learning we want to give a basic understanding of machine learning to readers with a physics background. And we want to explain the context of our work, meaning tokamaks and disruptions, to readers with a computer science background. To this end, in the next section 2 we present the different elements of the models used in our work, such as shapelets, multi-layer perceptron, and recurrent networks. We then continue the explanation of the workings of a tokamak we started in the introduction, to finally explain plasma instabilities, causes of disruptions, and their consequences. Section 4 and 5 contain the results of the two tasks we conducted, i.e. binary and multi-class disruption prediction on JET data from the years 2000 to 2008, and impurity accumulation detection at JET with a transfer learning application to a small dataset from ASDEX-Upgrade. In section 6 we conclude our work, and outline plans for future work.

2 Theory and fundamentals

Our study revolves around the use of Shapelets in neural networks for disruption prediction tasks, using different datasets. In this chapter, we give an overview of the tools used and describe disruptions. We first present Shapelets and the Shapelet Transform. Then we explain the architecture of feed-forward neural networks as well as recurrent networks. The numerous methods we used to improve the neural networks' performances are listed and described. Finally, we briefly present the physics of disruptions and where the different strategies to protect the machines such as avoidance, prevention, and mitigation apply.

Notation

Throughout this thesis lower case variables v represent scalar values, bold variables $\mathbf{v}, \mathbf{V}$ represent vectors or matrices. The indices of vector and matrices start at 0. $t_{i:j}$ defines the vector $(t_i, \dots, t_j)^T$

2.1 Shapelets

Shapelets were first introduced in [41] as a new primitive for data mining, particularly for time-series. In many data mining tasks with time-series the discriminating part is only a subsequence. The shapelets are time-series $\mathbf{S} = (t_0, \dots, t_{N-1})^T$ used to identify such subsequences via the following metric

$$d\,(T, S) = \min_{0 \leq i < M-N} \sqrt{\sum_{j=0}^{N-1} (T_{i+j} - S_j)^2} \tag{2.1}$$

for a time-series T of length M and a shapelet S of length N. It is the euclidean distance at the best matching location which is also referred to as shapelet similarity or transform.

This first work enumerated all possible subsequences of a data set to find the most discriminative ones. Multiple improvements have been proposed to speed up the enumeration using cheap upper-bound to prune candidate shapelets efficiently [42], ref. [43] optimized the shapelet computation by storing parts of the shapelet transform expression. [44] introduced the Fast Shapelet algorithm which reduces the dimensions of the shapelets using SAX words [45]. The Ultra-Fast Shapelet [46] avoids the exhaustive search of shapelets by randomly selecting a smaller number of candidates.

Some of the works listed above introduced a local standardization to the shapelet metric, also called z-normalization. [43] defined the normalized shapelet similarity as

$$d\left(T,S\right)=\min_{0\leq i<M-N}\sqrt{\sum_{j=0}^{N-1}\left(z\left(T_{i+j},\mathbf{T}_{i:i+N-1}\right)-z\left(S_j,\mathbf{S}_{0:N-1}\right)\right)^2} \tag{2.2}$$

with

$$z(X,\mathbf{X})=\frac{X-\mu_{\mathbf{X}}}{\sigma_{\mathbf{X}}} \tag{2.3}$$

with $\mu_{\mathbf{X}}$ and $\sigma_{\mathbf{X}}$ the mean and standard deviation of $\mathbf{X}=\{X_0,...,X_{N-1}\}$.

For univariate time-series [46] normalizes only the shapelet, not the local slice of the input time-series, while for multivariate time-series they also used equation 2.1.

There is no agreement in the literature on whether to use shapelets with normalization or not. In this thesis, unless specified otherwise, we used z-normalization as described in equation 2.2.

Reference [47] proposed to skip the enumeration of candidate shapelets altogether and use gradient descent to learn them. The first step in doing so is to show the differentiability of the shapelet transform. In equation 2.1, the distance is differentiable, but not the minimum function. Ref. [47] replaced the minimum with a soft minimum:

$$\text{soft-min}\left(\mathbf{Y}\right)=\frac{\sum_{j=0}^{M-N-1}Y_j e^{\alpha Y_j}}{\sum_{k=0}^{M-N-1}e^{\alpha Y_k}} \qquad \mathbf{Y}\in\mathbb{R}_+^{M-N}, \qquad \alpha\in\mathbb{R}_- \tag{2.4}$$

This work applies a logistic sigmoid function $\sigma\left(Y\right)=\left(1+e^{-Y}\right)^{-1}$ on the output of the Shapelet transform for its prediction task, and computes the final prediction as a linear combination of the logistic sigmoid outputs. Any function can be applied after the shapelet transform as long as it is differentiable. Ref. [48] embedded the shapelet transform as the first layer of a multi-layer percetron.

Two drawbacks of the shapelets identified in the literature are the enumeration of shapelets, which is solved by gradient-descent learning [47], and the shapelets' resemblance to the original data. When enumerating through candidate shapelets that were extracted from the data, as in [41–44, 46], the shapelets naturally resemble the data, but when learned, we do not have any control over the shapelets anymore. For example, in order to describe a raise, leading to a plateau in time-series, the learned shapelet can over-shoot and produce a peaked signal. This drawback does not necessarily affect the results, but the reduces the interpretability of the resulting shapelets. Also, generalization might be improved if the shapelets keep the properties of the data.

Ref. [49] used an adversarial training architecture to periodically correct the shapelets in order to make them resemble the original data. The method works but we observe in several cases that the regularization technique makes the shapelet smoother. So to control the shapelet regularity, at no extra cost, avoiding the sensitive parallel learning of a secondary generative model, we discretized the shapelets. Our shapelets are constructed using n degrees-of-freedom (DoF). We then interpolate between the points and this interpolation method defines the regularity of our shapelets.

In chapter 3, we thoroughly study the difference between standard shapelets and our spline-based shapelets, as well as the similarity between convolutions, signed cosine similarity measure, and our shapelets when the three use local normalization.

2.2 Neural network architecture

In this section, we present most of the tools used by our models. At the core, we use neural networks, which are often simply thought of as a number of cells and layers. In order to get competitive performance, a variety of tricks and tweaks both in the neural network architecture and in the training procedure are however necessary. In particular, we used dropout, batch-normalization, regularization, sharpness-aware minimization, pruning, label smoothing, and noise augmentation. In the following, we present the two versions of neural networks we used, i.e. multi-layer perceptrons and recurrent neural networks. We will explain how neural networks are trained and derive the back-propagation formula [50] to compute the gradient of the network. Finally, we will describe the various modifications we used and listed above.

2.2.1 Artificial neurons

Before we introduce any neural network architecture, we have to understand the basic element which is the artificial neuron.

The artificial neuron borrows from the brain's functioning mechanism. The neurons in the brain communicate with each other by releasing neurotransmitters from their synapses. The neurotransmitters are released after an electrical impulse traveled through the axon. This is called the activation of the neuron. Similarly, the artificial neurons have a number of inputs, which are combined linearly. The result is then passed through an activation function, whose output can be connected to other neurons.

2.2.2 Layers

From this base element, we can define a layer. A layer is a group of artificial neurons that have the same inputs. The way each neuron of the layer is connected to the input can define the type of layer. A dense layer connects all values of the inputs to all neurons. The number of neurons present in a layer defines the width of the layer.

Additionally, a bias neuron can be added to each layer. It adds a trainable constant value, allowing for shifts in the input distribution of the neurons. The output of a neuron is given in equation 2.5

$$\mathbf{o} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{2.5}$$

σ is a non-linear activation function such as the sigmoid function. Without the activation function, the neuron produces a simple linear combination of its input. A neural network could only learn linear functions regardless of the width and depth of the network. One or multiple layers form an artificial neural network.

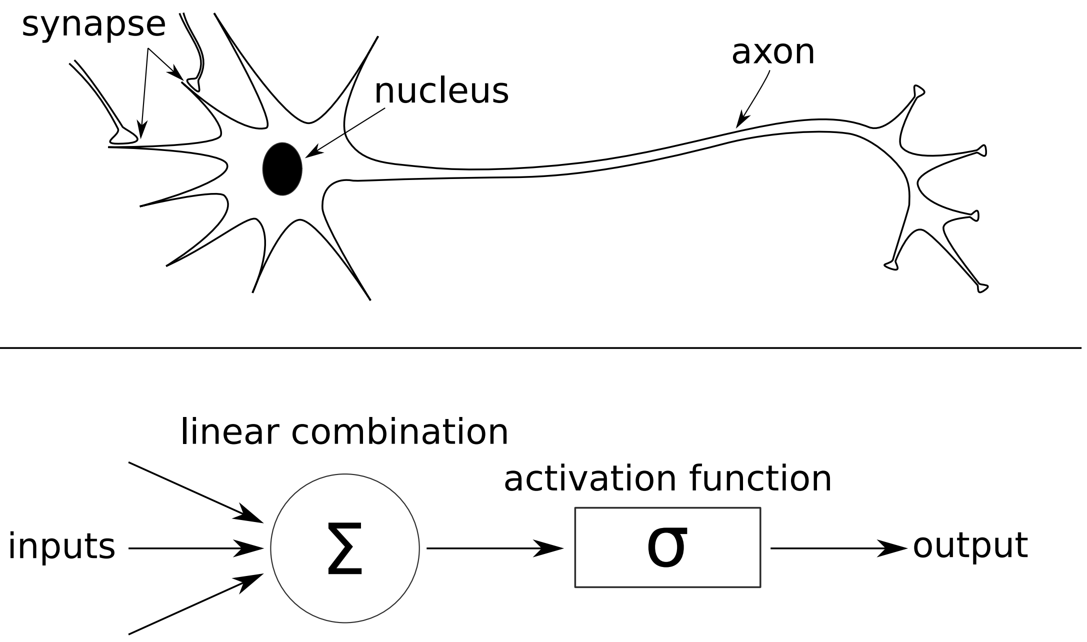


Figure 2.1: Representation of a biological neuron structure (top) and an artificial neuron used in machine learning (bottom).

2.2.3 Multi-layer perceptron

The multi-layer perceptron is the simplest artificial neural network architecture. It is made of a number of stacked layers, with the smallest configuration consisting of three layers: an input layer, one hidden layer, and the output layer. Figure 2.2 displays the representation of a multi-layer perceptron that computes the XOR function.

The XOR function is a basic function that is not linearly separable. It is a simple example to illustrate the need for the non-linear function in the artificial neuron. Adding an activation function, such as $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ or $\text{ReLU}(x) = \max(0, x)$ introduces enough non-linearity for the network to learn any continuous function, as stated by the universal approximation theorem [52].

2.2.4 Recurrent cells

A second class of neural networks is called recurrent neural networks (RNN). In such a model, the output of the neurons is fed back to themselves as input. This change in input processing compared to MLPs makes the processing of data series an integral part of the model. The ordering of the data is taken into account by the recurrent connections. The output of a recurrent neuron is

$$\mathbf{a}^t = \mathbf{b} + \mathbf{V}\mathbf{h}^{t-1} + \mathbf{W}\mathbf{x}^t \tag{2.6}$$

$$\mathbf{h}^t = \tanh\left(\mathbf{a}^t\right) \tag{2.7}$$

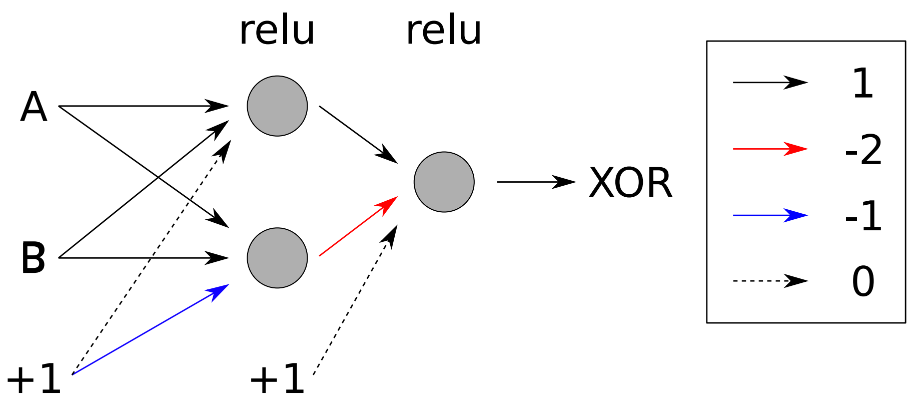


Figure 2.2: Multi-layer perceptron that computes the XOR function on A and B. Weights from ref. [51]. The weights of the connections are given by the arrows. Both the hidden and output layers have a ReLU activation function.

The only difference to a non-recurrent neuron is the addition of the recurrent term $\mathbf{V}\mathbf{h}^{t-1}$.

Gated RNN

The recurrent connections can be seen as stacked layers sharing the same weights with each intermediate layer giving the output at the intermediate time-steps. As with deep neural networks, such an architecture suffers from vanishing gradients, i.e. the training steps taken by the model during training become extremely small. The gradients become smaller and smaller when being passed through deeper and deeper layers. Hence old information will not have an impact on the predictions which means that in practice, the simple recurrent architecture is only able to remember recently seen values.

This motivated the appearance of gated recurrent cells. The gates introduce multiplicative operations with $[0, 1]$ values to allow or block the propagation of information, working as gates. The multiplicative values are based on the inputs as per other neurons.

We will first present the gated recurrent unit[53] (GRU), although it appeared later than LSTM. But its simpler construction might help to understand the LSTM architecture.

The gated recurrent unit has the following construction:

$$\mathbf{r}^t = \sigma\left(\mathbf{b} + \mathbf{V}_r\mathbf{h}^{t-1} + \mathbf{W}_r\mathbf{x}^t\right) \tag{2.8}$$

$$\mathbf{u}^t = \sigma\left(\mathbf{b} + \mathbf{V}_u\mathbf{h}^{t-1} + \mathbf{W}_u\mathbf{x}^t\right) \tag{2.9}$$

$$\mathbf{c}^t = \tanh\left(\mathbf{b} + \mathbf{V}_c\left(\mathbf{r}^t \odot \mathbf{h}^{t-1}\right) + \mathbf{W}_c\mathbf{x}^t\right) \tag{2.10}$$

$$\mathbf{h}^t = \mathbf{u}^t \odot \mathbf{h}^{t-1} + \left(1 - \mathbf{u}^t\right) \odot \mathbf{c}^t \tag{2.11}$$

with $\odot$ the element-wise product. The GRU unit starts by computing two gates $\mathbf{r}^t$ and $\mathbf{u}^t$ (equations 2.8 and 2.9) from the current input $\mathbf{x}^t$ and the previous output $\mathbf{h}^{t-1}$. $\mathbf{c}^t$ given by equation 2.10 is a candidate output based on the input $\mathbf{x}^t$ and a certain percentage, defined by the gate $\mathbf{r}^t$, of the previous output $\mathbf{h}^{t-1}$. $\mathbf{r}^t$ defines how strong the recurrence effect is for $\mathbf{c}^t$. The final output (equation 2.11) is a weighted average of the previous output, by-passing the information from the current input, and the candidate output $\mathbf{c}^t$ which combines both the current input and a certain amount of the previous output. The coefficient for the weighted average of the final output is given by the gate $\mathbf{u}^t$. The gates control the flow of information between the past information stored in $\mathbf{h}^{t-1}$ and the current input $\mathbf{x}^t$. The GRU unit exposes $\mathbf{h}^{t-1}$ directly to the output. Which is one of the main differences to the next gated unit, the Long-Short Term Memory[54].

The Long-Short Term Memory (LSTM) unit has the following construction:

$$\mathbf{f}^t = \sigma\left(\mathbf{b} + \mathbf{V}_f\mathbf{h}^{t-1} + \mathbf{W}_f\mathbf{x}^t\right) \tag{2.12}$$

$$\mathbf{i}^t = \sigma\left(\mathbf{b} + \mathbf{V}_i\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t\right) \tag{2.13}$$

$$\mathbf{c}^t = \tanh\left(\mathbf{b} + \mathbf{V}_c\mathbf{h}^{t-1} + \mathbf{W}_c\mathbf{x}^t\right) \tag{2.14}$$

$$\mathbf{o}^t = \sigma\left(\mathbf{b} + \mathbf{V}_o\mathbf{h}^{t-1} + \mathbf{W}_o\mathbf{x}^t\right) \tag{2.15}$$

$$\mathbf{s}^t = \mathbf{f}^t \odot \mathbf{s}^{t-1} + \mathbf{i}^t \odot \mathbf{c}^t \tag{2.16}$$

$$\mathbf{h}^t = \mathbf{o}^t \odot \tanh\left(\mathbf{s}^t\right) \tag{2.17}$$

It is very similar to the GRU unit. The first three equations are the same, up to the scaling of $\mathbf{h}^{t-1}$ in equation 2.10 which does not appear in equation 2.14. The difference to GRU lies in the last three equations 2.15, 2.16 and 2.17 where a more complicated combination of the current input and past information is performed through the use of an additional gate.

Equations 2.12, 2.13 and 2.15define the three gates of the LSTM called the forget gate, input gate, and output gate. Equation 2.11 was previously the output of the GRU unit. In the LSTM the similar equation 2.16 gives a cell state. The combination is not a weighted average between the previous cell state $\mathbf{s}^{t-1}$ and the candidate output $\mathbf{c}^t$, but both are weighted independently by two different gates.

Finally, the output of the LSTM unit (equation 2.17) is given by the product of the hyperbolic tangent of the new cell state $\mathbf{s}^t$ and the output gate $\mathbf{o}^t$.

An LSTM allows for much finer control of the flow of information by using one additional gate and gating the final output.

As mentioned previously, the gated recurrent unit was developed after the Long-Short Term Memory as a simplification of the latter. Ref. [55] compared different recurrent architectures and concluded that, with the same number of weights, the LSTM and GRU units offer similar performances. In ref. [56] the author used a performance-cost ratio to compare the two gated units and found that GRU outperformed LSTM on a Natural Language Processing task. Comparing with the same number of nodes, [57] found a slight advantage regarding performances for the LSTM, with the GRU very close behind. Overall there does not appear to be a clear agreement on which architecture performs better, but more a consensus that both architectures offer a performance-cost trade-off that has to be considered on a case-by-case basis.

2.2.5 Loss function

A loss function in the field of optimization defines how well a system is performing. The loss function brings down the performance of said system to a single scalar number. This definition is very broad, applying to *systems* and *performances* since many actions in our lives can be modeled as optimization problems.

In the case of supervised learning, the loss function will be used to define how close the predictions of the system, i.e. our network, are to the ground truth. Directly taking example from our classification problem, let's assume we have for input $\mathbf{X} \in \mathbb{R}^N$ a prediction $\hat{\mathbf{y}} \in [0,1]^2$ with y_0 and y_1 the probability of $\mathbf{X}$ being respectively of the first and second class. The true probabilities are $\mathbf{y} \in \{0,1\}^2$. The Cross-Entropy, equivalent to the log-likelihood, defines the error as

$$\mathcal{L}_{\text{CE}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = -\left(y_0 \log\left(\hat{y_0}\right) + y_1 \log\left(\hat{y_1}\right)\right) \tag{2.18}$$

For the neural networks in this thesis, we use an extension of the cross-entropy called focal loss [58]. The focal loss adds two parameters to the cross-entropy to deal with large class imbalances. Originally proposed for object detection in pictures, the goal of the focal loss is to allow a model to focus on the rare appearances of the class of interest, instead of concentrating effort on perfectly learning all the parts of the image where our object is *not* located. To do so, the first parameter included is $\gamma \in \mathbb{R}_+$. The modified cross-entropy reads:

$$\mathcal{L}_{\text{CE-}\gamma}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = -\left(y_0 \left(1 - \hat{y_0}\right)^\gamma \log\left(\hat{y_0}\right) + y_1 \left(1 - \hat{y_1}\right)^\gamma \log\left(\hat{y_1}\right)\right) \tag{2.19}$$

For $\gamma = 0$, we recover the original cross-entropy. For $\gamma > 0$, we are negatively impacting the error by the confidence of the prediction. For a very confident prediction with $\hat{y_0} = 0.95$, with a true prediction $y_0 = 1$, we are multiplying the error by 0.05^γ, essentially reducing it to zero. The model will not try to increase the score from 0.95 to 1 as much as with standard cross-entropy. With increasing γ, the model will focus less on increasing already good predictions (e.g. above 0.8), and consequently more on cases where the prediction is lower (e.g. at 0.4).

Figure 2.3 compares the standard cross-entropy loss function to the focal loss with different γ parameters.

The second parameter is $\alpha \in [0,1[^C$ to weight the entire classes, with C the total number of classes.

Equation 2.20 gives the general focal loss formulation for C classes.

$$\mathcal{L}_{\text{focal loss}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = -\sum_{c=0}^{C} y_c \alpha_c \left(1 - \hat{y_c}\right)^\gamma \log\left(\hat{y_c}\right) \tag{2.20}$$

2.2.6 Back-propagation

In machine learning, the training process is the resolution of an optimization problem. Optimization problems can be split into two types: convex optimization and non-convex optimization, referring to the properties of the function to optimize. For a convex problem, the exact solution can be found but non-convex problems are much harder to solve. Many models that are considered to be simpler than neural networks only require convex optimization, such as the Support Vector Model [59, 60]. This is an advantage that neural networks do not have. Neural networks require to solve a highly non-convex optimization

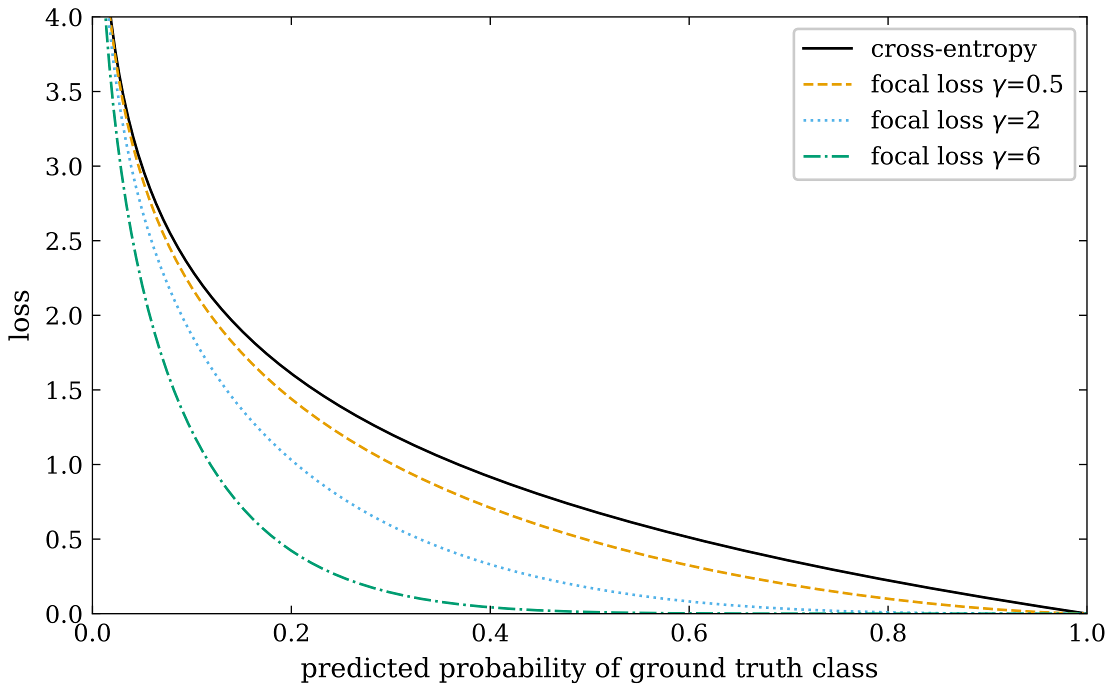


Figure 2.3: Comparison of the cross-entropy loss function and focal loss for different γ parameters. Close to zero, for the biggest error between prediction and true label the losses are similar. When the prediction approaches the ground truth value 1, the focal loss penalizes the error less than the cross-entropy. This allows the model to focus on badly classified examples instead of optimizing already good predictions.

problem with many local minimums. Many iterative gradient-based solvers exist for non-convex problems. In order to make use of those algorithms, one needs the gradient of the network. The back-propagation [50] method allows to efficiently compute the gradient of each layer of a network.

In the derivation of the gradient formula, the superscript denotes the layer number. A single subscript is the variable index for a specific layer. For weights we use two subscripts, the first being the index of the outgoing neuron and the second the index of the incoming neuron.

Each layer is a standard fully connected layer with bias

$$z_i^{l+1} = b_i^{l+1} + \sum_{j=0}^{m^{l+1}-1} w_{ij}^{l+1} x_j^l \tag{2.21}$$

with m^l the number of incoming connections called the width of the hidden layers. The layer index $l = 0$ corresponds to the inputs x_j^0 which does not have weights therefore the weights and bias layer index start at 1.

The input of the next layer is obtained after the activation function

$$x_i^{l+1} = \mathrm{ReLU}\left(z_i^{l+1}\right) \tag{2.22}$$

Before we can use back-propagation we start with a forward propagation. Forward propagation is the iterative process of passing the input x^0 through all layers of the network.

The network returns x^n with n the depth of the network. After this forward path, the loss of the network is calculated by comparing the output to the ground truth y. For our example, the loss function is the half mean squared error.

$$E = \frac{1}{2d} \sum_{i=0}^{d-1} (x_i^n - y_i)^2 \tag{2.23}$$

with d the number of outputs.

We calculate the gradient of the loss E with respect to the weight w_{ij}^l. The gradient with respect to the bias b_i^l can be obtained following the same steps with minor changes.

For the weight w_{ij}^l, according to the chain rule

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} \tag{2.24}$$

The partial derivative of equation 2.2.6 is

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = x_j^{l-1} \tag{2.25}$$

$$\delta_i^l \equiv \frac{\partial E}{\partial z_i^l} \tag{2.26}$$

Equation 2.2.6 defines the term called error, and we will show that it depends on the error term of the next layers. The output layer being the last, the error term needs to first be calculated there and the recurrence will allow computing the error terms of previous layers successively, hence the name *back-propagation*.

For the error of the last layer δ_i^n we have

$$\delta_i^n = \frac{\partial}{\partial z_i^n}\left(\frac{1}{2d}\sum_{j=0}^{d-1}\left(x_j^n - y_j\right)^2\right) \tag{2.27}$$

$$\delta_i^n = \frac{\partial}{\partial z_i^n}\left(\frac{1}{2d}\sum_{j=0}^{d-1}\left(\mathrm{ReLU}\left(z_j^n\right) - y_j\right)^2\right) \tag{2.28}$$

$$\delta_i^n = \frac{1}{d}\left(\mathrm{ReLU}\left(z_i^n\right) - y_i\right)\mathrm{ReLU}'\left(z_i^n\right) \tag{2.29}$$

$$\delta_i^n = \frac{1}{d}\left(x_i^n - y_i\right)\mathrm{ReLU}'\left(z_i^n\right) \tag{2.30}$$

As $\mathrm{ReLU}(x) = \max(0, x)$ its derivative is undefined at 0. In practice, the derivative of ReLU is implemented as

$$\mathrm{ReLU}'(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

We now have for the partial derivative of the loss with respect to the last layer n

$$\frac{\partial E}{\partial w_{ij}^n} = \frac{1}{d}\left(x_i^n - y_i\right)\mathrm{ReLU}'\left(z_i^n\right)x_j^{l-1} \tag{2.31}$$

where all terms are available from the forward pass and only ReLU' has to be applied to z_i^n.

Finally, δ_i^l for any intermediate layers is obtained by applying the chain rule in the multivariate case to equation 2.2.6.

$$\delta_i^l = \sum_{k=0}^{m^{l+1}} \frac{\partial E}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_i^l} \tag{2.32}$$

$$\delta_i^l = \sum_{k=0}^{m^{l+1}} \delta_k^{l+1}\frac{\partial}{\partial z_i^l}\left(b_k^{l+1} + \sum_{i=0}^{m^{l+1}-1} w_{ik}^{l+1}\mathrm{ReLU}\left(z_i^l\right)\right) \tag{2.33}$$

$$\delta_i^l = \mathrm{ReLU}'\left(z_i^l\right)\sum_{k=0}^{m^{l+1}} \delta_k^{l+1}w_{ik}^{l+1} \tag{2.34}$$

Having computed δ_i^n and using the recurrent equation 2.2.6 the partial derivative of the loss can be computed from the last to the first layer as

$$\frac{\partial E}{\partial w_{ij}^l} = x_j^{l-1} \mathrm{ReLU}' \left(z_i^l \right) \sum_{k=0}^{m^{l+1}} \delta_k^{l+1} w_{ik}^{l+1} \tag{2.35}$$

The error obtained can be back propagated through the network to obtain the gradient at each layer. Hence, for any architecture based on individually differentiable layers, the back-propagation algorithm allows to compute the gradient of the network. The gradient can then be used by gradient-based methods to train the network.

2.2.7 Sharpness-aware minimization

Sharpness-aware minimization [61] is a fairly recent development for neural network optimization. It stems from the discovery that poor testing performances of models perfectly trained on training data can be linked to the shape of the loss landscape, see [62].

The notion of flat minima as ideal targets for neural networks can first be found in ref. [63, 64] For simplicity, we will call a minimum in a flat (respectively sharp) region of the loss landscape a flat (respectively sharp) minimum. The idea from [62] is that a model trained towards a flat minimum will translate to similar performances on testing data, while a sharp minimum leads to worse generalization on testing data.

We will give an intuitive explanation based only on the training data.

Given the input $\mathbf{x} \in \mathbb{R}^N$, its true label probability $\mathbf{y} \in [0,1]^C$ with C the number of classes and the prediction $f_\mathbf{w}(\mathbf{x}) = \hat{\mathbf{y}} \in [0,1]^C$. For the sake of the argument, $f_\mathbf{w}$ is a linear combination of its inputs. We have a test data point $\mathbf{x}' = \mathbf{x} + \epsilon$ for a small ϵ. Test data follows the same distribution as training data. For the two close points $\mathbf{x}, \mathbf{x}'$ we expect similar output. Let's assume $f_\mathbf{w}(\mathbf{x}) = f_\mathbf{w}(\mathbf{x}')$. The small variation ϵ added to the input has the same effect as a perturbation on the weights $\mathbf{w}$, i.e. there exist $\mathbf{w}'$ such that $f_\mathbf{w}(\mathbf{x}') = f_{\mathbf{w}'}(\mathbf{x})$. This would mean that given a trained model, the test performance can be estimated by the prediction on the train data by a perturbed version of the given model. In the training loss landscape, this means that the test performance point is in the neighborhood of the trained model's loss.

Thus if a small perturbation of the model drastically lowers the performances on the same data, the model will generalize poorly to other data. Therefore a flat minimum will generalize better than a sharp minimum.

Ref. [61] suggests reformulating the neural network optimization from

$$\min_\mathbf{w} \mathcal{L}(f_\mathbf{w}) \tag{2.36}$$

which does not distinguish a sharp minimum from a flat minimum, to

$$\min_\mathbf{w} \max_\epsilon \mathcal{L}(f_{\mathbf{w}+\epsilon}) \tag{2.37}$$

which searches for the minimum that has the smallest maximum in its neighborhood ϵ.

With some assumptions, [61] shows that adding an uphill gradient ascent steps and applying the gradient at this position to the original starting point approximates the gradient $\boldsymbol{\nabla}_w \mathcal{L}(f_{\mathbf{w}+\epsilon})$ and can thus be used to solve equation 2.37 by gradient descent.

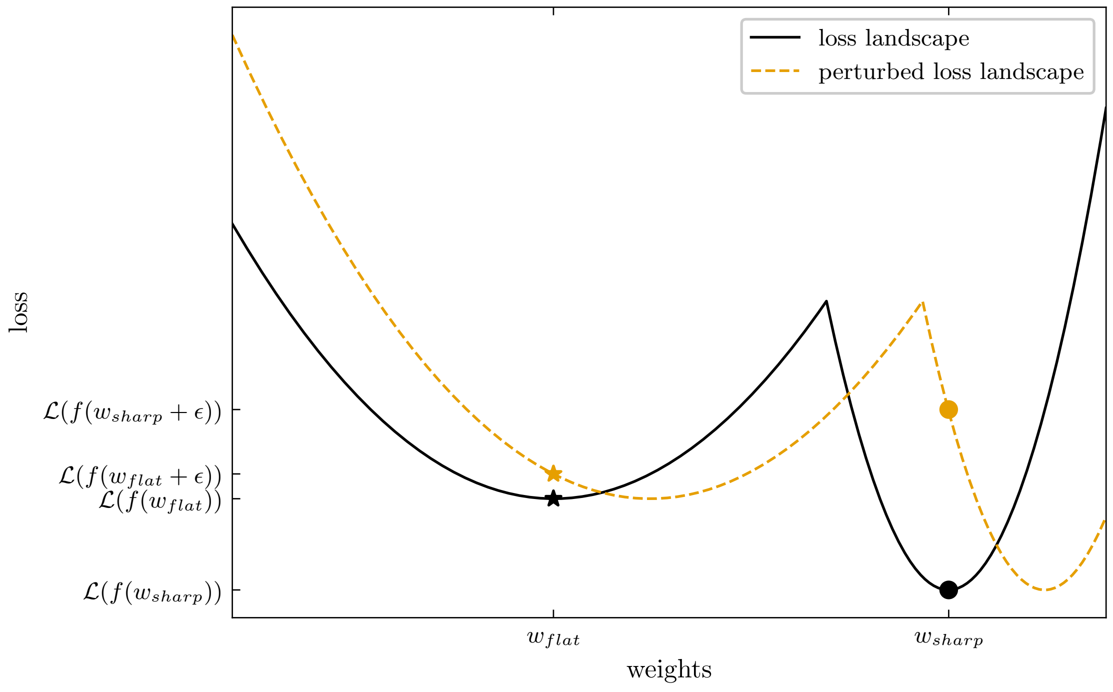


Figure 2.4: Illustration of a loss landscape containing both a flat and sharp local minimum and the perturbed loss landscape for the weight perturbation ϵ. The black star represents the loss of the model $f(w_{flat})$ with the weights corresponding to the flat minimum, while the black dot represents the loss of the model $f(w_{sharp})$ with weights corresponding to the sharp minimum. We see that the perturbation is less detrimental to the model at the flat minimum than the model at the sharp minimum.

Further developments have been made with among others Adaptative-SAM [65] which adapts the maximization region for each parameter depending on their scale. G-SAM [66] maximizes a slightly different objective called surrogate gap showing better generalization. Sharpness-aware minimization adds a forward-backward pass for each epoch resulting in twice the computation for training. Ref. [67] suggests to only maximize a subset of the parameters selected either randomly or by selecting the parameters most sensitive to the sharpness. This method is, therefore, more efficient than the original SAM and the authors report no loss in performance. Stochastic Scheduled Sharpness-Aware Minimization (SS-SAM) [68] updates all parameters but stochastically determines at which training epoch to apply a classic optimization step or the SAM strategy. All those recent advances appeared in the last two years. This field is rapidly evolving and we expect a generalization and combination of the best methods to be developed and added to popular machine learning libraries in the next years. For the work in this thesis, we used the original SAM method.

2.2.8 Dropout

Neural networks are very powerful non-linear models, which can easily suffer from over-fitting. Over-fitting arises when a model performs very well during training but is not able to do as well on test data. The network did not learn the general distribution behind the data but rather memorized every single instance in the training set. Methods to reduce over-fitting rely on the fact that very close data points most likely have similar labels. A slight change in the input should not change the labels too much. One of the methods to introduce slight changes is called dropout [69]. During training, neurons and their connections are deactivated at random. The remaining set of neurons can be seen as a new, slightly different model. For testing, the weights are multiplied by their dropout probability. The final model is a weighted combination of all sub-models generated via dropouts.

2.2.9 Batch normalization

A covariance shift in data is known to hurt the predictive performances of a model[70]. Similarly, the covariance shift of individual layers during the training of a neural network is detrimental to its performance. Ideally, the input of a neural network would be normally distributed. This facilitates training for multiple reasons. The scale of the weights depends on the scale of the inputs. With certain non-linear activation functions (sigmoid, hyperbolic tangent) the derivative can become very small, far away from 0. These issues appear at each layer of the network. Ref. [71] introduces batch normalization. In order to reduce the shift in distribution between layers, their outputs are normalized using the current batch. In parallel, a moving average and standard deviation are computed for each batch seen during training. After this normalization, a learnable shift and scaling is applied.

At inference time, the batch statistics are replaced by the moving average and standard deviation, keeping the output independent of the different batch samples.

Although this covariance shift explanation was stated by ref. [71] and has been accepted by the community since, recent works have challenged this claim [72, 73]. It would appear that the reason batch normalization works is not due to the covariance shift, but no definitive explanation has been found.

2.2.10 Weight regularization

As mentioned in section 2.2.9, the scale of the input matters. Similarly, the scale of the weights is important too. Large weights will cause a sudden change in outputs for small changes in the inputs. This goes against the assumption that two close input points have similar labels. A standard method that applies to many other models, not just neural networks, is to penalize the loss function via the scale of the weights. The regularization loss reads

$$\mathcal{L}_{\mathrm{R}}\left(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}\right) = \mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) + \|\mathbf{w}\| \tag{2.38}$$

with $\mathbf{y}$, $\hat{\mathbf{y}}$ the true labels and predictions. $\mathbf{w}$ the trainable weights subject to regularization of the model. $\|.\|$ a norm, usually L1 or L2. $\mathcal{L}$ is any loss on the predictions.

2.2.11 Pruning

The process of removing connections or neurons from a network, called pruning, can be used to reduce the computational cost of the model with the goal of retaining the same accuracy. Another use of pruning is to reduce the complexity of the model which might lead to better generalization.

In our case, pruning is used as a selection mechanism [74]. Finding the most representative shapelets is difficult, see section 2.1. Our strategy is to use learning, plus initializing a large number of shapelets which are slowly pruned during training.

Tensorflow provides a pruning layer but it cannot be used with the time distributed layer required for our application to time-series. Our implementation consists of using local 1D convolutions of width 1 after each shapelet. The absolute scale of each weight can be seen as the importance of each shapelet. By using an L1 weight normalization we add to the loss of the model a term dependent on the norm of the weights. Large weights will result in a larger loss and smaller weights in a smaller loss. With L1 weight normalization applied, the weights with the least importance will be shrunk towards 0. We then fix the weights to 0 of the smallest weights to reach our desired pruning sparsity rate. The pruning rate is defined as a final sparsity. The sparsity at each training step is given by linear interpolation between no sparsity at the beginning and the final sparsity at the end of the training.

2.2.12 Noise augmentation

As large neural networks tend to over-fit, i.e. learn the training data too precisely, we would be interested in never showing twice the same example. In a perfect world with infinite data and compute resources, we would always feed new data to the network. Unfortunately, with the finite amount of data available for our task, this is not an option.

Data augmentation tackles this issue by altering available data to generate new, slightly different, data points. In the same way a picture of a dog should still be labeled as a dog picture after a 5 degrees rotation, adding small random noise to the data series should not change their label.

In our case, we know that diagnostics can have a 5% error margin. We, therefore, add 5% random noise to every input before the model trains on it.

2.2.13 Label smoothing

Continuing with the goal to increase the generalization of our models, we will now look at the labels. As seen in section 2.2.5 we are not always interested in a model predicting with perfect certainty a few correct labels, and being wrong the rest of the time. We would rather have a model less certain, but more often producing reasonable predictions.

Label class smoothing

To slow down the learning before it reaches a certainty of 1.0 on a prediction we can modify the loss function as seen earlier with the focal loss but we can also modify the labels. For cross-entropy based loss functions, such as the focal loss, the output is a vector of size C, the number of classes. Each element gives the probability of the class, for this particular data point. This one-hot encoded vector created from discrete labels contains either 0 or 1. Label smoothing[22, 75, 76] replaces these hard targets, with the soft target $\frac{\epsilon}{k-1}$ and $1-\epsilon$, k being the number of classes, see figure 2.5.

Time smoothing

We just saw how the target label probabilities are softened for a single data point. Similarly, with time-dependent data temporal smoothing can be applied, as in ref. [77]. When predicting for a single class over time the following probabilities $\{0, 0, 0, 1, 1\}$, the exact time of the transition might not be as important as detecting the event at all. Both $\{0, 0, 0.5, 1, 1\}$ and $\{0, 0, 0, 0.5, 1\}$ predictions would be satisfying. In both cases, the difference to the ground truth is the same. The model would keep trying to fit the training data and its sharp transition better, possibly at the cost of generalization. To avoid it, we apply a Gaussian filter with parameter σ over the label probabilities in time, effectively smoothing out the transition between classes. See figure 2.5.

2.2.14 Hyper-parameter optimization

In machine learning, we distinguish between parameters and hyper-parameters. Parameters are trainable and learned during training. In the case of back-propagation [22], the parameters are learned through gradient descent. Hyper-parameters are usually non-differentiable with respect to the model's output and define among others the model architecture, training process, or data processing.

Machine learning methods as introduced in section 1.5 search for a function f that minimizes or maximizes an objective. The domain to which f belongs depends on the method used. The different methods such as linear interpolation, polynomial interpolation, Support-Vector-Machine, or Neural-Network, all define different function spaces the model f will belong to. The complexity of such models ranges from the simple linear interpolation

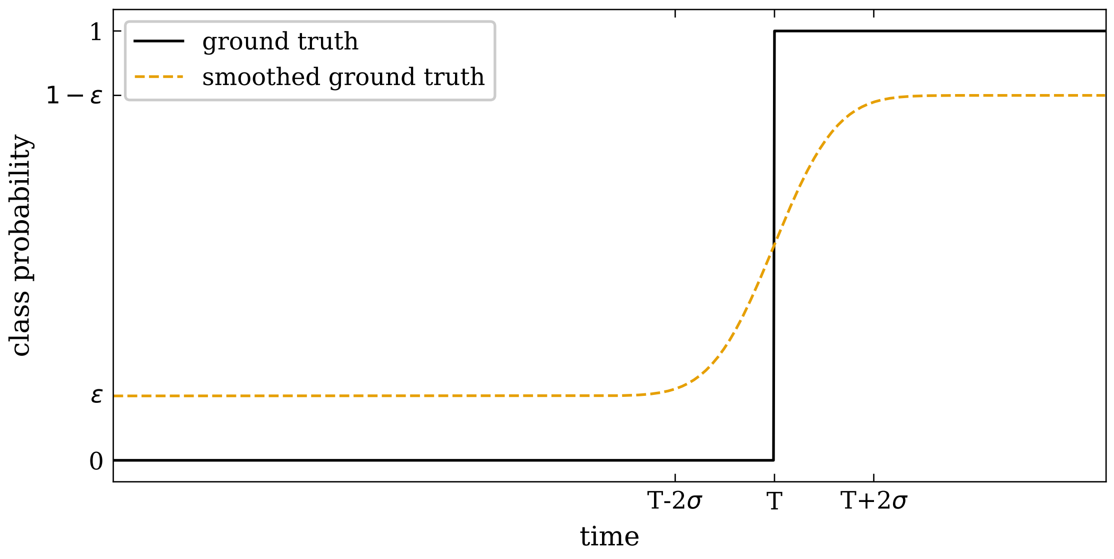


Figure 2.5: Probability of the ground truth class and smoothed probability. Probabilities are weighted per label by a factor ϵ, and in time using a Gaussian kernel of parameter σ.

with 2 parameters to learn and no hyper-parameter, to neural networks with a number of trainable parameters only limited by the hardware and training data available (GPT-3 [29] the largest language processing model from OpenAI has 175 billion parameters) and 10, 100, or even more hyper-parameters. The great flexibility of complex models such as neural networks comes at the cost of a high number of hyper-parameters. The task of hyper-parameter optimization consists of finding the best non-trainable parameters and is still an active field of research today [78–80].

For large models, it is sometimes impractical to try multiple parameters [80], but for smaller models with a smaller computational cost, we can use non-gradient based optimization methods.

A well know gradient-free optimizer is Bayesian Optimization with Gaussian Processes (BO-GP). From a set of points with known objective values, a Gaussian process is built to estimate the value of the objective function at unknown locations. Combining the estimated mean value at each location with its uncertainty, a new point that satisfies both a good exploration of the domain and the search of the optimum is picked. After the evaluation of the new point, a new Gaussian process is fitted and the process is repeated until a stopping criterion is met.

2.3 Tokamak detail

In section 1.2 we introduced today's most promising device to produce net energy and ultimately electricity with nuclear fusion, the tokamak [13]. We gave a simple description of its geometry and how the plasma is confined using magnetic fields.

In this section, we give further information about the tokamak. We present the limiter

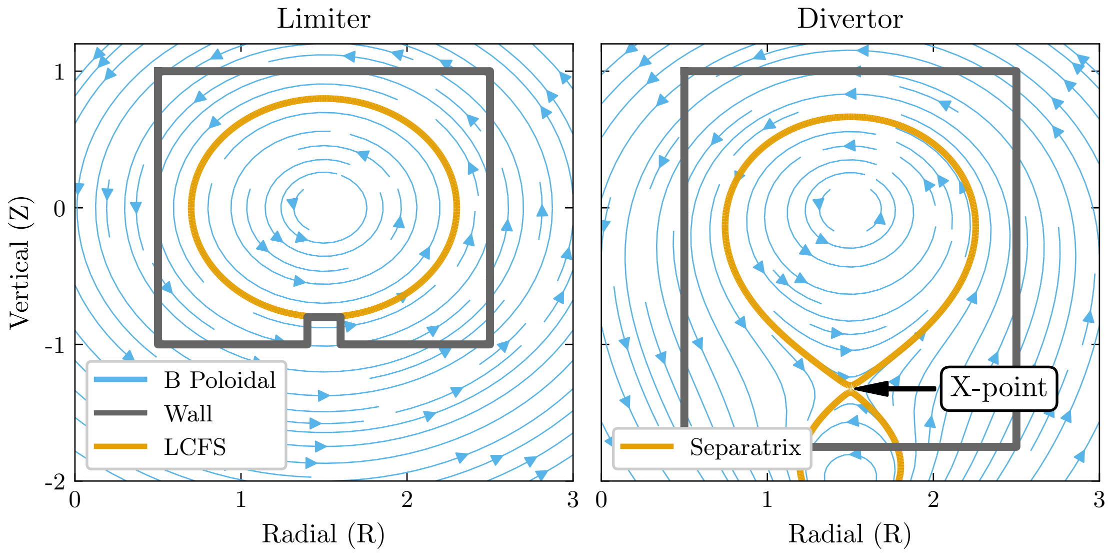


Figure 2.6: Illustration of a limiter and divertor configuration. The code used from [6] approximates the divertor configuration by using two wires and Ampere's circuital law.

and divertor configuration. We then introduce the safety factor which is used to describe the field lines shapes on the flux surfaces and gives stability criteria. We describe the signals used to observe the experiments, i.e. how can measurements be made in the extreme environment that is the inside of a tokamak. Lastly, we give an overview of a tokamak experiment with the current ramp-up/down and flat-top phases.

2.3.1 Limiter and divertor configuration

Tokamaks make either use of a limiter or a divertor. In the limiter configuration, the last closed field line (LCFS) touches the limiter plate, see figure 2.6. The limiter physically restricts the width of the plasma, but the constant collisions with the plasma cause substantial erosion and introduce impurities in the plasma. With a divertor configuration, the plasma's last closed flux surface is not touching the walls or a limiter. The magnetic field is shaped to create a separatrix and one or more X-points, see figure 2.6. The divertor gives an exhaust for particles while keeping the experiment running. The plasma collides with the divertor but contrary to a limiter it is located further away from the plasma core and leads to fewer impurities.

2.3.2 safety factor

As mentioned in section 1.2, the plasma is confined inside a tokamak by the combination of a toroidal and a poloidal magnetic field. The resulting twisted magnetic field lines are characterized by their number of poloidal turns n and their number of toroidal turns m. Their ratio defines the safety factor $q = \frac{m}{n}$.

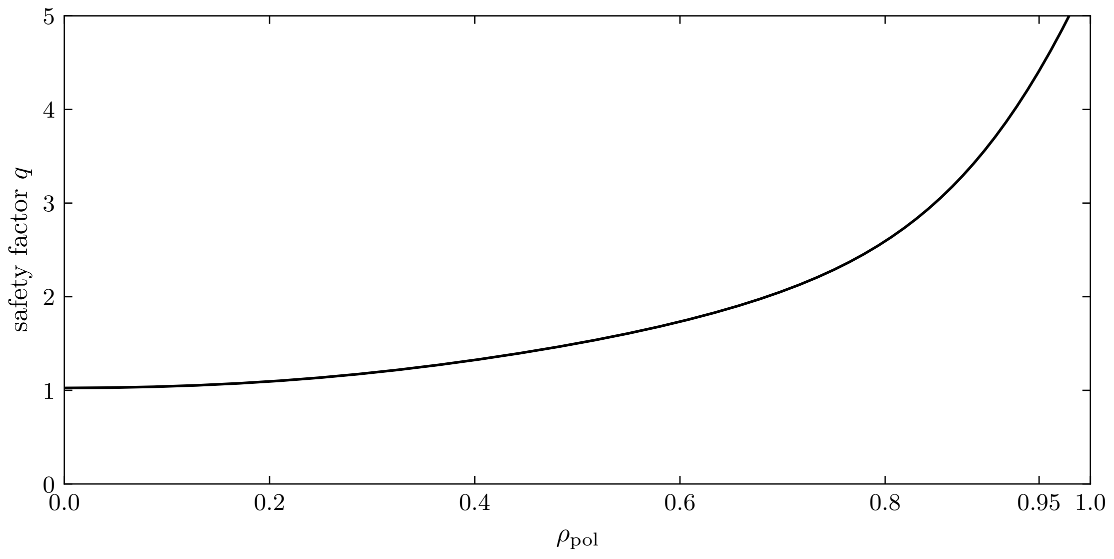


Figure 2.7: Example of safety factor profile q. q is monotonically increasing and tends towards infinity at the separatrix, i.e. normalized poloidal radius $\rho_{pol} = 1$. The reference value often used is q_{95} taken at $\rho_{pol} = 0.95$. In this example $q_{95} = 4.4$. $q = 1.5$ is located at $\rho_{pol} = 0.5$. $q = 2$ is located at $\rho_{pol} = 0.68$. These two radii give to position of the $(3, 2)$ and $(2, 1)$ modes respectively.

In standard cases, q grows monotonically from the magnetic axis, the center of the plasma, towards the edge. At ASDEX-Upgrade typical values are between 4 and 6 at the normalized poloidal radius $\rho_{\text{pol}} = 0.95$.

An example of a safety factor profile is given in figure 2.7. The ρ_{pol} axis used to display the safety factor profile gives the radial coordinate in the plasma from 0 on the magnetic axis to 1 at the separatrix. With ρ_{pol} being calculated as

$$\rho_{pol} = \sqrt{\frac{\Psi - \Psi_{\text{axis}}}{\Psi_{\text{sep}} - \Psi_{\text{axis}}}} \tag{2.39}$$

with Ψ the poloidal magnetic flux and Ψ_{axis}, Ψ_{sep} the values of the flux at the axis and separatrix respectively. This coordinate is better suited to complex magnetic field geometries as seen in figure 2.6 for the divertor case than the standard (normalized) radial distance to the magnetic axis. The distance to the magnetic axis is only appropriate for circular poloidal cross-sections. Later on, we will use the safety factor in the description of instabilities.

2.3.3 diagnostics

A critical element of any experiment is the measurements. We explained how hot the plasma is and the need for a complex magnetic confinement due to the extreme conditions. It follows that no measurement device can be introduced far into the plasma.

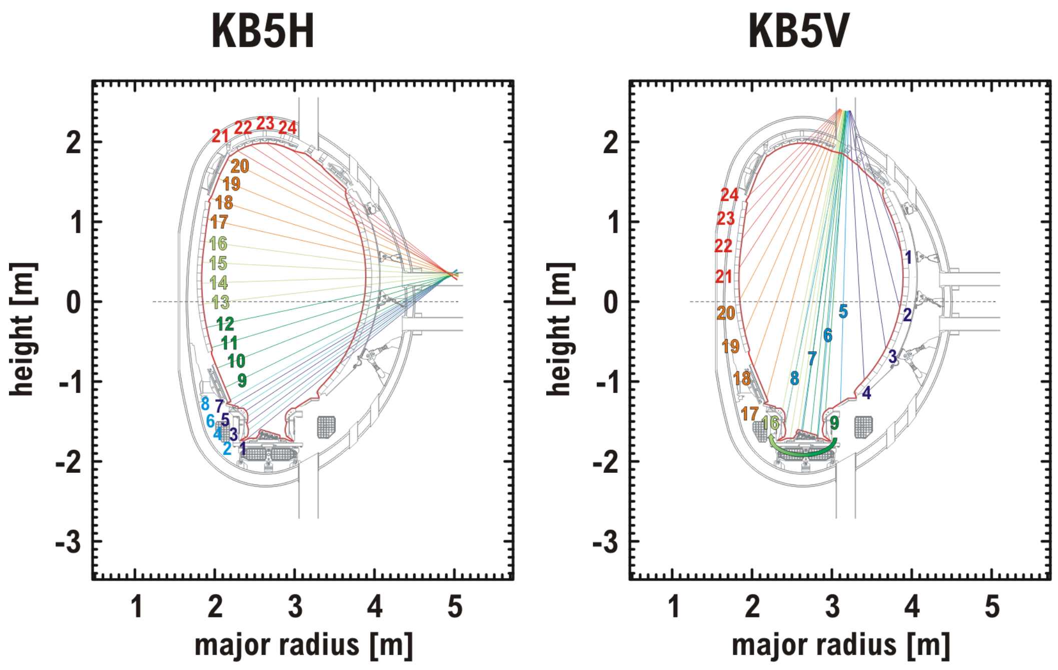


Figure 2.8: Line of sight for the bolometer cameras KB5H and KB5V.

The evaluation of quantities in the tokamak chamber is an active field of research. Many diagnostics in tokamaks work with lines of sight, see figure 2.8. The quantities can be integrated over the line of sight or from specific resonant locations. Tomography is used to reconstruct a solution for a poloidal cross-section from integrated lines of sight[81, 82] Diagnostics are based on laser interferometry, reflectometry, or bolometer cameras among others.

Laser interferometry measures the phase shift of the laser passing through the plasma. It is a line averaged metric used to measure the density of the plasma. Reflectometry uses wave reflections in a similar fashion.

Bolometer cameras work on a different principle. To measure the radiation coming from the plasma, bolometer cameras expose a metallic (gold at JET [83]) circuit to the plasma. As the resistance of metals is a function of the temperature, the resistance in the exposed gold circuit will change with the absorbed energy coming from the plasma.

Additionally, to know which section of the line of sight crossed the plasma the magnetic field must be known. The reconstruction of the magnetohydrodynamics during a discharge [84] is a non-trivial task [85].

The plasma does not behave the same way in all directions. The temperature, current or density gradients are orders of magnitude larger in the radial axis than toroidally. A quantity of interest for many signals is the plasma profile.

As an approximation, some symmetries are used for certain diagnostics. As the particles are trapped on magnetic surfaces, the quantities on magnetic surfaces have very little variations. Similarly, the toroidal dispersion speed is extremely fast. It follows that a one-dimensional radial profile for given quantities gives an accurate representation of the plasma.

Later on, we will use peaking factors. As we want the core of the plasma to be as hot as possible, ideal profiles are often very peaked. If the core cools down, the performance of the plasma is degraded even if the temperature closer to the edge stayed constant. To measure this effect the peaking factor implements the ratio of the quantity at the core to the quantity away from the core. The definition of the core varies. Ref [86] uses $r = 0.25$ as the edge of the core for temperature and density, while $r = 0.1$ is used for radiation.

2.3.4 Current ramp-up, ramp-down and flat-top

To finish the description of the tokamak we will explain the main phases of a discharge. First, the toroidal field is created. Then the elements that will form the plasma are injected into the vessel as gas. Today's experimental tokamaks use helium, hydrogen or deuterium, and in rare cases tritium. Power plants are planned to use deuterium and tritium. Next, the current in the central solenoid is ramped-up to generate the plasma current which induces the poloidal magnetic field. Using the different poloidal magnets the plasma is shaped, the X-point is created and the plasma collides with the divertor at the proper location.

This first phase is called the ramp-up. It is followed by the flat-top phase where the plasma current is at the discharge's predefined value. This is the main phase of the discharge. In a power plant fusion would happen during this phase which would last for hours. At JET a flat-top phase lasts around 20 seconds, while at ASDEX-Upgrade it is less than 10 seconds. Finally, the discharge comes to an end, the plasma current and energy are ramped down and the machine is safely turned off, it is the ramp-down phase. In figure 2.9 the three phases are marked over the plasma current.

Most of the tokamak studies focus on the main flat-top phase, but the ramp up and ramp down still need to be carefully designed to avoid any physical limits [87].

For disruption prediction studies it is common to truncate the discharges to their flat-top phases.

2.4 Disruptions

We presented the ideal development of a discharge but a discharge can become unstable and disrupt before the end of the ramp-down phase. Disruptions can be characterized by three phases. First comes an unstable phase, where the plasma exhibits abnormal behavior compared to normal operations. After these instabilities, a thermal quench and a current quench can occur. They are fast drops in temperature and current respectively. In the following, we give two examples of the destabilizing events that can occur during a discharge before describing the thermal quench and current quench. The runaway electrons that appear after a disruption are introduced next. Finally, we present the three phases which describe the different actions taken to protect the machine and stabilize the plasma, namely disruption avoidance, prevention, and mitigation.

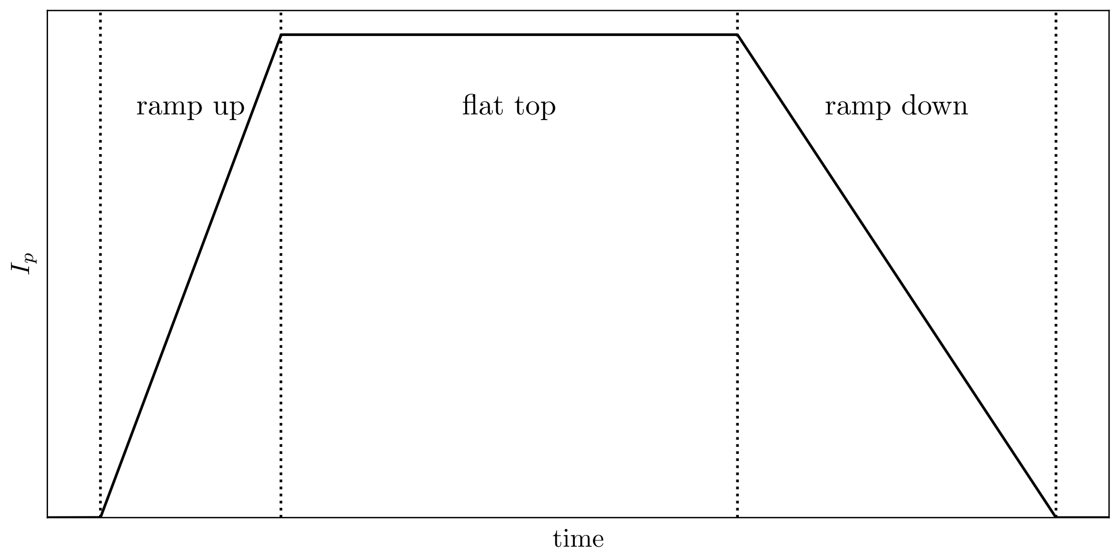


Figure 2.9: Current evolution from start to finish of a discharge. The first phase is called ramp up, where the plasma is formed and the current is brought to its target value. It is followed by the flat-top phase in which fusion would occur. Finally the energy and current in the plasma need to be lowered safely to terminate the discharge, it is the ramp down.

2.4.1 Instability

Before the plasma disrupts, it can show a variety of different behavior. Ref. [88] analyzed over 2000 disruptions at JET between the years 2000 and 2010 to investigate the pre-disruption phase. This work defines four elements to describe instabilities: an event, a chain of events, a root cause, and a class. An event is a single physical or technical event such as the cooling of the edge, or a failure in one of the machines' components. The second definition, the chain of events, is the list of successive events from the start of instability until the disruption. One can note that some chains of events happen more often than others, and some individual events are almost always followed by a few specific events. In other words, although any chain is possible, some paths are more likely to occur than others. The start of the instability phase, i.e. the first event identified in the chain is called the root cause. The root cause is important as it is the earliest event that can be avoided to keep the plasma away from disruptions.

For the disruption classes, ref. [88] clustered identical or similar chains of events. The classes give a higher level of description of the events leading to the disruptions.

There are a few caveats to this study. The authors note that each event *may* lead to a disruption, but does not necessarily. Some events can be overlapping but the chains only show one at a time. The study does not include timestamps for the events. Finally, the list of events is limited to those identifiable from the signals. This last statement seems trivial, but the fact that the plasma is only observed through a set of diagnostics limits any study to the information available from them.

In this study, the distinction is made between physics-related events and technical events. Examples of the technical events are the Vertical Stability control problem (VS) and Neutral Beam Injection problem (NBI). After a vertical stability control problem, very few discharges have had an emergency shut-down (an event called STOP), almost all discharges were followed by a Vertical Displacement Event (VDE) where the plasma moves up (or down) and hits the walls. This chain of events has occurred in both intentional and unintentional disruptions. In the case of an intentional disruption, a vertical stability control problem can be caused to obtain a VDE. Vertical Displacement Events are extensively studied in preparation for ITER. The ITER specifications set the limit of unmitigated VDE disruptions at full power at 1 or 2 for the lifetime of the machine [89, 90].

A problem with the Neutral Beam Injection, the main heating source at JET with a maximum power of 25MW, was always followed by the *Too little AUXiliary power* (AUX) event, which is then followed either by a *density control* (NC) event or a *Radiative Collapse* (RC).

In both VS and NBI events, the following events became physics-related events.

For physics-based events examples are *general (rotating) n=1 or 2 MHD* (MHD), *Internal Transport Barrier* (ITB) or *Impurity Accumulation*. We will explain the MHD event as it is a very common event in the disruptions studied by ref. [88], as well as the impurity accumulation event. The impurity accumulation event was not very common at JET until 2011 as the walls were made of carbon. After the change to a metallic wall, a sharp increase in disruptions due to tungsten impurity accumulation was observed [91, 92].

MHD event

As a note, MHD is the abbreviation of magnetohydrodynamics which refers to the vast field of study of electrically conducting fluids, such as plasmas, and their magnetic properties. We only use the term MHD as the name of the event defined by ref. [88]. This MHD event is one of many instabilities from magnetohydrodynamics that are outside the scope of this thesis. For an in-depth explanation of the magnetohydrodynamics phenomenon we refer the reader to ref. [93] and ref. [94].

The MHD event occurs in many disruptions. It is the presence of tearing modes [93] in the plasma. We briefly present tearing modes and their consequences on confinement in our context of disruption.

Ideally, the magnetic field in the plasma is made of nested closed flux surfaces, see figure 2.10a. Due to resistivity and current or pressure gradients, the field lines can tear and reconnect. The newly created 3-dimensional magnetic structures are called magnetic islands. An illustration of the appearance and growth of a tearing mode is depicted in figure 2.10 where a mode with poloidal mode number $m = 4$ is shown. As islands change the magnetic confinement and impair plasma performance, tearing modes are an important topic of research.

The magnetic confinement constrains the plasma to flow on the flux surfaces. On a flux surface, the pressure and temperature are constant and there is very little radial transport. The temperature profile can be very peaked with high values in the core. With the appearance of tearing modes, the plasma flows on flux surfaces that are no longer at

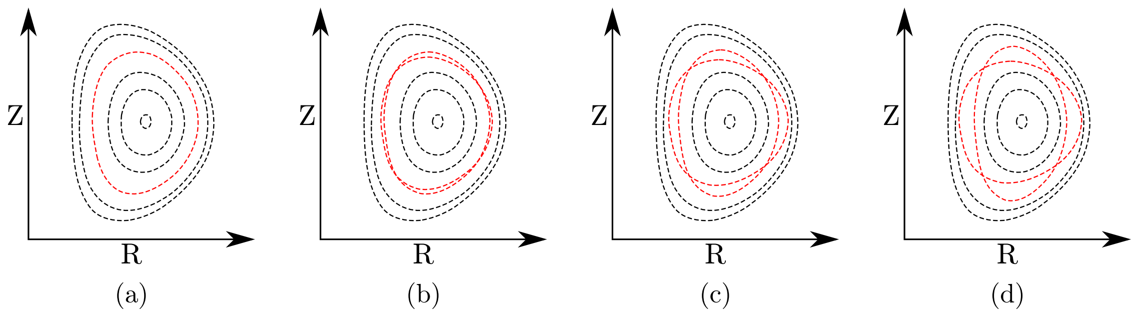


Figure 2.10: Poloidal cuts of the magnetic field showing the growth of mode islands over time from left to right.

constant ρ_{pol} but centered at the coordinate of the resonant flux surface. The temperature is therefore constant on the island creating radial transport and cooling down the plasma, as illustrated in figure 2.11. This figure shows the plateau created from the constant temperature of the islands' flux surfaces. As a result of this plateau, the temperature at the core is lowered.

The tearing modes are characterized by the poloidal and toroidal numbers (m, n) introduced in section 2.3.2. They describe the path of the island toroidally and poloidally, meaning how many turns around the plasma the mode is doing before coming back on itself. The behavior changes according to (m, n).

Although the safety factor profile is continuous, the modes are only located on resonant flux surfaces, i.e. flux surfaces located at rational values of q. In figure 2.12 we illustrate two field lines, one at a rational safety factor $q = \frac{2}{1}$, the other at an irrational safety factor $q = \frac{2\pi}{3}$. As we can see the field line on the resonant surface winds around the torus in a finite number of toroidal turns. A perturbation of the plasma on that closed field line is propagated along this line of finite length. In the case of the irrational q the field line takes infinitely many turns to wind around the torus, hence the perturbation is propagated along an open field line of infinite length making its effect on the plasma very small.

A (m, n) mode is located at the flux surface $q\left(r\right) = \frac{m}{n}$. The safety factor profile being a monotonically increasing function in standard cases, a $(3, 2)$ mode will be located closer to the axis of the plasma than a $(3, 1)$ mode.

The (m, n) mode numbers also give us the tendency of the mode to grow. The smaller numbers require less energy to grow than higher numbers. We will only describe a simple idea to understand this. Considering that traveling along a straight path requires less energy than a curved path, the modes with the sharpest changes in direction will require the most energy. With smaller numbers, we can see figure 2.13 that the path is less sharp than higher numbers, hence requiring less energy for the particles to travel long, and for the same energy the mode can grow bigger.

The risk caused by a tearing mode regarding disruptions is dependent on how close it is to the walls and how big it is. The smaller mode numbers will grow more but be further away from the walls, while the higher mode numbers will grow less and be closer to the wall. As those two properties go against each other, the worst modes numbers are the

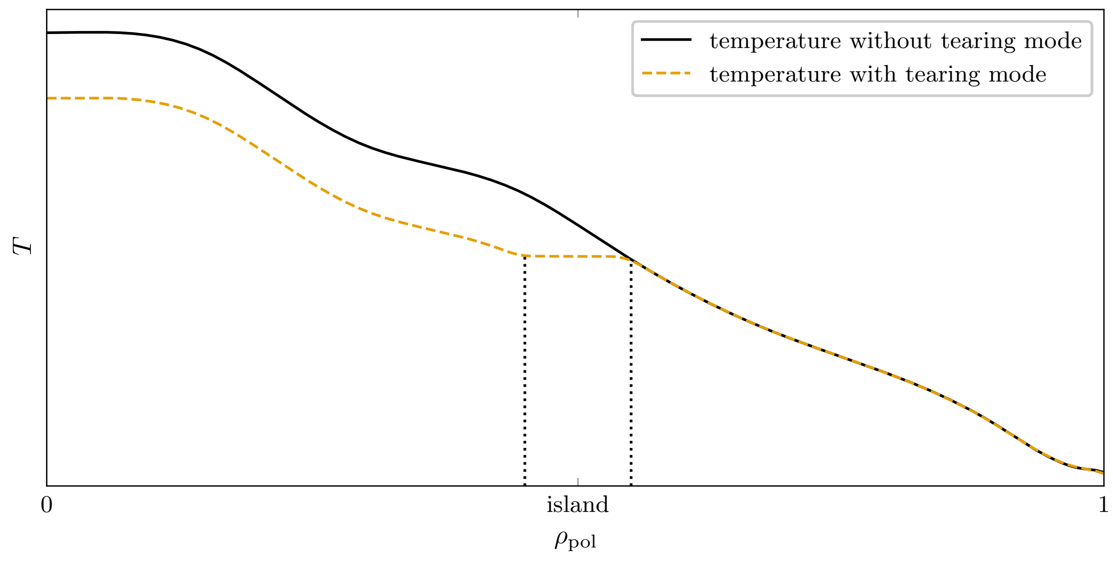


Figure 2.11: Temperature profile of a plasma with and without tearing mode. With a tearing mode, the island causes radial transport which flattens the profile. As a result, the temperature in the plasma core is lower than without tearing modes.

ones balancing a relatively important growth while being close enough to the walls.

Modes start by rotating along with the plasma. Due to the magnetic perturbation they cause, the induced current in the wall causes them to slow down until eventually locking, i.e. not moving with respect to the wall. This is the Mode Locking event which happens at the end of most of the chain of events before the disruption.

Tearing modes are very common and are a good example of the complexity of the event classification. A tearing mode can and will happen simultaneously with other events. Multiple tearing modes with different mode numbers can also co-exist and coupling might appear. Having only one event label per time-slice is therefore an approximation of what is happening in the plasma. Identifying the individual events is therefore harder, and even experts can disagree on some cases.

Impurity Accumulation

The Joint European Torus (JET) tokamak was first built with carbon walls. In 2011 the carbon walls were replaced by metallic walls. As they are the same walls as planned for ITER this new configuration of JET was named JET-ILW for ITER-like wall.

During the carbon phase, a relatively rare impurity accumulation event was identified by ref. [88]. An analysis of disruptions with the metallic walls revealed a new impurity accumulation event [92]. At JET-C (Carbon wall JET), the impurity events were due to low-Z impurities causing edge radiation which differ from the high-Z impurity accumulation causing core radiation at JET-ILW. The low-Z and high-Z distinction refers to the atomic number of the atoms. Carbon has an atomic number Z of 6, tungsten has an

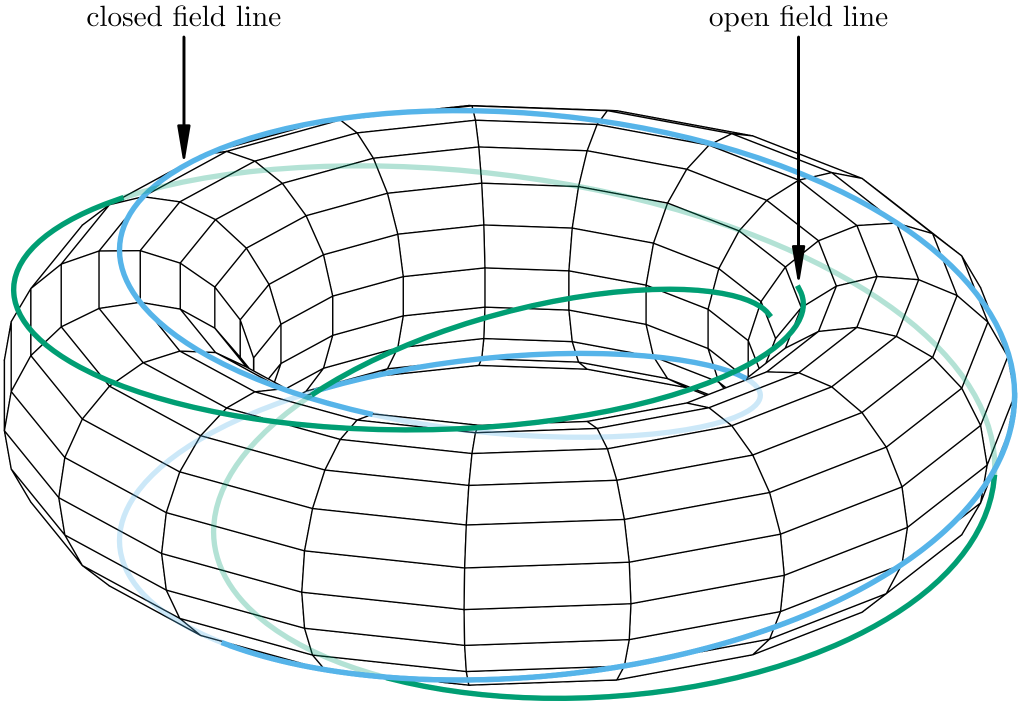


Figure 2.12: Diagram of the field lines for a ratio $q = \frac{2}{1}$ and irrational $q = \frac{2\pi}{3}$ safety factor. The field line for a rational safety factor winds around the torus in a finite number of turns, while the open field line would require an infinite number of turns, hence its name *open*. The torus represents the resonant surface at $q = \frac{2}{1}$ on which the closed field line lies. The open field line with a higher q is further away from the torus.

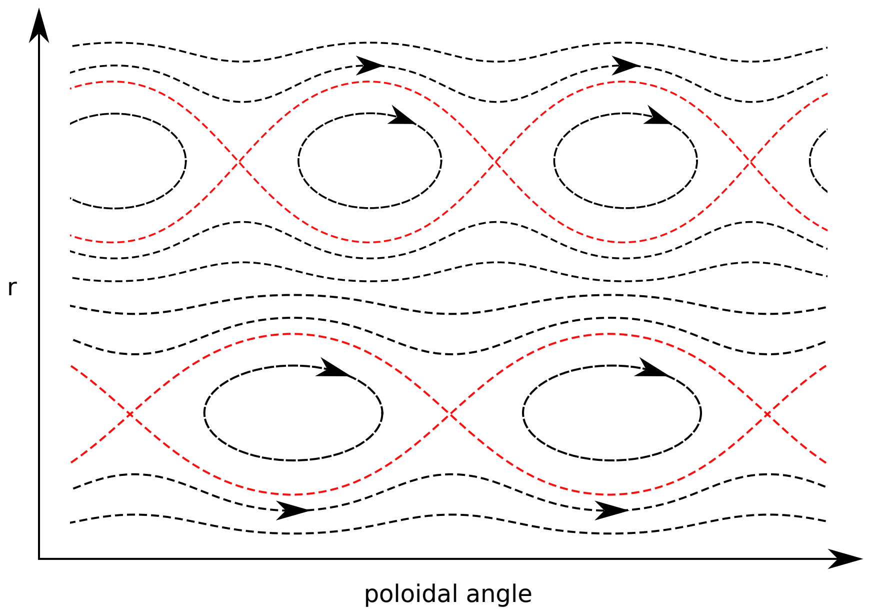


Figure 2.13: Diagram of the field lines in the presence of island chains from two modes, for example $(3, 2)$ at lower r and $(2, 1)$ above at higher radius r. The field lines are taken at a fixed height relative to the plasma axis.

atomic number of 74.

The two impurity accumulation events are clearly different as shown by early works on event classifications at JET-ILW [95].

The high-Z impurity event is the accumulation of tungsten in the core of the plasma. It leads to increased radiation from the core lowering the temperature and increasing the density. The current density profile is broadened resulting in instabilities such as tearing modes. After the locking of the modes the plasma disrupts.

Ref. [96] calculated that the required triple-product τ_ϵ for ignition increases by 20% for a tungsten concentration in the plasma of 3×10^{-5}, while at 1.9×10^{-4} ignition becomes impossible.

2.4.2 Thermal quench

Most of the time the thermal quench is the first major event of the disruption. It consists of a sudden loss of heat in the plasma. A large fraction of the stored energy is expelled towards the walls as heat. The heat loads can and have melted plasma-facing components in existing tokamaks. The short timescale of this event is responsible for the damages caused. For context, the thermal quenches are orders of magnitude shorter than the confinement time τ_ϵ. At JET, the thermal quench is in the orders of milliseconds, while τ_ϵ is in the orders of hundred milliseconds [97, 98].

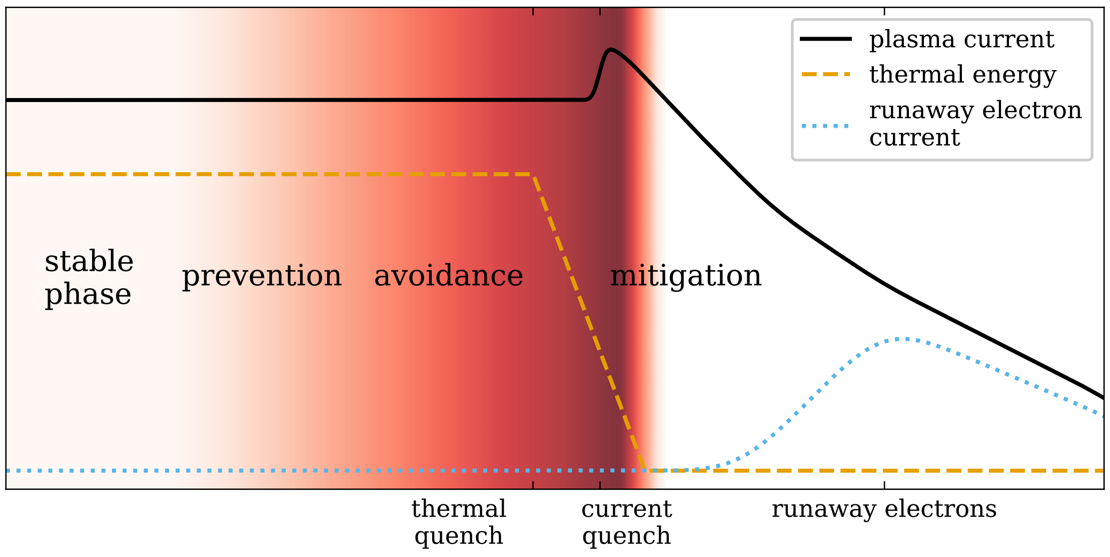


Figure 2.14: Evolution of plasma parameters over time during a disruption. The plasma starts in a stable phase. It transitions to an unstable phase. Disruption prevention aims at controlling the plasma to stay away from instabilities or bring the plasma back to a stable regime. Once the instabilities grow, we might not be able to recover the plasma anymore. Disruption avoidance could decide for a safe shut-down of the machine. Once the disruption is inevitable, mitigation can be employed. The goal of mitigation actions is to reduce the impact of the disruption.

2.4.3 Current quench

The resistivity of the plasma is proportional to $T^{-\frac{3}{2}}$ [13]. As a result of the temperature drop during the thermal quench, the resistivity of the plasma increases significantly and the current decays. During the current quench, the last amount of energy in the plasma is released in different forms. Toroidal eddy current is induced in the walls from the movement of the magnetic field [99]. Similarly, toroidal and poloidal Halo currents [100, 101] appear as the plasma moves towards the wall and makes contact with it. The eddy currents flow exclusively through the walls while the halo current flows partly through the plasma and partly through the walls. As a result of these large induced currents and the magnetic field, significant electromagnetic loads are applied to the tokamak structure. These forces are estimated at hundreds of kilo-Newton during major disruptions at ITER [102].

2.4.4 Runaway electrons

From the Current quench, the electrons' acceleration can surpass the energy loss through collisions. Such electrons accelerate until reaching relativistic speeds, i.e. speeds comparable to the speed of light. These high energetic particles, called runaway electrons (RE), store a significant proportion of the plasma's energy. Runaway electrons with over 50% of the plasma current have been observed at JET [93]. The substantial energy in the runaway electron beam can be lost to the plasma-facing components in just a few milliseconds resulting in damaged and melted wall components [103–105]. The high risk caused by runaway electrons is being heavily investigated but as shown in figure 2.14 they appear after the current quench and managing them is part of the mitigation effort. In our work, we aim at predicting the occurrence of a disruption or disruptive event and we will not consider REs.

2.4.5 Disruption avoidance

As trivial as it is, the first step to avoid a disruption is to avoid the instabilities in the first place. For many reasons, during research, the plasma parameters are pushed into unstable regions. Efforts are being made to identify such regions and include control loops to steer the plasma away from these instabilities [106]. The field of avoidance is tightly linked to plasma control [107]. Attempts to control plasma in tokamaks with highly non-linear models such as neural networks have been made as early as 1995 [108]. There is also a learning curve for the operators of a tokamak. Ref. [88] shows a downward trend of disruption rate at JET from 20% disruption rate in the middle of the 1980s to less than 5% between 2007 and 2009, which can be partly attributed to a better management of the machine.

2.4.6 Disruption prevention

The second step, if the instability region is reached, is to be able to prevent the disruption from happening. This requires the identification of the incoming disruption. As discussed in section 2.4.1, not all instabilities lead to disruptions. Once an incoming disruption is predicted, different actions can be taken to shut down the machine safely.

As for disruption avoidance, the field of disruption prevention has seen in recent years a number of machine learning based approaches [109–117]. But at JET the disruption alarm system is still only based on simple thresholds and does not include machine learning. Similarly, ITER will have multiple alarm systems with some only threshold-based and some allowing less interpretable black-box models.

2.4.7 Disruption mitigation

Lastly, after all of the above, if a disruption would occur we would need to mitigate its damages to a minimum. The disruption mitigation systems include massive gas injection [118] or shattered pellet injection [119, 120] as planned for ITER, see ref. [121]. Both aim at cooling down the plasma by rapidly injecting material into the vessel, therefore, reducing the amount of energy released during the disruption. Additionally, the increased collisionality reduces the appearance of runaway electrons.

Studies are investigating the use of passive coils [122] to disrupt the magnetic field and deconfine the runaway electrons faster before they can accumulate too much energy. A great benefit of this method is that they do not require an alarm system as they do not require to be powered on by a power supply. The disruption will induce the current into this non-axisymmetric coil which will in turn perturb the magnetic field. Hence relying only on physics to operate, without the need for actuators.

3 Spline shapelets and similarities with normalized convolutions

In this chapter, we cover two points. Firstly, we show the formulation of our spline shapelets and benchmark it on the UCR 2018 database. Secondly, we discuss the similarity between normalized shapelets, convolutional layers, and the recently introduced sharpened cosine similarity.

3.1 Spline shapelets formulation

With the change from exhaustive enumeration of shapelets extracted from the data to shapelets learned through gradient descent, a degradation of shapelet similarity to the data has been observed [49, 123]. The interpretability of the shapelets is worsened by the lack of resemblance to the data. Additionally, it is suspected that the generalization of the model could also be lowered. Ref. [124] also notes that using too many shapelets reduces the interpretability, in the same way an ensemble-classifier is harder to analyze than a single model. The authors, therefore, aim at generating a small number of shapelets.

The different techniques to learn data-like shapelets [49, 123] show improved interpretability using complex methods requiring the training of an additional adversarial or regularizing neural network. The adversarial network attempts to differentiate between a shapelet and the data. The main network's goal is to produce shapelets that the adversarial network cannot differentiate from the data while also achieving its initial goal, such as classification. The regularizer network directly modifies the shapelets to fit the data better.

We observe that the shapelets obtained with these methods have a certain regularity that matches the regularity of the data better than standard shapelets. From this observation, we suppose that learned shapelets could resemble the data if they were restricted during training to a subset of functions with a similar regularity as the data. To enforce this regularity directly into the shapelets we propose to use a small number of degrees of freedom and a reconstruction that fulfills the required regularity. This avoids building a parallel discriminator network, making the models much simpler. The resulting shapelets are not as constrained, but will have the desired regularity. Instead of the numerous free parameters of a discriminator network in the case of Adversarial regularization, our only parameters are the number of degrees of freedom and the interpolation method. Additionally, the optimization process of learning the shapelets now applies to the reduced number of degrees of freedom instead of the full-length shapelets, making it more efficient.

3.1.1 Spline shapelet transform

We present the computation of the normalized shapelet metric with our spline interpolation and show that the normalization computation can be done efficiently without reconstructing the full-length shapelets during training.

Section 2.1 defines shapelets and the shapelet transform. Recall that the formulation of the normalized shapelets reads

$$d\left(T,S\right)=\min_{0\leq i<M-N}\sqrt{\sum_{j=0}^{N-1}\left(z\left(T_{i+j},\mathbf{T}_{i:i+N-1}\right)-z\left(S_j,\mathbf{S}_{0:N-1}\right)\right)^2} \tag{3.1}$$

with

$$z(X,\mathbf{X})=\frac{X-\mu_{\mathbf{X}}}{\sigma_{\mathbf{X}}} \tag{3.2}$$

with $\mu_{\mathbf{X}}$ and $\sigma_{\mathbf{X}}$ the mean and standard deviation of $\mathbf{X}=\{X_0,...,X_{N-1}\}$.

We restrict the full-length continuous shapelets $\mathcal{S}$ to splines with n degrees of freedom. There is a mapping $\mathbf{C}\in\mathbb{R}^{m\times n}$ with $n<m$ from the coefficients $\hat{\mathbf{S}}\in\mathbb{R}^n$ of the basis elements of the spline space to the discretized full-length shapelet $\mathbf{S}\in\mathbb{R}^m$.

$$\mathbf{S}=\mathbf{C}\hat{\mathbf{S}} \tag{3.3}$$

for some interpolation matrix $\mathbf{C}\in\mathbb{R}^{m\times n}$ with $n<m$ the number of degrees of freedom. The $\hat{}$ notation is used for the spline weights of a given shapelet. The restriction of the full-length time-series S to $\hat{S}$ is done using the left-inverse matrix $\mathbf{D}\in\mathbb{R}^{n\times m}$ with

$$\mathbf{DC}=\mathbf{I} \tag{3.4}$$

$\mathbf{D}$ exists if and only if the rank of $\mathbf{C}$ is equal to its number of columns and less than its number of rows. We do have $n<m$ by definition of $\mathbf{C}$. If the n splines are chosen to be linearly independent we then have $\operatorname{rank}(\mathbf{C})=n$ and we are guaranteed the existence of the left-inverse $\mathbf{D}$.

Using this interpolation we now show the computation of the shapelet metric. We use upper-case $\mathbf{T}$ and $\mathbf{S}$ for the full-length time-series and shapelets, and the lower-case $\mathbf{t}$ and $\mathbf{s}$ for their normalized version.

Starting from equation 3.1 we have

$$d(T,S)=\min_{0\leq i<M-N}\sqrt{\sum_{j=0}^{n-1}t_{i+j}^2-2\sum_{j=0}^{n-1}\left(t_{i+j}s_j\right)+\sum_{j=0}^{n-1}s_j^2} \tag{3.5}$$

$$d(T,S)=\min_{0\leq i<M-N}\sqrt{n-2\mathbf{t}_{i:i+n-1}\mathbf{C}\hat{\mathbf{s}}+n} \tag{3.6}$$

with $\hat{\mathbf{s}}=\frac{\hat{\mathbf{S}}-\mu_{\mathbf{S}}}{\sigma_{\mathbf{S}}}$ the coefficients for the normalized shapelet $\mathbf{s}$.

Assuming we know the input $\mathbf{T}$, i.e. the shapelet transform is the first layer of our network, $\mathbf{t_{i:i+n-1}}\mathbf{C}$ can be computed before training and stored. Only the dot product between vectors of size n is left to be computed.

The normalization of $\mathbf{T}$ can also be computed in advance. $\mathbf{S}$ needs to be normalized during training, but the full reconstruction of $\mathbf{S}$ is not necessary to obtain $\mu_{\mathbf{S}}$ and $\sigma_{\mathbf{S}}$.

$$\mu_{\mathbf{S}} = \frac{\mathbf{1}_m^T \mathbf{S}}{m} = \frac{\mathbf{1}_m^T \mathbf{C}\mathbf{s}}{m} \tag{3.7}$$

$$\sigma_{\mathbf{S}} = \frac{\mathbf{S}^T\mathbf{S} - 2\mu_{\mathbf{s}}\mathbf{1}_m^T\mathbf{S} + \mu_{\mathbf{s}}^2}{m} = \frac{\mathbf{s}^T\mathbf{C}^T\mathbf{C}\mathbf{s} - 2\mu_{\mathbf{s}}\mathbf{1}_m^T\mathbf{C}\mathbf{s} + \mu_{\mathbf{s}}^2}{m} \tag{3.8}$$

with

$$\mathbf{1}_m^T\mathbf{C} \in \mathbb{R}^n \qquad \mathbf{C}^T\mathbf{C} \in \mathbb{R}^{n\times n}$$

being pre-computed. The m by n matrix $\mathbf{C}$ and its left-inverse $\mathbf{D}$ do not need to be stored.

As $\mathbf{t_{i:i+n-1}}\mathbf{C}$ can be stored and both $\mu_{\mathbf{S}}$ and $\sigma_{\mathbf{S}}$ can be expressed without $\mathbf{S}$, the shapelet metric computation is independent from the shapelet length m during training.

3.1.2 Basis function

The properties desired for the shapelets such as periodicity or constraint derivatives ($f'' =$ 0,...), can be set by the correct choice of basis elements. The full-length continuous shapelet $\mathcal{S}$, with $n+1$ degrees of freedom at positions $\{\tau_0, ..., \tau_n\}$ is given by

$$\mathcal{S}(x) = \sum_{i=0}^{n} f_i(x; \tau_0, ..., \tau_n) \tag{3.9}$$

The interpolation matrix $\mathbf{C}$ is computed by discretizing each f_i into $\{f_i(x_0), \ldots, f_i(x_m)\}$.

For splines, we implemented the Gaussian radial basis function kernel, B-splines, natural cubic splines, and natural periodic cubic splines, see figure 3.1. B-splines can be erratic at the boundaries [125]. The boundaries have contributions from a single node which means, with high degrees polynomials, the second, or higher order derivative can be large which we observed on the shapelets. The natural cubic spline constrains the second and third derivative to zero to address this issue. The Gaussian kernel does not have a compact support which for a small number of degrees of freedom results in small oscillations. As shown in figure 3.1 the Gaussian kernels cannot reproduce the $f(x) = 1$ function and only converges to it with the number of nodes going to infinity. For these reasons, we preferred the natural cubic spline over the Gaussian functions although they performed similarly on our tests. Finally, the natural periodic spline was implemented to illustrate the various properties that can be enforced, but not used on real data as periodicity is not required on our datasets.

3.1.3 Benchmark

To evaluate the new formulation of the shapelets we compare the standard shapelets to our spline-shapelets on the UCR 2018 benchmark [126]. We are particularly interested in the shape of the shapelets learned with both methods and whether or not they correspond

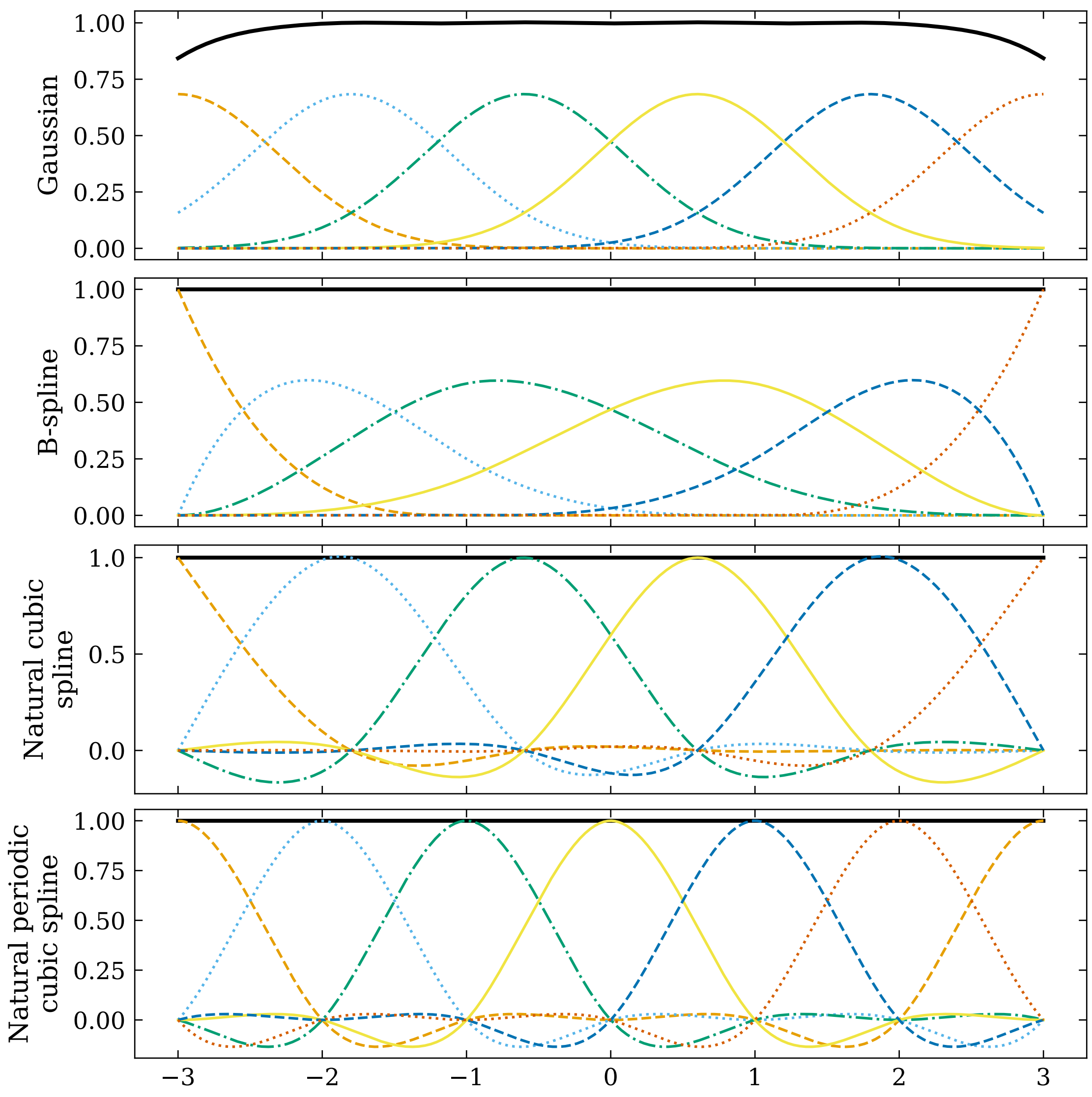


Figure 3.1: Basis functions on an evenly spaced grid with 6 degrees of freedom (DoF). The solid bold black line is the sum of the basis functions. For the Gaussian kernels $\sigma = \frac{\text{DoF}+1}{12}$. B-splines are of the 3rd order. All basis except for the Gaussians can represent the constant function $f(x) = 1$.

to the shapelets obtained by [49, 123]. We hypothesize that the shapelets learned on a reduced number of degrees of freedom perform similarly to the full shapelets, while resembling the data better, assuming the spline and degrees of freedom are chosen correctly.

UCR 2018 datasets

The UCR 2018 benchmark is widely used for testing time series classification (TSC) algorithms. It combines time-series from different domains such as spectrographs, electrocardiographs (ECG), or motion sensors. Some sets include varying length time series, which are padded with NaNs (not a number). We replaced them with zeros. The benchmark is made of 128 datasets and it is recommended to use all of them to avoid cherry-picking. With our methodology, only datasets with at least one class with more than 2 examples in the training set could be used. We, therefore, excluded Fungi, which has only 1 example for each of the 18 classes, and PigAirwayPressure, PigArtPressure, and PigCVP as the three of them only have 2 examples per class. Two datasets are excluded because of their length namely HandOutlines with a thousand training samples of length 2709 and the StarLightCurves dataset with a thousand training samples and 8236 testing samples of length 1024. Our models are applied to the resulting 122 out of the 128 datasets.

For this evaluation the neural networks are kept very simple: a first layer of shapelet transform with two different lengths, followed by the dense output layer. Similarly to the models used in ref. [49, 123], for the classifier model. All neural networks in this thesis are implemented with TensorFlow [127].

Hyper-parameter search

We use Gaussian processes to optimize the number of degrees of freedom, the length of the shapelets, and the learning rate.

The degrees of freedom are limited to the range $[3, 8]$ for all datasets. We arbitrarily chose this range without prior knowledge of the datasets.

In the search space, the two shapelet lengths have symmetrical effects. A network with shapelets length (s_1, s_2) is equal to the network with lengths (s_2, s_1). To avoid wasting half the resources searching the square of possible shapelet lengths, we project it to a triangle [128] such that $s_2 \leq s_1$ while keeping a distribution close to uniform in the triangle. The lengths are calculated as a fraction of the time series in the range $[0.1, 0.7]$. The search space for the learning rate l is $[-5, 0]$ and the learning rate is then set as 10^l.

During the hyper-parameter search, the models are trained for 1000 epochs, with early stopping if no improvement was made in the last 100 epochs. The batch size is set to 8 for all datasets.

The hyper-parameter search initialization uses 30 evaluations with a quasi-random scrambled Sobol sequence. The search is limited to ten hours or 60 total evaluations with a minimum of 3 hours of run-time. The smaller datasets will do many more evaluations than 60 in the 3 hours. Some data sets will reach 60 evaluations between 3 and 10 hours, while the largest datasets will be interrupted at 10 hours with a small number of evaluations. We considered it a reasonable trade-off given the task.

Results

In table 3.1 we report on the performance of our model with plain shapelets, i.e. a number of degrees of freedom equal to the length of the shapelet, and our spline-shapelet. The splines used for the benchmark are the natural cubic splines. For context, we include the accuracy of eight models from the review of deep learning time-series classification algorithms [129]. On average the plain shapelets have a rank of 4.79 and the spline shapelets 5.13. The plain shapelets model places first in 12 of the benchmarks and the spline shapelets in 11 of the benchmarks.

	resnet	fcn	cnn	mlp	mcdcnn	twiesn	tlenet	encoder	Plain Shapelets	Spline Shapelets
ACSF1	**0.916**	0.898	0.334	0.558	0.226	0.592	0.100	0.444	0.570	0.350
Adiac	0.833	**0.841**	0.393	0.391	0.620	0.428	0.023	0.318	0.642	0.642
AllGestureWiimoteX	**0.741**	0.713	0.411	0.477	0.261	0.522	0.100	0.475	0.494	0.510
AllGestureWiimoteY	**0.794**	0.784	0.479	0.571	0.420	0.600	0.100	0.509	0.550	0.556
AllGestureWiimoteZ	**0.726**	0.692	0.375	0.439	0.287	0.516	0.100	0.396	0.394	0.444
ArrowHead	0.838	**0.843**	0.717	0.784	0.678	0.689	0.303	0.630	0.623	0.554
Beef	0.753	0.680	**0.767**	0.713	0.507	0.527	0.200	0.707	0.633	0.667
BeetleFly	0.850	0.910	0.900	0.880	0.630	0.790	0.500	0.620	0.500	**0.95**
BirdChicken	0.880	**0.94**	0.710	0.740	0.540	0.620	0.500	0.510	0.750	0.500
BME	**0.999**	0.836	0.947	0.905	0.896	0.819	0.333	0.827	0.887	0.887
Car	**0.917**	0.913	0.777	0.783	0.700	0.737	0.317	0.587	0.767	0.717
CBF	**0.996**	0.994	0.959	0.869	0.908	0.896	0.332	0.977	0.859	0.991
Chinatown	0.978	**0.98**	0.977	0.872	0.945	0.825	0.726	0.966	0.962	0.953
ChlorineConcentration	**0.853**	0.817	0.608	0.800	0.662	0.554	0.533	0.583	0.571	0.561
CinCECGtorso	**0.838**	0.829	0.749	**0.838**	0.801	0.288	0.250	0.748	0.621	0.686
Coffee	**1**	**1**	**1**	0.993	0.979	0.979	0.507	0.886	0.964	0.964
Computers	0.806	**0.819**	0.539	0.558	0.590	0.641	0.500	0.630	0.648	0.592
CricketX	**0.799**	0.794	0.535	0.591	0.513	0.627	0.074	0.644	0.715	0.641
CricketY	**0.81**	0.793	0.582	0.598	0.521	0.652	0.085	0.639	0.690	0.679
CricketZ	0.809	**0.81**	0.501	0.629	0.484	0.643	0.062	0.651	0.631	0.646
Crop	0.743	0.738	0.670	0.618	0.687	0.489	0.042	**0.76**	0.664	0.630
DiatomSizeReduction	0.301	0.346	**0.954**	0.909	0.646	0.914	0.301	0.880	0.284	0.931
DPOutlineAgeGroup	0.718	0.718	0.758	0.647	0.729	0.705	0.433	**0.761**	0.719	0.748
DPOutlineC	0.770	0.760	**0.772**	0.727	0.759	0.711	0.583	0.724	0.768	0.728
DPTW	0.663	**0.695**	0.671	0.610	0.685	0.591	0.285	0.694	0.691	0.619
DodgerLoopDay	0.150	0.143	0.313	0.160	0.305	**0.593**	0.160	0.488	0.512	0.475
DodgerLoopGame	0.710	0.768	0.816	0.865	**0.877**	0.716	0.478	0.810	0.522	0.522
DodgerLoopWeekend	0.952	0.904	0.974	0.978	0.978	0.954	0.739	**0.983**	0.957	0.971
Earthquakes	0.712	0.725	0.709	0.727	**0.748**	**0.748**	**0.748**	0.740	**0.748**	**0.748**

ECG200	0.874	0.888	0.816	**0.914**	0.838	0.874	0.640	0.884	0.810	0.840
ECG5000	0.935	0.940	0.928	0.930	0.933	0.922	0.584	**0.941**	0.928	0.925
ECGFiveDays	0.966	0.985	0.874	0.973	0.800	0.723	0.497	0.842	0.995	**1**
ElectricDevices	**0.728**	0.706	0.686	0.593	0.653	0.605	0.242	0.702	0.646	0.655
EOGHorizontalSignal	**0.599**	0.565	0.410	0.432	0.424	0.392	0.083	0.337	0.445	0.467
EOGVerticalSignal	0.445	0.446	0.370	0.418	0.361	0.344	0.083	0.319	**0.489**	0.445
EthanolLevel	**0.758**	0.484	0.558	0.386	0.339	0.483	0.250	0.555	0.252	0.672
FaceAll	0.867	**0.938**	0.774	0.794	0.720	0.673	0.080	0.794	0.722	0.762
FaceFour	**0.955**	0.930	0.905	0.836	0.711	0.857	0.295	0.852	0.864	0.898
FacesUCR	**0.954**	0.943	0.873	0.831	0.775	0.641	0.143	0.867	0.886	0.870
FiftyWords	**0.74**	0.646	0.624	0.708	0.611	0.518	0.125	0.658	0.679	0.655
Fish	**0.981**	0.961	0.855	0.848	0.720	0.878	0.126	0.734	0.891	0.931
FordA	**0.937**	0.914	0.896	0.816	0.863	0.555	0.510	0.928	0.923	0.802
FordB	**0.813**	0.772	0.749	0.707	0.698	0.512	0.503	0.777	0.754	0.623
FreezerRegularTrain	**0.998**	0.997	0.987	0.906	0.973	0.946	0.500	0.760	0.992	0.994
FreezerSmallTrain	0.832	0.683	0.739	0.686	0.688	0.917	0.500	0.676	**0.986**	0.784
Fungi	0.177	0.018	**0.961**	0.863	0.051	0.439	0.063	0.934	-	-
GestureMidAirD1	**0.698**	0.695	0.534	0.575	0.518	0.549	0.038	0.528	0.669	0.631
GestureMidAirD2	**0.668**	0.631	0.518	0.545	0.500	0.575	0.038	0.480	0.608	0.538
GestureMidAirD3	0.340	0.326	0.317	0.382	0.278	0.275	0.038	0.368	0.438	**0.50**
GesturePebbleZ1	**0.901**	0.880	0.844	0.792	0.769	0.840	0.163	0.821	0.785	**0.901**
GesturePebbleZ2	0.777	0.781	0.778	0.701	0.720	0.843	0.184	0.796	0.867	**0.873**
GP	0.991	**1**	0.948	0.928	0.907	0.989	0.493	0.784	0.967	0.967
GPAgeSpan	**0.997**	0.996	0.912	0.934	0.887	0.965	0.494	0.890	0.892	0.905
GPMaleVersusFemale	0.992	**0.997**	0.977	0.980	0.952	0.988	0.525	0.978	0.987	0.978
GPOldVersusYoung	**0.989**	**0.989**	0.922	0.941	0.926	0.975	0.524	0.923	0.940	0.921
Ham	0.758	0.707	0.720	0.699	0.718	**0.768**	0.514	0.682	0.733	0.600
HandOutlines	**0.914**	0.799	0.883	**0.914**	0.904	0.661	0.584	0.857	-	-
Haptics	**0.51**	0.490	0.375	0.425	0.399	0.410	0.208	0.394	0.432	0.393
Herring	0.600	0.644	0.531	0.491	0.572	0.625	0.594	0.513	**0.688**	0.625
HouseTwenty	**0.983**	0.982	0.815	0.734	0.741	0.899	0.580	0.901	0.782	0.689

InlineSkate	**0.377**	0.332	0.289	0.335	0.203	0.309	0.156	0.241	0.202	0.196
InsectEPGRegularTrain	0.998	**0.999**	0.716	0.646	0.643	0.881	0.474	0.694	0.739	0.920
InsectEPGSmallTrain	0.372	0.218	0.682	0.627	0.460	0.681	0.474	0.627	**0.691**	0.687
InsectWingbeatSound	0.499	0.392	0.585	0.604	0.587	0.435	0.091	**0.63**	0.598	0.582
ItalyPowerDemand	0.962	0.963	0.954	0.953	**0.966**	0.871	0.499	0.964	0.949	0.958
LargeKitchenAppliances	0.901	**0.903**	0.660	0.470	0.429	0.784	0.333	0.721	0.773	0.723
Lightning2	**0.78**	0.734	0.649	0.682	0.646	0.692	0.541	0.715	0.721	0.672
Lightning7	**0.827**	0.825	0.647	0.616	0.559	0.608	0.260	0.696	0.699	0.740
Mallat	**0.974**	0.967	0.923	0.923	0.861	0.666	0.123	0.818	0.839	0.849
Meat	0.990	0.803	0.913	0.893	0.787	0.970	0.333	0.787	**1**	0.900
MedicalImages	0.770	**0.778**	0.671	0.719	0.627	0.649	0.514	0.664	0.628	0.643
MelbournePedestrian	0.909	**0.912**	0.813	0.863	0.840	0.730	0.100	0.884	0.851	0.851
MPOutlineAgeGroup	0.545	0.535	0.534	0.522	0.558	0.578	0.571	0.577	**0.617**	0.597
MPOutlineC	0.826	0.795	0.744	0.755	0.796	0.743	0.570	0.752	**0.828**	0.801
MPTW	0.495	0.501	0.551	0.536	0.562	0.569	0.286	**0.597**	0.558	0.552
MSRegularTrain	**0.973**	0.955	0.850	0.907	0.860	0.715	0.270	0.818	0.883	0.827
MSSmallTrain	**0.917**	0.893	0.743	0.841	0.709	0.705	0.270	0.751	0.826	0.884
MoteStrain	0.924	**0.936**	0.885	0.855	0.691	0.809	0.539	0.872	0.918	0.909
NIFetalECGThorax1	0.941	**0.958**	0.861	0.915	0.902	0.486	0.029	0.873	0.901	0.870
NIFetalECGThorax2	0.944	**0.953**	0.896	0.918	0.910	0.517	0.029	0.899	0.925	0.892
OliveOil	**0.847**	0.720	0.400	0.653	0.400	0.840	0.400	0.400	0.833	0.800
OSULeaf	**0.98**	0.979	0.482	0.560	0.419	0.628	0.182	0.554	0.628	0.744
PhalangesOutlinesC	**0.845**	0.818	0.799	0.756	0.795	0.656	0.613	0.745	0.735	0.688
Phoneme	**0.333**	0.328	0.095	0.094	0.122	0.132	0.113	0.174	0.108	0.117
PickupGestureWiimoteZ	0.704	**0.744**	0.608	0.604	0.412	0.616	0.100	0.496	0.560	0.540
PigAirwayPressure	**0.406**	0.172	0.061	0.065	0.024	0.163	0.019	0.053	-	-
PigArtPressure	**0.991**	0.987	0.099	0.105	0.029	0.669	0.019	0.097	-	-
PigCVP	**0.918**	0.831	0.071	0.076	0.015	0.519	0.019	0.066	-	-
PLAID	**0.94**	0.904	0.641	0.625	0.625	0.533	0.161	0.728	0.750	0.691
Plane	**1**	**1**	0.962	0.977	0.952	**1**	0.143	0.964	0.981	**1**
PowerCons	0.879	0.863	0.960	**0.977**	0.929	0.852	0.500	0.971	0.939	0.933

PPOutlineAgeGroup	0.847	0.825	0.812	0.849	0.839	0.839	0.488	**0.872**	0.824	0.834
PPOutlineC	**0.92**	0.907	0.807	0.730	0.866	0.817	0.684	0.768	0.835	0.825
PPTW	0.773	0.761	0.777	0.767	0.775	0.784	0.341	0.791	0.756	**0.795**
RefrigerationDevices	**0.53**	0.497	0.437	0.377	0.391	0.510	0.333	0.461	0.395	0.459
Rock	0.552	0.632	**0.88**	0.852	0.628	0.456	0.364	0.784	0.700	0.740
ScreenType	0.615	**0.622**	0.398	0.402	0.406	0.427	0.333	0.414	0.413	0.440
SHGenderCh2	0.824	0.816	0.800	0.822	0.846	0.768	0.470	**0.885**	0.858	0.870
SHMovementCh2	0.439	0.476	0.455	0.435	0.429	0.426	0.167	0.484	**0.496**	0.469
SHSubjectCh2	0.739	0.742	0.790	0.817	**0.841**	0.664	0.200	**0.841**	0.836	0.802
ShakeGestureWiimoteZ	0.880	**0.884**	0.580	0.548	0.516	0.864	0.100	0.756	0.720	0.680
ShapeletSim	**0.782**	0.706	0.497	0.513	0.498	0.546	0.500	0.510	0.500	0.500
ShapesAll	**0.926**	0.894	0.617	0.776	0.599	0.643	0.017	0.679	0.755	0.747
SmallKitchenAppliances	**0.781**	0.777	0.626	0.380	0.471	0.674	0.333	0.773	**0.781**	0.683
SmoothSubspace	**0.98**	0.975	0.976	**0.98**	0.963	0.849	0.333	0.964	0.953	0.940
SonyAIBORS1	**0.961**	0.958	0.690	0.692	0.655	0.725	0.429	0.729	0.429	0.827
SonyAIBORS2	0.975	**0.98**	0.831	0.831	0.804	0.635	0.617	0.844	0.913	0.880
StarLightCurves	**0.972**	0.965	0.926	0.950	0.936	0.849	0.577	0.907	-	-
Strawberry	**0.98**	0.975	0.952	0.959	0.958	0.911	0.643	0.959	0.935	0.911
SwedishLeaf	0.963	**0.967**	0.884	0.845	0.841	0.837	0.064	0.902	0.848	0.842
Symbols	0.893	**0.955**	0.808	0.836	0.644	0.798	0.174	0.754	0.953	0.935
SyntheticControl	**0.997**	0.989	0.987	0.973	0.953	0.879	0.167	0.973	0.970	0.970
ToeSegmentation1	0.957	**0.961**	0.598	0.589	0.559	0.882	0.526	0.706	0.939	0.868
ToeSegmentation2	0.894	0.889	0.752	0.745	0.649	0.794	0.815	0.702	0.869	**0.90**
Trace	**1**	**1**	0.952	0.806	0.902	0.934	0.240	0.740	**1**	**1**
TwoLeadECG	**1**	0.999	0.877	0.753	0.806	0.949	0.500	0.784	0.976	0.991
TwoPatterns	**1**	0.870	0.991	0.948	0.976	0.875	0.259	**1**	0.988	0.945
UMD	**0.99**	0.988	0.960	0.949	0.842	0.835	0.333	0.771	0.951	0.951
UWGLibraryAll	0.861	0.818	0.922	**0.954**	0.928	0.581	0.128	0.937	0.941	0.932
UWGLibraryX	**0.781**	0.754	0.721	0.768	0.726	0.608	0.127	0.771	0.771	0.780
UWGLibraryY	0.666	0.642	0.626	**0.699**	0.639	0.497	0.121	0.676	0.679	0.669
UWGLibraryZ	**0.749**	0.727	0.630	0.697	0.645	0.573	0.121	0.684	0.727	0.700

Wafer	**0.998**	0.997	0.961	0.996	0.992	0.916	0.892	**0.998**	**0.998**	0.991
Wine	0.722	0.611	0.519	0.541	0.500	**0.744**	0.500	0.556	0.611	0.574
WordSynonyms	**0.617**	0.561	0.568	0.599	0.470	0.506	0.219	0.557	0.506	0.495
Worms	0.761	**0.782**	0.416	0.457	0.426	0.475	0.429	0.566	0.506	0.442
WormsTwoClass	0.748	0.743	0.551	0.608	0.590	0.556	0.571	0.683	0.701	**0.779**
Yoga	**0.867**	0.837	0.786	0.856	0.741	0.626	0.536	0.753	0.554	0.536
Average rank	**2.51**	3.25	5.74	5.37	6.71	6.07	9.49	5.23	4.79	5.13
Number of Top-1	**64**	29	6	8	4	5	1	11	12	11
	resnet	fcn	cnn	mlp	mcdcnn	twiesn	tlenet	encoder	Plain Shapelets	Spline Shapelets

Table 3.1: Accuracy on the UCR 2018 benchmark for our model with plain shapelets and spline shapelets along with eight models from the literature. A few names have been abbreviated to accommodate for space restrictions. C:Correct, DP:DistalPhalanx, GP: GunPoint, MP:MiddlePhalanx, MS:MixedShapes, NI:NonInvasive, PP:ProximalPhalanx, RS:RobotSurface, SH:SemgHand, WG:WaveGesture. The average rank includes the 6 datasets we could not evaluate our algorithm on, namely Fungi, HandOutlines, PigAirwayPressure, PigCVP and StarlightCurves. We filled in the accuracy for the shapelet models on these datasets as zero, ranking last by default. For the average rank in this table, we consider that our models rank last on the 6 missing datasets with an accuracy of 0.

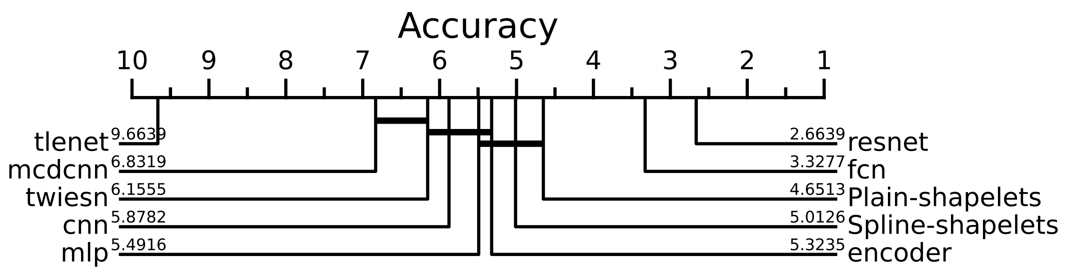


Figure 3.2: Critical difference diagram with Wilcoxon-Holm post-hoc analysis. Horizontal connections indicate models that are not statistically different. We can see that the shapelet models place third and fourth, and are not statistically different on the UCR 2018 benchmark. For this diagram, the six missing datasets for the shapelet models are excluded.

Critical difference diagram

The critical difference diagram with Wilcoxon-Holm post-hoc analysis is given in figure 3.2.

It uses a model pair-wise comparison of ranks to determine models with significant differences. The clusters represented by horizontal connections between models represent models that are not statistically different from one another. As we can see the Plain-shapelets model and Spline-shapelets models are not statistically different in our benchmark. This validates our hypothesis that our splines with a restricted number of degrees of freedom do not significantly change the performance of the model if the correct number of degrees of freedom is used. The correct number of degrees of freedom is intrinsically linked to the dataset and task. We, therefore, include it as a hyper-parameter that is optimized for each dataset individually.

Shapelets analysis

To visualize the shapelets we follow the method used by [49, 123]. They employ the gradient-class activation map (Grad-CAM) [130] to display the most important shapelet per time series. The most important shapelet is the one with the highest explanatory capacity regarding the classification. This shapelet has the highest gradient for the correct classification of the example at hand.

The class activation map uses a linear model after the last convolutional layers of a neural network. The importance of one of the feature maps is given by the scale of the weights in the linear transformation with large weights associated with an important feature map.

Grad-CAM generalizes the class activation map to any architecture following the last convolutional layers. The importance α_k^c of the feature map k to the classification c is defined as the average pooled partial derivative of the class output, with respect to the features map pixels.

$$\alpha_k^c = \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \frac{\partial y^c}{\partial F_{ij}^k} \tag{3.10}$$

In our case with 1D time-series and the shapelet metric with a global softmax, the feature F^k is a scalar. Therefore the index i^c of the most important shapelet to the prediction class c is given by equation 3.11.

$$i^c = \arg\max_k \frac{\partial y^c}{\partial F^k} \tag{3.11}$$

The location of the shapelet in the time series is given by the location of the minimum in the shapelet transform formula.

The partial derivative is computed using the automatic differentiation of TensorFlow. As the gradient depends on the input, the importance is calculated per input time-series.

For a good overview of the impact of the splines, we look at 4 datasets. A first dataset on which both shapelet methods perform well. Two sets where only one of our two models performs well. Lastly, a dataset on which none of the shapelet models perform well.

On the first dataset, GestureMidAirD3, the plain shapelets ranked second out of the 10 methods and the spline shapelets ranked first. There are some slight differences in the learned shapelets, but both captured the data's smoothness appropriately, see figure 3.3.

On the second dataset, BeetleFly, the plain shapelets ranked last out of the 10 methods, while the spline shapelets ranked first. In this case, the plain shapelets did not converge to shapelets resembling the data at all as shown in figure 3.4. The difference is remarkable. The spline shapelets accurately capture the sharpness of the local extremums in the time series.

On the third dataset, Meat, the plain shapelets ranked first, while the spline shapelets ranked fifth. The shapelets learned by the plain shapelet method are much longer than the shapelets from the spline shapelet method, see figure 3.5. At the same length, the restricted number of degrees of freedom would constrain the smoothness too much given the data it is trying to match. It illustrates the importance of learning the appropriate smoothness to produce shapelets with the correct basis.

Lastly, we show a case where both methods performed poorly. On the Yoga dataset, the plain shapelets ranked eighth and the spline shapelets ninth. It seems that both methods simply were not able to do correct classifications even though their shapelets are arguably not far from the data as shown in figure 3.6. The normalized shapelets could simply not be suited to this dataset.

We showed that unconstrained shapelets do not match the original data as well as our spline shapelets if the spline type and degrees of freedom are chosen correctly. The performance difference of the models is not significant, as seen with the critical difference diagram, and both perform quite well compared to state-of-the-art architectures.

Compared to the adversarial regularization introduced by [49, 123], although our constraint is not as strong, we obtain relatively smooth shapelets at no cost. The only parameter to tune is the number of degrees of freedom. Depending on the user's knowledge of the data, it could also simply be set and not searched. With degrees of freedom

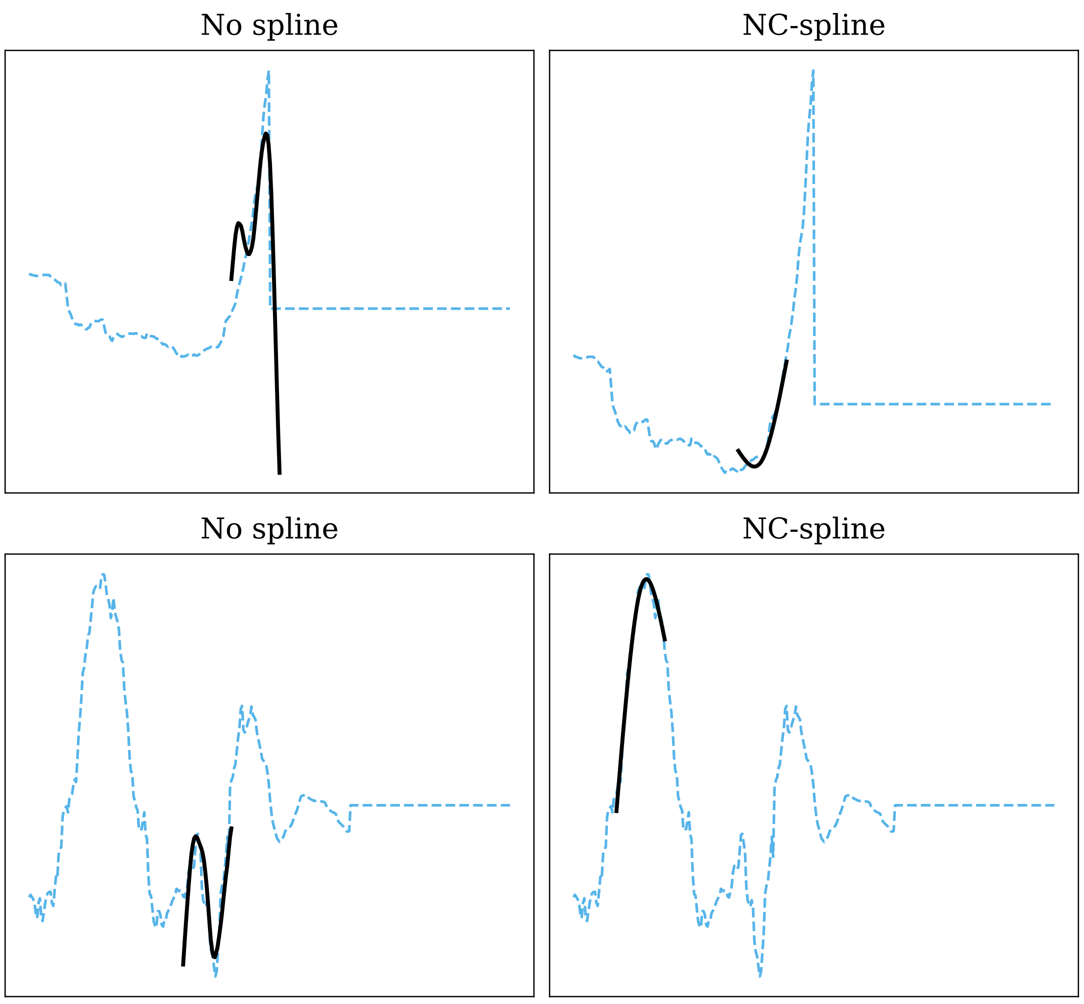


Figure 3.3: Comparison of the most discriminative shapelet (black) on two samples belonging to different classes from the GestureMidAirD3 dataset. The non-constrained shapelet model is on the left and our shapelet model constructed with natural cubic splines on the right.

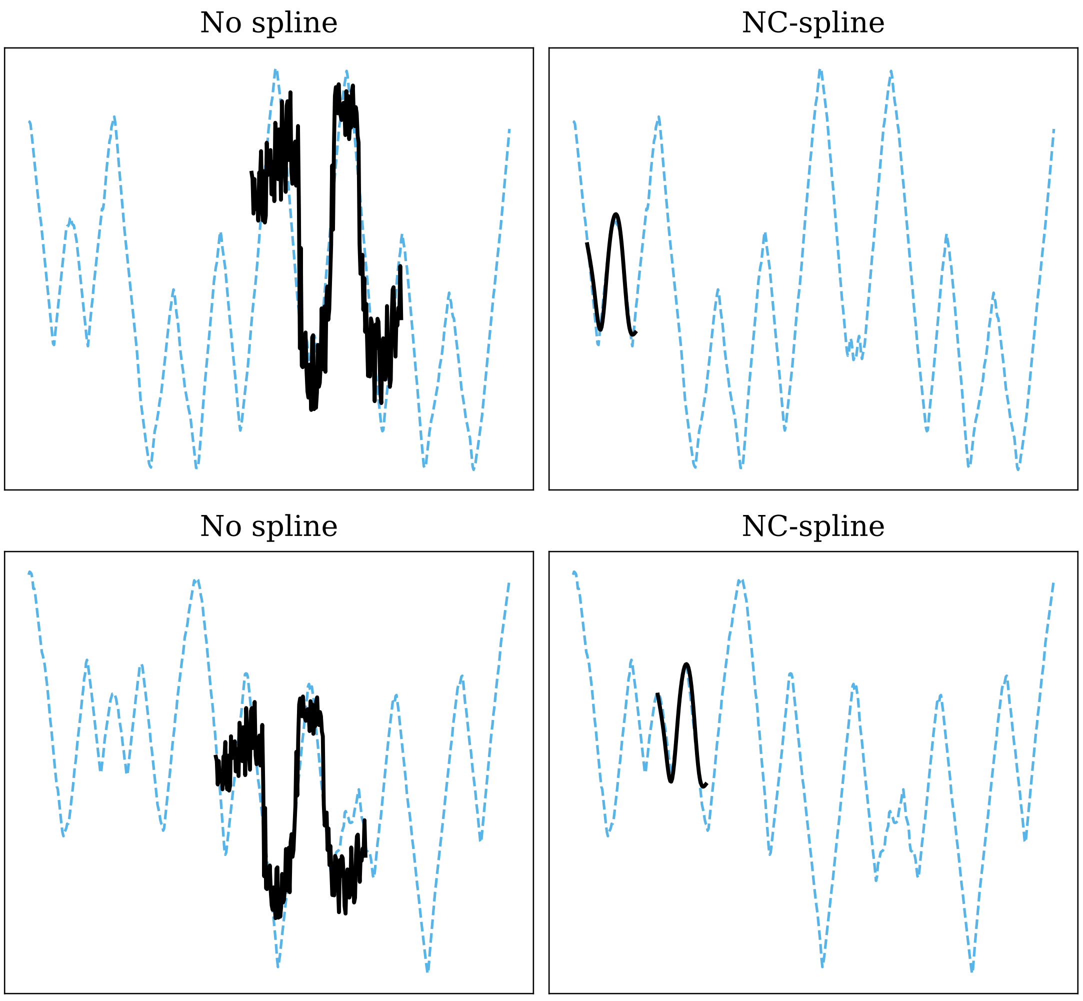


Figure 3.4: Comparison of the most discriminative shapelet (black) on two samples belonging to different classes from the BeetleFly dataset. The non-constrained shapelets are on the left and our shapelets constructed with natural cubic splines on the right.

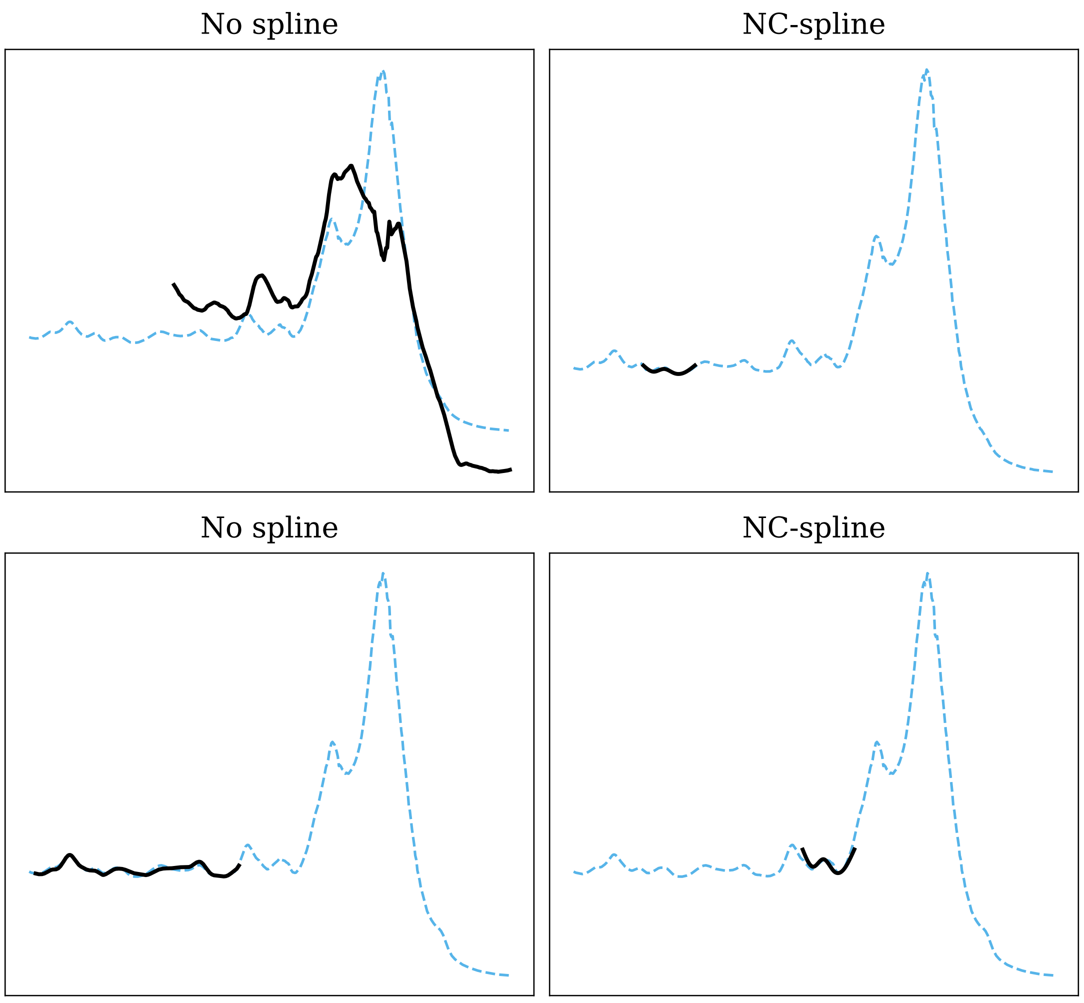


Figure 3.5: Comparison of the most discriminative shapelet (black) on two samples belonging to different classes from the Meat dataset. The non-constrained shapelets are on the left and our shapelets constructed with natural cubic splines on the right.

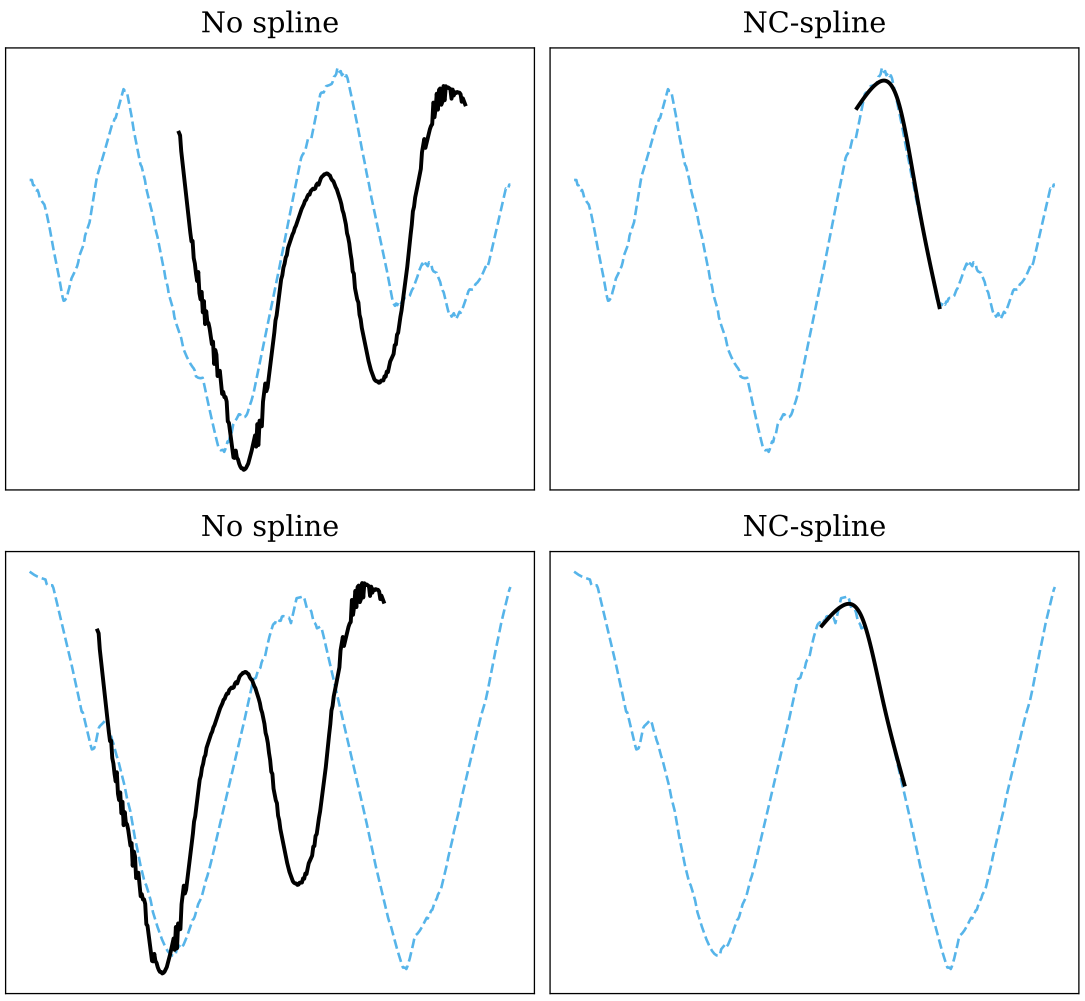


Figure 3.6: Comparison of the most discriminative shapelet (black) on two samples belonging to different classes from the Yoga dataset. The non-constrained shapelets are on the left and our shapelets constructed with natural cubic splines on the right.

much smaller than the length of the shapelet, our method even speeds up the training. The speed-up is due to pre-computation reducing the data from the original space to the spline knots. It is only valid if the shapelet transform is the first layer of the neural network. This restriction is not problematic as to our knowledge shapelets have only been used as first layers.

3.2 Convolutions and Sharpened Cosine Similarity

To finish this chapter we want to address the similarity and differences between shapelets, convolutional layers, and the sharpened cosine similarity. Convolutions [131] improve on the fully connected layer by processing local information which makes them much more efficient on data where data location matters as in pictures, or time series. As our work is focused on time-series and to simplify notations we will only use 1-dimensional convolutions in the following.

3.2.1 Convolutional layer

The convolutions for neural networks convolve the input with a trainable filter producing a feature map as follows

$$y_i = \mathrm{ReLU}\left(\mathbf{w}^T \mathbf{x}_{i-n:i+n} + b\right) \tag{3.12}$$

for a single filter $\mathbf{w}$ and bias b. $\mathbf{x}_{i-n:i+n}$ is the local patch from the input of length $2n+1$. The same $\mathbf{w}$ is applied to each position i. The activation function, ReLU in this case adds non-linearity to the filter.

A pooling operation is used to reduce the resolution of the feature map and make the convolution shift invariant. The pooling operations commonly used are maximum or average pooling. Again, this operation is local with the pooling applied to individual patches.

$$p_j = \frac{1}{2m+1} \sum_{i=j-m}^{j+m} y_i \tag{3.13}$$

for an average pooling of width $2m+1$ over the feature map $\mathbf{y}$. Less commonly used is the global maximum pooling, a maximum pooling over the entire feature map.

For convolutional layers, both the filter and pooling sizes are relatively small with the most common sizes being 3 and 5. Larger receptive fields are achieved by stacking convolutional layers.

3.2.2 Sharpened cosine similarity

In ref. [132] the authors replace the dot product of convolutional layers with the cosine similarity

$$y = \frac{\mathbf{w}^T \mathbf{x}}{|\mathbf{w}|\,|\mathbf{x}|} \tag{3.14}$$

The cosine similarity introduces the normalization by the scale of both the weight vector $\mathbf{w}$ and the input $\mathbf{x}$. Further work investigated the use of the cosine similarity for few-shot learning [133] and zero-shot learning [134]. Ref. [135] introduces the sharpened cosine similarity

$$y = \operatorname{sign}\left(\mathbf{w}^T\mathbf{x}\right)\left(\frac{\mathbf{w}^T\mathbf{x}}{\left(|\mathbf{w}|+q\right)\left(|\mathbf{x}|+q\right)}\right)^p \tag{3.15}$$

which introduces the sign $\left(\mathbf{w}^T\mathbf{x}\right)$ to preserve the sign of the cosine transform, as well as the trainable coefficients p and q.

The parallel development of the normalized shapelets and the sharpened cosine similarity shows the need for robust and transferable models for complex applications such as zero-shot, few-shots, and transfer learning.

All three methods, convolutions, sharpened cosine similarity, and shapelets identify local features by sliding a kernel over the data. They use pooling methods to reduce the scale of the feature map and implement different normalization schemes. If one standardizes the convolution layer operation we obtain

$$\frac{\left(\mathbf{w}^T-\mu_{\mathbf{w}}\right)\left(\mathbf{x}-\mu_{\mathbf{x}}\right)}{\sigma_{\mathbf{w}}\sigma_{\mathbf{x}}} \tag{3.16}$$

which is to an affine transformation and a square root equal to the normalized shapelet operation

$$\sqrt{(n-2\mathbf{ts}+n)} \tag{3.17}$$

from equation 3.6 as $\mathbf{t}$ and $\mathbf{s}$ are also standardized. The indices are ignored for simplicity.

The cosine similarity brings both $\mathbf{w}$ and $\mathbf{x}$ to unit vectors while standardization makes them 0 mean and standard deviation 1. The shapelets are traditionally longer than the kernels of convolutional layers or cosine similarity. Shapelets are supposed to capture the discriminative features from the input data in their entirety while the convolution layers stack multiple layers with small kernels to capture larger features through the combination of the layers. The resulting networks with many layers are referred to as deep neural networks. Tuning the number of layers is the equivalent of tuning the length of the shapelets.

The shapelets used in this work are therefore very similar to traditional convolutional layers except for the length of the kernels and the normalization. The normalized shapelets use a different normalization as the sharpened cosine similarity but both attempt to increase the feature extraction generalization.

4 Disruption prediction on JET data in its carbon wall configuration

The first work for this thesis focused on the application of shapelet-based neural networks to disruption prediction, targeting early signs of disruptions for prevention. We compared the predictive capacity of the shapelets to different state-of-the-art methods from the literature. We chose two models. A first model with already a lot of research and optimization [109, 110, 136–139], albeit for the slightly different task of disruption prediction up to 10 milliseconds before the disruption. This system was built to detect as many disruptions as possible, without the goal to detect early instabilities. The second model represents a more modern approach making use of state-of-the-art neural networks[111]. We selected those two architectures because they both proved to be effective at their tasks, and are very different. This allows for a good comparison to our models.

A difficulty with any disruption study is the complexity and time-consuming task of data labeling. It takes experts a substantial amount of time to identify the different events that lead to a disruption. We, therefore, relied on the work of Peter de Vries [88] who manually analyzed over two thousand disruptions at JET. The choice of tokamak and time period is restricted to the experiments present in ref. [88]. The data used for this study comes from the Joint European Torus (JET) tokamak in its carbon-wall configuration.

4.1 Disruption prediction and disruption type identification

The prediction of incoming disruptions has been investigated at different tokamaks. As there can be large differences between tokamaks, predictors are not easily applied to different machines. This is why most research effort has been conducted using distinct datasets, making comparison difficult. We now briefly describe a few of the methods that have been explored.

The Advanced Predictor Of DISruption (APODIS) [109, 110] system is developed at JET. It uses support-vector machines (SVM) on specifically designed features. It performs very well for short-time predictions and its architecture allows it to be extremely fast, which is a necessary condition for real-time predictions. The APODIS predictions are available in the JET database but they are not currently actively used to interrupt discharges. One of our models implemented for comparison is based on the APODIS architecture. We describe it in further detail in section 4.3.1.

Ref. [140], also developed at JET, uses Generative Topographic Mapping (GTM) to cluster the time-slices. It then identifies the region with stable plasma conditions, disruptive ones, or different types of disruptions. This mapping can be used to project new discharges and the type of the closest nodes are used to classify the new data. A

2-dimensional projection is used which limits the model's performance but allows for excellent visualization of the clusters and projections. To obtain predictions of the levels of the less interpretable APODIS model, this GTM-based approach uses a quite long assertion time of around 200 milliseconds, a detection threshold that depends on the time, and adds a threshold based on the locked-mode signal.

Trees and random forests have been used at the Alcator C-Mod, DIII-D, and EAST tokamaks [113, 141–143]. These studies present an extensive analysis of the results between the different machines.

Although neural networks have seen a rise in popularity in recent years, disruption prediction models using multi-layer perceptrons were some of the first machine-learning models used for disruption predictions [144, 145]. The early 2000s did not have the resources available today and recent work showed impressive results using deep recurrent neural networks on enormous datasets [111, 146].

In parallel with these efforts on disruption prediction, the question of disruption type identification was addressed. Knowing that a plasma will disrupt is required to protect the machine but does not allow to recover the plasma, or take specific actions other than stopping the experiment. The identification of the disruption types brings insight into the predictions, can help the post-discharge analysis, and could be used to recover the plasma through targeted actions instead of stopping the experiment.

Generative Topographic Mapping has also been used for this purpose, see ref. [95, 147]. Since the clusters are identified in an unsupervised manner, the disruption types can easily be included by simply labeling the training time-slices. This work makes use of the classes identified by ref. [88]. Similarly, using the geodesic distance on Gaussian manifolds, ref. [148] predicts disruption types at JET with 85% accuracy compared to the manual classification from ref. [88]. An important difference to note between these works and ours is the inclusion of the disruption type classification into the disruption prediction model. The disruption type identification models proposed in the literature often rely on a secondary model to first detect disruptive time-slices.

4.2 Dataset

The main dataset in this chapter pre-dates the installation of the tungsten ITER-like wall (ILW). The vast majority of the discharges classified by ref. [88] come from the years 2000 to 2008. The dataset contains a chain of events for each disruption. Those events are ordered but do not have timestamps. From unintentional discharges, i.e. not triggered by the experimentalists, we selected those starting from one of the eight most common root causes: edge localized modes (ELM), Greenwald limit (GWL), internal transport barrier (ITB), low density (and low q) (LON), Edge q close to rational (> 2) (QED), negative central magnetic shear (MSH), MHD, and neoclassical tearing mode (NTM). As we do not have timestamps and therefore do not know when the root cause started and ended we decided to label the discharges by their last event, as shown in table 4.1.

This results in ten labels namely the vertical displacement event (VDE), mode locking (ML), radiative collapse (RC), general $n = 1, 2$ MHD instability (MHD), internal kink mode (KNK), low-q or $q_{95} \sim 2$ (LOQ), Multifaceted Asymmetric Radiation From the

Discharge	Chain of event	Selected	Assigned label
58167	ITB → PRP → KNK	Yes	KNK
58172	ITB → PRP → MHD	Yes	MHD
59426	NTM → VST → VDE	Yes	VDE
61167	NTM → ROT → ML	Yes	ML
62513	ELM → MHD → ML	Yes	ML
61636	HUM → LON → ML	No	-

Table 4.1: Example of the data selection process and label assignment for different discharges at JET. Discharges are first selected according to their root cause. The 8 most common root causes between 2000 and 2008 were used: ELM, GWL, ITB, LON, QED, MSH, MHD, and NTM. The label assigned is then the last event identified before the disruption. HUM is the label for the human error event. For the other events not specified and for the full description of the chain of events please refer to ref. [88].

Class	Train	Validation	Test
safe	20	90	90
VDE	9	3	4
ML	43	15	15
RC	9	3	4
MHD	5	2	2
KNK	16	6	6
LOQ	4	2	2
other	2	2	2

Table 4.2: Number of discharges per label for the train, validation and test split. *Other* includes the following types: MAR, WAL, NTM, RCY.

Edge - MARFE (MAR), plasma too close to the wall (WAL), Neo-classical tearing mode (NTM), and high recycling (RCY). We will use a train, validation, and test split for the training and evaluation of the different models which means that at least three discharges per label are required. Four labels did not meet this requirement: MAR, WAL, NTM, and RCY. Instead of removing them, we aggregated the discharges under a single *other* label. It will make the classification harder but such a situation can arise when building models for ITER.

The data is split into three subsets. The models train on the train set, the hyperparameters are tuned using the validation set and the performance of the trained models is evaluated on the test set. The number of discharges per set is listed in table 4.2.

4.2.1 Time to disruption

Correctly identifying the start of the unstable phase preceding a disruption is another difficult task that illustrates the complexity of disruption prediction. Although the disruption time can be defined with a certain change of current, $T_d = 5\,\text{MA/s}$ at JET, the start of the unstable phase does not have such a definition. We list a few approximate criteria from the literature. Ref. [109] defines as unstable the last 90 ms before the disruption time T_d. The authors exclude all time-slices belonging to $[T_d - 1000, T_d - 90]$ from the training set, and uses the time-slices prior to $T_d - 1000$ as stable. Safely terminated discharges are also used for stable samples.

Ref. [95] uses the last 210 ms of disruptive discharges for the unstable phase. In contrast to these small time windows close to T_d, [111, 146] uses the last 10 s of disruptive discharges as unstable. Using such a long time window is only made possible by the large amount of data used by the authors. To place these time scales into context a single discharge lasts from a few seconds, up to twenty seconds at JET, and the thermal quench and current quench are of the orders of milliseconds and tenths of milliseconds respectively. Ref. [149] provides details on the timescale and energy losses in disruptions at JET.

Ref. [150] developed a method for automatic detection of the start of the unstable phase using statistical indicators. It is a promising approach but we found that on our data there is still a threshold to be optimized for each discharge individually instead of on the global dataset. The original method used signals that are not available on our discharges, for example peaking factors computed from newer bolometer cameras [83], therefore our unsuccessful experience might be related to the discrepancy in signals.

For our prevention task an earlier disruption time $\tilde{T_d}$ is manually selected for all discharges which is at least 17 ms earlier than T_d and on average 123 ms. the previous two seconds from $\tilde{T_d}$ are considered unstable. Up to $\tilde{T_d} - 2\,\text{s}$ all time-slices are considered stable.

4.2.2 Signals

Today's tokamaks, as experimental devices, have tens to hundreds of diagnostics [151]. The plasma state is observed through the various signals coming from the many diagnostics varying in accuracy, frequency, localization, and so on. Not every diagnostic is useful for every task, hence the first step of any data-driven approach in fusion is to select the appropriate diagnostics.

The signals for this dataset were selected with a few requirements. They had to be close to the signals used by other work in the literature, such as the signals from APODIS see ref. [109, 110]. We made sure the signals were available both at JET and ASDEX-Upgrade, to prepare for potential follow-up work on a cross-tokamak application. Ideally, the signals would be accessible in real-time to give a good overview of the capabilities of the models for a live application in a tokamak.

At our stage of research and development, it is not of first importance to have real-time signals as we are not investigating if our models can be applied in a live environment, but whether or not the method works. Also, having post-experiment analysis tools is a must to prepare for future discharges and design future tokamaks. Therefore a good interpretable disruption prediction model can be useful even if it does not only make use of real-time diagnostics.

We selected seven signals: plasma current, locked mode activity, safety factor q95, plasma inductance, height of magnetic axis, core-line integrated density, and normalized beta. The only signal we did not mention yet or is not necessarily understandable from the name outside the field of fusion is the normalized beta.

β is a measure of the plasma performance, see equation 4.2.2, defined as the ratio of the mean plasma pressure $\langle p \rangle$ and the magnetic pressure $\frac{B^2}{2\mu_0}$ where B is the mean total magnetic field strength and μ_0 the magnetic permeability in a vacuum.

$$\beta = \frac{\langle p \rangle}{B^2/\left(2\mu_0\right)} \tag{4.1}$$

The normalized beta β_N, see equation 4.2.2, is normalized with the minor radius a, the toroidal magnetic field B_T, and the plasma current I_p.

$$\beta_N = \beta \frac{aB_T}{I_p} \tag{4.2}$$

It is used to define how close the plasma is to physical limits as the Greenwald density limit [152] or magneto-hydrodynamic instabilities [93].

As some signals exhibit different behaviors during ramp-up and ramp-down compared to the flat-top phase, and disruption prediction is mainly focused on the flat-top phase we excluded the ramp-up and ramp-down on each discharge.

The signals are uniformly re-sampled with a 500Hz sampling frequency as it matches the cycle frequency of the JET real-time network (ATM) [153]. The resampling is causal meaning that the new value s_i of a signal at time t_i is computed using the previous values $\{s_0, ...s_i\}$ and not values from later time-steps. In this way, we are not leaking information from the future to current time-steps.

The data is standardized with mean 0 and standard deviation 1 using the training set. The mean and standard deviation of the validation and test set are scaled using the training set's values.

4.2.3 Problem formulation

We formulate the classification problem as a sequence-to-sequence problem. For each vector $x_t^i \in \mathbb{R}^7$ of discharge i at time-step t, we want to assign a discrete label $y_t^i \in Y$ with the sample space $Y = \{\text{safe}, \text{disruptive}\}$ (or $Y = \{\text{safe}, \text{VDE}, \text{ML}, \text{RC}, \text{MHD}, \text{KNK}, \text{LOQ}, \text{other}\}$). The labels are first one-hot encoded, turning a discrete target y_t^i into a vector $\hat{z}_t^i \in \{0,1\}^2$ (or $\hat{z}_t^i \in \{0,1\}^8$). $\hat{z}_t^i$ is the probability of each label with their sum being equal to one. The discrete probabilities are then smoothed as described in section 2.2.13 resulting in the continuous target probabilities $z_t^i \in [0,1]$. The three models trained for this study approximate $f \in C\left(\mathbb{R}^{\text{in}}, \mathbb{R}^{\text{out}}\right)$ such that $f(x_t^i) = z_t^i$. $\mathbb{R}^{in}$ will be described for each model individually. $\mathbb{R}^{out}$ is either $\mathbb{R}^2$ or $\mathbb{R}^8$ for the binary and multi-class case respectively.

4.2.4 histogram

In order to observe the 7-dimensional data and compare the different classes, we start by looking at the one-dimensional histograms. Figure 4.1 displays the histogram of the 7 signals split between the safe time-slices in blue, and disruptive time-slices in red.

The top and bottom 1% of the data have been discarded to build the histograms. The data is made of over 90% of safe time-slices we, therefore, displayed the density of the histograms to enhance visualization.

As we can see from the 6 signals, excluding the locked mode amplitude, the values taken by both classes are almost the same, albeit with a different, but non-zero, probability.

The locked mode is slightly different. Around 0.0004T the density of the safe time-slices in blue becomes extremely small and goes to 0 for higher values, while the density for disruptive samples remains between 0.2 and 0.3%. But the majority of the values from the disruptive time-slices still have a similar distribution to the stable time-slices.

It is well known that the locked-mode amplitude can be used very effectively to predict disruptions with a small warning time[154, 155]. As discussed in section 2.4.1 the vast majority of disruptions at JET end with a locked mode. Since this is the last event before disruption and we consider the last two seconds of a disruptive discharge as unstable, the histogram matches the expectations. The data from the first 1.5-1.9 seconds of these signals are still in the range of stable operations, although other instabilities might already have appeared. Only a few time-slices, the last ones, do not follow the stable distribution. As we are interested in earlier signs of incoming disruptions, we included a wider time window as the disruptive region than the time window for which the locked mode amplitude becomes larger.

Figure 4.1 shows a trend with higher normalized β and lower plasma internal inductance L_i being linked to more disruptive than safe discharges. Again, this holds proportionally to the safe discharges but cannot be used alone to identify disruptions.

4.2.5 t-SNE

We explained the complexity of the event classification in chapter 2 section 2.4.1 and the visualization of the histograms in section 4.2.4 for the binary safe-disruptive splits illustrated it showing no clear separation.

Ref. [112] used the t-SNE[156] algorithm to demonstrate the importance of the time component for disruption prediction. By comparing the t-SNE visualization on single time-slices with the t-SNE plot of samples concatenating multiple time-slices, the authors showed the clusters appear significantly clearer in the latter case.

To get a sense of the multi-class classification results we can expect, we used t-SNE visualization in the same way, i.e. concatenating multiple time-slices for each sample point. The goal is to use an unsupervised method to reveal clusters in our data.

With 7 signals, a single point is represented by a vector of size $N * 7$, aggregating N time-slices. We add a stride parameter s. Our vector contains N individual timestamps, spaced by s milliseconds. This drastically reduces the computational cost of t-SNE.

Coloring is done a posteriori.

Figure 4.2 shows the t-SNE plot of all classes. We identify roughly 4 clusters. The top right corner contains almost exclusively safe samples. Below it, to the right, we can find a cluster of disruptive time-slices made of ML, LOQ, RC, and *other*. At the bottom is a cluster with a very high proportion of KNK disruptive samples. Lastly in the upper

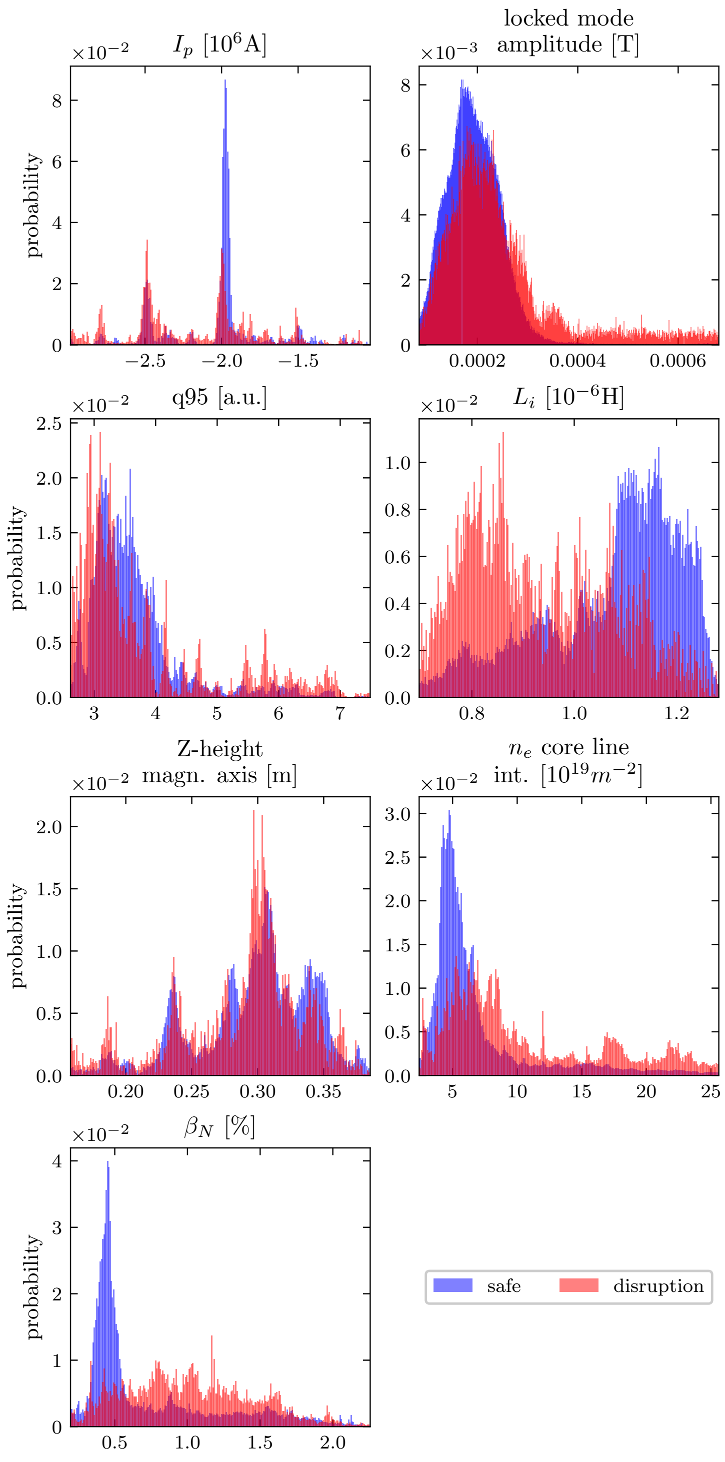


Figure 4.1: Density histograms of the seven signals from the carbon dataset. The stable samples are shown in blue while the disruptive samples are in red. Disruptive samples are from the last two seconds of a disruptive discharge.

half, from the center towards the left, we find a mix of almost all classes both safe and disruptive except KNK.

In comparison with the 1-dimensional histogram, we do find patterns, validating the fact that a data-based approach is appropriate and that multiple signals discriminate better. But the different classes are strongly entangled.

Since we are visualizing all classes at the same time and managed to isolate some clusters, we next display the t-SNE plots of fewer classes.

In figure 4.3 we display the result of the t-SNE visualization using a subset of the classes. We see clearly that the locked-mode class (ML) is present around most other types of disruptions. The mixing of the classes shows the difficulty of the task at hand. Two similar disruptions might share labels but the single label assigned to them does not reflect it. This is obviously a flaw and emphasizes the importance of having good labeled data in the fusion community to build robust data-based models.

4.3 Models

Most disruption prediction studies involving machine learning use different datasets therefore their performance metrics cannot be directly compared. For a fair comparison of the different architectures on the task we defined, built, and trained on our data two other models named A-SVM and LSTM, with similar architectures as ones in the literature.

4.3.1 A-SVM: APODIS-replica, stacked Support-Vector Machine

The first model we use for comparison is based on APODIS [109, 110]. Its purpose is to protect the machine from damages and therefore predict disruptions with enough time for mitigation actions to take place. At JET the minimum time necessary for mitigation actions is defined as 10 ms. The model is aimed at real-time application which requires a prediction time under 2 milliseconds. It makes use of SVMs, and the input features were hand-designed, combining expert knowledge with machine learning. Ref. [136] compared two feature extraction methods: one using all time-slices of 30 milliseconds windows, the second extracting a feature based on the frequency domain for each 30 milliseconds window. They report very close performances between the two methods with slightly better results for the frequency domain feature. Following this work, APODIS combined both methods with a slight change using the mean of the signals instead of their raw values for the first feature. For the first method, APODIS computes the mean of the time window instead of concatenating all values. The size of the time window was also changed to 32 milliseconds.

The input is a selection of diagnostics available in real-time, which are further processed to extract the previously mentioned features from the time-series. SVMs being not suited to handle time sequences, the time component of the data is included in the feature processing. Each data point is constructed from a 32 milliseconds time window and the input consists of three consecutive 32 milliseconds windows: $(\mu_{0:32}, \sigma_{0:32})$, $(\mu_{32:64}, \sigma_{32:64})$, and $(\mu_{64:96}, \sigma_{64:96})$. Each time window is passed through a separate SVM. The output classification of the three SVMs is passed on to a fourth SVM which outputs the final classification.

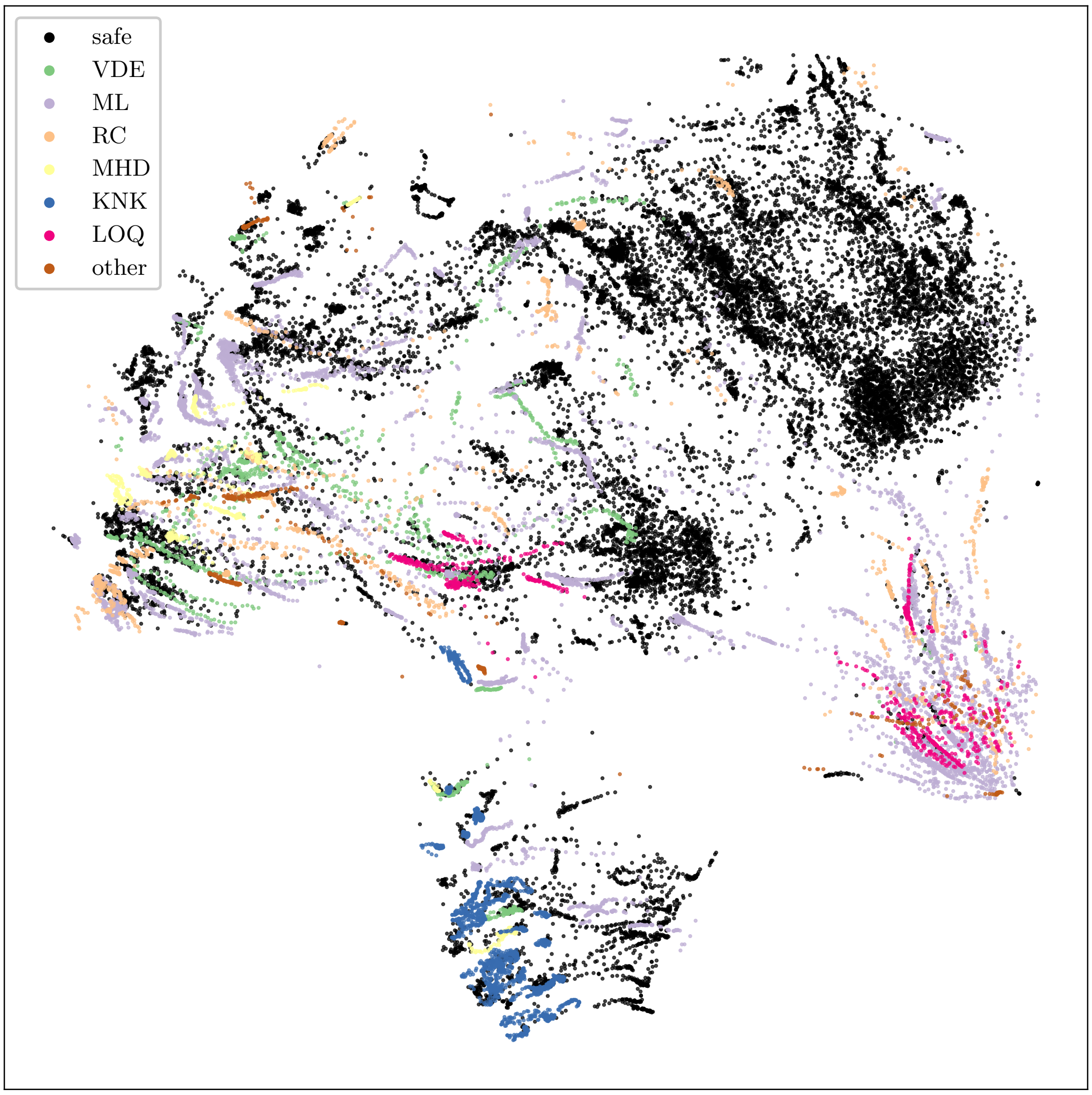


Figure 4.2: t-SNE plot of the JET carbon dataset. Each sample is a 10×7 dimensional point combining 10 time-slices with a stride of 20 milliseconds. The safe class is sub-sampled by a factor of 100 resulting in 18957 samples. All disruption classes are sub-sampled by a factor of 10 resulting in 13466 samples. The t-SNE algorithm is run with 1500 iterations, perplexity set to 10^{12}, and 250 iterations of early exaggeration. The coloring of the samples is down a posteriori.

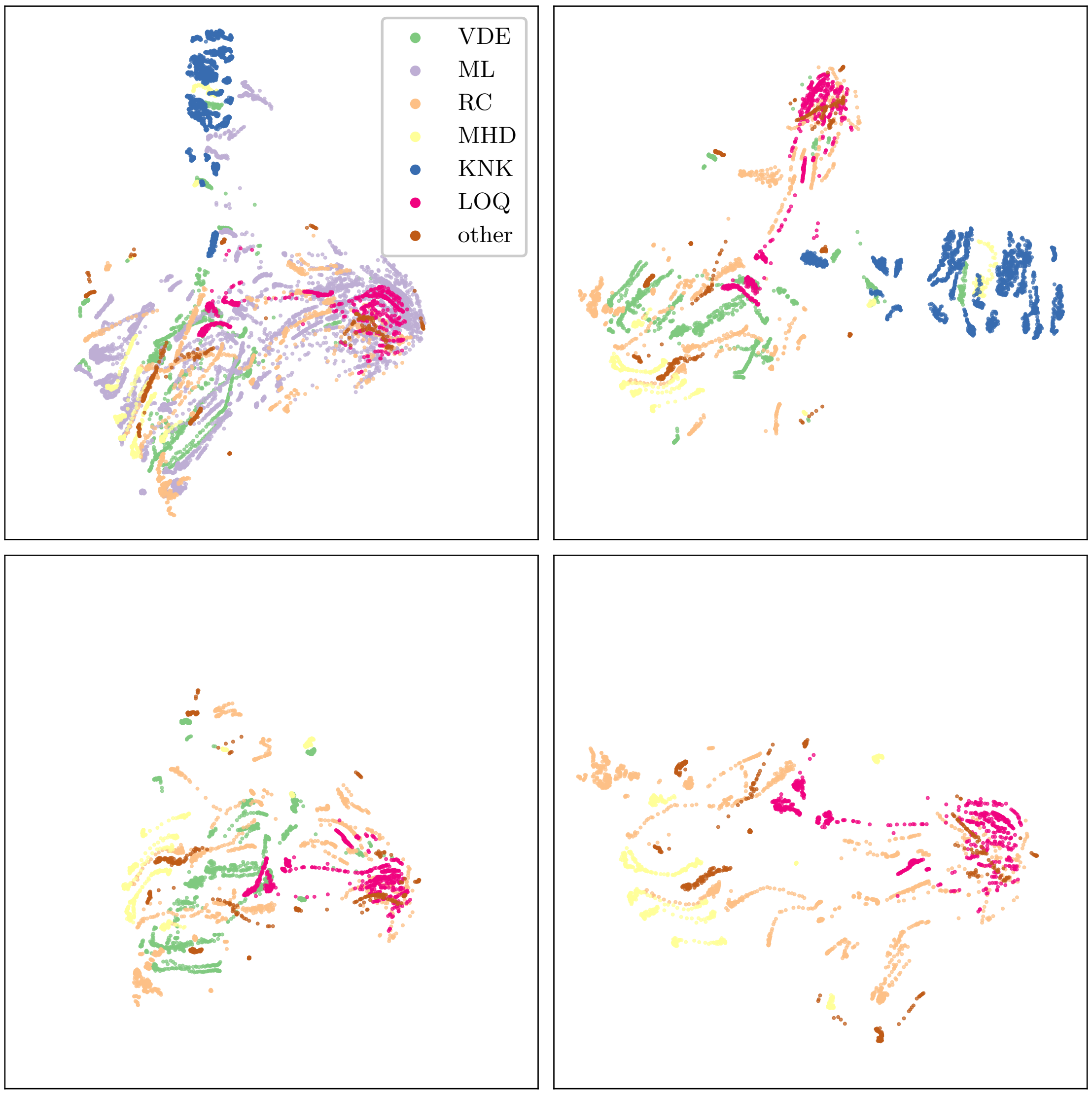
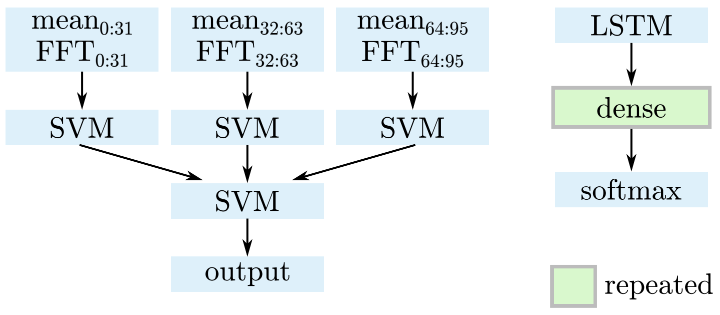


Figure 4.3: t-SNE plot of the JET carbon dataset for hierarchical subsets of the classes. Each sample is a 10×7 dimensional point combining 10 time-slices with a stride of 20 milliseconds. The coloring of the samples is down a posteriori.



$\text{mean}_{0:31}$ $\text{FFT}_{0:31}$
$\text{mean}_{32:63}$ $\text{FFT}_{32:63}$
$\text{mean}_{64:95}$ $\text{FFT}_{64:95}$
SVM
SVM
SVM
SVM
output
LSTM
dense
softmax
repeated

Figure 4.4: Model architecture of A-SVM (left) and LSTM (right). The layer in green can be repeated. The number of repetitions is defined by a hyper-parameter.

A-SVM is implemented using Scikit-learn [157].

4.3.2 LSTM: A classic neural network architecture for time series

As a second model for our benchmark, we implemented a standard LSTM network. Recently, attention networks [158] have surpassed LSTMs, but they remain a prominent architecture for time series. Ref. [111] used an LSTM network to do binary classification on the biggest dataset ever used in the field. They detail their efficient use of powerful computing clusters to train the models on close to 9000 discharges.

Our model is made of a first layer of LSTM cells, followed by a fully connected layer. Finally, another dense layer gives the prediction. The architecture replicates the one used by ref. [111] in its simpler version. Their bigger model used convolution filters to process 1-dimensional signals when available. As for some of their data, the discharges from our period, 2000 to 2008, did not have the specific 1-dimensional profiles.

The LSTM network makes use of dropouts, weight regularization, and label smoothing. The loss function is set to the focal loss. The model is trained using back-propagation [22] and the ADAM optimizer [159].

A common issue with LSTMs is that the internal states can need a few iterations to get set, it needs a "warm-up" time. Values can be fed to the network before the first prediction. We opted for another solution. The first 10 timesteps are ignored. It does not put the LSTM model at a disadvantage compared to the other two models since they too cannot produce predictions on the first few time-slices.

4.3.3 SHP-MLP: Shapelet-based multi-layer perceptron

Our model combines a shapelet transform layer with pruning and a simple multi-layer perceptron. Three blocks of 30 shapelets per input signal compose the first layer of the model. Each block has a different size to capture different event sizes in the signals. The lengths used are 30, 60, and 120 milliseconds. The vector of all shapelet distances is concatenated to the mean of the input signal. It is followed by a batch normalization layer. We add a pruning layer, allowing for a selection of the shapelets during training by reducing the weights connecting to the less important shapelets to zero. One fully connected layer and a dense output layer finalize the model.

parameter	range
Safe under-sampling	$[0.05, 1.0]$
VDE over-sampling	$[1, 5]$
ML over-sampling	$[1, 3]$
RC over-sampling	$[1, 5]$
MHD over-sampling	$[1, 5]$
KNK over-sampling	$[1, 5]$
LOQ over-sampling	$[1, 5]$
Other over-sampling	$[1, 5]$
γ_1	$[10^{-6}, 10^{6}]$
C_1	$[10^{-6}, 10^{6}]$
γ_2	$[10^{-6}, 10^{6}]$
C_2	$[10^{-6}, 10^{6}]$
safe sub-sampling	$[0.15, 0.85]$
label smoothing	$[0, 0.35]$
time smoothing	$[0, 30]$

Table 4.3: Hyper-parameters for the A-SVM model. The first layer of support-vector machines share the coefficients γ_1 and C_1, while γ_2 and C_2 are the coefficients of the last aggregating support-vector machine. The over-/under-sampling parameters are applied to discharges. The safe sub-sampling applies to individual slices including the safe time-slices of discharges that ultimately disrupt.

parameter	range
Safe under-sampling	$[0.05, 1.0]$
VDE over-sampling	$[1, 5]$
ML over-sampling	$[1, 3]$
RC over-sampling	$[1, 5]$
MHD over-sampling	$[1, 5]$
KNK over-sampling	$[1, 5]$
LOQ over-sampling	$[1, 5]$
Other over-sampling	$[1, 5]$
number of LSTM cells	$[16, 256]$
number of dense layers	$[1, 2]$
number of dense units	$[32, 256]$
learning rate	$[10^{-3}, 10^{0.4}]$
label smoothing	$[0, 0.35]$
time smoothing	$[0, 30]$

Table 4.4: Hyper-parameters for the LSTM model. For the parameters defined as 10^x the parameter search is done in log-space, i.e. on the exponent.

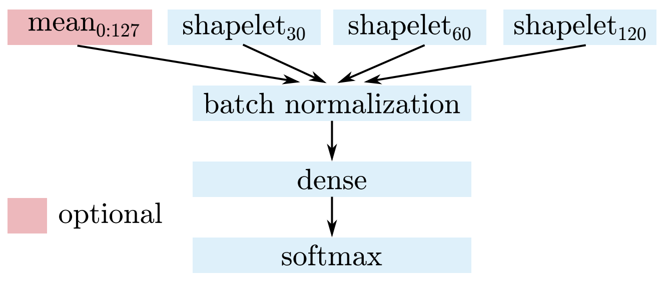


Figure 4.5: Model architecture of our SHP model. The red layer is optional depending on whether the model uses the mean of the signals as input or not.

The input of our SHP-MLP model is a slice of 128 milliseconds. The 128 milliseconds are analyzed by the shapelets and the model returns a single prediction corresponding to the last timestamp of the 128 milliseconds sequence.

SHP-MLP makes use of dropouts, weight regularization, pruning, and label smoothing. As with the LSTM model, the loss function is set to the focal loss. The model is also trained using back-propagation and the ADAM optimizer.

4.4 Results

The results are split into three sections. First, we present the results for the binary classification task. For this task, the models are either trained directly on the binary dataset or trained on the multi-class data and evaluated on the binary task. We then show the results on the multi-class task, where only models trained on multi-class data apply. Finally, to better understand the predictive differences, these two sections are followed by an ablation study.

4.4.1 Binary classification

In this section, we report on the predictive results for the binary prediction task. The models are trained in a supervised manner, with a prediction label for each time-slice, but the predictions shown here are per discharge. Assuming a safe-disruption classifier is used to trigger either a safe shut-down of the machine or disruption mitigation mechanisms, as soon as one time-slice is classified as disruptive the experiment will be interrupted. Meaning that a discharge can be:

1. Successful and terminate properly (true negative)
2. Successful but interrupted (false positive)
3. Disruptive and interrupted (true positive)
4. Disruptive and not interrupted (false negative)

The orders of magnitude are 10^6 data points, from 10^2 individual discharges.

The results are shown in table 4.6. The models with suffix *binary* are trained on binary data, while the models without suffixes are trained using multi-class labels.

parameter	range
Safe	$[0.05, 1.0]$
VDE	$[1, 5]$
ML	$[1, 3]$
RC	$[1, 5]$
MHD	$[1, 5]$
KNK	$[1, 5]$
LOQ	$[1, 5]$
Other	$[1, 5]$
number of shapelets	$[2, 30]$
shapelet stride	$[1, 5]$
degrees of freedom	$[3, 15]$
soft-min α	$[10^{-15}, 10^{-2}]$
dense units	$[32, 512]$
kernel regularizer	$[10^{-8}, 10^{0.2}]$
activity regularizer	$[10^{-8}, 10^{0.2}]$
sparsity	$[0.8, 0.99]$
epochs	$[20, 60]$
learning rate	$[10^{-3}, 10^{0.4}]$
label smoothing	$[0, 0.35]$
time smoothing	$[0, 30]$

Table 4.5: Hyper-parameter list for our shapelet model SHP. See table 4.3 for parameter description. Soft-min α refers to the soft-min parameter α used in the differentiable shapelets [47]. Kernel regularization and activity regularization is applied to the dense layers. Sparsity defines the percent of shapelets pruned at the last epoch. The sparsity at intermediate epochs is linearly interpolated from 0 at the start. For all parameters defined as 10^x, the parameter search is done in logspace, i.e. on the exponent.

Metrics description

The first metric in table 4.6 is the area under the receiver operating curve (AUC - ROC) [160]. For a predictor with continuous outputs, such as our models, a threshold has to be defined to obtain the discrete classes. This threshold offers a trade-off between detecting all disruptions at the cost of many false alarms and detecting few disruptions but without any false alarms. The receiver operating curve draws, for every possible threshold, the false positive rate (FPR) and true positive rate (TPR) pair. Both rates are in $[0, 1]$. The integral of the ROC curve (AUC - ROC) lies between 0 and 1. A score of 0.5 indicates a random classifier on balanced data. The higher the value, the better. With an AUC - ROC of 1 corresponding to a perfect classifier. A model predicting binary labels at random has an AUC - ROC of 0.5. The AUC - ROC below 0.5 indicates the model is predicting the opposite of the true labels.

The ROC curves only use the true positive rate (TPR) and false positive rate (FPR) and are therefore not taking the imbalance in the dataset into account. For this purpose, the third column of table 4.6 contains the area under the Precision-Recall curve (AUC - PR) [161]. Here recall is a synonym for true positive rate. Precision is the ratio of true positive samples in all the positive predictions, i.e. out of all the disruptions, the model detects how many are actual disruptions. A random classifier has an AUC-PR equal to the ratio of positive samples in the data, i.e. with $X\%$ disruptions in the dataset, a random classifier's AUC-PR would be X.

Threshold selection

Those two metrics judge the performance of the model for all possible thresholds. While the number of disruptive time-slices is orders of magnitude smaller than the number of safe time-slices, the number of discharges for each of the two classes is similar. To pick an optimal threshold θ^*, we can therefore use the geometric mean

$$\theta^* = \arg\min_{\theta} \mathrm{TPR}_\theta \times (1 - \mathrm{FPR}_\theta)$$

The geometric mean, true positive rate, and false positive rate at θ^* at reported in table 4.6.

hyperparameter runs

For a fair comparison of all models, we made sure the hyper-parameter tuning converges to stable results by running each tuning three times. Each run starts with a different random seed. The hyper-parameter search with the multi-point expected improvement (q-EI) BO-GP is initialized with 50 quasi-random points, using a scrambled Sobol sequence. The initialization is followed by fifty steps with eight parallel evaluations. The hyper-parameter tuning is implemented with the BoTorch library [162].

As will be shown in the next section the results from the final models are stable, but to avoid favoring one model over another, we combine the three runs of each model into a voting committee. The prediction threshold is set for each model individually and the

Model	AUC-ROC	AUC-PR	G-mean	True positive rate	False positive rate
A-SVM binary	0.91±0.01	0.82±0.03	0.92±0.01	86.70±5.37	22.60±5.23
LSTM binary	0.86±0.01	0.75±0.02	0.85±0.01	71.43±4.05	19.97±4.80
SHP binary	**0.98**±0.00	**0.96**±0.01	**0.95**±0.01	**97.10**±0.00	**14.43**±5.51
A-SVM	0.88±0.02	0.75±0.05	0.84±0.02	80.97±9.41	19.27±10.55
LSTM	0.84±0.01	0.71±0.02	0.83±0.01	79.03±7.15	29.63±7.05
SHP	**0.97**±0.01	**0.93**±0.01	**0.94**±0.01	**90.47**±3.57	**9.23**±3.67

Table 4.6: Mean±standard deviation of the Area Under the Curve (AUC) for both receiver-operating curve (ROC) and Precision-Recall (PR) curve, Geometric-mean (G-mean), true positive rate (TPR), and false positive rate (FPR), for the 3 instances of each models. The validation set is used to define the decision threshold optimizing the binary geometric mean of the true positive and false positive. All other metrics, i.e. AUC-ROC, AUC-PR, TPR and FPR are from the previously unseen test set. Results are reported for the binary classification task for binary models denoted by the "binary" suffix and multi-class models without suffix.

discrete labels are used for votes. The the label with the majority is assigned as final prediction for the committee voting ensemble.

For the multi-class prediction, if models disagree on the disruption type the time slice is considered disruptive but no disruption type is assigned and the time slice is not included in the multi-class statistics.

Results of the individual hyper-parameter tuning runs

We start by analyzing the results of the training of individual models (table 4.6). We can see that in all cases the best performances are obtained with our SHP models. Best cases meaning the lowest value in false positive rate and higher in all other metrics.

The standard deviation indicated as $\pm\sigma$ is very small for AUC-ROC, AUC-PR, and the geometric mean, indicating the robustness of the hyperparameter tuning. The standard deviation for the true positive rate and the false positive rate is higher. With the almost constant geometric mean, this indicates that the lowering of the true positive rate goes in pair with the lowering of the false positive rate. No model shows a significantly worse TPR and FPR than average, at the same time.

Comparing the performance of both binary-trained and multi-class-trained models on a binary task, we are interested in the variation of the results. Is it harmful to the prediction to distinguish the disruptions per event, or on the contrary, do the models benefit from it. Looking at the AUC-ROC, AUC-PR, and geometric mean, we see a small drop in performances for models trained on the multi-class dataset.

Model	G-mean	TPR	FPR
A-SVM binary	0.81	0.83	0.2
LSTM binary	0.75	0.71	0.21
SHP binary	**0.94**	**0.97**	**0.08**

Table 4.7: Results of binary committee-voting models on the binary task. All metrics, including the geometric mean are from the unseen test set.

Result of the committee voting ensembles

The committee voting results are shown in table 4.7. The A-SVM and LSTM true positive rate and false positive rate stay similar to the average of all runs. While the true positive rate of the SHP binary model is also close to the average, the false positive rate is reduced significantly from 14% to only 8%. The three models do not wrongly classify the same time-slices. So the committee voting removes many of those wrong predictions. This is only true in the case of SHP binary, as the multi-class model SHP does not see the same improvement. For the multi-class SHP, both the true positive rate and false positive rate increase slightly from 90.47% and 9.23% to 94% and 11%.

4.4.2 Multi-class classification

Knowing that the models perform relatively well on the simpler binary task, we investigate the multi-class prediction capability. For multiple reasons, judging the performances of the multi-class classifiers is a more complex task than binary classifiers. First of all, the continuous predictions need to be turned into a single label requiring multiple thresholds. A multivariate ROC curve is more complex to analyze. Additionally, the previous trade-off between safe discharges interrupted too early and non-detected disruptions is now between multiple classes which is harder to measure.

Since not all labels have the same meaning there is a natural binary split between the disruption labels and the safe label. We used this property in the following strategy. A single threshold θ^* is defined for the binary safe-disruptive task. For predictions, first, a time-slice is classified as safe or not. In a second step and for the disruptive predictions only, we look at the highest score given by the model for a disruptive class, see equation 4.3.

$$y_{\text{class}} = \begin{cases} 0 & y_0 > 1 - \theta^* \\ \underset{i \in [\![1, n-1]\!]}{\arg\max}\, y_i & \text{else} \end{cases} \tag{4.3}$$

With $y \in [0,1]^n$ the prediction vector and y_0 the probability of the safe label. The result y_{class} is the index of the predicted class.

Confusion matrices

To analyze the confusion matrices we will first describe the ideal/target confusion matrix. The first row, corresponding to the safe label should have a one in the safe prediction. All safe discharges should be classified as such. All other categories should be 0, as no safe

Model	G-mean	TPR	FPR
A-SVM	0.81	0.83	0.2
LSTM	0.74	0.8	0.31
SHP	**0.91**	**0.94**	**0.11**

Table 4.8: Results of multi-class committee-voting models on the binary task. All metrics, including the geometric mean are from the unseen test set.

SHP	A-SVM	LSTM	SHP-only
2.96	**2.47**	2.97	3.41

Table 4.9: Frobenius distance to the ideal matrix.

discharge should be classified as disruptive. The equation 4.3 would give a true prediction for only one label per discharge. With one disruptive label per discharge, the diagonal of the confusion matrix should be ones.

Most disruptive discharges start with a stable phase meaning that they have some time-slices labeled as safe. It translates to the first column of ones. For a few disruptions, the flat-top phase does not reach two seconds before disrupting. In that case, the two seconds prior to disruption used as warning time means the entire discharge is classified as disruptive. It is the case for several KNK disruptions. All other elements should be 0. The ideal matrix given our methodology is given in figure 4.6.

Our models are trained to produce the ideal matrix. This is much more restrictive than what happens in reality with multiple events succeeding each other, or even overlapping. This simplification is due to the lack of time stamps for the events. For the results, we can expect multiple disruptive types per discharge. Having the ideal matrix would be a sign of over-fitting.

For comparison, in the hand-written digits dataset MNIST, the best models have over 99.8% accuracy, and it is well known that the last few digits are very bad and non-recognizable by humans. Therefore a model with 100% accuracy could be described as over-fitting for these few digits that should not be identified.

In our case, we know the target function is the right direction for our model to train, but not the best.

The multi-class task results are presented in figure 4.7.

For the models SHP, A-SVM, and LSTM the first column is made mostly of ones as expected. Only SHP and LSTM have predictions for KNK discharges that do not include a single safe time-slice. Regarding the *other* class, it is unlikely that the discharges are classified as such since only six discharges with four different labels constitute this class. No models use the *other* label but the discharges are still identified as disruptive under different labels.

The distance between each matrix to the ideal matrix is listed in table 4.9. The A-SVM model performs best, followed by SHP and LSTM with very close values, SHP-only is ranked last.

For the matrices in figure 4.7, each model has a different false positive rate as they are individually optimized to maximize the geometric mean of the TPR and FPR. The

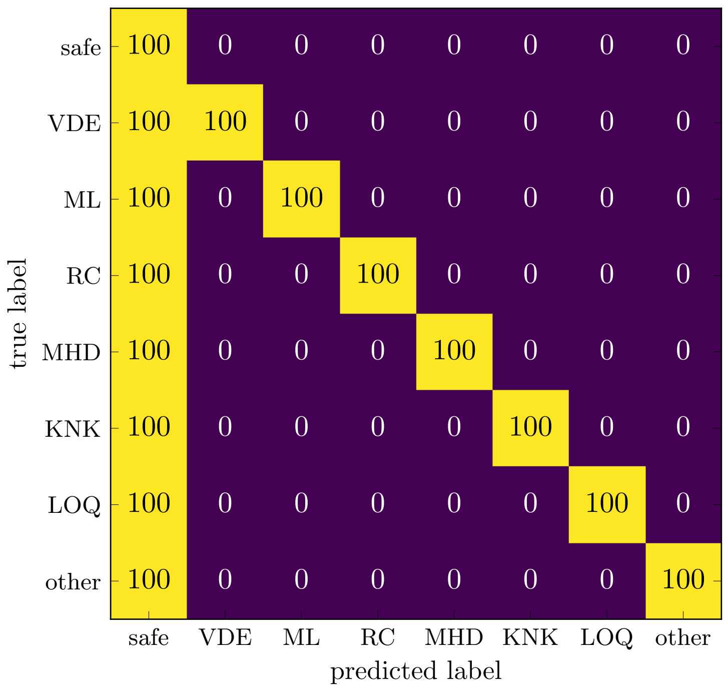


Figure 4.6: Ideal confusion matrix for the training data. Values are percentages of the classes. All safe discharges should only be labeled as safe. All disruptive discharges should have a single disruptive label as defined in section 4.2.3. Additionally because only the last two seconds of disruptive discharges are considered disruptive, a safe label should be assigned for earlier time-slices resulting in a the first column with safe labels for all disruptive discharges.

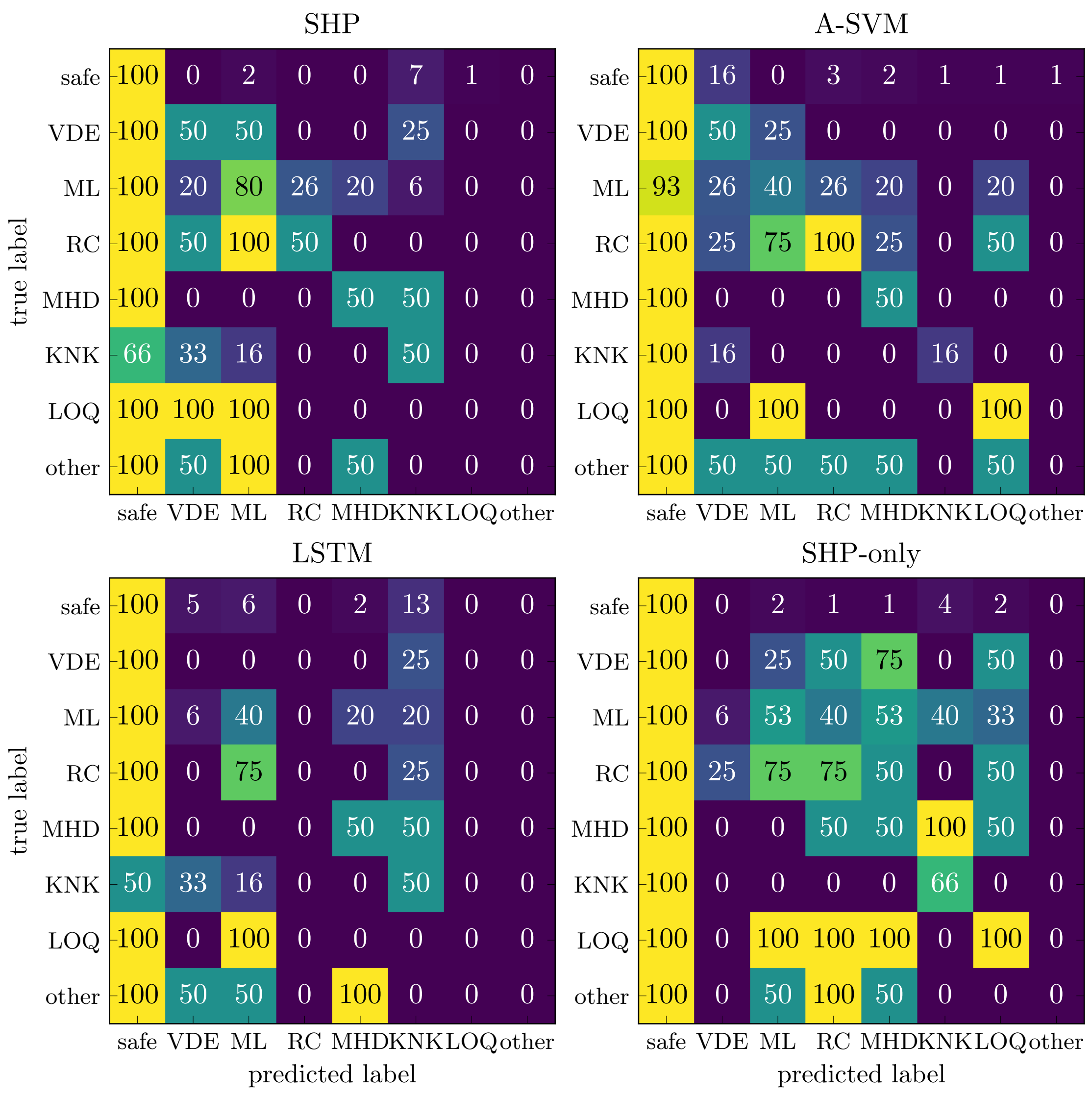


Figure 4.7: Confusion matrix for the different committee-voting models on the unseen test data. Using the disruptive threshold, time-slices are first considered safe or disruptive as per the binary classification task. If a time-slice is disruptive, it is assigned the multi-class label with the highest probability. Although we train with a single disruptive class per discharge having a transition between two labels might not be physically incorrect. As most discharges are longer than two seconds, they start with safe labels before being labeled as disruptive. We correctly find very high percentages of safe labels for all discharges.

Model	TPR	FPR
A-NN binary	0.91	0.17
SHP-only binary	0.94	0.08
A-NN	0.97	0.22
SHP-only	0.94	0.10

Table 4.10: Binary true positive rate (TPR) and false positive rate (FPR) For the A-NN and SHP-only models from the ablation study, trained for binary and multi-class classification. The "binary" suffix denotes the models trained for the binary task.

A-SVM disruption type prediction comes at the cost of close to twice the false positive rate of the SHP model. The false positive rate of the LSTM is three times higher than for SHP although they have very similar distances to the ideal matrix.

4.4.3 Ablation study

We showed that on our dataset, the Shapelet-based neural network outperforms the A-SVM and LSTM models. We note that there is a similarity between the A-SVM architecture and SHP. Both take as input the mean of the input signals, concatenated to a second feature. To explain the difference in performances, the key factor could be the feed-forward neural network in SHP compared to the SVMs in A-SVM, and not the shapelets themselves. To test this hypothesis, two models were built. One uses the features from A-SVM, i.e. the mean of the signal and the standard deviation of the Fourier transform coefficient, and a feed-forward neural network, called A-NN. The second model, called SHP-only, only uses the shapelets, not the mean of the signals, and is also followed by a feed-forward neural network.

The results are shown in table 4.10.

We compare the performances only on the binary task, but for both binary and multi-class models. The lowest true positive rate, at 0.91, is obtained by A-NN binary, combined with the second highest false positive rate. The SHP-only and SHP-only binary models obtain similar results with both 0.94 TPR and respectively 0.10 and 0.08 FPR. A-NN obtains the best TPR at 0.97 but the worst FPR at 0.22. Compared to SHP-only (binary) it gains only 0.03 in TPR for a 0.12 (0.14) worse FPR. This is more than double the false positive rate of the shapelet models for very few disruptions detected. The shapelet-based model remains better, even without the mean of the signal as input.

We want to emphasize this point, SHP-only normalizes the input as part of the normalized euclidean distance calculation and has no information regarding the absolute value of the signals. Meaning that this predictor is capable of binary safe-disruptive predictions without knowing the absolute values of the signals. This can be key to building predictors for new machines, such as ITER. The signals can have a different scale, with for example much higher current, but the behavior captured by the normalized shapelets is similar allowing for accurate cross-tokamak predictions without re-training.

Unfortunately, the type of disruption cannot be predicted by the shapelets alone.

5 Cross-tokamak impurity accumulation detection at JET and ASDEX-Upgrade

In this chapter, we report on the application of our shapelet-based models to the task of disruptive impurity accumulation detection. As mentioned in section 2.4, the main cause of disruption at JET changed after the installation of the metallic wall. High-Z impurity accumulation became the most common cause of disruption and is expected to also appear at ITER due to its similar tungsten wall. This makes the prediction of such disruptions a priority.

To evaluate our models we used two different machines: JET and ASDEX-Upgrade. Because of the low number of discharges annotated for this task coming from ASDEX-Upgrade, we used the JET data both for training and as a first proxy for cross-tokamak evaluation. We then applied the different models to the ASDEX-Upgrade discharges without any changes. This zero-shot transfer learning illustrates the scenario of training a model on existing machines and applying it to ITER on day one, i.e. before any knowledge is obtained from operating ITER.

We present the datasets, their signals, and how we designed our training, validation, testing, and cross-tokamak splits. The datasets' characteristics are first observed through histograms and t-SNE visualization. From this initial study, we found that using a simple threshold on a specific signal might produce a very basic and accurate predictor. This predictor will be used as a baseline compared to our shapelet models. We compare our models with and without access to the signals' absolute values and show that as hypothesized on the carbon wall data, the normalized shapelets produce good results with regard to cross-tokamak applications where other methods fail.

5.1 Cross-tokamak prediction

One of the ambitious goals of disruption prediction for ITER is to have a reliable predictor before the machine starts operating. In machine learning, this setup is known under the term transfer learning and has shown to be a particularly difficult task, often requiring tailored and task-specific solutions.

A key assumption underlying all applications of machine learning, except transfer learning, is that the data is independent and identically distributed. Applying machine learning to a set of physics experiments that differ in their hardware specifications, parameter regimes, etc. would immediately break this fundamental assumption if done naively. And even if the operational space of multiple machines can be mapped onto the same latent space there may still exist machine-specific limits that are not universal, which also makes predictions very hard. Facing this challenge head-on and building on the existence of good predictors for single machines, research is starting to turn towards cross-tokamak transfer learning.

There exist multiple strategies to build neural networks for new machines. These include training them on existing machines and directly applying them to a new one, an approach called zero-shot transfer learning [111, 113, 163]. Or adding a few so-called *"glimpses"* of the new machine [111, 114] into the training set. Alternatively, novel models can be trained on-the-fly, a strategy referred to as *"from scratch"* where models are quickly updated in between discharges [115, 139, 164, 165].

The zero-shot transfer learning methods build a model on a given machine and apply it directly to a newer machine without any changes. It is the hardest task and is often used in the literature to show a worst-case scenario before introducing a different method. The next method consists of adding *"glimpses"* of the new machine to the training data of the model. For this, a small percentage of the discharges from the target dataset is excluded and used to tune the models. This method often shows drastic improvements over the zero-shot transfer learning. The third method, *"from scratch"* is fundamentally different. For this strategy, small models capable of learning in just a few minutes are trained between the experiments. As there can be half an hour or more between two experiments a model can be trained using the data that was just produced. In the case of a disruption, the model should be retrained to detect it. This method also addresses the issue of performance deterioration over time. Since discharge parameters change over time as physicists explore new scenarios, the various predictors built on older data slowly become obsolete. The *"from scratch"* strategy only uses the last n discharges and removes older ones. Finally, combining two methods ref. [116] trains models with the update strategy *"from scratch"* but initialized with an existing device.

A study of the influence of discharges from a new machine to complement the dataset from existing machines has been presented in ref. [112] using the three tokamaks Alcator C-Mod, DIII-D, and EAST. The combined data from two machines are used to build a predictor for the third device, including different scenarios where some data from the third tokamak is also available as *"glimpses"*. Ref. [117] carried out a similar study with data from the tokamaks J-TEXT and EAST while developing a more advanced feature extractor in their neural network.

5.2 JET-ITER like wall and ASDEX-Upgrade datasets

For this study, we compiled three datasets. Two from JET in its ITER-like wall configuration and one from ASDEX-Upgrade. The first dataset from JET referred to as the JET-Early dataset, contains 130 discharges from the year 2012. The second dataset, referred to as JET-Late, contains 302 discharges from the years 2015 to 2020. The last dataset from ASDEX-Upgrade referred to as the AUG dataset, contains 19 discharges from the years 2013, 2020, and 2021.

For these datasets, three disruptive events are identified resulting in four possible labels: safe, impurity accumulation, edge cooling, and pre-disruptive. Each event comes with its timestamps, contrary to the data used in chapter 4. In this work, we only predict the impurity accumulation class and therefore aggregate the three other labels under a single

Class	Train	Validation	Test
safe	12	13	13
impurity accumulation	30	31	31

Table 5.1: Number of discharges per label for the train, validation and test split for the JET-Early dataset.

one: safe. In this chapter, the safe label is a *non-impurity accumulation time-slices*, i.e. safe time-slices as well as time-slices with the edge cooling and pre-disruptive label.

As all predictors have an aging effect, i.e. performance degrades over time as the discharges' scenarios evolve we use JET-Early to build our models, JET-Late to test them for slightly different discharges but still on the same machine and lastly, we analyze the predictions on the AUG dataset to evaluate a true cross-tokamak application.

5.2.1 JET-Early

The JET-Early dataset is made of 130 discharges from number 81852 (01.17.2012) to number 83635 (16.07.2012) and is similar to the dataset used in ref. [140]. While in the previous chapter discharges containing only safe labels did not disrupt at all, in this dataset the safe label is used as anything that is not of the type of events we are looking for. For example, discharge number 81990 does not have a single impurity accumulation, edge cooling, or pre-disruptive label but still disrupts due to a density limit.

We reiterate, in this chapter *safe* means *anything other than the disruptive event we investigate*. JET-Late and AUG use the same labeling as described here.

26 discharges contain exclusively safe labels, 92 contain a disruptive impurity accumulations event, 11 contain a disruptive edge cooling event and 28 contain a pre-disruptive event. First, we insist on *disruptive* events as some discharges might have impurity accumulation which does not completely degrade the plasma and would therefore not be classified as an impurity accumulation we are interested in. Second, the numbers per class do not add up to the 130 discharges as some discharges can have an edge cooling event followed by an impurity event, counting twice in the list above.

In terms of time-slices, 595587 (88.2%) are of the safe label, 72814 (10.8%) impurity accumulations, 1255 (0.7%) Edge cooling and 5896 (3.3%) pre-disruptive.

The discharge count of the train, validation, and test set for this dataset is given in table 5.1.

5.2.2 JET-Late

JET-Late is made of 302 discharges from number 89063 (11.17.2015) to number 98005 (18.09.2020). 146 contain only safe labels, 73 contain disruptive impurity accumulations, 26 contain disruptive edge cooling and 57 contain pre-disruptive labels. From these 302 discharges, 1269745 (93.9%) time-slices are labelled safe, 36934 (2.7%) are of the impurity accumulation class, 11184 (0.8%) edge cooling and 34056 (2.5%) pre-disruptive.

There is no training split for this dataset as all discharges are used to evaluate the models trained on JET-Early.

5.2.3 AUG

In the AUG dataset, the fourteen discharges between number 29284 to 30006 are from 2013 and five between number 37380 and 39348 are from 2020 and 2021. All nineteen discharges have disruptive impurity accumulations. This small dataset does not include discharges without impurity accumulations.

14076 (67.6%) time-slices are of the safe label and 6737 (32.4%) are of the impurity accumulation label. The small number of discharges gives for a much larger proportion of impurity accumulation labels.

As discharges at ASDEX-Upgrade are shorter than at JET, some time-series are not long enough for our models. Those short discharges are padded with their last value and random noise to fit the minimum length.

Again no training split is required for this dataset.

5.2.4 Signals

Twelve signals available both at JET and ASDEX-Upgrade were selected for these datasets:

1. Safety factor q_{95}
2. Normalized beta β_N
3. Plasma inductance L_i
4. Electron density peaking factor N_e
5. Normalized total radiated power
6. Greenwald fraction
7. Electron temperature peaking factor T_e
8. LH power fraction
9. Power fraction
10. Core radiation peaking factor
11. Divertor radiation fraction
12. Plasma current error fraction

The $q95$, β_N, and L_i are similar to the signals selected for the carbon wall data in chapter 4 and similarly, the signals are selected such that they are also available at ASDEX-Upgrade. These signals are normalized to be more robust to machine differences and plasma conditions. For example, the electron density is normalized with the Greenwald limit [152]. Peaking factors are core-vs-all metrics, see section 2.3.3.

Multiple power-related diagnostics were added as they are relevant for the disruptive classes of interest. The plasma current is normalized with the target plasma current as

$$I_{\text{error}} = \frac{I_p - I_{\text{target}}}{I_{\text{target}}} \tag{5.1}$$

The LH power fraction describes the ratio of power input to the power needed to transition from the Low (L) confinement mode to the High (H) confinement mode. The H-mode was discovered in 1982 at ASDEX-Upgrade [166]. It is an improved magnetic confinement mode that enables higher core temperature to be achieved. Scaling laws can give a necessary power above which the plasma can reach the H-mode [167].

All signals are made machine independent either by the nature of the signal itself, e.g. β_N, or by normalizing to other quantities of the plasma, e.g. using the Greenwald fraction instead of the density. This is made to facilitate the transfer between the two machines but as we will show some discrepancies persist. To build and evaluate our different models the data is normalized using the training split of JET-Early as reference.

5.3 Problem formulation

For this study, we focus only on the disruptive impurity accumulation events. We used the same binary formulation as for the carbon but instead of aggregating the disruption together, we aggregate the safe, edge cooling, and pre-disruptive labels together. We want to identify the disruptive impurity accumulation event against all other events.

We again formulate the classification problem as a sequence-to-sequence problem. For each vector $\mathbf{x}_t^i \in \mathbb{R}^{\text{input}}$ of discharge i at time-step t, we want to assign a categorical label $y_t^i \in Y$ with the sample space $Y = \{\text{safe}, \text{impurity accumulation}\}$. For the input dimension, we designed predictors with a single dimension or use hyper-parameter tuning to define which signals to use. The labels are first one-hot encoded, turning the categorical target y_t^i into a vector $\hat{\mathbf{z}}_t^i \in \{0,1\}^2$. $\hat{\mathbf{z}}_t^i$ gives the probability of each label with the sum of the probabilities equal to one. The discrete probabilities are then smoothed as described in section 2.2.13 resulting in the continuous target probabilities $\mathbf{z}_t^i \in [0,1]$.

As we have time-stamps for the disruptive events in the database we are not trying to classify the end of a disruptive discharge as disruptive until it disrupts. We target only the time windows containing the impurity accumulation event which can stop before the end of the discharge. For many discharges, the time between the end of the impurity event and the disruption remains very short.

5.4 Histogram and t-SNE visualization

5.4.1 Histogram analysis

We start by looking at the histogram of the individual signals, see figure 5.1. Contrary to the carbon wall data, this dataset presents a much clearer split between impurity accumulations and safe labels. Although most signals show a reasonable difference between the two labels, the core radiation peaking from the bolometer cameras stands out. Almost all safe labels have core radiation values below five while most of the impurity accumulation labels are above five. Since we are interested in the high-Z impurity accumulation in the core of the plasma which will strongly radiate and the core radiation signal measures the ratio of radiation coming from the core compare to the rest of the plasma the results match our expectations.

Compared to the carbon wall data, we are expecting quite high performance from the predictors as we already see a good separation of the single signal histograms.

The histograms of the JET-Late dataset exhibit similar properties for half of the signals, while for the other half either one class or both have different distributions (figure 5.2). The q_{95} also has its highest density between 3 and 4 but has many more high values, with higher densities above 5. For the internal inductance, L_i the impurity accumulation density is very similar, but the safe samples shifted to higher and lower values with much fewer samples between 1×10^{-6} H and 1.2×10^{-6} H which was the interval with the highest density for the JET-Early dataset. The Greenwald fraction is more concentrated between 0.25 and 0.85. The LH power threshold has many more values above 2 for the safe samples. The core radiation is very similar, except for a few impurity accumulation samples around 2.5. In the JET-Early dataset, the plasma current error fraction was mostly negative with values in $[-0.05, 0]$ while the plasma current error fraction is centered around 0.

We show the histograms for the discharges from ASDEX-Upgrade in figure 5.3. If the individual signals have the same distribution as the data from JET, the cross-tokamak prediction is trivial. If no signals have the same distribution either a combination of the signals or other feature invariant with respect to the absolute values of the signals, such as the normalized shapelets, would be required to be trained at JET and applied at ASDEX-Upgrade.

By design, the two tokamaks operate with different q_{95} profiles but the physics behind it is the same for both machines. The different signals differ from the two JET dataset to various degrees but one substantial difference is in the core radiation histogram. For JET-Early and JET-Late values below five belong to safe time-slices with a very high probability while on the AUG dataset the transition between safe and impurity accumulation labels happens at around 1.5. The simple Threshold model is therefore highly unlikely to be appropriate for cross-tokamak applications. For ITER, it could only be used after sufficient data has been gathered from experiments. We will later see how these differences affect our models.

5.4.2 t-SNE visualization

For the t-SNE visualization (figure 5.4) we colored each time-slice with the original labels, including the edge cooling label and pre-disruptive label. The 4 classes are quite well separated. The pre-disruption labels are split into two groups, one clustered around safe labels, and a second closer to the impurity accumulation label. The distinct separation of the classes from t-SNE also confirms that the classification on JET-Early should be fairly accurate. We note the small number of edge cooling labels compared to the other classes.

5.5 Models

We built five models for this study. As the core radiation signal exhibits very good predictive capability, the first model is a simple threshold over the core radiation signal. Next are our four shapelet-based models. Two are also only using the core radiation signals, while the two other use hyper-parameter tuning to select the best set of signals.

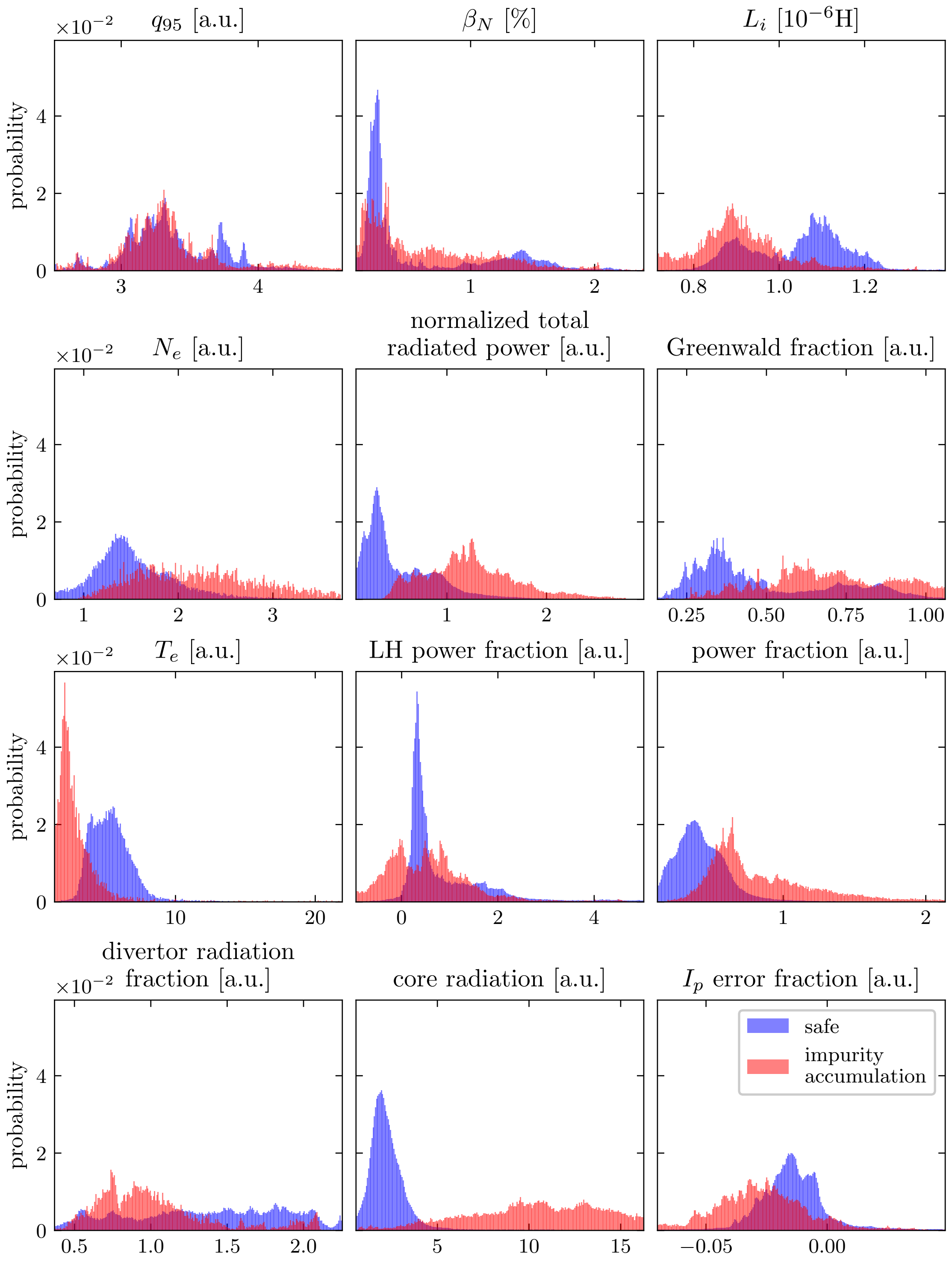


Figure 5.1: Histograms for the JET-Early dataset.

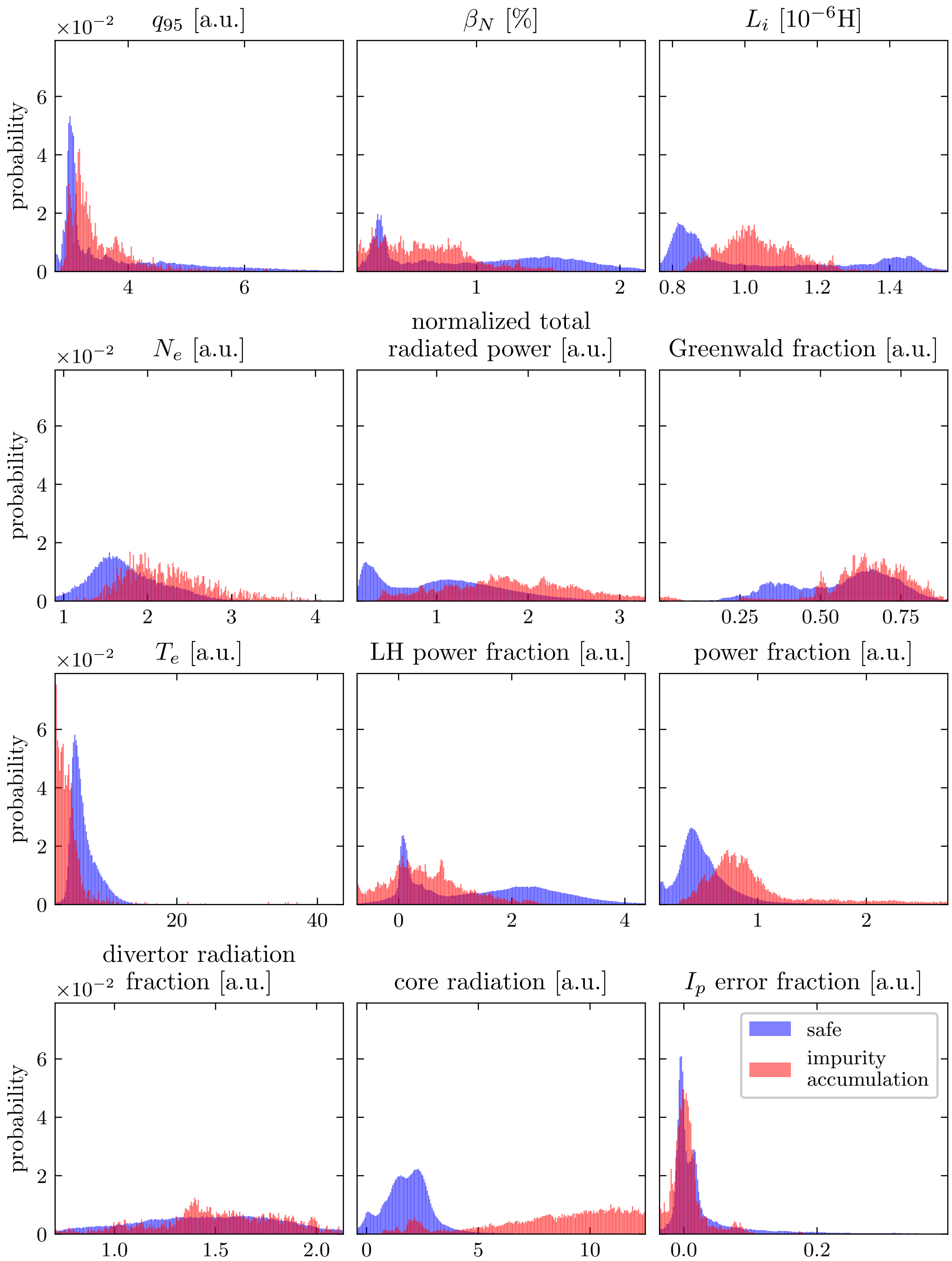


Figure 5.2: Histograms for the JET-Late dataset.

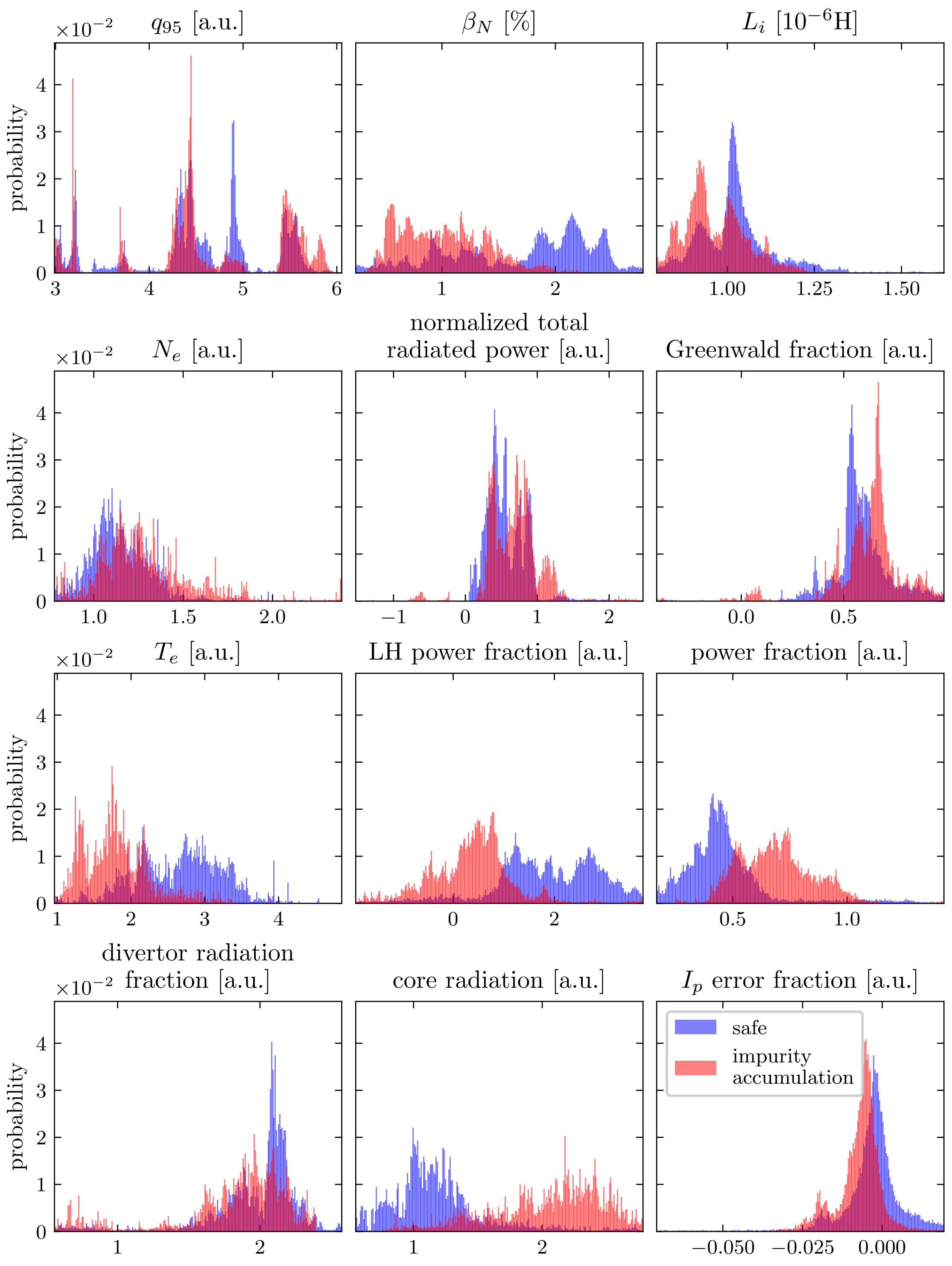


Figure 5.3: Histograms for the AUG dataset.

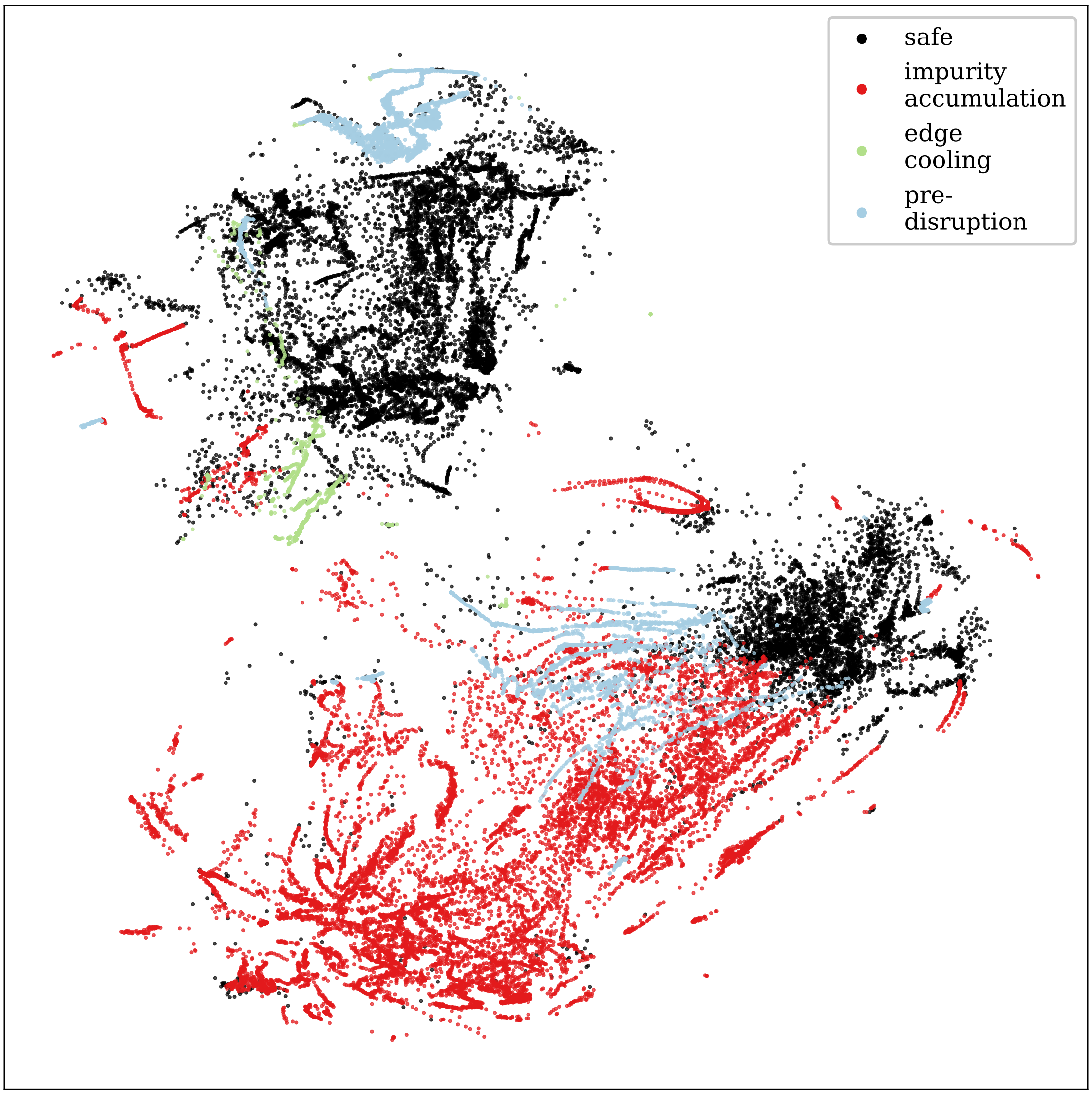


Figure 5.4: t-SNE plot of the JET-Early dataset. Each sample corresponds to a 12 dimensional point from a single time-slice. Coloring of the samples is done a posteriori.

For each pair, one model uses the mean of the input signal time window while the other exclusively uses the normalized shapelets.

5.5.1 Threshold model

The core radiation signal is constructed from the bolometric camera's signal which measures the radiation coming from the plasma. The cameras named KB5H and KB5V have twenty-four lines of sight horizontally and vertically respectively. Height lines are aimed at the divertor region, while the other sixteen cover the rest of the plasma, see section 2.3.3.

This signal is the peaking factor of the radiated power measured by the bolometer cameras. The peaking factor of plasma profiles is defined by ref. [86] as a metric of the difference in temperature, density, or radiation in the plasma core, against the rest of the plasma. This is of particular interest as a few phenomena are related to changes in the core. A hollow density or temperature profile, i.e. a core density or temperature being lower can be a sign of a poorly performing plasma. Similarly, higher radiation coming from the core than the rest of the plasma is a sign of high-Z impurity accumulation which can lead to disruptions.

For instance, the destabilization of the plasma by high-Z impurity that leads to disruptions is characterized by increased core radiation and the hollowness of the temperature profile. Which is measured by an increase in the peaking factor of the radiated power and a decrease in the peaking factor of the electron temperature.

In section 5.4.1 we saw that the core radiation signal is strongly correlated with the impurity accumulation class. So much so that a simple threshold could be able to correctly detect this class. We therefore create a basic model, named Threshold, by selecting the threshold θ such that the following classification is optimized:

$$y_{\text{impurity}} = \mathbb{1}_{\text{core radiation} \geq \theta} \tag{5.2}$$

The training and validation set is used to optimize a single hyper-parameter: the smoothing of the core radiation signal. The search space is $[-500, 500]$ with the sign defining the filter: positive for mean filtering, negative for median filtering. The window size is the absolute value. The best filter is a causal mean filtering of 19 time-slices, equal to 38 milliseconds. It is very close to the 40 milliseconds used in [140], but they used median filtering.

5.5.2 Shapelet-based models

Two other models are built on the single core radiation signal alone. They are both shapelet-based neural networks with one having access to the mean of the input signal named SHP-1 Mean and the other using exclusively the shapelet transform on the input named SHP-1.

In chapter 4 the mean of the input signals was concatenated to the shapelet transform output. To avoid having the mean parallel to the shapelets and to enforce a stronger coupling we modify the inclusion of the input means. Inspired by the Squeeze-and-excitation block [168], we multiply the output of the shapelet transform by a non-linear combination of the mean values. The model architecture is shown in figure 5.5.

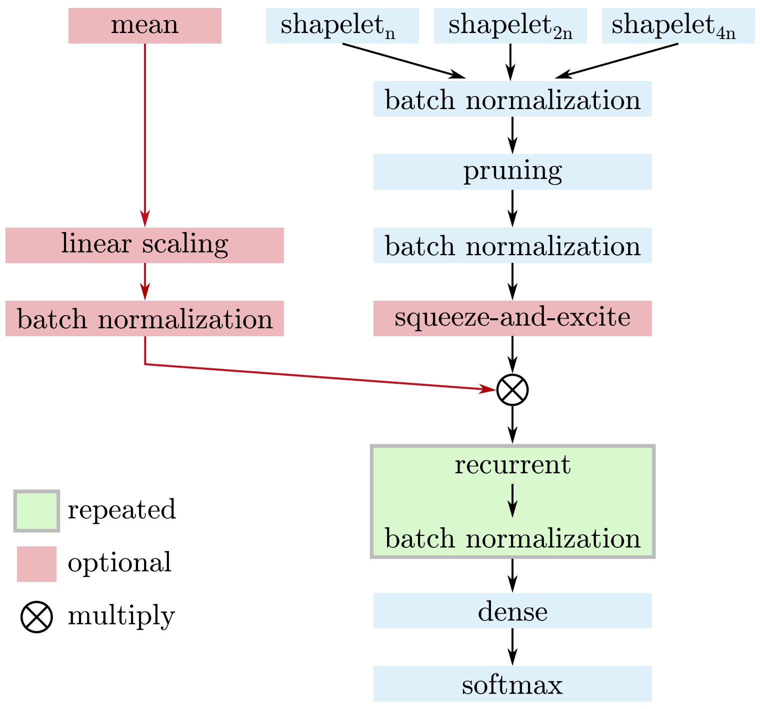


Figure 5.5: Model architecture of the shapelet-based model for impurity accumulation detection. The red layers are optional. Layers in green are repeated according to a hyper-parameter. The shapelet subscript indicates the length of the shapelet. The shapelets of length $2n$ and $4n$ are included if they are smaller than the time window of the shapelet transform.

The second half of the network is common to the version without mean. It consists of one or multiple recurrent layers with batch normalization and a dense output layer. Both recurrent and dense layers use dropout regularization. The list of hyper-parameters can be found in table 5.2. Regarding the shapelet length, the hyper-parameter is in the range $[100, 500]$ but in case the difference between *input length* and *LSTM time-steps* is smaller than the shapelet length, the shapelet length is reduced to match the available time window.

The last two models, named SHP-12 Mean and SHP-12, are the same as the previous two shapelet-based neural networks except that they take multiple signals as input. The signals are selected through hyper-parameter tuning. The list of hyper-parameters can be found in table 5.2.

5.5.3 Shapelet heuristic initialization

Choosing an appropriate shapelet initialization can improve the learning process. Multiple methods have been proposed, they can be initialized directly from random sequences in the training data, using the centroids of a K-Means clustering of the training data [169] or using a first shapelet discovery method on a quantized version of the training data [124].

In chapter 4 a simple random initialization is used, while in this chapter we can make use of the time-stamps to select one candidate shapelet at each class transition, i.e. start and end of an impurity accumulation event. The centroids of the K-Means clustering of this set of candidates can be used as initialization for the shapelets. A hyper-parameter defines whether to use this heuristic initialization or random values.

5.5.4 Training metric

The neural networks' training is performed with the focal loss, back-propagation, and the ADAM optimizer [159] with a 30 epochs early stopping patience. For early stopping, as to select the final models, we can use global metrics over the entire dataset instead of losses defined on individual batches. The metric we optimize is

$$\mathcal{L}_{\text{global}}(\theta) = \sqrt{\text{F1}_{\text{slice}}(\theta)} + \sqrt{\text{F1}_{\text{discharge}}(\theta)} \tag{5.3}$$

$\text{F1}_{\text{slice}}(\theta)$ is the F1 score per time-slice and $\text{F1}_{\text{discharge}}(\theta)$ is the F1 score per discharge. We want the model to learn to predict the time-slices correctly, while still having good performance per discharge. The opposite would be for example a model that predicts perfectly the impurity accumulation, but a few time-slices of every single safe discharge is wrongly classified as impurity accumulation. Those few time-slices would not change the per slice F1 score significantly as it is just a time-slices compared to hundreds of thousands. To avoid it we combine both metrics.

5.6 Results

In this section, we start by looking at the models' performance through the true and false positive rates for the three different datasets. We will then explain how we can inter-

parameter	range
input length	$[300, 750]$
batch size	$[8, 64]$
LSTM time-steps	$[50, 650]$
input median filtering	$[0, 50]$
normalization	{z-normalization, yeo-johnson}
prediction filtering size and type	$\{[0, 200], \{\text{mean}, \text{median}\}\}$
include diagnostic	$\{\text{False}, \text{True}\}$[11]
safe undersampling	$[0.05, 1.0]$
impurity accumulation oversampling	$[1.0, 5.0]$
safe weight	$[0.1, 1.0]$
impurity accumulation weight	$[1.0, 20.0]$
data augmentation noise level	$[0.0, 0.1]$
number of shapelets	$[3, 15]$
shapelet length	$[100, 500]$
degrees of freedom	$[3, 5]$
batch normalization	$\{\text{False}, \text{True}\}$
squeeze-and-excite	$\{\text{False}, \text{True}\}$
squeeze-and-excite ratio	$[0.1, 2.0]$
multiple shapelet lengths	$\{\text{False}, \text{True}\}$
shapelet initialization heuristic	$\{\text{False}, \text{True}\}$
recurrent cell	$\{\text{GRU}, \text{LSTM}\}$
number of recurrent layers	$[1, 3]$
number of recurrent cells	$[8, 128]$
recurrent dropout rate	$[0.0, 0.5]$
dense dropout rate	$[0.0, 0.5]$
kernel regularizer	$[10^{-8}, 10^{0.2}]$
activity regularizer	$[10^{-8}, 10^{0.2}]$
sparsity	$[0.0, 0.9]$
sharpness-aware minimization ρ	$[0.1, 0.25]$
focal loss γ	$[0.1, 5.0]$
epochs	$[80, 300]$
learning rate	$[10^{-4}, 10^{1.3}]$
label smoothing	$[0.0, 0.35]$
time smoothing	$[0.0, 50.0]$

Table 5.2: Hyper-parameter list for the shapelet-based neural networks. The *input length* defines the time window size of the input to the network. *LSTM time-steps* defines the number of time-steps before the gradient is updated, this method is known as truncated back-propagation through time. The difference between *input length* and *LSTM time-steps* is the length of the sequence on which the shapelet transform is applied. *include diagnostic* contains a boolean value for each diagnostic defining whether it is used or not. The *include diagnostic* hyper-parameter is not present for SHP-1 and SHP-1 Mean. For all parameters defined as 10^x, the parameter search is done in logspace, i.e. on the exponent.

	Per-slice		Per-discharge	
	TPR	FPR	TPR	FPR
Threshold	91.3	1.8	100	30.8(4)
SHP-1	63.4	2.7	100	15.4(2)
SHP-12	71.6	3.6	100	38.5(5)
SHP-1 Mean	92.6	1.9	100	30.8(4)
SHP-12 Mean	87.4	2.5	100	23.1(3)

Table 5.3: True positive rate (TPR) and False positive rate (FPR) for all classifier on the JET-Early dataset. Threshold is the classifier described in equation 5.2. There are 13 safe discharges meaning each false positive equates to a 7.7% increase in the false positive rate. The number of wrongly classified discharges is in parenthesis.

pret the predictions by analyzing the learned shapelets and finally visualize the models' decisions on a few discharges.

5.6.1 JET-Early results

The Threshold model performs already very well as shown by its results in table 5.3. The per slice results take every single time-slice into account, while the per discharge results count the number of discharges with or without the impurity accumulation task as single elements. The Threshold model presents a true positive rate per slice of 91.3%, a very low false positive rate per slice of 1.8%, but a higher 30.8% false positive rate per discharge. The test set of the JET-Early data collection only contains thirteen safe discharges. A 30.8% false positive rate is therefore a misclassification of four out of thirteen safe discharges. The per discharge true positive rate is 100%.

The results of the one-signal neural networks SHP-1 and SHP-1 Mean on the JET-Early dataset are in table 5.3. Starting with SHP-1 Mean the results are almost identical to the Threshold model with a 92.6% true positive rate and 1.9% false positive rate per slice, and the same four per discharge false positives. The per discharge true positive rate is 100%.

The model without mean, SHP-1 differs quite a bit with only 63.4% true positive rate per slice and 2.7% false positive rate per slice. Although the per slice performance is worse, the per-discharge performance is actually better with the same 100% true positive rate and only 2 (15.4%) false positive discharges. For one of the two discharges, discharge number 83346, the same 750 milliseconds as the other models are classified as impurity accumulation. For the second discharge, number 83403, they classify the same 50 milliseconds at the end of the discharge, but the shapelet model without mean also misclassifies 83 milliseconds at 15.2 seconds into the discharge. The higher false positive rate mostly comes from the model not turning back to a safe label after an impurity accumulation event and from a few events that are wrongly detected as disruptive impurity accumulations. The lower per slice true positive rate while still having a perfect per-discharge positive detection comes from events that are not detected in their entirety.

For SHP-12 and SHP-12 Mean the results on the JET-Early dataset table 5.3 show in

	Per-slice		Per-discharge	
	TPR	FPR	TPR	FPR
Threshold	86.1	1.4	97.5	17
SHP-1	55.8	2.8	98.7	58.7
SHP-12	62.5	8.4	94.9	74.0
SHP-1 Mean	86.0	1.5	97.5	16.6
SHP-12 Mean	85.5	4.1	100	40.4

Table 5.4: True positive rate (TPR) and false positive rate (FPR) for all classifier on the JET-Late dataset. Threshold is the classifier described in equation 5.2.

both cases deterioration of the per slice performance compared to the models with a single signal. SHP-12 has three more per discharge false positives than SHP-1 and SHP-12 Mean has one less per discharge false positives than SHP-12.

5.6.2 JET-Late results

The first application outside the JET-Early dataset the models were trained on concerns the JET-Late data. As we saw from the histograms the two datasets are very similar but still contain some differences.

There are significant differences in true and false positive rates when applying the models trained with JET-Early to the JET-Late dataset, see table 5.4.

All models have around 5 to 10% lower per slice true positive rates with a deterioration in per slice false positive rate that is most notable in SHP-12 and SHP-12 Mean. The per discharge true positive rate is still extremely high with the only model below 97.5% being SHP-12 at 94.9%. The per-discharge false positive rate on the other hand is significantly worse. Furthermore, with 229 discharges without the impurity accumulation label, the false positive rate can be better estimated than for JET-Early. The Threshold and SHP-1 Mean models have very similar predictions with a per discharge false positive rate of 17% and 16.6% respectively. SHP-1, SHP-12, and SHP-12 Mean have much higher rates. As we will explain later the SHP-1 model is detecting a second event linked to plasma confinement mode transition that can cause impurities to accumulate in the core. This event was not anticipated when building the models and illustrates the complexity of labeling tokamak discharges and the limitations of simple global metrics such as the true and false positive rates. A detailed discharge analysis allows for a better understanding of the capabilities of the models.

5.6.3 AUG results

Lastly, we investigate a true cross-tokamak prediction task by applying our models to the AUG data. From the histograms, we already noted important differences in the signals compared to JET, although we tried to select normalized and machine-independent quantities. As shown in table 5.5, the Threshold, SHP-1 Mean and SHP-12 Mean models, i.e. all models using the absolute values of the signals, have a 0% true and false positive rate. These models are failing and cannot detect any impurity accumulation events. The

	Per-slice	
	TPR	FPR
Threshold	0	0
SHP-1	80.4	43.5
SHP-12	83.8	80.9
SHP-1 Mean	0	0
SHP-12 Mean	0	0

Table 5.5: True positive rate (TPR) and false positive rate (FPR) for all classifier on the AUG dataset. Threshold is the classifier described in equation 5.2.

two models based only on shapelets, SHP-1 and SHP-12 detect some of the events with 80.4% and 83.8% per slice true positive rate, but with a very high false positive rate especially for SHP-12. SHP-1 has a per slice false positive rate of 43.5% and SHP-12 has a per slice false positive rate of 80.9%. These numbers are obtained from only nineteen discharges, all having impurity accumulations. A larger number of discharges including discharges without impurity accumulations would result in more significant statistics and allow for per-discharge analysis.

Next, we analyze the models and explain how SHP-1 and SHP-12 differ resulting in the large difference in the false positive rate on AUG.

5.6.4 Interpretability

A benefit of the shapelets is their interpretability. The transformation from the shapelet metric is easy to understand and the matching between the shapelet and data can be visualized. Although these statements are true for early shapelet-based models [41] where a decision tree was applied to the shapelet metric, the inclusion of shapelets into neural networks can deteriorate this interpretability. Additionally, learning the shapelets [47] has the unintended consequence that the shapelets no longer directly come from the training data, but can now take any shape. To enforce regularity in the shapelets and therefore make them better fit the data we used a restricted number of degrees of freedom and constructed the shapelets with splines. Our implementation of the shapelets is extensively covered in chapter 3.

To visualize the shapelet contribution to the predictions we combine two methods. As in [49, 123] we use grad-CAM [130] to compute the importance of each shapelet and display the shapelets at the time with the highest activity. Additionally, we use grad-CAM to color the time-series with their importance at each time-step, as done in [129].

The importance of a signal at a given time-step is computed by combining the contribution of each shapelet using the Grad-CAM. The grad-CAM originally developed for images defines the importance of a feature map as

$$\alpha_k^c = \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \frac{\partial y^c}{\partial F_{ij}^k} \tag{5.4}$$

with α_k^c the importance of the feature map F^k to the prediction y^c, calculated as the average of the partial derivative for the nm pixels. The heatmap giving the importance of a pixel from the input image to the prediction for a class c is then

$$L_{ij}^c = \text{ReLU}\left(\sum_{k=0}^{p-1} \alpha_k^c F_{ij}^k\right) \tag{5.5}$$

which is the sum over the feature maps of a pixel value of a feature map multiplied by the importance of this feature map. The ReLU function is applied to keep only positive contributions to the prediction.

For our shapelet-based models applied to time-series the heatmap is computed as

$$L_t^c = \sum_{k=0}^{p-1} \text{ReLU}\left(\alpha_k^c F_t^k \mathbf{M}_t\right) \tag{5.6}$$

with t the time-slice index, α_k^c the importance of the k-th shapelet and F_t^k the result of the shapelet transform. $M_t \in \{0, 1\}$ indicates whether or not the shapelet, at its best location, overlaps with the time-slice t.

Using shapelets and recurrent layers means the contribution of the shapelets is slightly different than for the original Grad-CAM. Here the index t corresponds to the time index for which the shapelet transform returned a value but the shapelet transform is also computed over a sequence. So we have a global time index t which goes from $l_{\text{in}-\text{shp}}$, the length of the shapelet transform time window, to the end of the discharge, and a local time index t' which gives the position of the shapelet in the shapelet transform. If at time-step t the best local position of the shapelet is found at t', it is possible that the best position of the shapelet at the global time-step $t+1$ is $t'-1$. When computing L_t^c the same shapelet can therefore contribute multiple times to the same location as it kept matching the same location while the global time-step t was moving forward. It is captured by $\mathbf{M}$ which can be non-zero at multiple locations. We included the ReLU function inside the sum to directly exclude negative contributions of individual shapelets.

The colors displaying the importance of the signals in this chapter are computed with c as the index of the impurity accumulation class. We only show the positive contribution of the shapelets to the impurity accumulation class, not the safe class.

Analysis of the learned shapelets

We first start by visualizing the shapelets learned by the different models. As SHP-12 and SHP-12 Mean have 90 and 66 shapelets, respectively, we only show the shapelet for the core radiation signal which is used by SHP-1 and SHP-1 Mean. The shapelets are shown in figure 5.6 for SHP-1, 5.7 for SHP-1 Mean, 5.8 for SHP-12, and 5.9 for SHP-12 Mean.

Comparing the length of the shapelets of the models with and without mean we see that SHP-1 and SHP-12 used longer shapelets of length 194 and 220 time-steps (388 and 440 milliseconds respectively). SHP-1 Mean uses the shortest shapelets at 34 time-steps and SHP-12 Mean is the second shortest at 125 time-steps.

We can also see that the shapelets from models without mean, figure 5.6 and 5.8 evolve more over the training epochs than the shapelets of the models with mean, see figure 5.7 and 5.9.

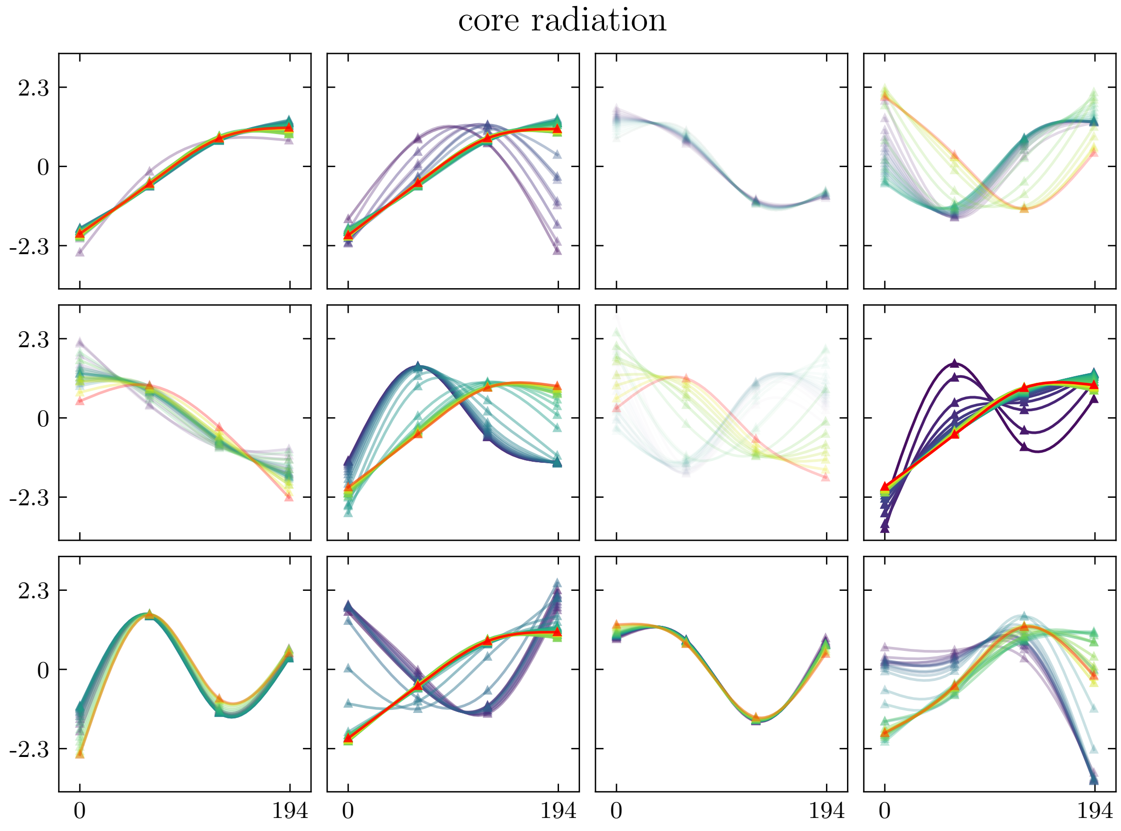


Figure 5.6: Shapelets learned by SHP-1. Shapelets are shown for all training epochs and in red for the last epoch. The triangular markers are the coefficients for each spline. The transparency of the shapelet is based on the multiplicative coefficient of the shapelet monitored by the pruning layer. The more transparent the less the shapelet contributes to the output.

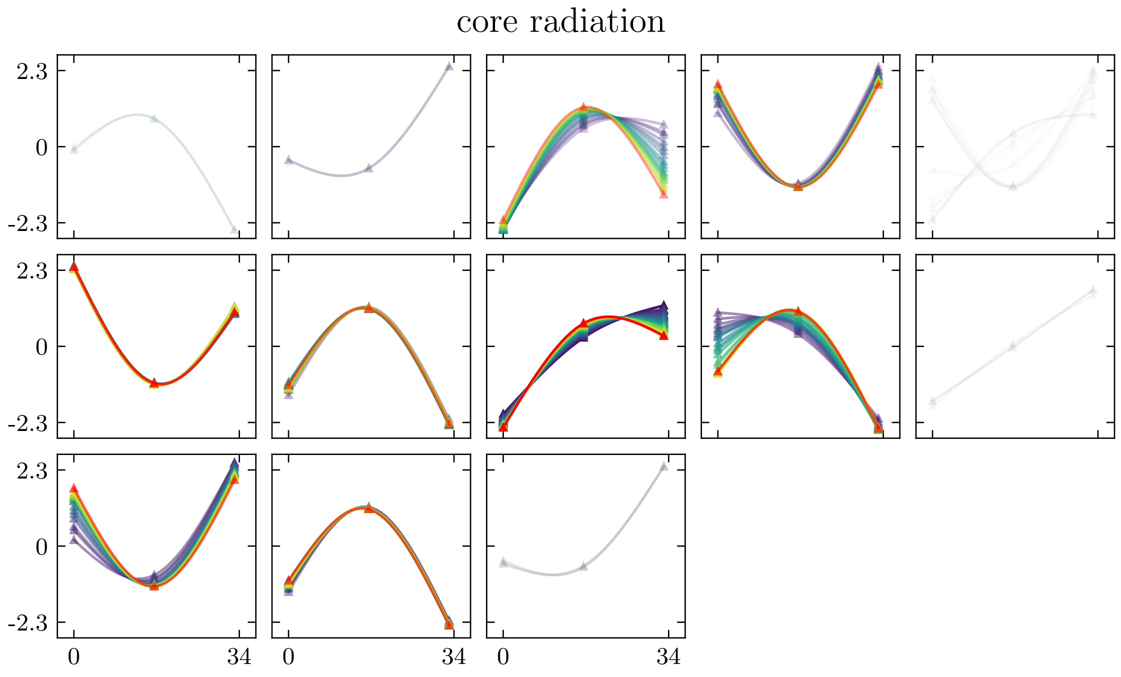


Figure 5.7: Shapelets learned by SHP-1 Mean. Shapelets are shown for all training epochs and in red for the last epoch. The triangular markers are the coefficients for each spline. The transparency of the shapelet is based on the multiplicative coefficient of the shapelet monitored by the pruning layer. The more transparent the less the shapelet contributes to the output.

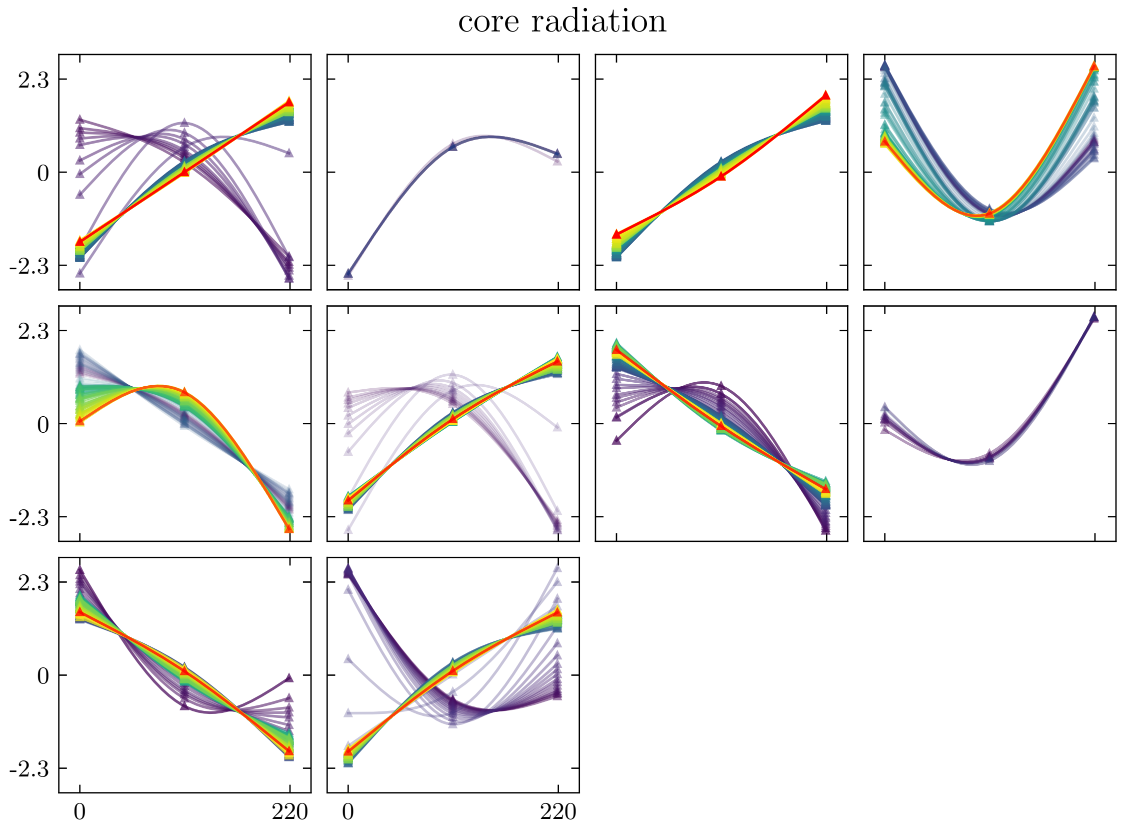


Figure 5.8: Shapelets learned by SHP-12. Shapelets are shown for all training epochs and in red for the last epoch. The triangular markers are the coefficients for each spline. The transparency of the shapelet is based on the multiplicative coefficient of the shapelet monitored by the pruning layer. The more transparent the less the shapelet contributes to the output.

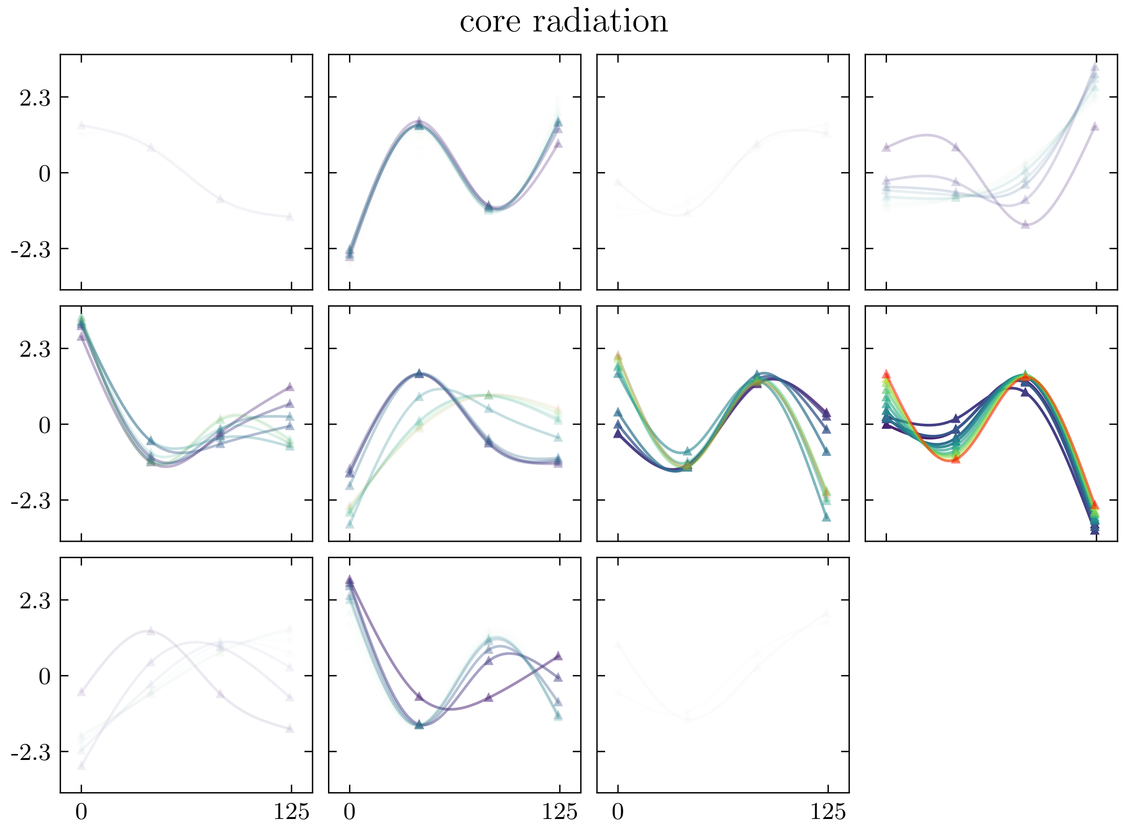


Figure 5.9: Shapelets learned by SHP-12 Mean. Shapelets are shown for all training epochs and in red for the last epoch. The triangular markers are the coefficients for each spline. The transparency of the shapelet is based on the multiplicative coefficient of the shapelet monitored by the pruning layer. The more transparent the less the shapelet contributes to the output.

SHP-12 Mean stands out as the model making the least use of the shapelets on the core radiation profile compared to the three others. But it should be noted that SHP-1 and SHP-1 Mean only use the core radiation signal so these two models do not have other signals to rely on.

Looking at the shapelets themselves we see figure 5.6 that for SHP-1 most shapelets converged towards shapelets with a positive derivative which matches with the increase radiation in the core of the plasma as high-Z impurities accumulate. A similar convergence is observed in figure 5.8 for SHP-12. This behavior is not apparent in models using the mean of the signals. It can indicate that SHP-1 Mean and SHP-12 Mean might either not use the core radiation signal at all or, more likely, only the mean of its value.

Analysis of the shapelets-data matches

After observing the learned shapelets we now look at the data the shapelets matched to. For this, we display the shapelets on top of the subsequences from the data they best matched to, for each discharge. We use the transparency of the subsequence to represent the importance of the shapelet when it matched to this subsequence. The visualization for SHP-1, SHP-1 Mean, SHP-12 and SHP-12 Mean can be seen in figures 5.10, 5.11, 5.12, and 5.13 respectively.

Three shapelets of SHP-1 clearly match a rise in the signal with a larger first derivative for three-quarters of the shapelet and a smaller, almost zero first derivative for the last quarter. We do not implement measures against redundant shapelets as this model illustrates. It is an interesting lead for future work that could reduce the number of shapelets and help with interpretation and visualization.

The top right shapelet in figure 5.10 is consistently matching similar data subsequences as the first two shapelets although it is a completely different shapelet with a negative first derivative for the first three quarters and a positive first derivative for the last quarter. The matching subsequences are the opposite of the shapelet. This behavior appears with normalized kernels and data. The two extreme cases when matching a shapelet to data are either the data is very close and the normalized shapelet measure tends toward zero or the normalized data is the opposite of the shapelet and the measures tend towards $2\sqrt{n}$ with n the length of the shapelet. Ref. [135] noted this behavior on the sharpened cosine similarity and attributed to it the space efficiency of the network stating that a single kernel can detect both identical and opposite patterns.

SHP-1 Mean, see figure 5.11, shows a completely different behavior. All shapelets except one seem to match the same data. As with SHP-1 a large amount of the data seem to be subsequences with constant positive first derivatives, but not exclusively. One other less notable trend is the opposite with decreasing subsequences. Overall the shapelets do not appear to match the data closely.

Interestingly SHP-12, see figure 5.12, shows close matches between the shapelets and the data but the matches for the core radiation signal are different from SHP-1. We do not see the same subsequences with positive derivatives. Two shapelets clearly match data with negative derivatives. A third shapelet shows the same tendency although less clearly. A fourth shapelet matches data that would correspond to subsequences with a positive constant second derivative with a minimum in the middle. These shapelets are

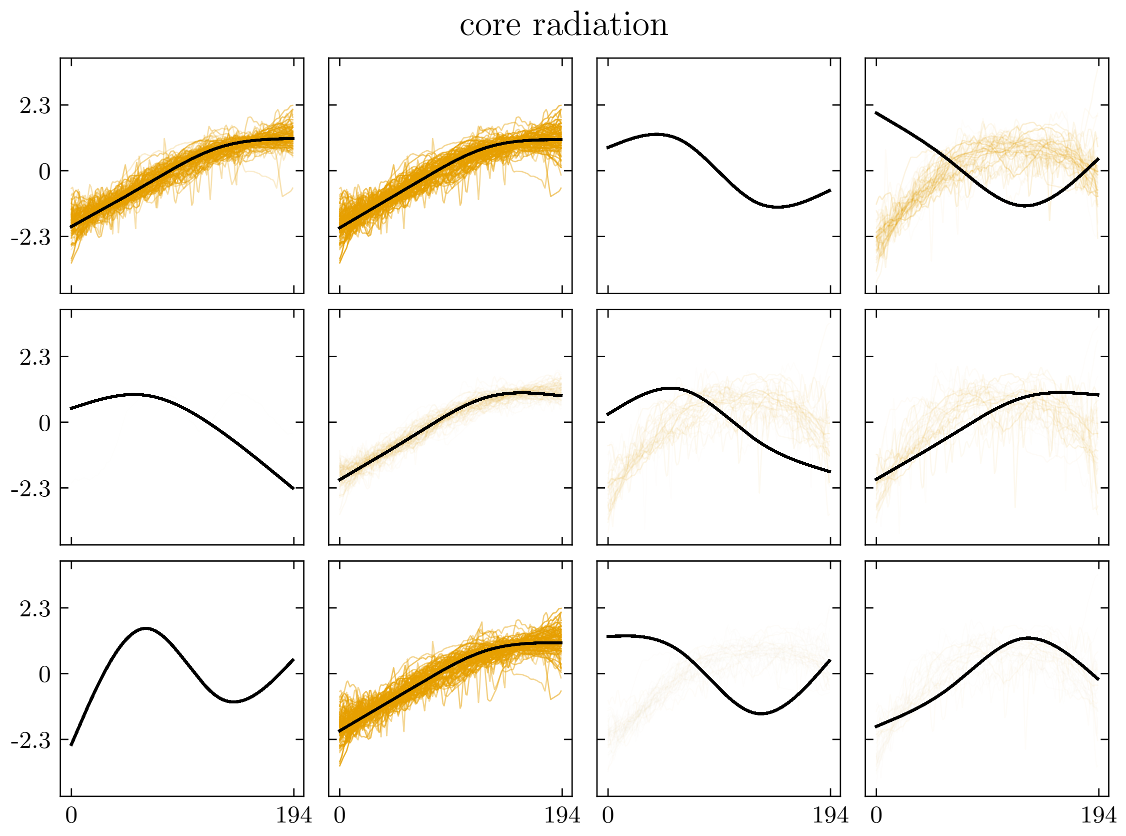


Figure 5.10: Shapelets (black) and the matched data subsequences with the best discriminative capacity (orange) for SHP-1. The transparency of the data subsequences indicates the importance of the shapelets when they best matched this subsequence. One subsequence is extracted per discharge.

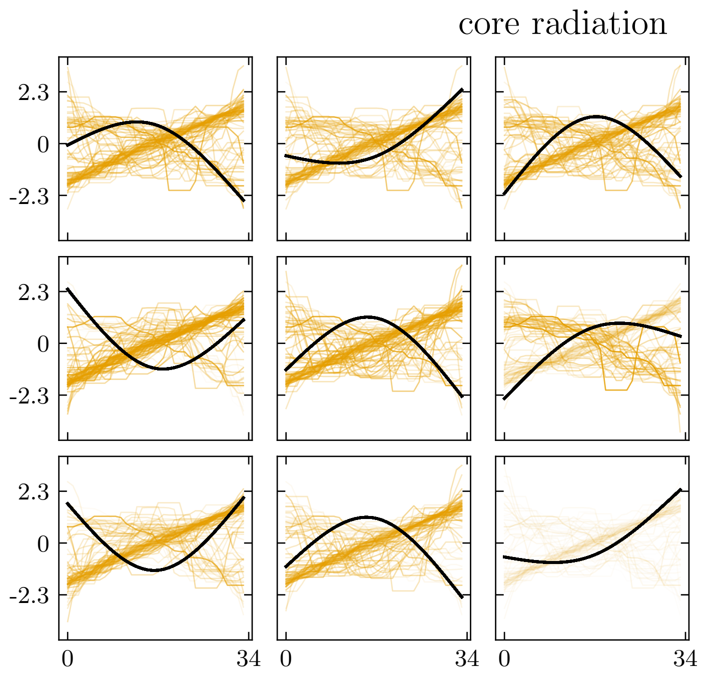


Figure 5.11: Shapelets (black) and the matched data subsequences with the best discriminative capacity (orange) for SHP-1 Mean. The transparency of the data subsequences indicates the importance of the shapelets when they best matched this subsequence. One subsequence is extracted per discharge.

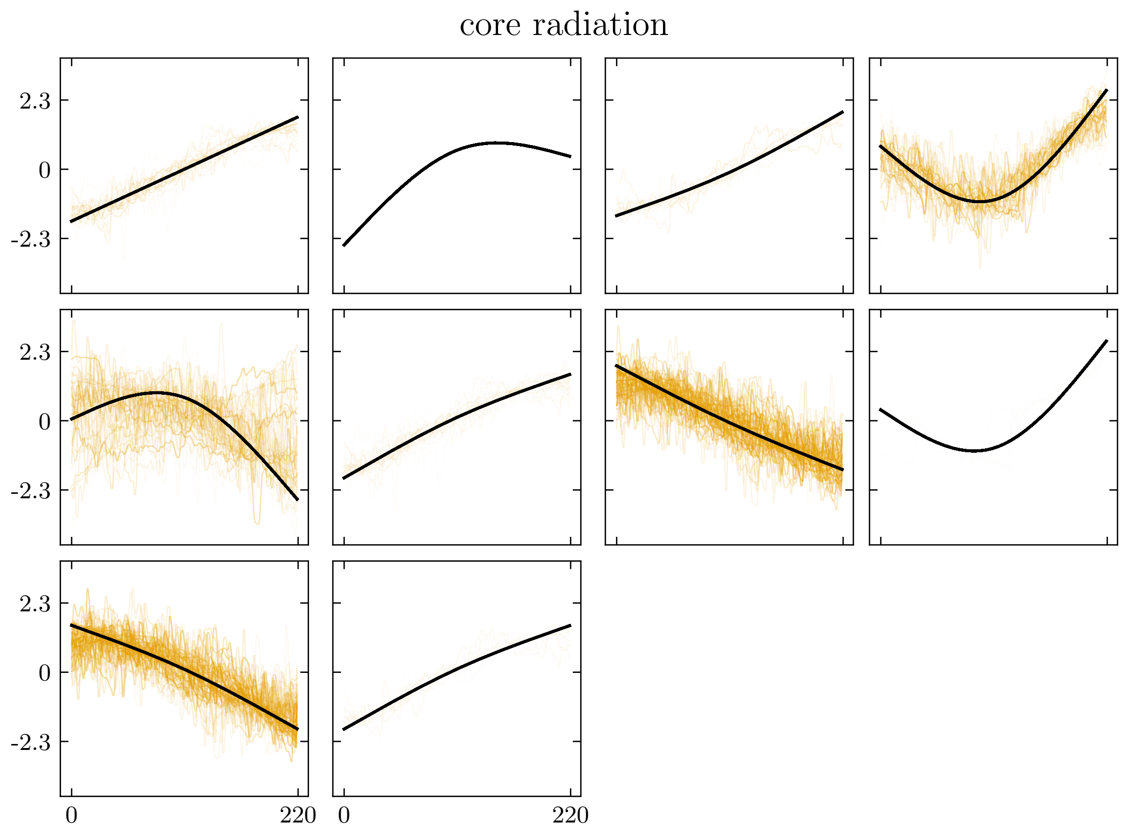


Figure 5.12: Shapelets (black) and the matched data subsequences with the best discriminative capacity (orange) for SHP-12. The transparency of the data subsequences indicates the importance of the shapelets when they best matched this subsequence. One subsequence is extracted per discharge.

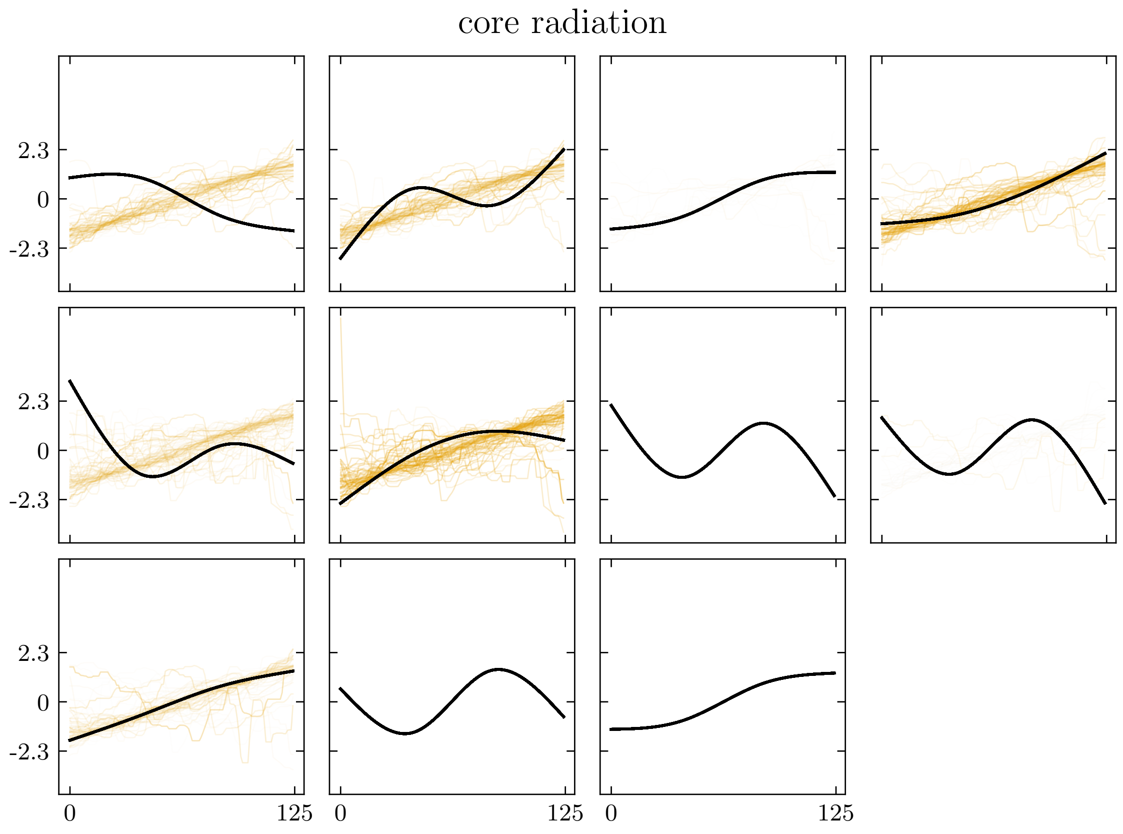


Figure 5.13: Shapelets (black) and the matched data subsequences with the best discriminative capacity (orange) for SHP-12 Mean. The transparency of the data subsequences indicates the importance of the shapelets when they best matched this subsequence. One subsequence is extracted per discharge.

not expected for impurity accumulation detection. The model might detect the impurity accumulations from the other signals and use the core radiation differently than SHP-1.

Lastly, we look at the subsequences matched by the shapelets of SHP-12 Mean. Again this model mostly identifies the sequences with constant positive derivatives but the shapelet themselves do not have the same shape. We do not see the shapelets being close to the average of all data subsequences matched as for SHP-1 or SHP-12. As this was also observed for SHP-1 Mean we attribute it to the use of the mean of the signals in the models.

5.6.5 Predictions analysis

In the final section of this chapter, we put together the model predictions and visualization methods to analyze a few discharges. The signals are colored according to their importance to the impurity accumulation prediction. The shapelets are placed at their most important position to the prediction which also overlaps with the most important time-slice of the signal and normalized with respect to the subsequence they matched. Their transparency is proportional to their importance at that location. For the Threshold model, the heatmap is either zero if the signal is below the classification threshold or one, the maximum, if the signal is above the classification threshold.

JET-Late discharge number 90319

The first discharge we selected is number 90319 from the JET-Late dataset for which all models except SHP-12 provide an excellent classification. Figure 5.14 shows the Threshold model detects the impurity accumulation event at the end of the discharge correctly, although slightly in advance compared to the ground truth.

The SHP-1 model produces almost identical predictions, see figure 5.15. We can see the importance of the signal increase every time it rises with the light green color and the most important location for the shapelets is the early rise in core radiation between 10 and 10.5 seconds resulting in the same early classification as the Threshold model.

The same results are observed in figure 5.16 for the SHP-1 Mean model. As we visualized in the previous sections the shapelets of this model do not provide much information. The important section of the signal is between 10.3 seconds and the end of the signal and follows the variations of the signal quite well which also illustrates that the absolute value of the signal seems more important than the normalized shapelets.

SHP-12 Mean is in agreement with the previous three models and labels the same section as impurity accumulation but again as for SHP-1 Mean the shapelets do not seem to provide much information on the importance of the signals.

SHP-12 is the only model detecting wrong events with the first detection around 8.4 seconds and a second around 10 seconds, see figure 5.18. For the first event, the heatmap and placement of the shapelets seem to indicate that relative increases in the temperature peaking factor T_e at 7.3 and 8.15 seconds and plasma current I_{err} at 7.38 and 7.82 seconds caused the classification. This behavior is already observed on JET-Early as shown in figure 5.19 for the discharge number 82657. Right before the second 18 the heatmap and shapelet placement indicates the importance of the T_e and I_{err} signals. Just after

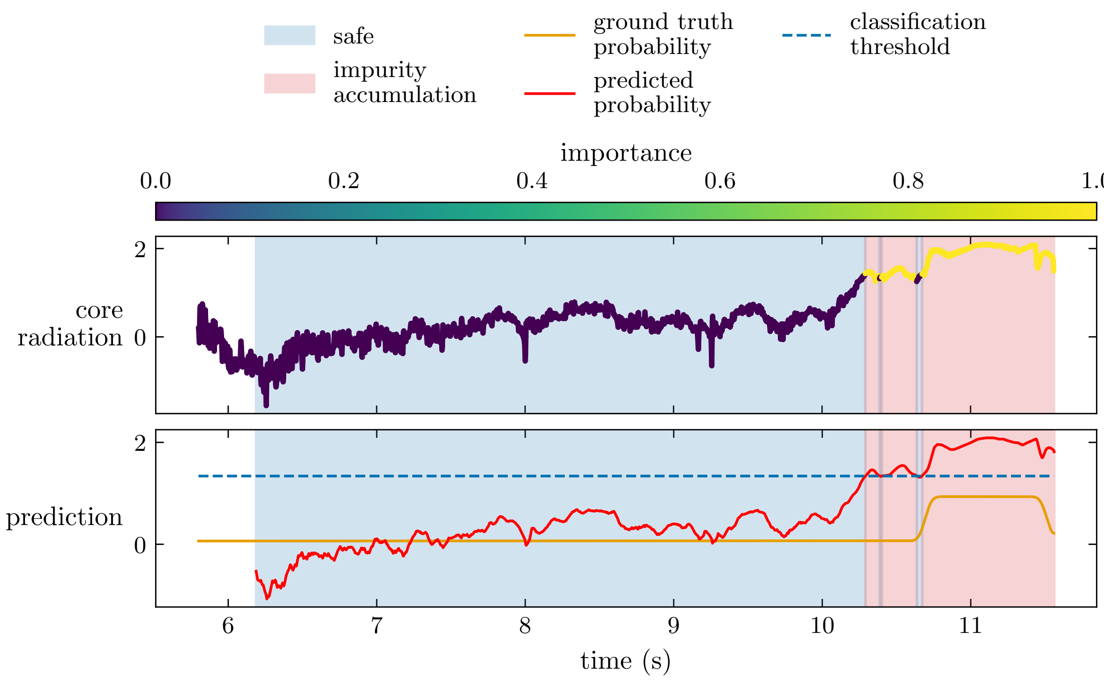


Figure 5.14: Prediction of the Threshold model on discharge number 90319 from the JET-Late dataset.

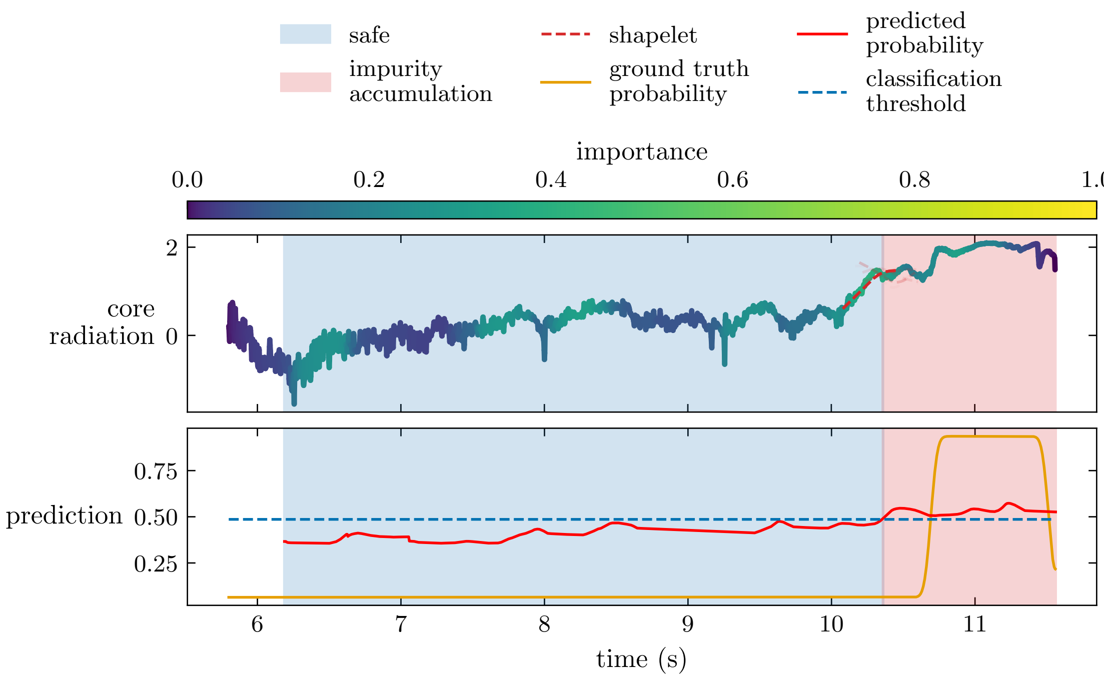


Figure 5.15: Prediction of the SHP-1 model on discharge number 90319 from the JET-Late dataset.

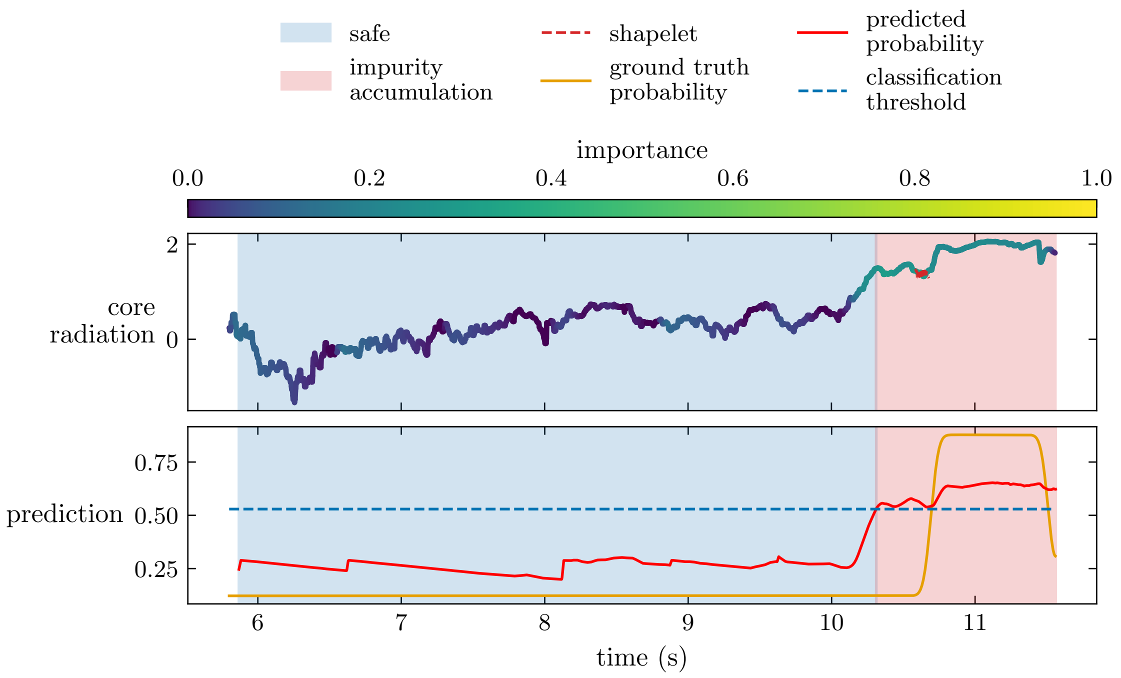


Figure 5.16: Prediction of the SHP-1 Mean model on discharge number 90319 from the JET-Late dataset.

18 seconds, we see a light green color for the core radiation color. We learn from this visualization, together with figure 5.12 showing the pattern learned by the shapelets for the core radiation that the SHP-12 model detects events that often occur before and after the start of the impurity radiation. After a sharp increase in core radiations, the is often a slight drop. Although the drop in T_e after 18 seconds in figure 5.19 is what experts use to identify the cooling of the plasma core the model learned to identify a small rise before. This could explain the worse false positive rate per discharge of SHP-12 on JET-Early and JET-Late, see table 5.3 and 5.4. As the physical description of high-Z impurity accumulation events does not include the behavior described by SHP-12 may be a different loss metric for the model or more data would steer the model to another solution. If this behavior persists and performance improves with larger datasets the predictions should be further analyzed to understand the link between the signals and the event.

For SHP-1 we observe that a lot of false positives in JET-Late are transitions from the high confinement mode to the low confinement mode. In discharges where the power is only marginally above the H-L power threshold [167] a step-down in the auxiliary heating by neutral beam injection (NBI) causes an H-L back transition. These events can be abrupt and depending on the plasma conditions, they might cause an accumulation of impurities in the core. It does not necessarily lead to disruptions but it is a very early event from which the plasma can further deteriorate. In that sense, SHP-1 predicts impurity accumulations that do not only directly lead to disruptions but also link to H-L back transition and could therefore be used as a first layer for disruption avoidance. Adapting the model for this case would require changing the labeling to include the

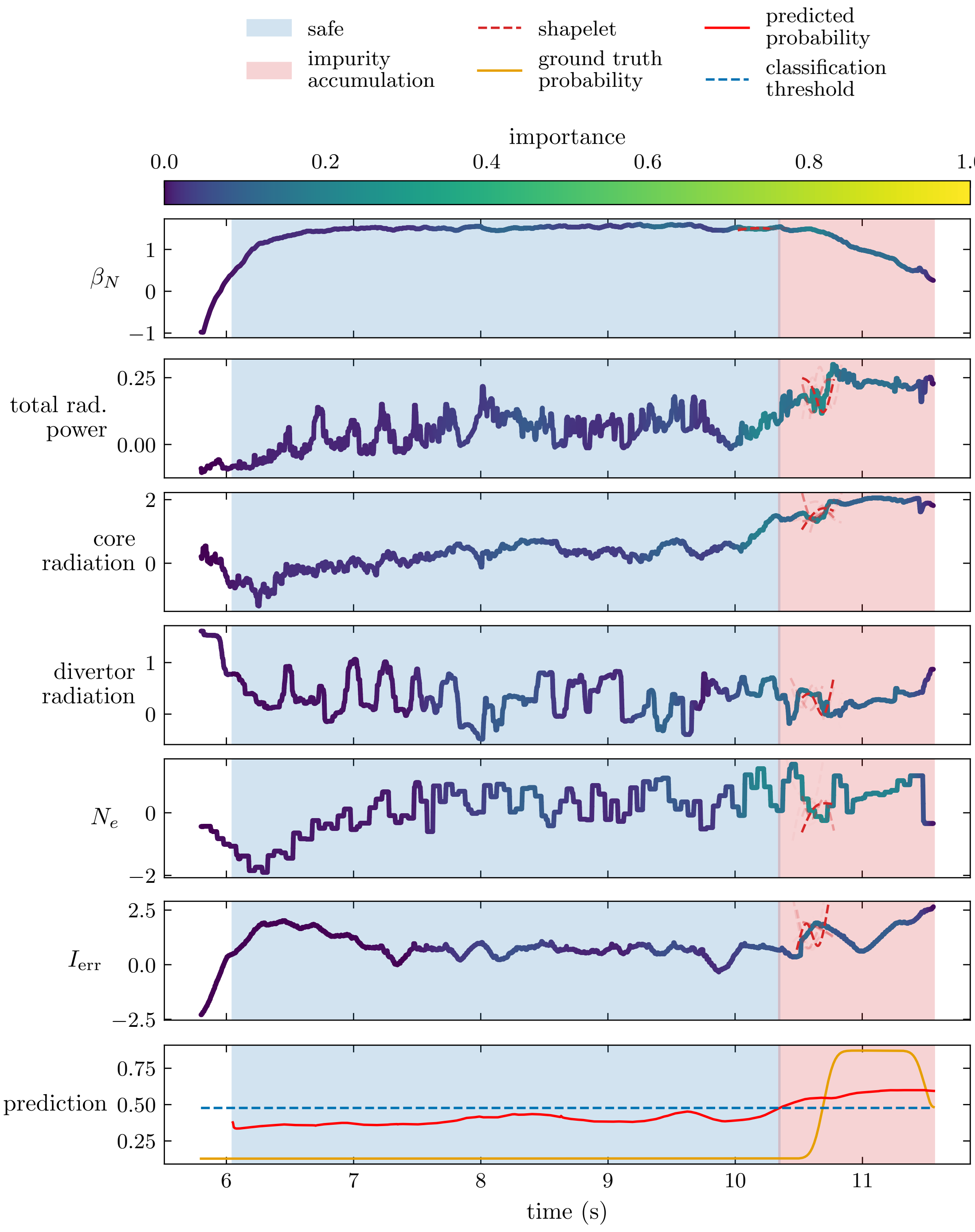


Figure 5.17: Prediction of the SHP-12 Mean model on discharge number 90319 from the JET-Late dataset.

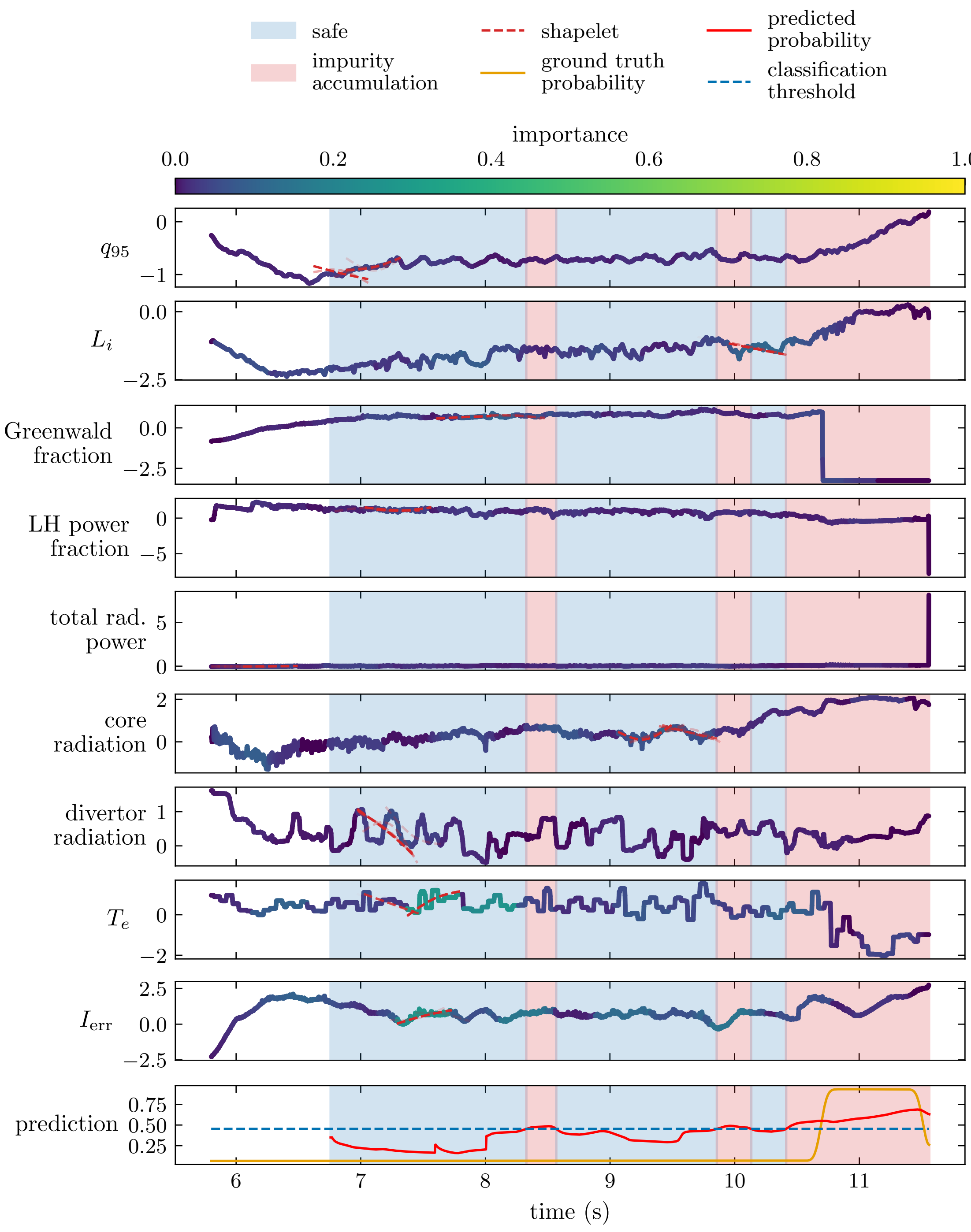


Figure 5.18: Prediction of the SHP-12 model on discharge number 90319 from the JET-Late dataset.

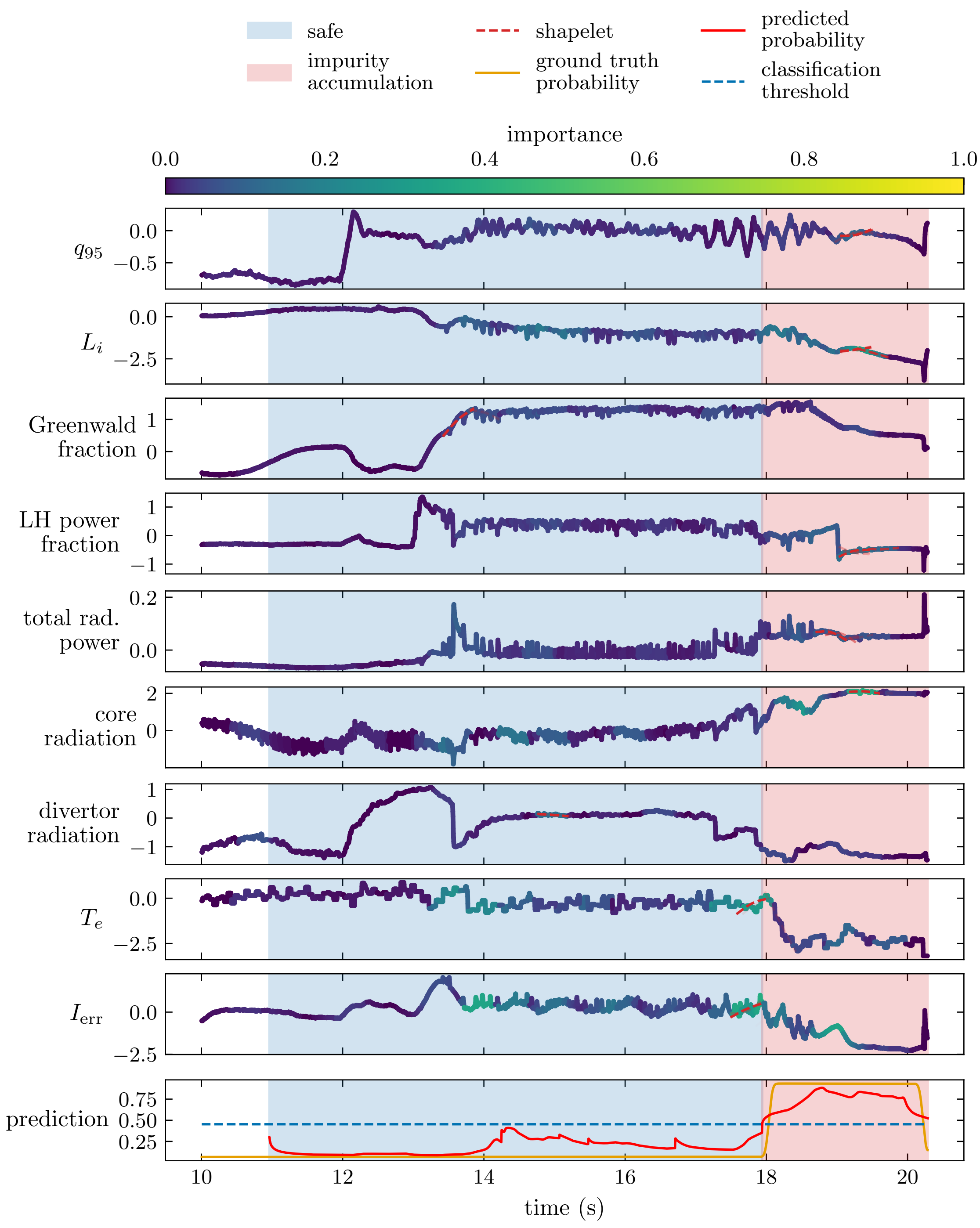


Figure 5.19: Prediction of the SHP-12 model on discharge number 82657 from the JET-Early dataset.

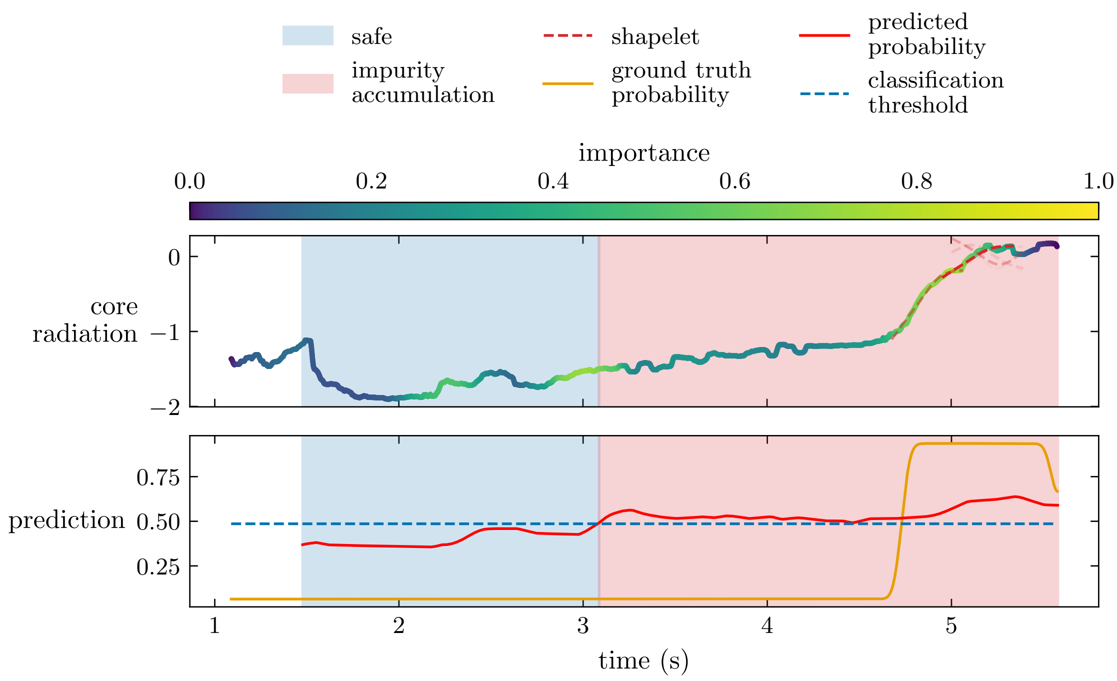


Figure 5.20: Prediction of the SHP-1 model on discharge number 29284 from the AUG dataset.

impurity accumulations that result from abrupt H-L transitions.

AUG discharge number 29284

Finally, we show two examples from the AUG dataset. On this dataset, all models that include the absolute values of the signals failed with a predicted impurity accumulation probability never crossing the detection threshold. Only SHP-1 and SHP-12 labeled any time-slices as impurity accumulations.

In figure 5.20 we see the application of SHP-1 to the AUG discharge number 29284. From the heatmap, we see again that SHP-1 is identifying the increase in core radiation. After 3 seconds, the model identifies the increase in core radiation and wrongly labels the time-slices as impurity accumulations. The real significant rise starting at 4.7 seconds is identified as the most important part of the signal for this discharge.

For SHP-12, see figure 5.21 the same patterns are observed as for JET-Late and JET-Early. The drop in core radiation at 1.5 seconds simultaneously with the rise in T_e are deemed important as well as I_{err} slightly later. Although these signals are clearly not linked to impurity accumulation in this discharge SHP-12 wrongly classifies the impurity accumulation already from 2.3 seconds on.

AUG discharge number 37851

The results for discharge 37851 are better and both models detect the event, SHP-12 slightly later than SHP-1. We see figure 5.22 the correct placement of the shapelet at

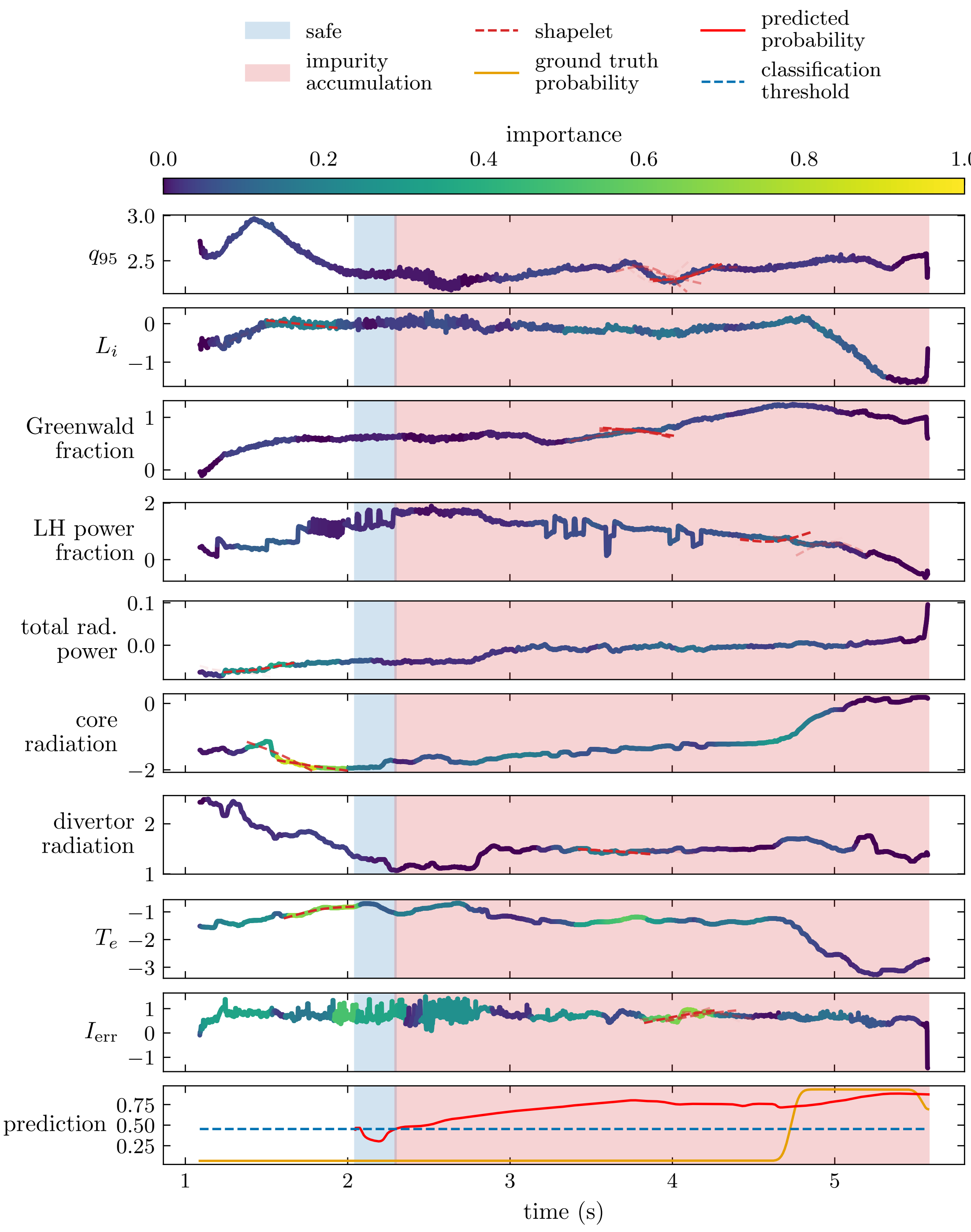


Figure 5.21: Prediction of the SHP-12 model on discharge number 29284 from the AUG dataset.

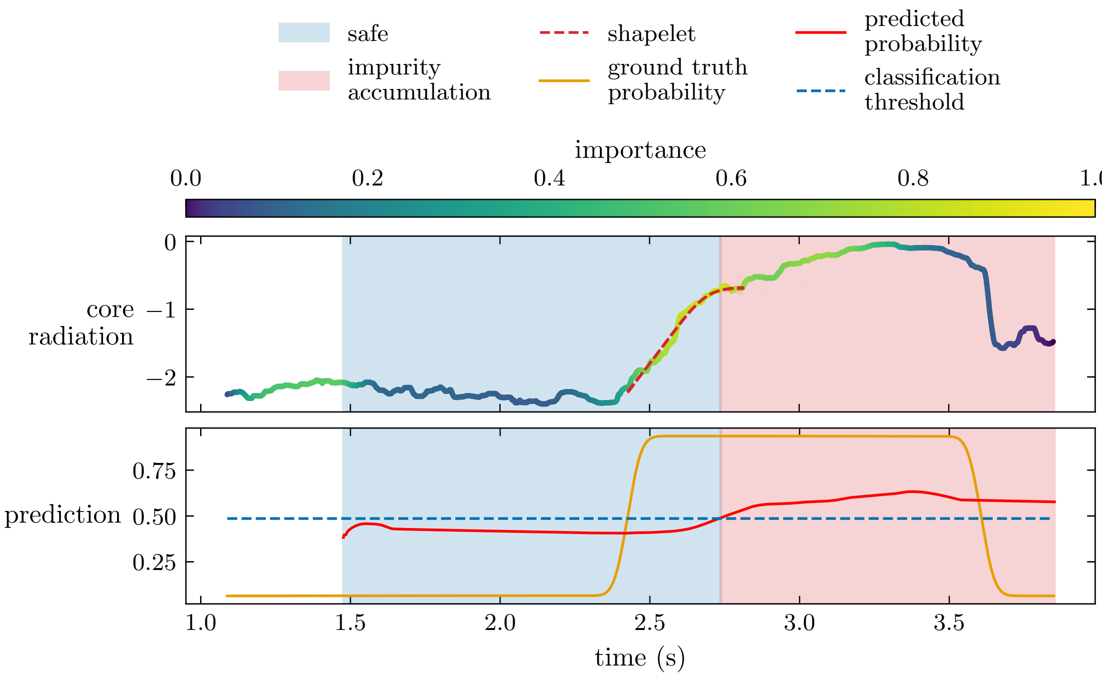


Figure 5.22: Prediction of the SHP-1 model on discharge number 37851 from the AUG dataset.

the beginning of the impurity accumulation event for SHP-1. The T_e and I_{err} signals are dominant in the decision of the SHP-12 model as seen in figure 5.23.

5.6.6 Conclusion

We have built two models just using the shapelets, SHP-1, and SHP-12, which behave very differently. The results on the JET-Early dataset show very similar per slice true positive and false positive rates. SHP-12 has three more per discharge false positives than SHP-1 which only has two false positives. The trend remains on JET-Late with 58.7% per discharge false positive rate for SHP-1 and 74% for SHP-12. Using neural networks it is often surprising to see worse performance when adding input data, but thanks to the interpretability of our architecture we were able to identify the difference between the two models. SHP-12 learned a specific pattern that happens right before and after the traditional explanation of plasma behavior when high-Z impurities accumulate in the core. With high-Z impurities in the core, we observe a rise in core radiation which is learned by SHP-1, while SHP-12 learned the slight drop in core radiation after the increase. It combines it with a rising T_e peaking factor and I_{err} to identify the impurity events. Regarding the T_e peaking factor, the main consequence of the impurity accumulation is a lowering which happens after the rise detected by SHP-12. Although the drop is more significant the model learned the small rise that might appear right before. Different explanations around physical phenomena can be interesting but in this case, we see that this description of the impurity accumulation event did not match the JET-Late and AUG

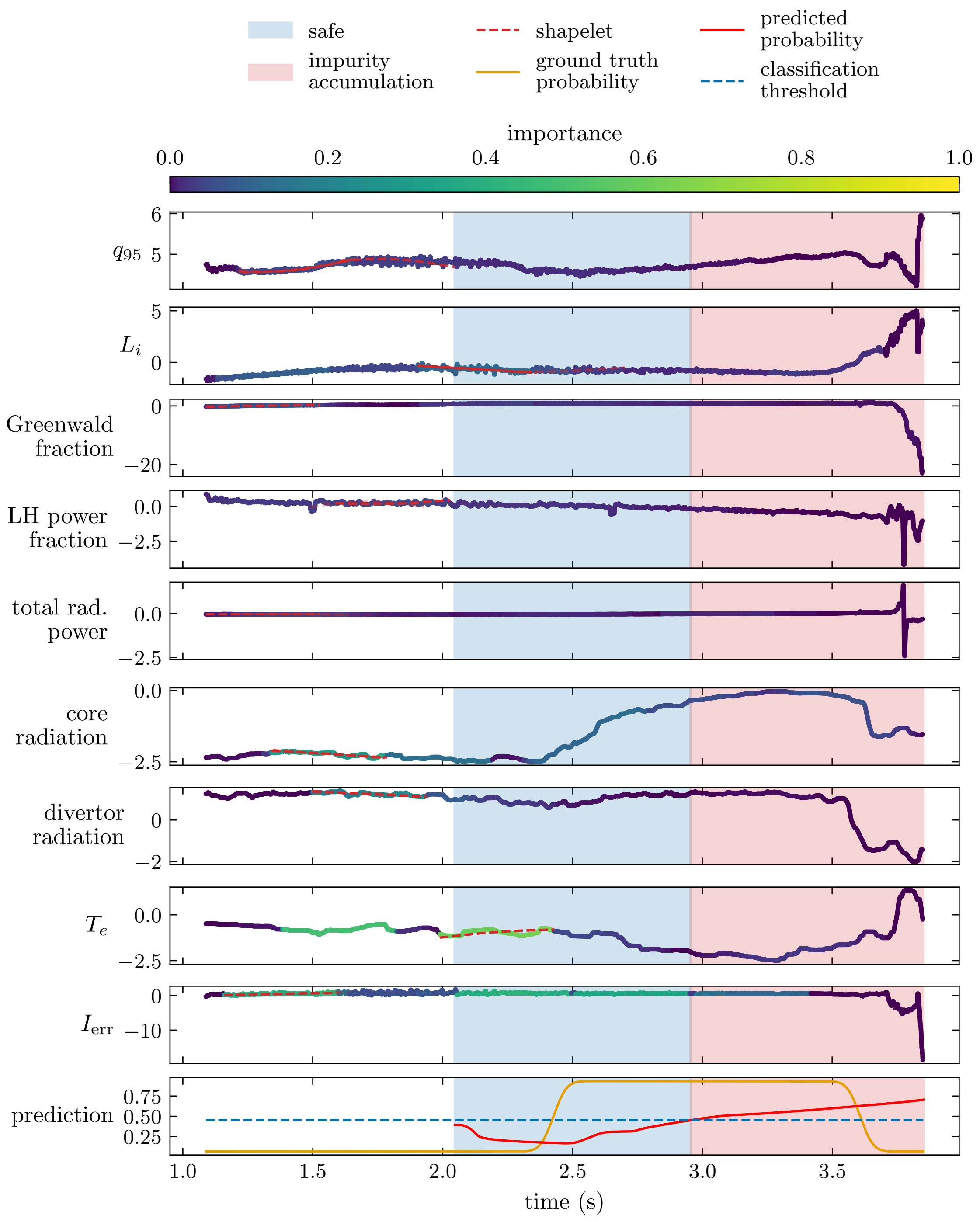


Figure 5.23: Prediction of the SHP-12 model on discharge number 37851 from the AUG dataset.

datasets.

For SHP-1 we observe that many false positives are due to another physical event, specifically, the H-L back transition after the NBI turns off while the power is just marginally above the H-L threshold.

SHP-1 Mean is producing almost identical results to the Threshold model. They both label the time-slices as impurity accumulations when the core radiation peaking factor signal cross a fixed value.

SHP-12 Mean is similar but also includes other signals. As with SHP-1 Mean the shapelets do not appear to be used by SHP-12 Mean.

Although the interpretable architecture restricts the liberty of the model which can come at a small performance cost it allows us to understand the inner workings of the two models and can help in deciding which model should be applied in which circumstances. We saw that for the transfer task to AUG, SHP-1 does outperform SHP-12 for exactly the reasons identified on JET-Early and JET-Late. And for a zero-shot transfer application, the Threshold model cannot be used leaving the SHP-1 as the best of the five models.

6 Conclusions and Outlook

In this thesis, we investigated the use of shapelet-based neural networks for disruption prediction. Specifically, binary and multi-class disruption prediction at JET and ASDEX-Upgrade. We proposed a different formulation of the shapelet transform that makes use of spline interpolation to control the regularity of the learned shapelets and decrease computational costs. First, we created a dataset with data from JET in its carbon wall configuration. Our second task was the study of event detection for impurity-driven instabilities. Impurity accumulations became the first cause of disruption at JET after the installation of the ITER-like wall (ILW). As the name indicates, ITER will have similar walls, therefore impurity accumulation disruptions are to be expected. The prediction of such events is critical to guarantee a safe operation of the ITER tokamak without major disruptions. Using a small sample of discharges from ASDEX-Upgrade, we evaluated our models on a zero-shot transfer learning task which corresponds to a scenario where a predictor is built before the completion of ITER and applied on day one.

Shapelet transforms are comparable to convolutional neural network cells, with a few key differences. While convolutional kernels are kept small, usually not longer than 5 pixels or time-steps, and multiple stacked layers are used to capture larger patterns, shapelets are of the length of the pattern to be identified and a single layer is used. The second difference is local normalization. The normalized shapelet transform normalizes both the input local patch and the kernel to zero-mean and standard deviation of one. Lastly, the Euclidean distance is used in the case of shapelets, while convolutions use the dot product. We showed that if the local z-normalization is applied to convolutions, the square root of the dot product is equal to the shapelet transform up to a constant $2 * N$ with N the dimension of the kernels.

The recently developed sharpened cosine transform is similar to the normalized shapelets as it introduces the normalization by the norm of the input and kernel. The parallel development of the normalized shapelets and sharpened cosine similarity illustrates the need for more robust operations as the standard convolutions, especially with the growing number of zero-shot and transfer learning applications.

The bigger length of the shapelets increases interpretability as patterns are captured in a single operation. Unfortunately, stability and resemblance to the data of the learned shapelets are reduced. In practice, the shapelets can be a lot noisier than the data they trained on. To address this issue, we proposed to learn C^2 shapelets by using cubic spline interpolation with a small number of degrees of freedom. The effectiveness of our method is demonstrated on the UCR2018 dataset, showing no statistical differences in performance with the standard shapelets and producing shapelets that better resemble the data. This new formulation of the shapelets has the added benefit of reduced computational cost during training as the optimization is independent of the full length of the shapelets and only scales with the number of degrees of freedom.

Our first study is based on the classification from Peter de Vries of JET disruptions between 2000 and 2010, prior to the installation of the metallic wall at JET. A set of 167 disruptions with 7 different labels were selected, along with 200 safe shots from the same period. We compared our shapelet based neural-network with two standard architectures from the literature. First, stacked support-vector machines with specifically designed input features. Secondly, a neural network using long-short term memory (LSTM) cells designed for time series data. For each model, three independent versions are trained to account for the variations due to random initialization in both the hyperparameter tuning and the neural network weights. The final models are built by using majority voting on the three independent versions.

The tasks we evaluated the models on are a binary classification of the time-slices as safe or disruptive and a multi-class classification where the type of disruption has to be predicted as well. On the binary classification tasks, our shapelet-based model outperformed the other two models. The results are more difficult to analyze for the multi-class task due to several restrictions when building the dataset, such as no available time-stamps for the different classes and a single class per time-slice while different classes could be overlapping. We analyzed the trends in the confusion matrices and found that the support-vector machine model and our model performed similarly while the LSTM neural network performed worse.

In order to understand which elements of the models contributed to differences in performance, we conducted an ablation study. A model combining the input features from the support-vector machine model with a multi-layer perceptron is constructed to evaluate the impact of the support-vector machine models compared to the complex neural networks. This model performed better than the original architecture from the literature, but not as well as our shapelet-based model. We conclude that although the neural network contributes to better performances, it is not the only element necessary to match our model. Our model used both shapelets and the mean of the input signals over a recent time window. The second model of the ablation study is similar, except the connection with the mean of the signals is removed, leaving only the shapelets as connections to the input. This model could not identify the different disruption classes but perform just as well as the original model on the binary task. The shapelets use local normalization, meaning that our model can predict disruptions without the need of the absolute value of the signals. This could be a key element toward cross-tokamak disruption predictors where models using the absolute value of the signals fail due to the difference between machines, even if normalized physical quantities are used.

After the change from carbon to metallic walls at JET, the main cause of disruptions became too high-Z impurity accumulations in the core. As the walls of ITER will be similar to JET, the prediction of the impurity accumulation events that lead to disruptions is of high importance. This problem requires accurate predictions of impurity accumulations on existing devices, but also the ability to predict such events on unseen data, i.e. cross-tokamak predictions.

To simulate such an environment we used three datasets from two different tokamaks.

The first two datasets come from two different periods of the JET operations. JET-Early, the first dataset coming from the year 2012, and JET-Late, the second from the years 2015 to 2020. Lastly, we used nineteen discharges with disruptions after impurity accumulations at ASDEX-Upgrade. This last dataset is named AUG.

Twelve signals deemed relevant to our task were selected. The physical quantities were normalized to make them less machine-dependent and allow for an easier cross-tokamak transfer learning.

The neural-network architecture from the JET-carbon application was improved by the addition of recurrent short-long term memory (LSTM) cells. Different versions of the model called SHP-12 Mean and SHP-12, with and without the inclusion of the sliding mean of the input signals were implemented. Additionally, from a simple statistical analysis of the data, one signal was identified to be strongly correlated with the impurity accumulation events: the core radiation peaking factor. It was expected as high-Z impurity accumulation at the center of the plasma drastically increases the radiation compared to the rest of the plasma, which is the behavior captured by an increase of the core radiation peaking factor. We therefore also implemented two models using only the core radiation peaking factor as input, named SHP-1 Mean and SHP-1. As a simple baseline, we also built a model, called Threshold, based on a simple rule: whether the core peaking radiation factor crosses a certain threshold.

The training and hyper-parameter tuning is performed on JET-Early. The models are evaluated on the test set of the JET-Early dataset as well as on the entirety of the JET-Late and AUG datasets without any modification. This zero-shot transfer learning is the hardest task for cross-tokamak predictions and is often used as a mediocre baseline before introducing relaxations where data from the new machine is leaked to the training sets. We strictly focused on the harder zero-shot transfer learning task.

The simple Threshold model performed extremely well on JET-Early and JET-Late with the SHP-1 and SHP-12 performing worse. The SHP-1 Mean is almost identical to the Threshold model, and the SHP-12 Mean model is in between SHP-1 Mean and the models without mean.

We thoroughly analyzed our models as they are designed with interpretability in mind. The learned shapelets can be retrieved and analyzed to see the features identified by the models. We found that the models with the inclusion of the mean bypassed the shapelets and mostly used the absolute value of the input signals, while the models without the mean clearly identified specific shapes to predict the impurity accumulation events. It allowed us to identify that SHP-1 correctly detects the rise in the core radiation peaking factor as a discriminative feature. On the other hand, we found that SHP-12 went against the expectations and learned variations in the signals that occur just before and after the start of the impurity accumulation event. These features appeared to be enough to provide correct predictions on JET-Early but transferred poorly to JET-Late. The analysis also showed that SHP-1 is consistently detecting a second event that was not considered when building the datasets. When the power input is just slightly above the H-L power threshold and the neutral beam injection cuts off, an H-L back transition can occur. As a result of the plasma changes, high-Z impurities can accumulate in the core. After those events, the plasma conditions are not as deteriorated as for the events identified in our datasets, but could be very informative for early signs of instability in a context of disruption avoidance instead of prediction.

The last evaluation on AUG showed that SHP-1 Mean, SHP-12 Mean, and Threshold (the three models using the absolute value of the signals) cannot predict impurity accumulations on another machine as they did not detect a single time-slice. The SHP-12 model obtains too many false alarms which are triggered after the identification of the particular features we consider irrelevant. The SHP-1 model performs best, but also with a number of false alarms as the local normalization makes it unable to distinguish between a slight increase in core radiations and very significant ones. A larger set of discharges from ASDEX-Upgrade, also including ones without any impurity accumulation event is now necessary to further estimate the performances of the SHP-1 model and to refine our architecture in preparation for ITER.

6.1 Outlook

As a recurrent theme in data-based applications, the next step would be to evaluate our models on a larger dataset from ASDEX-Upgrade and include other machines. The first benefit would be better statistics for the evaluations instead of the ones obtained on the nineteen discharges from ASDEX-Upgrade. The second benefit is to test the cross-tokamak predictions, starting from a different machine than JET, or training on multiple machines before applying the model to an unseen one. This would best represent the scenario for ITER where data from all existing machines should be used to design the best predictors. This requires a massive effort from the entire community to label discharges in a consistent way, identify common signals between the machines, and the proper machine-independent normalization of the physical quantities.

A comparison of the normalization scheme between the normalized shapelet transform and the sharpened cosine similarity would be interesting for transfer-learning applications. Replacing the single shapelet layer with multiple layers of sharpened cosine similarity would reduce the interpretability of the models. It is always a trade-off between performance and interpretability, and moving from the shapelets to slightly less interpretable methods to increase predictive capability would also be a natural continuation of this thesis. But importantly, further work should as much as possible attempt to reuse similar data, and therefore produce comparable statistics.

Lastly, an adaptation of our zero-shot transfer learning scenario to the *"glimpses"* strategy would be an important study, as it is important for ITER to not only have predictors on day one but also to update them as experiments are conducted. The strategy *"glimpses"* includes a few discharges from a new machine to the training data of an older machine to guide the learning process. It has been shown to improve the results over the harder zero-shot transfer learning scenario.

Bibliography

[1] John Cook, Naomi Oreskes, Peter T Doran, William RL Anderegg, Bart Verheggen, Ed W Maibach, J Stuart Carlton, Stephan Lewandowsky, Andrew G Skuce, Sarah A Green, et al. Consensus on consensus: a synthesis of consensus estimates on human-caused global warming. *Environmental Research Letters*, 11(4):048002, 2016.

[2] Ralph F Keeling and Charles D Keeling. Atmospheric monthly in situ co2 data-mauna loa observatory, hawaii. *Scripps CO2 program data. UC San Diego Library Digital Collections*, 2017.

[3] Paola Arias, Nicolas Bellouin, Erika Coppola, Richard Jones, Gerhard Krinner, Jochem Marotzke, Vaishali Naik, Matthew Palmer, G-K Plattner, Joeri Rogelj, et al. Climate change 2021: The physical science basis. contribution of working group14 i to the sixth assessment report of the intergovernmental panel on climate change; technical summary. 2021.

[4] R Aymar, P Barabaschi, and Y Shimomura. The iter design. *Plasma physics and controlled fusion*, 44(5):519, 2002.

[5] B Bigot. Progress toward iter's first plasma. *Nuclear Fusion*, 59(11):112001, 2019.

[6] Thomas Alfred John Body. *Development of Turbulence Simulations for the Edge & Divertor and Validation against Experiment.* PhD thesis, Technische Universität München, 2022.

[7] FC Schuller. Disruptions in tokamaks. *Plasma Physics and Controlled Fusion*, 37 (11A):A135, 1995.

[8] A. S Eddington. The internal constitution of the stars. *The Observatory*, 43(853): 341–358, 1920.

[9] Marcus Laurence Elwin Oliphant, Paul Harteck, and Ernest Rutherford. Transmutation effects observed with heavy hydrogen. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 144(853):692–703, 1934.

[10] J Ongena, R Koch, R Wolf, and H Zohm. Magnetic-confinement fusion. *Nature Physics*, 12(5):398–410, 2016.

[11] Cathey Cevallos et al. *Non-linear magnetohydrodynamic simulations of edge localised modes (ELMs).* PhD thesis, Technische Universität München, 2021.

[12] S Zheng, DB King, L Garzotti, E Surrey, and TN Todd. Fusion reactor start-up without an external tritium source. *Fusion Engineering and Design*, 103:13–20, 2016.

[13] John Wesson and David J Campbell. *Tokamaks*, volume 149. Oxford university press, 2011.

[14] Jet Team et al. Fusion energy production from a deuterium-tritium plasma in the jet tokamak. *Nuclear Fusion*, 32(2):187, 1992.

[15] M Keilhacker, A Gibson, C Gormezano, PJ Lomas, PR Thomas, ML Watkins, P Andrew, B Balet, D Borba, CD Challis, et al. High fusion performance from deuterium-tritium plasmas in jet. *Nuclear Fusion*, 39(2):209, 1999.

[16] A Bock, E Fable, R Fischer, M Reich, D Rittich, J Stober, M Bernert, A Burckhart, H Doerk, M Dunne, et al. Non-inductive improved h-mode operation at asdex upgrade. *Nuclear Fusion*, 57(12):126041, 2017.

[17] T Fujita, Y Kamada, S Ishida, Y Neyatani, T Oikawa, S Ide, S Takeji, Y Koide, A Isayama, T Fukuda, et al. High performance experiments in jt-60u reversed shear discharges. *Nuclear Fusion*, 39(11Y):1627, 1999.

[18] Samuel E Wurzel and Scott C Hsu. Progress toward fusion energy breakeven and gain as measured against the lawson criterion. *Physics of Plasmas*, 29(6):062103, 2022.

[19] G Federici, C Bachmann, L Barucca, C Baylard, Wolfgang Biel, LV Boccaccini, C Bustreo, S Ciattaglia, F Cismondi, V Corato, et al. Overview of the demo staged design approach in europe. *Nuclear fusion*, 59(6):066013, 2019.

[20] Miroslav Kubat and Kubat. *An introduction to machine learning*, volume 2. Springer, 2017.

[21] Peter Harrington. *Machine learning in action*. Simon and Schuster, 2012.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, chapter Injecting Noise at the Output Targets, page 239. MIT Press, 2016. http://www.deeplearningbook.org.

[23] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687, 2020.

[24] M Emre Celebi and Kemal Aydin. *Unsupervised learning algorithms*. Springer, 2016.

[25] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] Diego Ardila, Atilla P Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine*, 25(6):954–961, 2019.

[27] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[28] Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

[29] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[30] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[32] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[33] T Wakatsuki, T Suzuki, N Hayashi, N Oyama, and S Ide. Safety factor profile control with reduced central solenoid flux consumption during plasma current ramp-up phase using a reinforcement learning technique. *Nuclear Fusion*, 59(6):066022, 2019.

[34] T Wakatsuki, T Suzuki, N Oyama, and N Hayashi. Ion temperature gradient control using reinforcement learning technique. *Nuclear Fusion*, 61(4):046036, 2021.

[35] Jaemin Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Feedforward beta control in the kstar tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

[36] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[37] Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. *Physics-based Deep Learning*. WWW, 2021. URL `https://physicsbaseddeeplearning.org`.

[38] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[39] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. Interpretability of machine learning-based prediction models in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10 (5):e1379, 2020.

[40] Radwa ElShawi, Youssef Sherif, Mouaz Al-Mallah, and Sherif Sakr. Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. *Computational Intelligence*, 37(4):1633–1650, 2021.

[41] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

[42] Lexiang Ye and Eamonn Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery*, 22(1):149–182, 2011.

[43] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162, 2011.

[44] Thanawin Rakthanmanon and Eamonn Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2013.

[45] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[46] Martin Wistuba, Josif Grabocka, and Lars Schmidt-Thieme. Ultra-fast shapelets for time series classification. *arXiv preprint arXiv:1503.05018*, 2015.

[47] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[48] Roberto Medico, Joeri Ruyssinck, Dirk Deschrijver, and Tom Dhaene. Learning multivariate shapelets with multi-layer neural networks. In *ACDL-Advanced Course on Data Science & Machine Learning*, 2018.

[49] Yichang Wang, Rémi Emonet, Elisa Fromont, Simon Malinowski, Etienne Menager, Loïc Mosser, and Romain Tavenard. Learning interpretable shapelets for time series classification through adversarial regularization. *arXiv preprint arXiv:1906.00917*, 2019.

[50] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[51] Daniel Jurafsky and James H. Martin. Speech and language processing, third edition draft. Accessed on August 16th 2022. URL https://web.stanford.edu/~jurafsky/slp3/.

[52] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[53] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[56] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pages 98–101. IEEE, 2020.

[57] Apeksha Nagesh Shewalkar. Comparison of rnn, lstm and gru on speech recognition data. 2018.

[58] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[59] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[60] Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289, 2009.

[61] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[62] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1): 1–42, 1997.

[65] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.

[66] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*, 2022.

[67] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021.

[68] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Ss-sam: Stochastic scheduled sharpness-aware minimization for efficiently training deep neural networks. *arXiv preprint arXiv:2203.09962*, 2022.

[69] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[70] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[72] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[73] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*, 2018.

[74] Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Ruizhe Ma, and Rafal Angryk. On the mining of the minimal set of time series data shapelets. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 493–502. IEEE, 2020.

[75] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[76] Jiaheng Wei, Hangyu Liu, Tongliang Liu, Gang Niu, and Yang Liu. Understanding generalized label smoothing when learning with noisy labels. *arXiv preprint arXiv:2106.04149*, 2021.

[77] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[78] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[79] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.

[80] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

[81] Francisco Matos, Vlado Menkovski, Alessandro Pau, Gino Marceca, Frank Jenko, TCV Team, et al. Plasma confinement mode classification using a sequence-to-sequence neural network with attention. *Nuclear Fusion*, 61(4):046019, 2021.

[82] F Matos, V Menkovski, F Felici, A Pau, F Jenko, TCV Team, EUROfusion MST1 Team, et al. Classification of tokamak plasma confinement states with convolutional recurrent neural networks. *Nuclear Fusion*, 60(3):036022, 2020.

[83] Alexander Huber, K McCormick, P Andrew, P Beaumont, S Dalley, J Fink, JC Fuchs, K Fullard, W Fundamenski, LC Ingesson, et al. Upgraded bolometer system on jet for improved radiation measurements. *Fusion Engineering and Design*, 82(5-14):1327–1334, 2007.

[84] LL Lao, H St John, RD Stambaugh, AG Kellman, and W Pfeiffer. Reconstruction of current profile parameters and plasma shapes in tokamaks. *Nuclear fusion*, 25 (11):1611, 1985.

[85] M Gelfusa, A Murari, I Lupelli, N Hawkes, P Gaudio, M Baruzzo, M Brix, T Craciunescu, V Drozdov, A Meigs, et al. Influence of plasma diagnostics and constraints on the quality of equilibrium reconstructions on joint european torus. *Review of Scientific Instruments*, 84(10):103508, 2013.

[86] A Pau, A Fanni, B Cannas, S Carcangiu, Gianluca Pisano, G Sias, P Sparapani, M Baruzzo, A Murari, F Rimini, et al. A first analysis of jet plasma profile-based indicators for disruption prediction and avoidance. *IEEE Transactions on Plasma Science*, 46(7):2691–2698, 2018.

[87] F Imbeaux, J Citrin, J Hobirk, GMD Hogeweij, F Köchl, VM Leonov, S Miyamoto, Y Nakamura, V Parail, G Pereverzev, et al. Current ramps in tokamaks: from present experiments to iter scenarios. *Nuclear Fusion*, 51(8):083026, 2011.

[88] PC De Vries, MF Johnson, B Alper, P Buratti, TC Hender, HR Koslowski, V Riccardo, JET-EFDA Contributors, et al. Survey of disruption causes at jet. *Nuclear fusion*, 51(5):053018, 2011.

[89] Charles M Greenfield, Raffi Nazikian, Mark S Foster, John Canik, Jerry Hughes, Wayne Solomon, Max E Fenstermacher, Oliver Schmitz, Larry Baylor, and Gary Jackson. Fusion energy sciences workshop on transients in tokamak plasmas: Report on scientific challenges and research opportunities in transient research june 8-11, 2015. 2015.

[90] M Sugihara, S Putvinski, D Campbell, S Carpentier-Chouchana, F Escourbiac, Y Gribov, T Hirai, K Ioki, H Labidi, S Maruyama, et al. Disruption impacts and their mitigation target values. 2012.

[91] PC De Vries, G Arnoux, A Huber, J Flanagan, M Lehnen, V Riccardo, C Reux, S Jachmich, C Lowry, G Calabro, et al. The impact of the iter-like wall at jet on disruptions. *Plasma Physics and Controlled Fusion*, 54(12):124032, 2012.

[92] PC De Vries, M Baruzzo, GMD Hogeweij, S Jachmich, E Joffrin, PJ Lomas, GF Matthews, A Murari, I Nunes, T Pütterich, et al. The influence of an iter-like wall on disruptions at jet. *Physics of Plasmas*, 21(5):056101, 2014.

[93] Hartmut Zohm. *Magnetohydrodynamic stability of tokamaks*. John Wiley & Sons, 2015.

[94] Valentin Igochine et al. *Active control of magneto-hydrodynamic instabilities in hot plasmas*. Springer, 2015.

[95] Barbara Cannas, PC De Vries, Alessandra Fanni, A Murari, Alessandro Pau, Giuliana Sias, and JET Contributors. Automatic disruption classification in jet with the iter-like wall. *Plasma Physics and Controlled Fusion*, 57(12):125003, 2015.

[96] Th Pütterich, R Neu, R Dux, AD Whiteford, MG O'Mullane, HP Summers, et al. Calculation and experimental test of the cooling factor of tungsten. *Nuclear Fusion*, 50(2):025012, 2010.

[97] DP Schissel, JC DeBoo, KH Burrell, JR Ferron, RJ Groebner, H St John, RD Stambaugh, BJD Tubbing, K Thomsen, JG Cordey, et al. H-mode energy confinement scaling from the diii-d and jet tokamaks. *Nuclear fusion*, 31(1):73, 1991.

[98] M Maslov, A Boboc, M Brix, JC Flanagan, E Peluso, C Price, M Romanelli, et al. Energy and particle confinement in jet h-mode plasma. *Nuclear Fusion*, 60(3): 036007, 2020.

[99] Leonid E Zakharov, Sergei A Galkin, Sergei N Gerasimov, and JET-EFDA contributors. Understanding disruptions in tokamaks. *Physics of Plasmas*, 19(5):055703, 2012.

[100] O Gruber, K Lackner, G Pautasso, U Seidel, and B Streibl. Vertical displacement events and halo currents. *Plasma physics and controlled fusion*, 35(SB):B191, 1993.

[101] N Pomphrey, JM Bialek, and W Park. Modelling of the toroidal asymmetry of poloidal halo currents in conducting structures. *Nuclear Fusion*, 38(3):449, 1998.

[102] M Sugihara, M Shimada, H Fujieda, Yu Gribov, K Ioki, Y Kawano, R Khayrutdinov, V Lukash, and J Ohmori. Disruption scenarios, their mitigation and operation window in iter. *Nuclear Fusion*, 47(4):337, 2007.

[103] Cédric Reux, V Plyusnin, B Alper, D Alves, B Bazylev, E Belonohy, A Boboc, S Brezinsek, I Coffey, J Decker, et al. Runaway electron beam generation and mitigation during disruptions at jet-ilw. *Nuclear Fusion*, 55(9):093013, 2015.

[104] TC Hender, JC Wesley, J Bialek, A Bondeson, AH Boozer, RJ Buttery, A Garofalo, TP Goodman, RS Granetz, Y Gribov, et al. Mhd stability, operational limits and disruptions. *Nuclear fusion*, 47(6):S128, 2007.

[105] Boris N Breizman, Pavel Aleynikov, Eric M Hollmann, and Michael Lehnen. Physics of runaway electrons in tokamaks. *Nuclear Fusion*, 59(8):083001, 2019.

[106] M Maraschek, A Gude, V Igochine, H Zohm, E Alessi, M Bernert, C Cianfarani, S Coda, B Duval, B Esposito, et al. Path-oriented early reaction to approaching disruptions in asdex upgrade and tcv in view of the future needs for iter and demo. *Plasma Physics and Controlled Fusion*, 60(1):014047, 2017.

[107] M Lennholm, T Budd, R Felton, M Gadeberg, A Goodyear, F Milani, and F Sartori. Plasma control at jet. *Fusion Engineering and Design*, 48(1-2):37–45, 2000.

[108] Chris M Bishop, Paul S Haynes, Mike EU Smith, Tom N Todd, and David L Trotman. Real-time control of a tokamak plasma using neural networks. *Neural Computation*, 7(1):206–217, 1995.

[109] GA Rattá, Jesús Vega, A Murari, G Vagliasindi, MF Johnson, PC De Vries, and JET EFDA Contributors. An advanced disruption predictor for jet tested in a simulated real-time environment. *Nuclear Fusion*, 50(2):025005, 2010.

[110] Jesús Vega, Sebastián Dormido-Canto, Juan M López, Andrea Murari, Jesús M Ramírez, Raúl Moreno, Mariano Ruiz, Diogo Alves, Robert Felton, JET-EFDA Contributors, et al. Results of the jet real-time disruption predictor in the iter-like wall campaigns. *Fusion Engineering and Design*, 88(6-8):1228–1231, 2013.

[111] Julian Kates-Harbeck, Alexey Svyatkovskiy, and William Tang. Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *Nature*, 568 (7753):526–531, 2019.

[112] JX Zhu, Cristina Rea, Kevin Montes, RS Granetz, Ryan Sweeney, and Roy Alexander Tinguely. Hybrid deep-learning architecture for general disruption prediction across multiple tokamaks. *Nuclear Fusion*, 61(2):026007, 2020.

[113] Cristina Rea, RS Granetz, K Montes, Roy Alexander Tinguely, N Eidietis, JM Hanson, and B Sammuli. Disruption prediction investigations using machine learning

tools on diii-d and alcator c-mod. *Plasma Physics and Controlled Fusion*, 60(8): 084004, 2018.

[114] GA Rattá, J Vega, and A Murari. Viability assessment of a cross-tokamak aug-jet disruption predictor. *Fusion Science and Technology*, 74(1-2):13–22, 2018.

[115] A Murari, M Lungaroni, M Gelfusa, E Peluso, J Vega, and JET Contributors. Adaptive learning for disruption prediction in non-stationary conditions. *Nuclear Fusion*, 59(8):086037, 2019.

[116] A Murari, R Rossi, E Peluso, M Lungaroni, P Gaudio, M Gelfusa, G Ratta, J Vega, JET Contributors, and ASDEX Upgrade Team. On the transfer of adaptive predictors between different devices for both mitigation and prevention of disruptions. *Nuclear Fusion*, 60(5):056003, 2020.

[117] Wei Zheng, Fengming Xue, Ming Zhang, Zhongyong Chen, Chengshuo Shen, Xinkun Ai, Nengchao Wang, Dalong Chen, Bihao Guo, Yonghua Ding, et al. Transferable cross-tokamak disruption prediction with deep hybrid neural network feature extractor. *arXiv preprint arXiv:2208.09594*, 2022.

[118] DL Chen, B Shen, RS Granetz, JP Qian, HD Zhuang, L Zeng, Y Duan, T Shi, H Wang, Y Sun, et al. Disruption mitigation with high-pressure helium gas injection on east tokamak. *Nuclear Fusion*, 58(3):036003, 2018.

[119] M Hoelzl, D Hu, E Nardon, GTA Huijsmans, JOREK Team, and ASDEX Upgrade Team. First predictive simulations for deuterium shattered pellet injection in asdex upgrade. *Physics of Plasmas*, 27(2):022510, 2020.

[120] TE Gebhart, LR Baylor, MN Ericson, SJ Meitner, AL Qualls, and DA Rasmussen. Recent progress in shattered pellet injection technology in support of the iter disruption mitigation system. *Nuclear Fusion*, 61(10):106007, 2021.

[121] M Lehnen, D Campbell, D Hu, U Kruezi, TC Luce, S Maruyama, JA Snipes, R Sweeney, NW Eidietis, A Matsuyama, et al. R&d for reliable disruption mitigation in iter. In *Preprint: 2018 IAEA Fusion Energy Conf.*, pages EX–P7, 2018.

[122] Valerie A Izzo, Istvan Pusztai, Konsta Särkimäki, Andréas Sundström, Darren T Garnier, David B Weisberg, Roy Alexander Tinguely, Carlos Paz-Soldan, Robert Granetz, and Ryan Sweeney. Runaway electron deconfinement in sparc and diii-d by a passive 3d coil. *Nuclear Fusion*, 2022.

[123] Qianli Ma, Wanqing Zhuang, Sen Li, Desen Huang, and Garrison Cottrell. Adversarial dynamic shapelet networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5069–5076, 2020.

[124] Zicheng Fang, Peng Wang, and Wei Wang. Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 497–508. IEEE, 2018.

[125] Aris Perperoglou, Willi Sauerbrei, Michal Abrahamowicz, and Matthias Schmid. A review of spline function procedures in r. *BMC medical research methodology*, 19 (1):1–16, 2019.

[126] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[127] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[128] Eric Heitz. A low-distortion map between triangle and square. 2019.

[129] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[130] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[131] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[132] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, pages 382–391. Springer, 2018.

[133] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018.

[134] Kai Li, Martin Renqiang Min, and Yun Fu. Rethinking zero-shot learning: A conditional visual classification perspective. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3583–3592, 2019.

[135] Raphael Pisoni. Sharpened cosine distance as an alternative for convolutions, 2022. `https://rpisoni.dev/posts/cossim-convolution/`.

[136] GA Rattá, J Vega, Andrea Murari, Missy Johnson, and Jet-Efda Contributors. Feature extraction for improved disruption prediction analysis at jet. *Review of scientific instruments*, 79(10):10F701, 2008.

[137] GA Rattá, J Vega, A Murari, JET-EFDA Contributors, et al. Improved feature selection based on genetic algorithms for real time disruption prediction on jet. *fusion Engineering and Design*, 87(9):1670–1678, 2012.

[138] GA Rattá, Jesús Vega, A Murari, S Dormido-Canto, R Moreno, and JET Contributors. Global optimization driven by genetic algorithms for disruption predictors based on apodis architecture. *Fusion engineering and design*, 112:1014–1018, 2016.

[139] A Murari, M Lungaroni, E Peluso, P Gaudio, J Vega, S Dormido-Canto, M Baruzzo, M Gelfusa, and JET Contributors. Adaptive predictors based on probabilistic svm for real time disruption mitigation on jet. *Nuclear Fusion*, 58(5):056002, 2018.

[140] A Pau, A Fanni, S Carcangiu, B Cannas, G Sias, A Murari, F Rimini, et al. A machine learning approach based on generative topographic mapping for disruption prevention and avoidance at jet. *Nuclear Fusion*, 59(10):106017, 2019.

[141] Cristina Rea and Robert S Granetz. Exploratory machine learning studies for disruption prediction using large databases on diii-d. *Fusion Science and Technology*, 74(1-2):89–100, 2018.

[142] Christina Rea, KJ Montes, KG Erickson, RS Granetz, and RA Tinguely. A real-time machine learning-based disruption predictor in diii-d. *Nuclear Fusion*, 59(9): 096016, 2019.

[143] Kevin Joseph Montes, Cristina Rea, RS Granetz, Roy Alexander Tinguely, N Eidietis, OM Meneghini, DL Chen, Biao Shen, BJ Xiao, Keith Erickson, et al. Machine learning for disruption warnings on alcator c-mod, diii-d, and east. *Nuclear Fusion*, 59(9):096015, 2019.

[144] G Pautasso, Ch Tichmann, S Egorov, T Zehetbauer, O Gruber, M Maraschek, K-F Mast, V Mertens, I Perchermeier, G Raupp, et al. On-line prediction and mitigation of disruptions in asdex upgrade. *Nuclear Fusion*, 42(1):100, 2002.

[145] Barbara Cannas, Alessandra Fanni, E Marongiu, and P Sonato. Disruption forecasting at jet using neural networks. *Nuclear fusion*, 44(1):68, 2003.

[146] Ge Dong, Kyle Gerard Felker, Alexey Svyatkovskiy, William Tang, and Julian Kates-Harbeck. Fully convolutional spatio-temporal models for representation learning in plasma science. *Journal of Machine Learning for Modeling and Computing*, 2(1), 2021.

[147] Barbara Cannas, Alessandra Fanni, A Murari, Alessandro Pau, Giuliana Sias, and JET EFDA Contributors. Automatic disruption classification based on manifold learning for real-time applications on jet. *Nuclear Fusion*, 53(9):093023, 2013.

[148] Andrea Murari, P Boutot, Jesús Vega, M Gelfusa, R Moreno, Geert Verdoolaege, PC De Vries, JET-EFDA Contributors, et al. Clustering based on the geodesic distance on gaussian manifolds for the automatic classification of disruptions. *Nuclear Fusion*, 53(3):033006, 2013.

[149] V Riccardo, A Loarte, et al. Timescale and magnitude of plasma thermal energy loss before and during disruptions in jet. *Nuclear fusion*, 45(11):1427, 2005.

[150] E Aymerich, A Fanni, G Sias, S Carcangiu, B Cannas, A Murari, A Pau, et al. A statistical approach for the automatic identification of the start of the chain of events leading to the disruptions at jet. *Nuclear Fusion*, 61(3):036013, 2021.

[151] AJH Donné, AE Costley, R Barnsley, Henrik Bindslev, R Boivin, G Conway, R Fisher, R Giannella, H Hartfuss, MG Von Hellermann, et al. Diagnostics. *Nuclear Fusion*, 47(6):S337, 2007.

[152] Martin Greenwald. Density limits in toroidal plasmas. *Plasma Physics and Controlled Fusion*, 44(8):R27, 2002.

[153] R Felton, K Blackler, S Dorling, A Goodyear, O Hemming, P Knight, M Lennholm, F Milani, F Sartori, and I Young. Real-time plasma control at jet using an atm network. In *1999 IEEE Conference on Real-Time Computer Applications in Nuclear Particle and Plasma Physics. 11th IEEE NPSS Real Time Conference. Conference Record (Cat. No. 99EX295)*, pages 175–181. IEEE, 1999.

[154] G Sias, B Cannas, A Fanni, A Murari, A Pau, EUROfusion MST1 Team, et al. A locked mode indicator for disruption prediction on jet and asdex upgrade. *Fusion Engineering and Design*, 138:254–266, 2019.

[155] J Vega, A Murari, S Dormido-Canto, F Hernández, T Cruz, D Gadariya, GA Rattá, and JET Contributors. A linear equation based on signal increments to predict disruptive behaviours and the time to disruption on jet. *Nuclear Fusion*, 60(2): 026001, 2019.

[156] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[157] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[158] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[159] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[160] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[161] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[162] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html`.

[163] CG Windsor, G Pautasso, C Tichmann, RJ Buttery, TC Hender, JET EFDA Contributors, et al. A cross-tokamak neural network disruption predictor for the jet and asdex upgrade tokamaks. *Nuclear fusion*, 45(5):337, 2005.

[164] J Vega, A Murari, S Dormido-Canto, R Moreno, A Pereira, A Acero, JET-EFDA Contributors, et al. Adaptive high learning rate probabilistic disruption predictors from scratch for the next generation of tokamaks. *Nuclear Fusion*, 54(12):123001, 2014.

[165] S Dormido-Canto, J Vega, JM Ramírez, A Murari, R Moreno, JM López, A Pereira, JET-EFDA Contributors, et al. Development of an efficient real-time disruption predictor from scratch on jet and implications for iter. *Nuclear Fusion*, 53(11): 113001, 2013.

[166] Fritz Wagner, G Becker, K Behringer, D Campbell, A Eberhagen, W Engelhardt, G Fussmann, O Gehre, J Gernhardt, G v Gierke, et al. Regime of improved confinement and high beta in neutral-beam-heated divertor discharges of the asdex tokamak. *Physical Review Letters*, 49(19):1408, 1982.

[167] E Righi, D Bartlett, G Conway, JG Cordey, LG Eriksson, C Gormezano, JCM de Haas, L Horton, J Jacquinot, C Lowry, et al. Global and local conditions for the lh and hl transitions on jet. *Plasma physics and controlled fusion*, 40(5):721, 1998.

[168] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[169] Gilles Vandewiele, Femke Ongenae, and Filip De Turck. Gendis: Genetic discovery of shapelets. *Sensors*, 21(4):1059, 2021.

Acknowledgements

First, I want to thank my supervisor Frank Jenko, for the supervision and guidance. He was always supportive of my work and gave me the freedom to explore my ideas. This work wouldn't have been possible without the collaboration with Peter de Vries and Alessandro Pau. Peter allowed me to use his dataset of disruptions from JET between 2000 and 2008, which required an immense amount of work to manually annotate. Data-driven methods rely on such efforts. The discussions were very useful in understanding the issues and importance of disruption prediction. And I want to thank Alessandro for the help with the JET-ILW and ASDEX-Upgrade data, and for sharing his knowledge on impurity accumulations, data processing, and analysis.

For proofreading different sections of the thesis, I want to thank Michael Bergmann, Maximilian Müller, Constantin Gahr, Magdalena Bauer, Hannes Bergström, Peter Halldestam, and Christina Winkler.

Overlapping with some of the previous names, I am grateful for the colleagues from our "boar's head" office: Alessandro Di Siena, Karen Pommois, Francisco Matos, Andres Cathey, Michael Bergmann, Constantin Gahr, Robin Greif, Maximilian Müller, Hannes, and Peter.

I want to thank everyone that made my last 4 years in Munich what they have been with a special thanks to all the climbers.

And lastly, I want to thank my parents and my brother for their support during my life and this thesis.