



Technische Universität München

Max-Planck-Institut für Plasmaphysik

Deep Learning for Tomography, State Classification and Event Detection in Nuclear Fusion Plasmas

Francisco Duarte Pinto de Almeida Matos

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Hans-Joachim Bungartz

Prüfer der Dissertation: 1. Hon.-Prof. Dr. Frank Jenko

2. Prof. Dr.-Ing. Nils Thuerey

3. Prof. Dr. Vlado Menkovski

Eindhoven University of Technology

Die Dissertation wurde am 17.02.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 30.06.2021 angenommen.



Technische Universität München

Department of Informatics

Deep Learning for Tomography, State Classification and Event Detection in Nuclear Fusion Plasmas

Francisco Duarte Pinto de Almeida Matos

Supervisor: Hon.-Prof. Dr. Frank Jenko

17.02.2021



Acknowledgments

The work described in this thesis would not have been possible without the support of numerous people. I'd like to thank all the co-authors of my research papers: Andrea Pavone, Jakob Svensson, Tomas Odstrcil, Alessandro Pau, Federico Felici, Gino Marceca, and in particular, Vlado Menkovski, both for the many hours of discussions and conversations, and for his interest in being a part of the examining committee for this dissertation. Naturally, none of this work could have happened without the supervision of Frank Jenko, to whom I am grateful for giving me the opportunity, resources and guidance necessary to carry out this research. I would like to thank Nathan de Oliveira, Robert Brzozowski, Andres Cathey and Karl Stimmel for their help with reviewing the thesis and their suggestions for improvements; Sebastian Huber and Colin Benz for their help with translating necessary materials into German; Hans Bungartz for agreeing to be the chair of this thesis's examining committee; and Nils Thuerey for agreeing to be a committee member. Finally, but most importantly, I thank all of my friends and family, both in Munich and elsewhere, who were a source of strength and resilience while I carried out my PhD.

Abstract

Nuclear fusion experiments regularly generate large amounts of data that can constitute a rich source of information for new scientific and engineering insights. Yet in many cases, the complexity of that data is such that it is difficult to develop matching theoretical frameworks that can deliver such insights.

Concurrently, advancements in the field of artificial intelligence in the past few years mean that deep learning algorithms now exist which can automatically learn from complex, high dimensional data. This dissertation describes several applications of such algorithms to data analysis problems in nuclear fusion. Specifically, two projects are described.

One project concerns the development of models for automated detection of plasma confinement states and ELMs at the TCV tokamak. For several TCV discharges, the time traces of 4 different signals are collected, and are then used to produce corresponding sequences of events, based on the features existing in the data. Three different approaches are proposed, which use the same signals, but differ from each other with regards to the assumptions they make about the correlations existing in the data, and about how to model the outputs. The first two approaches are discriminative neural networks that produce a single sequence of outputs, while the third is a generative sequence-to-sequence model.

The second project builds up on previous approaches for tomography in fusion plasmas. At the ASDEX Upgrade tokamak, soft X-ray signals can be used to obtain the plasma emissivity profile through Gaussian process tomography. A particular step of this algorithm – Bayesian model selection – can be computationally expensive. Thus, a neural network model is proposed to perform this step, bypassing the need for iterative optimization.

Zusammenfassung

Fusionsexperimente generieren große Mengen an Daten, die zu wissenschaftlicher und technischer Innovation beitragen können. In vielen Fällen erschwert die hohe Komplexität der Daten jedoch die erfolgreiche Auswertung der Ergebnisse auf theoretischer Ebene.

Durch Fortschritte in der Anwendung Künstlicher Intelligenz entstanden über die letzten Jahre Deep-Learning-Algorithmen, die aus komplexen mehrdimensionalen Datenquellen automatisiert lernen können. Diese Dissertation beschreibt die Anwendung solcher Algorithmen zur Lösung von Datenanalyseproblemen im Bereich der Kernfusion anhand von zwei Projekten.

Ein Projekt behandelt die Entwicklung von Modellen zur automatisierten Ermittlung von Plasmaeinschlusszustände und ELMs am TCV Tokamak in Lausanne. Dabei werden vier verschiedene Signale bei mehreren TCV-Entladungen im zeitlichen Verlauf verfolgt, um dann korrespondierende Sequenzen von Ereignissen zu produzieren, die auf den aufgezeichneten Daten basiert. Es werden drei unterschiedliche Modelle vorgestellt, die dieselben Signale untersuchen, sich jedoch in den Annahmen über die Korrelationen im Datensatz und der Modellierung der ausgegebenen Daten unterscheiden. Die ersten beiden Modelle beruhen auf diskriminative neuronale Netzwerke, die eine einzige Ausgangssequenz produzieren, während das dritte Modell ein generatives sequence-to-sequence Netzwerk einsetzt.

Das zweite Projekt baut auf früheren Forschungsansätze aus der Plasmatomografie auf. Am ASDEX Upgrade Tokamak können weiche Röntgensignale dazu benutzt werden, das Plasmaemissionsprofil durch Gaußsche Prozesstomografie herzustellen. Insbesondere durch den Schritt der Bayesschen Modellselektion kann dieser Algorithmus eine hohe Rechenleistung beanspruchen. Daher wird vorgeschlagen, für diesen Schritt ein neuronales Netzwerkmodell zu benutzen, um die Notwendigkeit iterativer Optimierung zu umgehen.

Contents

I. Introduction and Theory	1
1. Introduction	3
2. Background	9
2.1. Fusion Experiments	10
2.2. Soft X-ray tomography at ASDEX Upgrade	12
2.2.1. Plasma tomography	13
2.2.2. The SXR diagnostic at ASDEX Upgrade	14
2.3. Plasma confinement at TCV	17
2.3.1. High confinement and Edge Localized Modes	17
2.3.2. TCV Diagnostics	19
3. Methodology	21
3.1. Background concepts	22
3.1.1. Types of learning	22
3.1.2. Classification and regression	23
3.2. Deep Learning	23
3.2.1. Neural Networks	24
3.2.2. Training	26
3.2.3. Overfitting and regularization	29
3.2.4. Convolutional neural networks	30
3.2.5. Recurrent neural networks	34
3.2.6. Sequence-to-sequence models	37
3.2.7. Attention mechanism	42
3.3. Gaussian process regression	43
3.4. Concluding remarks	46

II. Publications	49
4. Summary	51
5. Deep learning for Gaussian process soft X-ray tomography model selection in the ASDEX Upgrade tokamak	55
5.1. Introduction	56
5.2. Background	58
5.2.1. Computed Tomography	58
5.2.2. SXR tomography at ASDEX Upgrade	60
5.2.3. Regularization-based Methods	61
5.2.4. Deep Learning-based Methods	62
5.2.5. Gaussian Process Tomography	63
5.3. Methods	68
5.3.1. Soft X-Ray Data	68
5.3.2. Dataset Generation	69
5.3.3. Deep Learning Model	72
5.4. Results	74
5.4.1. Neural Network	74
5.4.2. Sample Reconstructions	76
5.4.3. Model complexity and data fit	77
5.4.4. Discussion	80
5.5. Conclusions	81
6. Classification of tokamak plasma confinement states with convolutional recurrent neural networks	85
6.1. Introduction	86
6.2. Previous work	88
6.3. Background	90
6.3.1. Low, dither and high plasma confinement modes	90
6.3.2. Edge Localized Modes	90
6.4. Methods	91
6.4.1. Problem formulation and approach	91
6.4.2. Data and event features	94
6.4.3. Model training	98

6.4.4. Model design	101
6.4.5. Data split	103
6.5. Evaluation metrics	105
6.5.1. ROC curve	105
6.5.2. Kappa statistic	106
6.6. Results	107
6.6.1. CNN	108
6.6.2. Conv-LSTM	109
6.6.3. Discussion	111
6.7. Conclusions	113
7. Plasma Confinement Mode Classification Using a Sequence-to-Sequence Neural Network With Attention	117
7.1. Introduction	118
7.2. Background	121
7.2.1. Sequence-to-sequence models	121
7.2.2. Attention	123
7.3. Methods	124
7.3.1. Problem Formulation	124
7.3.2. Data engineering	126
7.3.3. Model architecture	127
7.3.4. Dataset Preparation	132
7.4. Results	134
7.5. Discussion	136
7.6. Conclusions	139
III. Conclusions and outlook	143
8. Conclusions	145
List of Figures	149
List of Tables	153

Contents

Bibliography	155
IV. Appendices	171
A. Original Publications	173

Part I:
Introduction and Theory

1. Introduction

Over the past decades, research has been carried out on developing nuclear fusion as a viable source of energy for the world. Contrary to other, currently existing sources of energy (such as fossil fuels and nuclear fission), fusion promises to be both cleaner and safer, yet also virtually inexhaustible. Generically, a nuclear fusion reaction consists in the fusing of light atomic nuclei to produce a heavier nucleus, with subatomic particles as by-products. In most fusion reactions, there is also a release of energy, and it is this that could potentially be harvested for humanity's needs.

Fusion is the natural process through which stars produce energy. An example of this is the proton-proton (p-p) cycle, which is the sequence of nuclear reactions that predominates in stars like the sun. The cycle begins with the fusion of protons (hydrogen nuclei), and ends with the release of energetic subatomic particles and helium nuclei (so-called α -particles), with several intermediate steps involving different types of fusion reactions.

For two atomic nuclei to fuse, they must collide, so that the strong nuclear force can bind their protons and neutrons together. This in turn requires that the nuclei be sufficiently energetic to overcome their mutual Coulomb repulsion (a product of the electric charges of their protons). The kinetic energies required for two nuclei to fuse are higher than the binding energy of their electrons, and therefore, a fusion fuel is ionized, i.e. the nuclei and electrons of the fuel's constituent atoms separate[1]. Under these conditions, the fuel is said to be a plasma. Generally, fusion can happen across a wide range of temperatures, with the probability of a successful fusion reaction (the so-called fusion cross-section) varying as a function of the velocity of the involved nuclei.

The Lawson criterion[2] establishes the conditions under which a fusion fuel is ignited, that is, releasing energy to its surrounding environment while also keeping enough energy to sustain the fusion processes within itself. The crite-

tion states that, for a fuel to be ignited, the product of a plasma's density, its temperature, and the rate at which it releases energy into its environment (the so-called confinement time) should be above a certain threshold.

From a research point of view, the question is then how to best engineer fusion processes in a controlled setting or reactor on Earth, with a view to energy extraction. Such an endeavor would have several differences with regards to stellar fusion. Firstly, the density conditions existing in stellar cores, which are a product of the enormous gravitational pressure in those environments, cannot be replicated. As a result, achieving the Lawson criteria to extract energy from a plasma in a reactor would require extremely high temperatures, higher than those in stars — on the order of 100 million degrees Kelvin[3]. In addition, a stellar core is a rather inefficient energy producer — the fusion reactions happening there typically have low cross-sections, i.e. nuclear collisions are, relatively speaking, rare. This is, however, offset by the size of celestial bodies, which means that even at low values of energy production per volume, the total power output of a star is enormous. A working reactor would require a much greater efficiency, i.e., it would need to produce much more energy per volume.

For these reasons, the most promising type of fusion reaction for terrestrial energy production is that of deuterium-tritium (D-T). This reaction has a relatively high cross-section and releases significant amounts of energy at a (compared with other reactions) low temperature, while generating only energetic neutrons and helium as waste products. Furthermore, deuterium and tritium are both isotopes of hydrogen, an abundant element; deuterium is readily available on Earth, while tritium can be produced from lithium.

Several proposals exist for how to engineer fusion processes for energy extraction on Earth. One of them, that of magnetic confinement fusion (MCF), proposes to use strong external magnetic fields to confine plasma inside a toroidal chamber. Two main concepts for such chambers are currently being investigated (the tokamak and the stellarator) which differ mainly with regards to how the magnetic field is generated [4]. Another approach, that of inertial confinement fusion, instead proposes to use small pellets of D-T fuel, and then heat them, for example, with lasers, to the point where the nuclei are sufficiently energetic to fuse. In this thesis, the problems described are relative to tokamaks.

In order to produce energy with a MCF reactor, the total amount of power

injected into the vessel to heat the plasma must be smaller than the output power produced by the fusion reactions. In an ideal scenario for tokamak operation, an initial burst of energy, provided, for example, by microwaves, neutral beams or electric current, would heat up the fuel, turning it into a plasma and, through the use of powerful external coils, a magnetic field would be produced to keep the plasma in place. The energy produced by the fusion reactions would then be harvested for human needs, but also keep the plasma ignited, allowing for further reactions while waste products would be regularly extracted.

Unfortunately, when subject to the temperatures required for fusion, plasmas are highly susceptible to instabilities, which can lead to fast bursts of particles and energy that come into contact with the containment vessel. This means that, in a typical tokamak fusion experiment, when the plasma becomes too unstable, the experiment must be shut down to avoid damaging the reactor's walls. Up to the present, this has prevented the useful extraction of energy from fusion plasmas.

From a tokamak engineering point of view, an important goal is therefore to develop methods for keeping the plasma under control to allow for stable, long-pulse operation of the reactor, thus preserving a positive net balance of input versus output energy. This requires understanding the underlying physical phenomena that occur during a discharge. One example of such a phenomenon is the High confinement mode (H-mode): most modern tokamaks regularly execute discharges where the plasma can, at some point, transition into a state where the amount of particles and energy being transported to the outside of the plasma edge is dramatically reduced. Yet no model exists that can completely account for why this shift in the plasma's behavior happens.

On the other hand, a typical tokamak contains dozens of different diagnostics and measuring systems which provide information, both during and after an experiment, about the plasma's behaviour. These systems regularly produce enormous amounts of data, yet much of it is not used because of its sheer volume and the difficulty inherent in thoroughly analysing it. Nevertheless, this data frequently contains important information regarding phenomena happening in the plasma, with many events leaving clearly identifiable signatures in signal time traces. Thus, it would be useful to have tools allowing for automated analysis of this data, to extract new insights and information and facilitate sci-

entific discovery in nuclear fusion.

In the past decade, developments in the field of machine learning and artificial intelligence have led to a resurgence of interest in the area, after a long so-called "AI winter" lasting for several decades during which the field mostly stagnated. This change is mostly due to the emergence of new methods for training deep learning algorithms such as convolutional and recurrent neural networks. Though the principles behind these algorithms have been known and studied for a long time, practical applications were always hindered by their computational cost; it has only been recently, thanks to advancements in hardware, particularly GPU[5] and parallel computing, as well as the appearance of massive labeled datasets, that new architectures appeared which allow these algorithms to realize their full potential. Deep learning algorithms can efficiently find complex patterns and correlations in data, and are now in everyday usage in many fields, with self-driving cars, medical image analysis, and automated translation to name a few. Nevertheless, their usage for data analysis within the field of fusion research remains rather limited, though interest has picked up in the past few years.

The goal of this thesis is therefore to try to use some of the advancements of the past few years in the field of deep learning, and apply them to current problems in fusion which could be aided by the usage of data-centric approaches. In particular, this thesis focuses on the problem of Soft X-ray tomography in the ASDEX Upgrade tokamak, and on the problem of automated detection of plasma confinement states and Edge Localized Modes in the TCV tokamak, and makes a contribution to those problems through the usage of algorithms like convolutional and recurrent neural networks, not only as stand-alone architectures, but also as part of larger models like a sequence-to-sequence network or a gaussian process framework for regression. The thesis is structured as follows: Chapter 2 gives background information regarding TCV and ASDEX Upgrade, with a particular emphasis on the datasets collected from the two tokamaks which are used by the proposed models, as well as a description of the problems being modelled. Chapter 3 gives an overview of several concepts within the field of machine learning which are required to model the problems being analysed, and discusses convolutional neural networks, recurrent neural networks, as well as frameworks such as sequence-to-sequence models and gaussian process regres-

sion. The work done is then reported as three peer-reviewed journal articles, which are reproduced in this document. Chapters 5 and 6 refer to articles which had already been published at the time of the submission of this thesis; chapter 7 refers to an article which had been formally accepted for publication at the time of submission, and which was in the meantime published in its final form. Finally, chapter 8 discusses the conclusions of this work and its potential implications. The published forms of the three papers (with journal layout) can be found in the Appendices.

2. Background

A tokamak is a torus-shaped machine, inside of which fuel is heated up to temperatures that ionize it, turning it into a plasma throughout most of the vessel volume. The fuel itself is a gas mixture of hydrogen isotopes or helium, depending on the desired fusion reactions. Because a plasma is composed of electrically charged particles (the positively-charged nuclei and electrons), it can be confined through the application of a sufficiently strong magnetic field. In a tokamak, this field has two components: a toroidal component, imposed by external coils, and a poloidal component, produced by the inductively driven plasma electric current[3]. The resulting magnetic field lines, along which charged particles describe their trajectories, wind helically around the torus.

In a single deuterium-tritium reaction (which is planned to be the type of fusion reaction happening in future industrial reactors) a helium nucleus (α -particle with an energy of 3.54 MeV) and a neutron are produced. This neutron carries most of the energy (14.05 MeV) released in the reaction; because it has no charge, it is not affected by the tokamak's magnetic field, escapes the plasma, and is absorbed by, and heats, the vessel wall. In a future commercial reactor, that heat would then be extracted with a coolant, and used for electricity production. The α -particle contains the remaining reaction energy, but it remains in the plasma, because it has a positive charge. Eventually, its energy is dissipated into the surrounding particles through collisions (so called α -heating). In a scenario of continuous tokamak operation, it would be the energy from α -particles that would provide most of the necessary heating to preserve the plasma temperature, with the remaining energy being provided by an additional, external power source.

The wall surrounding the plasma chamber has a blanket made of a lithium compound. When the incoming neutrons from the plasma are absorbed, they react with the lithium to produce more tritium for the fusion reactions. The wall's

plasma-facing components are usually made from an element with a high melting point, to withstand the energies of the incoming particles. In addition, in modern tokamaks, the magnetic field lines are designed to carry waste products of the fusion reactions to a component called the divertor, which acts as a sort of tokamak exhaust. The divertor avoids the build-up of too many impurities, whether α -particles or stripped ions produced by plasma-wall interactions, and is more efficient than earlier schemes for particle removal such as limiters[6]. Figure 2.1 provides an illustration of a tokamak and its cross-section, as well as many of the terms described up until this point.

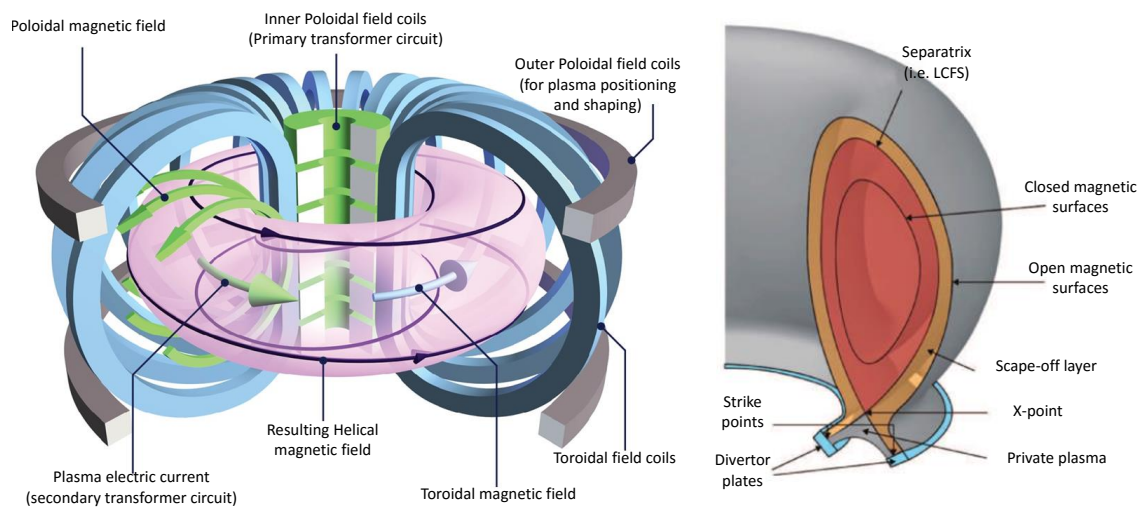


Figure 2.1.: Schematic drawing of a tokamak (left) and its 2D cross section (right). Adapted from EuroFusion figures database and reprinted from[7].

2.1. Fusion Experiments

A plasma discharge in a tokamak is commonly called a shot. During a shot, the plasma can be in several different states or modes. The two most studied are the Low (L) and High (H) confinement modes, though other ones, such as for example the dithering (D) or intermediate mode, have also been described[8]. When a shot begins, the plasma current is slowly increased up until a desired value, in a phase known as the ramp-up. Once that target is achieved, the plasma is consid-

ered to be in L-mode. However, given sufficient heating power, a phenomenon occurs whereby a heat and particle transport barrier develops in the plasma edge (the region close to the separatrix, shown on the right in Fig. 2.1). The sudden appearance of this barrier corresponds to a switch in the plasma state from L to H-mode; the barrier significantly reduces the transport of heat and particles from the core (the central plasma region) to the outside of the edge. From a fusion reactor design and operational perspective, this situation is ideal: once the plasma is in H-mode, further increases in input power yield comparatively better confinement, thus making the fusion process more efficient. The H-mode was discovered at the ASDEX tokamak[9], but it has since been observed in almost all tokamaks, and current plans envisage that in the future, tokamak plasmas will regularly run in that state.

Running a tokamak continuously, however, is challenging. Instabilities can develop in both L and H-mode plasmas that may provoke a total loss of power and confinement, which forces the termination of a discharge. For example, during H-mode, due to the reduced transport, there can be periodic bursts of particles and energy — edge localized modes, or ELMs — that threaten the plasma’s stability. Therefore, any future continuously running tokamak must have the tools necessary to accurately, and quickly, detect changes in the plasma, so as to respond in a way that keeps it confined.

Concurrently, during a plasma discharge, contact between the plasma and the surrounding vessel walls must be minimized. Due to the very high plasma temperatures, when such contact happens, traces of the wall materials can become diluted in the plasma, where they turn into impurities. These impurities can cool down the plasma and limit the efficiency of the fusion reactions[10] once their concentration level is above a certain threshold. They are not, however, without their advantages: for example, in a tokamak, impurities can also help reduce the power load on the divertor[11], by radiating away energy.

One way to obtain information about how impurities are distributed inside the plasma is by studying their radiation signatures, in particular when they are constituted mostly by elements with a high atomic number. In this case, the radiation emitted by the impurities will be clearly distinguishable from the background radiation emitted by the plasma. This requires obtaining the overall emissivity distribution in a poloidal cross-section of the tokamak, which can be

done with tomographic algorithms. However, in the fusion setting, tomography is a difficult problem because it is significantly under-determined: typically, due to the restrictive environment of the tokamak, only a handful of line-integrated measurements of the emitted plasma radiation are available. This means that typical plasma tomography algorithms require some sort of regularization or prior information. On top of that, these algorithms are often too slow to be used for purposes of real-time analysis and control.

The issues described so far apply in general to all tokamaks. This thesis describes work done for two of those machines: the Axially Symmetric Diverter Experiment (ASDEX) Upgrade, and the Tokamak à Configuration Variable (TCV). Both machines are mid-sized tokamaks equipped with many diagnostic systems designed to obtain information about physical phenomena occurring in the plasma during a discharge, with different diagnostics yielding information about different events and quantities. The remainder of this chapter describes the connection between plasma diagnostics and two problems: that of Soft X-ray (SXR) tomography on ASDEX Upgrade, and that of detection of plasma confinement modes on TCV. Those are the topics that were the focus of the research presented in this work, and they are elaborated upon in chapters 5, 6 and 7.

2.2. Soft X-ray tomography at ASDEX Upgrade

Tomography generically refers to imaging the interior of a body without having to section it. One way to achieve this is to focus some type of penetrating radiation on the body of interest, and measuring the attenuated radiation values on its opposite side — i.e., the object's *projection*. The mathematical formulation behind tomography has existed since at least 1917[12], but most applications of tomography only came about in the 1950s and 60s with the advent of modern computing[13].

When performing tomographic reconstruction of 2-dimensional object cross-sections with parallel rays, the mathematical framework used by traditional tomographic algorithms is the Fourier slice theorem[14]. It states that the 1-dimensional Fourier transform of a projection is a slice of the inverse 2-dimensional transform of the full tomographic reconstruction. Thus, a recon-

struction can be obtained by applying this Fourier transform in succession on many projections. In fact, it has been shown that, given an infinite number of projections, it is possible to fully reconstruct the internal properties of any given body. In practice, it is impossible to get an infinite number of projections, but medical applications, for example, can rely on as many as 10^5 of them[15], taken along different directions.

2.2.1. Plasma tomography

In fusion devices, as in other scientific domains, one is frequently interested in the distribution of certain physical quantities inside the reactor, without having direct access to them. For example, one might be interested in knowing the distribution of the electron density or the magnetic flux surfaces[16]. At ASDEX upgrade, one motivating factor for the usage of tomographic algorithms is to study the distribution of impurities in the plasma, namely tungsten, brought on by interactions between the plasma and the vessel wall[17]. Much like the plasma itself, these impurities radiate; however, they do so in different regions of the electromagnetic spectrum — in particular, in the soft x-ray (SXR) region. Therefore, obtaining the cross-sectional emissivity distribution in this spectral region can yield information about the distribution of certain chemical elements in the reactor. Conceptually, this problem is still tomography, because one wishes to obtain the internal properties of a body (the emissivity distribution inside the vessel) without direct access to it.

Nevertheless, there are some differences between plasma tomography, and that which is done in other applications. For starters, in fusion experiments, the tomographic projections are not obtained by focusing the vessel with radiation. Rather, the radiation emitted by the phenomena occurring inside the vessel is, itself, considered the tomographic projection. This can be measured, for example, with a bolometer system such as used in the JET[18] tokamak, or through arrays of diodes in ASDEX upgrade[19]. However, unlike the case in other applications, the number of measurements that can be made of the emitted plasma radiation is rather limited: for example, in devices such as JET, the bolometer diagnostic only has about 50 measurements (see Figure 2.2), while one is usually interested in obtaining cross-sectional distributions with a much larger dimensionality (i.e.

number of pixels). For that reason, the tomography problem in fusion plasmas is under-determined, meaning that a potentially infinite number of solutions for the cross-sectional emissivity exist that can correctly match the obtained measurements. Therefore, traditional plasma tomography algorithms require some sort of regularizing factor that constrains the solution to a physically realistic value.

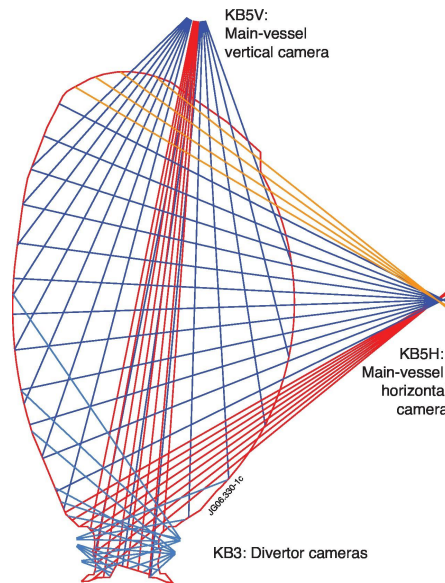


Figure 2.2.: Bolometer system installed at JET (Source: EUROfusion Figures Database)

2.2.2. The SXR diagnostic at ASDEX Upgrade

The ASDEX Upgrade tokamak has been operating since 1991. It is a medium-sized tokamak, with a major radius of 1.65m and a minor radius of 0.5m and a tungsten wall[20]. Though the machine is equipped with many different diagnostics, for the purposes of this thesis we focus on one in particular: the soft X-ray (SXR) system.

The SXR system at ASDEX-upgrade consists of 8 cameras, each with a diode array, that measure the radiation emitted by the plasma in the spectral interval ranging from 2.3keV to 13keV. The cameras are all laid out in the same poloidal plane of the tokamak, and generate a total of 208 volumes of sight through which

the plasma can be observed. In practice, the toroidal extent of the volumes of sight is minor, and it is sufficient to consider lines of sight (LOS) instead. Each LOS corresponds to a single sensor. The total radiation measured by a sensor, at any given point in time, corresponds to the line-integrated values of SXR radiation along its corresponding LOS.

The data from this diagnostic can be used for tomographic reconstruction of the plasma emissivity profile. Previous work on tomography at ASDEX Upgrade based on soft X-ray measurements was done with regularization-based methods. In particular, the work in [21] uses a minimum Fisher-based operator[22] for the regularization, which essentially finds solutions that tend to have strong gradients in areas of low intensity, but allows for large gradients (i.e. peaked plasma profiles) in areas where the intensity is highest. Because the minimum Fisher regularization is a function of the solution itself, it must be computed iteratively, usually until a minimum of a cross-validation is reached. This approach can produce excellent results for the reconstruction of the plasma emissivity profile, but is too slow for real-time purposes. On the other hand, work has been done on tomographic reconstruction for other machines using alternative frameworks such as deep neural networks[23, 24] and gaussian process regression[25]. Yet these methods have their own problems.

Deep neural networks can produce tomographic reconstructions at high speed (enough for real-time analysis purposes), but discriminative neural networks such as the ones used for tomography are unable to quantify the uncertainty in their reconstructions. Neural networks have been shown to be able to generalize even in very high-dimensional settings, but ultimately there are no guarantees about the correctness of the obtained reconstructions[26]. This is a problem inherent to this type of model, especially in a regression task. A well-chosen train and validation set allow one to diagnose a neural network's behavior, but it would be more interesting to have direct uncertainty estimates on solutions.

On the other hand, Gaussian process regression allows for obtaining uncertainties on the obtained tomographic profiles, and given some fixed model, can obtain reconstructions in real-time, while offering guarantees of always finding the best possible fit for the data given the model. This, however, opens up the door to over- (or under-) fitting, since the chosen model might either be too tight or too flexible for the given data. This particular problem can be solved through

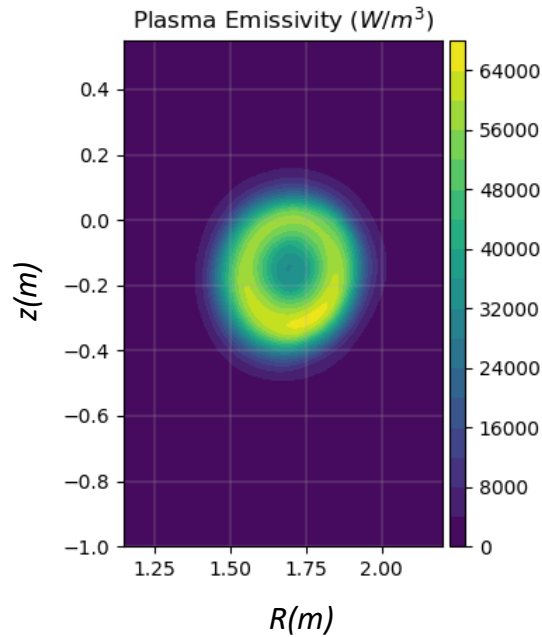


Figure 2.3.: Existing tomographic reconstruction at ASDEX Upgrade, from shot #31238 at $t = 5.2841$ s.

Bayesian formalism: different models can be directly compared with each other in terms of their evidence, and the best one will be guaranteed to have the optimal trade-off in terms of model simplicity and complexity. However, this last step can require computationally-intense sampling techniques; even in the cases where it is analytically solvable, it involves a series of matrix multiplications which, in high-dimensional data settings (as is the case in plasma tomography) significantly increase computational time, eliminating the possibility of obtaining tomographic reconstructions in real-time.

Thus, it would be interesting to develop a framework where the advantages of Gaussian process regression can be used together with the advantages of deep learning for tomographic reconstruction, while eliminating their respective disadvantages. Chapter 5 delves into further detail on the work done in this regard, as well as giving a more thorough description of the signal (SXR) data used.

2.3. Plasma confinement at TCV

Shortly after a tokamak discharge begins, the plasma is considered to be in Low (L) confinement mode. In this regime, the plasma's poloidal density profile is rather flat, because there are significant losses of particles and energy into the scrape-off layer[27]. However, at the TCV tokamak, as in others, given sufficient heating power, a phase shift can occur in the plasma: the appearance of the so-called edge transport barrier (ETB). This corresponds to a spontaneous, self-organizing phenomenon of the plasma itself, and is so called because this barrier significantly reduces the transport of particles and energy to outside of the plasma. The emergence of the ETB corresponds with the appearance of a large gradient in the plasma emissivity profile in the edge region. As time progresses, this gradient eventually spreads to the inner regions of the plasma. At that point, for the same input power, the plasma can store more energy and particles; it is for this reason that this new state is called high (H) confinement mode. Compared to L-mode, a plasma in H-mode can have up to double the confinement time[28].

In addition, it is also possible to observe an additional, intermediary state — the so called Dithering (D) mode. During a discharge, the ETB which characterizes the H-mode can collapse and reappear many times, with the plasma correspondingly switching between Low and High confinement. The plasma is considered to be dithering when the appearance and collapse of this barrier occur in quick succession, with a very high frequency, because it no longer exhibits characteristics of either L or H-mode.

2.3.1. High confinement and Edge Localized Modes

It is known[8] that the development of a tokamak plasma from Low (L) to High (H) confinement mode can be provoked by the application of a sufficient amount of heating power on the ionized gas (the so-called power threshold). However, many factors, such as for example the size of the tokamak and the plasma density[29], can influence the value of this threshold. The number of those factors is large enough, and the interactions between them complex enough, that to this day, no thorough explanation exists for why a plasma transitions from L to

H-mode. This can be particularly problematic for the operation of future machines. Current assumptions are that they will regularly run in H-mode, due to its enhanced confinement of energy and particles. However, due to the aforementioned problems, it is difficult to estimate, *a priori*, what the H-mode power thresholds for those machines will be. In addition, there are various explanations regarding the underlying microscopic phenomena responsible for the appearance of the ETB. One commonly accepted explanation[30] is that the transition into H-mode is caused by flows of particles that dampen the otherwise existing turbulence, thus increasing the overall stability and confinement of the plasma. Nevertheless, the exact mechanisms vary from experiment to experiment and from machine to machine, and no overarching theory exists. Attempts have been made to derive power scaling laws based on past experiments across multiple machines that might indicate what power thresholds might be necessary to trigger the H-mode in future experiments. Unfortunately, such power scaling laws have a high degree of uncertainty[31].

Due to the enhanced confinement of the H-mode and the associated increase in the plasma density in the edge region, periodic expulsions of particles and energy can occur, which are the so-called edge localized modes (ELMs). In this case, the H-mode is said to be ELMy, though ELM-free H-modes also exist[32]. ELMs can be useful because they allow for the periodic removal of impurities from the plasma, but they can also provoke very high power loads, particularly on certain tokamak components such as the divertor. Broadly, ELMs can be placed into three categories. Type I ELMs correspond to bursts that result from the plasma being close to its stability limit, and their frequency increases with input power. Type II ELMs have a lower intensity, and their frequency does not increase as a result of larger heating power. Type III ELMs correspond to the plasma being significantly below its stability limit, and their frequency decreases as heating power increases.

When the input power is close to the L-H power threshold, a clear transition from L to H-mode does not always happen immediately. In this case, a series of oscillating fluctuations in the plasma edge density occur. These cycles can be easily confused with the ELMs that are characteristic of the H-mode, particularly type III ELMs, due to the signatures they imprint on data, and also due to the fact that they occur for roughly the same input power values. Indeed, some

literature refers to this state as a "dithering H-mode"[8]. However, unlike the H-mode, when the plasma is dithering, its confinement compared to L-mode only improves marginally[33], a fact which justifies Dither's classification as a separate confinement mode.

2.3.2. TCV Diagnostics

It would be interesting to have algorithms and methods that allow for automated detection or prediction of ELMs, and for real-time monitoring of the plasma to detect changes between different confinement modes. Such algorithms could then be used by feed-back systems to steer the plasma into an operational space that maximizes confinement, for example through periodic, controlled expulsions of impurities and excess energy[34].

Changes between different plasma confinement modes, as well as ELMs, leave signatures in diagnostic time traces. These signatures are usually mostly visible in the emitted plasma radiation[33]. When the plasma switches from L to H-mode, the emitted radiation drops noticeably, though the actual slope of the descent (i.e. the speed with which the transition happens) can vary depending on the discharge. Similarly, ELMs can usually be seen as a series of spikes in the emitted radiation. Dithers can be observed as a series of oscillations, very similar to the highly frequent spikes of type III ELMs.

Like ASDEX Upgrade, TCV is a medium-sized tokamak[35], featuring a major radius of 0.88m and a minor radius of 0.25m. Its relatively high height-to-width ratio makes it an ideal machine for experimenting with varying plasma shapes; this might be important for future tokamak operation as some shapes might be better suited for efficient machine operation. At TCV, experts routinely use 4 different diagnostics when determining the plasma's confinement state and to pinpoint ELMs, on a post-experimental basis. These are the plasma's electric current, its electron density, its magnetic field, and the emitted radiation in a certain region of the spectrum. There has been previous work with using data-based approaches, using these same signals, for detection of switches between confinement states, and of ELMs. Many of those approaches, however — such as threshold based detectors[36, 37, 38], support vector machines[39, 40, 41, 42], and Kalman filters[43] — have several limitations. This is especially true when

comparing them to advanced deep learning methods like recurrent and convolutional neural networks, which excel at processing very high-dimensional data such as the timeseries involved in this task, and even more so when compared to generative deep learning models such as sequence-to-sequence networks. This motivates the use of those approaches for this task. That work, as well as a thorough description of the data which was here briefly discussed, can be found in Chapters [6](#) and [7](#).

3. Methodology

Many problems in engineering deal with the automation of systems. Many systems, while being complex, can nevertheless be handily described with rule-based approaches that constitute satisfactory solutions; for example, a robot working in a factory can be programmed with a set of instructions to do repetitive tasks.

However, many problems have a level of complexity which makes it difficult, if not impossible, to use this sort of approach. A typical example used in many tutorials in machine (and in particular, deep) learning is the task of designing an algorithm capable of distinguishing between different types of images — for example, distinguishing between two different types of animals in a picture (such as a dog and a cat). While this task is trivial for a human, it is not obvious how to design a rule-based, automated, system to carry it out. Such a system would require an extensive compilation of the inherent features of each class of animal, and furthermore, many of those features overlap with each other in non-trivial ways. For example, one could make the rule that a dog has features like eyes, ears and a nose — but so does a cat. One could then try to specify more fine-grained characteristics of those features for each type of animal, but then, how would one explicitly enunciate the differences between the eyes of a dog, and those of a cat? In this and many other problems, the amount of rules and exceptions is so large that one cannot have any hope of enunciating them thoroughly.

Fortunately, modern machine learning algorithms can solve this problem by attempting to find statistical patterns in data and discover any underlying features and rules by themselves. The specifics of each machine learning algorithm may vary, but the basic common theme is that these algorithms learn rules automatically through experience, in a process known as *training*. Training involves looking at many data samples, with the expectation that any existent statistical patterns in the data can be picked up by the algorithm, and later, successfully

used to accurately make *predictions* about new, unseen data.

This chapter begins with an overview of several topics transversal to data analysis and machine learning. This is followed by an overview of the particular type of machine learning algorithm — deep neural networks — which were used extensively for the work detailed in this thesis. Finally, the chapters wraps up with an overview of Gaussian process regression, an algorithm which, while not qualifying as machine learning *per se*, can also be employed in data analysis tasks, and was also used for work done in the context of this dissertation.

3.1. Background concepts

3.1.1. Types of learning

Broadly, within the field of machine learning, three different branches can be said to exist: *supervised* learning, *unsupervised* learning, and *reinforcement* learning[44].

Supervised learning refers to the process of training an algorithm on *labeled* data. An example of a labeled data point would be an image, and a corresponding encoded value (for example, a probability between 0 and 1) indicating what is present in the image: a dog or a cat. In a typical supervised learning task, an algorithm would look at many data samples of such images and learn the existing patterns that allow for an accurate prediction of the animal therein. Examples of supervised learning algorithms include support vector machines, regression trees, and many neural network architectures such as convolutional neural networks.

Unsupervised learning, on the other hand, requires no labeled samples. Typically, in this setting, one is interested in discovering existing patterns or features in the data. This can be done, for example, through a process of dimensionality reduction that attempts to force an algorithm to encode individual samples in a lower-dimensional space. The expectation is that forcing an algorithm to find such a lower-dimensional representation of the data entails finding the most salient features and patterns in it; this is the idea behind unsupervised neural network algorithms such as autoencoders. Other examples of unsupervised learning algorithms include clustering and principal components analysis[45].

Finally, reinforcement learning deals with *actions* to be taken as a function of input data. In this case, an algorithm can be thought of as an agent in an environment, who must make choices with regards to both its input, and its current state. Examples of such algorithms include Q-learning, SARSA and Deep Q-learning algorithms.

3.1.2. Classification and regression

Two distinct classes of problems typically exist within supervised machine learning: problems involving *classification*, and problems involving *regression*.

Classification problems describe situations where the output variable can be modeled as a series of discrete, distinguishable classes. One example is the aforementioned task of predicting what type of animal exists in an image. The classification can be binary (class A vs. class B), or multi-way (i.e., k distinct classes exist). These variables can be modelled in a variety of different ways; a typical way to do so is to use so-called "one-hot encoding", whereby a variable is encoded as a probability distribution, with a probability of 1 for the class to which the variable belongs, and a probability of 0 for all other classes.

Regression, on the other hand, deals with problems where the output variable is assumed to be continuous. An example here is the interpolation of a real-valued function (see Figure 3.1).

3.2. Deep Learning

Deep learning has emerged as an umbrella term broadly describing machine learning algorithms based on deep neural networks. Although machine learning and neural networks have been objects of research for a long time, it has only been recently that the more specific field of deep learning has emerged[46]. The main reason for this is a series of advancements, in the past years, which have enabled the appearance of neural network architectures with many layers (hence the term "deep"). This leap was made possible by advances in hardware, namely GPU computing, which enabled training and inference on new data within a reasonable time, but also by software developments such as new activation functions, and most importantly as far as supervised learning is con-

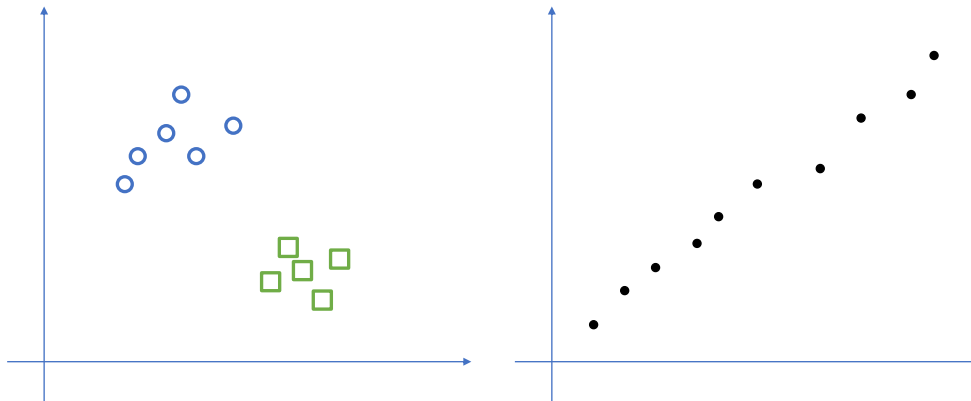


Figure 3.1.: Classification versus regression. In the figure on the left, we are interested in finding what class an input point maps to (either the class of circles, or the class of squares). On the right, we are interested in mapping an input to a real-valued output.

cerned, the appearance of massive labelled datasets with thousands, or millions, of samples.

3.2.1. Neural Networks

The central idea, and indeed the main component, of any neural network is the artificial *neuron*. In the most general formulation, a neural *network* consists in a series of layers of many neurons, with units in successive layers connected to each other. Typically, layers that neither process network inputs, nor produce network outputs (i.e. in the middle of a network) are called *hidden* layers. Many different types of layers exist; for now, we will limit ourselves to discussing fully connected layers. Sections 3.2.4 and 3.2.5 describe convolutional and recurrent layers and networks.

Roughly inspired by the functioning of biological neurons, an artificial neuron receives a vector of inputs x from the neurons of the previous layer to which it is connected. Each of these connections has an associated weight w , which is used to compute a weighted sum of the inputs. To this sum, a bias term b is added, and the neuron then merely computes its output o as $o = \phi(w^T x + b)$, where ϕ is the neuron's pre-defined *activation function*. This is illustrated in Figure 3.2.

Many types of activation functions exist, but non-linear activations are usu-

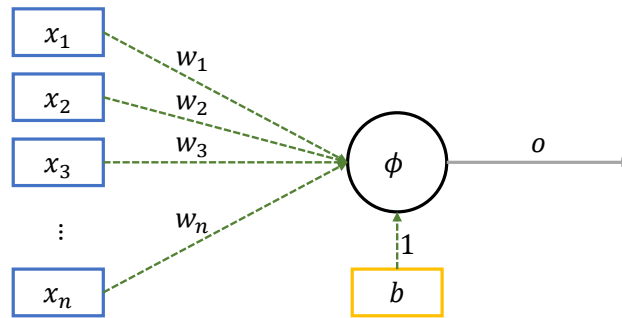


Figure 3.2.: Schema of an individual neuron. x_k denotes an output from a neuron in the previous layer, w_k denotes the weight of the connections, b denotes the bias, and o denotes the output of the represented neuron.

ally chosen to allow a network as a whole to approximate non-linear functions. In many modern neural network architectures, this is frequently the rectified linear unit (ReLU) function, defined as $\phi(x) = \max(0, x)$. Either way, activation functions must be differentiable, for reasons we will see later on.

In a fully connected layer, all neurons connect to all the outputs in the preceding layer; an example of this can be seen in Figure 3.3, where the hidden (yellow) layer neurons all map to the neurons in the first (blue layer). When several fully connected layers such as this are stacked together, the resulting neural network architecture is typically called a *multi-layer perceptron*. As we will see, stacking other types of layers, with different types of neurons, yields different neural network architectures; different architectures can be used to efficiently model different problems.

Although an individual neuron, by itself, performs a relatively simple computation, the composition of many layers of non-linear neurons in a network creates a powerful model, which is, in theory, a universal function approximator[47]. Thus, for any classification or regression task, a deep neural network can, at least in theory, and given an appropriate training set, learn the underlying, unknown, function which maps inputs to their correct outputs; this depends on the network weights and biases having the necessary values that generate that function. Finding those values requires that, like any machine learning algorithm, a neural network be trained. In addition, when designing a neural network, one must make an adequate choice for its *hyperparameters*. These are

manually set by the designer and ultimately depend on the task being modeled; they express choices made about the desired type of model and can include, but are not limited to, how many layers a network has, or what activation function(s) it uses.

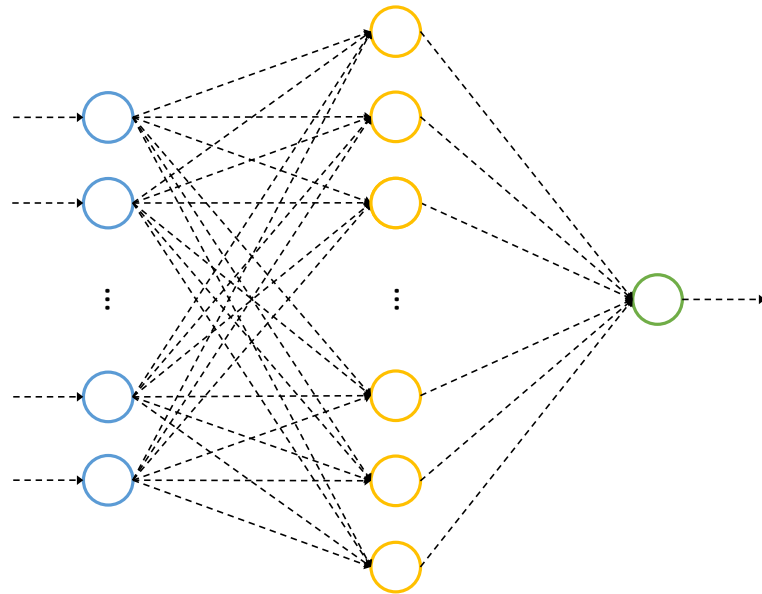


Figure 3.3.: Schema of a fully connected neural net (multilayer perceptron) with three layers: to the left, the input layer; in the middle, a single hidden layer; to the right, an output layer with a single neuron.

3.2.2. Training

A machine learning task can be described as attempting to discover, or approximate, an unknown (but assumed to exist) function $F(x)$; that could be for example the function, or set of rules, that correctly maps all possible images of cats and dogs to their respective animal class. A machine learning algorithm can be conceptualized as the family of functions from which $F(x)$ is drawn, with each individual function $f_{\theta}(x)$ in that family corresponding to a particular machine learning *model*, specified by its individual set of parameters θ . These parameters can be, for example, the weights in a linear regression model; in a neural network, they are the network's weights and biases. The goal of training a neural network is to use a finite set of datapoints drawn from $F(x)$ to find the network

weights that specify the model $f_{\hat{\theta}}(x)$, which is the best possible approximation of F given the data.

Consider a dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where n is the number of elements in D , and each element (x_k, y_k) in D is a mapping between two values. In the example above, x would be a vector of images, while y would be a matching vector whose entries would be values denoting an animal class (for example, $y_k = 0$ for a dog, and $y_k = 1$ for a cat).

We now treat D as the training set for a neural network. Training the network is done in a series of discrete steps; at each step, a different model is evaluated through a loss function $L(\theta)$, which returns a score indicating the mean distance between y and $f_{\theta}(x)$, the model under consideration. For example, the loss function can be a mean squared error:

$$L(\theta) = \frac{1}{n} \sum_{k=0}^n (y_k - f_{\theta}(x_k))^2. \quad (3.1)$$

In practice, many different loss functions exist; the appropriate choice of L depends on the underlying problem that is being modelled. The goal of the training process is to find

$$\hat{\theta} = \underset{\theta}{\operatorname{arg\,min}} L(\theta)$$

i.e., finding the parameters $\hat{\theta}$ of the model $f_{\hat{\theta}}(x)$ that minimizes the loss function and best approximates y , which is assumed to also best approximate $F(x)$. Finding $\hat{\theta}$ can be done with *gradient descent*, in conjunction with *backpropagation*.

Intuitively, the idea behind gradient descent is to, at each step t of the training process, compute the gradient ∇L , i.e., the partial derivatives of the loss function L with respect to all of the network's parameters θ . Through this process, at each training step, the gradient descent algorithm computes the direction in the network parameter space for which the loss function's slope is steepest, and then updates the network parameters in that direction. The update rule for the network parameters is simply

$$\theta_{t+1} = \theta_t + \Delta\theta,$$

where

$$\Delta\theta = -\eta \frac{\partial L(\theta)}{\partial \theta}.$$

Here, η is a pre-specified learning hyperparameter, the *learning rate*, and controls the size of each step $\Delta\theta$. The partial derivatives $\frac{\partial L(\theta)}{\partial \theta}$ must be computed for all the parameters in the network; it is here that backpropagation comes into play. At any given training step, the first thing to be computed is the loss's derivative with respect to the network's outputs. Assuming a mean squared error loss function as described in Equation 3.1, and a network with a single neuron in the final layer with output $o = \phi(wx + b)$, that derivative is simply

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o}(y - o)^2 = 2(y - o).$$

This value can then be recursively backpropagated layer by layer, via the chain rule, to compute the derivative of L with respect to all of the network's parameters. For example, if the activation function ϕ of the neuron in the final layer is merely a linear mapping, then the derivative of the loss with respect to that neuron's weight w would be

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial o} \frac{\partial o}{\partial w} = 2(y - o)x.$$

Derivatives for other weights and biases, in other layers, would be similarly recursively computed; here we show an example with a single output neuron, but this process can be extended for networks with many neurons and layers. From here, it also becomes clear why activation functions must be differentiable.

The idea is that in consecutively repeating this process, one will eventually approach a minimum of the loss function and therefore, the desired function $f_{\theta}(x)$. In general, in machine learning, the best one can achieve is to find a local (but not a global) minimum of the loss, though in practice, in many settings, this is enough to achieve good results. Many specific variations of gradient descent exist; for example, *stochastic* gradient descent computes parameter updates on randomly selected batches of training data, instead of the entire training set, and is proven to be more efficient than simple gradient descent, while achieving in most cases the same results[48]. Other optimization algorithms include

Adagrad, Adadelta, and RMSProp; a proper comparison between, and description of, all these algorithms is beyond the scope of this thesis but can be found in[45].

3.2.3. Overfitting and regularization

In the previous section, we assumed that the vector y in D contains enough information about the underlying, but unknown, function $F(x)$, such that we can compute a good approximation $f_{\hat{\theta}}(x)$ of F . Unfortunately, this assumption is often not valid; in many problems, the number of points in the training set is not sufficient to cover all possible regions of F , and as a result, the solution $f_{\hat{\theta}}(x)$, while minimizing the loss function on the training set, can diverge from the function we are interested in approximating. In this case, a machine learning model is said to be *overfitting*: while it can very accurately map inputs to outputs in the range of its training set, it cannot do so in the data domain outside of that set.

Typically, a way to monitor whether a model overfits its training data is to hold out another dataset, the *validation* set, which is not used for training; instead, it is merely used to monitor training. This can be done by checking whether, as the training progresses, the loss scores for the train and validation data remain similar. If the losses start to diverge significantly, this is good indication that the model is overfitting.

Several strategies exist to prevent overfitting. A simple strategy is that of *early stopping*: once the train and validation losses start diverging, training is stopped, and the best model up to that point is kept. Another option is to use regularization: a penalty term $R(\theta)$ is introduced into the loss function, which becomes

$$L(\theta) = \frac{1}{n} \sum_{k=0}^n (y_k - f_{\theta}(x_k))^2 + \lambda R(\theta). \quad (3.2)$$

In this case, the regularization function R can be, for example, the $L1$ or $L2$ Euclidean norm of θ . The choice of regularization function can yield different results. The factor λ controls the regularization strength; too strong regularization can prevent the discovery of a local minimum for the loss function (in which case a model is said to *underfit*); too weak regularization might end up having

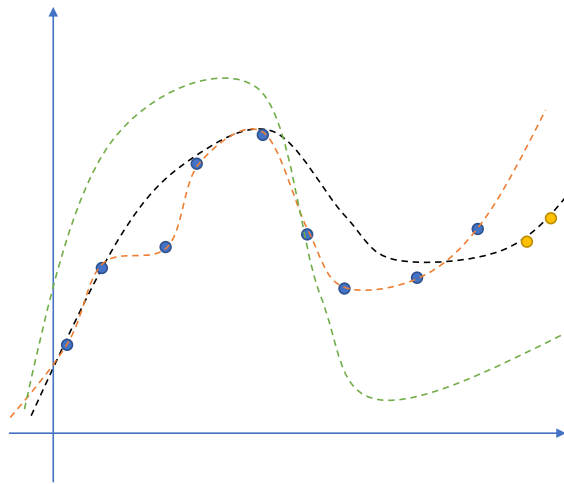


Figure 3.4.: Examples of overfitting (orange line), underfitting (green line) and a good fit (black line) to the underlying data (blue points are the training data, orange points are the validation data)

no effect in stopping overfitting.

Finally, one anti-overfitting strategy typically used with convolutional neural networks (which we detail in section 3.2.4) is the usage of *dropout*[49]: in this case, during training, certain parameters of the network are randomly switched off and on. In theory, this means that during training, the network cannot rely too much on any given parameter to accurately produce a prediction, since that parameter is sometimes unavailable. The idea is that this forces a network to find parameter values that are not too finely-tuned or restrictive, and in so doing, converging to a model which can generalize to outside the training set.

3.2.4. Convolutional neural networks

Different problems generate different types of data, in particular with regards to the correlations present. For example, in an image, any correlations can be expected to be mostly localized: in the picture of a dog, the feature "nose" can not be expected to be spread out in the image, but rather, it should be restricted to a certain region, with the pixels in that region being correlated to each other. Conversely, one can expect little correlation between pixels which are far apart. Another example is the classification of what number is present in an image, such as in the MNIST[50] digit dataset: when determining what number is in

an image, the individual features that make up a number can be assumed to be localized (see Figure 3.5).

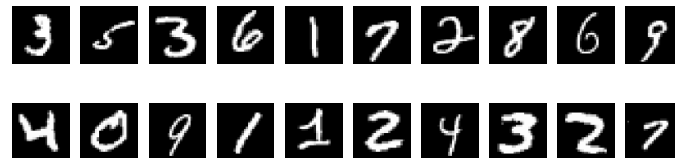


Figure 3.5.: Sample images from the MNIST digit database.

Many different types of architectures of neural networks exist, each with a different area of application. For data with localized correlations, deep convolutional neural networks (CNNs) have achieved the best performance[51]. Roughly inspired by the functioning of the human visual cortex, CNNs are typically composed of a series of convolutional layers interspersed with pooling and dropout layers, with some fully connected layers at the end. The first layer in the network receives the input, and the layers after it look for spatial features. Generally, the first layers look for lower level features (for example, eyes and nose in the picture of a dog) while the final layers combine those to detect higher level features (a dog's face).

In a convolutional layer, the neurons do not all individually connect to all inputs from the previous layer. Instead, the neurons all share the same *kernel* (i.e. weights), and the kernel only links certain neurons to certain inputs. To process the entire input, the kernel slides across it, activating in the regions where a feature of interest has been found. This property of convolutional layers, called *weight sharing*, cuts down the number of network parameters significantly when compared to a fully-connected network, and is one of the reasons behind the effectiveness of CNNs when processing image data.

The data processed by a CNN can have varying dimensionalities; furthermore, regardless of the data dimensionality, one can frequently describe the data as possessing several channels. For example, an image can be decomposed into its RGB components; in this case, the data fed to a CNN would have a depth of 3 (red, green blue), plus a dimensionality of 2, requiring 2-dimensional convolutions for processing. In a signal processing task, the data could have several

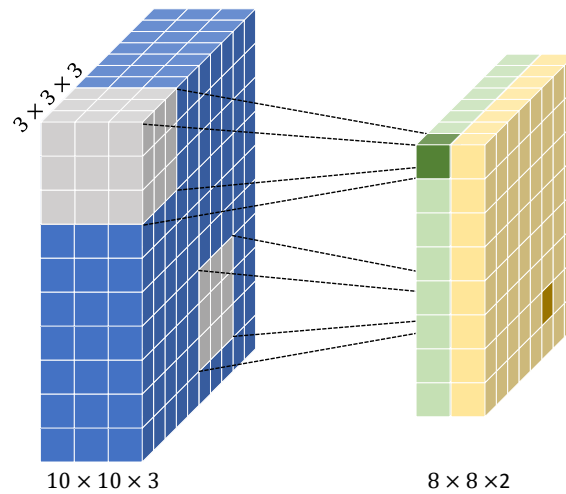


Figure 3.6.: Example of the operation performed by a CNN layer. 2 different $3 \times 3 \times 3$ kernels or filters are applied on the input (the blue shape). The kernels slide across the input, producing two outputs of smaller dimensionality (the convolution operation decreases the input size). Each output neuron in the second layer (i.e. the green and orange grids) processes a different region of the input, convolved by the kernel.

channels, corresponding, respectively, to different signals at the same moment in time, though each signal itself would merely be a 1-dimensional vector of scalar values, and thus, only 1-d convolutions would be required. For example, Figure 3.6 shows two filters being applied on a 3-channel input. In any case, regardless of the data dimensionality, CNNs are particularly suited for processing data with localized correlations.

One assumption behind the data processed by CNNs is that of translational invariance: the position of a certain feature (i.e., a certain localized correlation) in the data is not as important as the existence of the feature itself. To deal with this, CNNs frequently make use of pooling layers. These layers are not trained, but rather, are responsible for performing pre-defined operations on the outputs of the convolutional layers. For example, a max pooling layer will, upon receiving the outputs from a convolutional layer, select (and pass on to the next layer) only a fraction of the outputs with the largest activations, while discarding the remaining values. The idea here is that this operation eliminates information about feature location, while at the same time, selecting only the

most salient features (see Figure 3.7). This further cuts down the number of network parameters, helping to avoid overfitting.

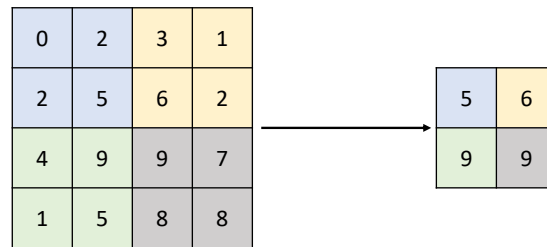


Figure 3.7.: Example of a max pooling layer operating on a 1-channel 2×2 input.

State of the art CNN architectures typically contain many convolutional and pooling layers; for example, the VGG net architecture[52] contains 5 successive convolutional and pooling layers, with each layer having hundreds of kernels. Most of these models achieve excellent results in supervised classification tasks on public image datasets.

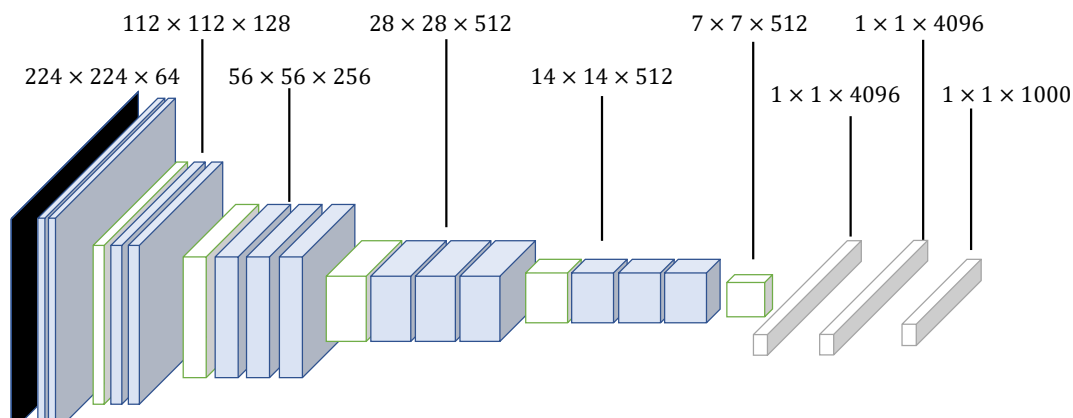


Figure 3.8.: Schema of the VGG network architecture. Blue denotes convolutional layers, green are pooling layers, gray are dense layers. Notice how as one moves deeper into the CNN, the layers become smaller and smaller.

We have so far given examples of using CNNs for types of data where points are considered to be independent from each other. In some domains, this independence assumption is not valid, and one can actually benefit from modelling the data not as a series of points which are independent from each other, but

rather, as elements or steps of a *sequence* that are correlated. An example of sequential data is that of a film, where a single frame does not exist in isolation, but instead, one can assume that a continuity, and therefore a correlation, exists from one frame to the next (even if a frame itself is just an image). It contrasts with random pictures of dogs, where one can assume that there is no correlation between different images. The notion of *time* is critical to make the subtle, yet nevertheless important, distinction between non-sequential and sequential data: with sequential data, one assumes that whatever patterns exist are not limited to a data point in itself; rather, these patterns are spread out across time (i.e. across the sequence). Thus, the accurate output for a certain point (step) in time depends not only on what occurs in that instant, but also on what occurs in other timesteps, particularly in the past.

As it turns out, CNNs can be adapted to process sequential data like this as well[53]; one merely has to model time as another input dimension, and can then design a corresponding CNN architecture. However, CNNs excel mostly in discovering localized features in data, and when used for sequence processing, they can be inefficient when dealing with sequences whose time dimension is very long, and when the temporal correlations are very far apart from each other[54]. As we will see in section 3.2.5, recurrent neural networks are a type of neural network architecture that is designed to deal precisely with these settings.

3.2.5. Recurrent neural networks

Many types of sequential data exist. One good example is that of natural language: when reading a text, the meaning of a word depends not only on the word itself, but also on the wider context and meaning of the words and sentences not only before, but potentially also after it. Unlike CNNs, recurrent neural networks (RNNs) explicitly assume that their input data is sequential (see Figure 3.9), and that any correlations can be spread out across time, and therefore are ideal for processing this type of data. They can be used, among other tasks, to map an entire sequence into a single output (for example, performing sentiment analysis on a text), or to map a sequence into another sequence (for example, performing automated translation)[55].

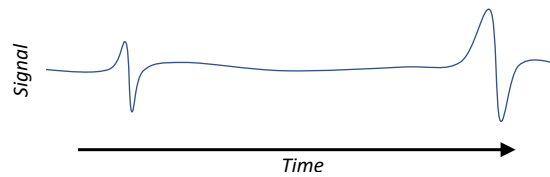


Figure 3.9.: Example of sequential data. The events in the signal (shown as the two spikes) are separated by a certain amount of time, yet they are nevertheless correlated with each other.

The key behind the functioning of RNNs is the recurrent neuron. Recurrent neurons have an internal *state vector*, normally denoted h , where they store information about past parts of the sequence which they are processing. Recurrent neurons work by reading their input data in a series of timesteps; at each step, they update their state vector to reflect the information they just saw; at every new step, when processing the input, they use the information stored in the state vector to process the incoming data, and in so doing, condition their output to the past inputs (Figure 3.10). The actual calculations performed at each timestep, for a simple recurrent neuron, are[56]:

$$h_t = \phi_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \phi_y(W_y h_t + b_y)$$

where:

- h_t is the hidden state vector at t ;
- ϕ_h and ϕ_y are activation functions;
- x_t and y_t are the input and output for timestep t ;
- and, W_h, W_y, U_h, b_h and b_y are trainable parameters.

Like any other artificial neuron, recurrent neurons must be trained. As it turns out, a modified backpropagation algorithm can also be applied to recurrent neurons; in this case, the algorithm is called backpropagation through time (BPTT). The name is due to the fact that, when training recurrent neurons, the flow of gradients occurs not only from output to input, but also into the past, such that the weights W and U can be updated.

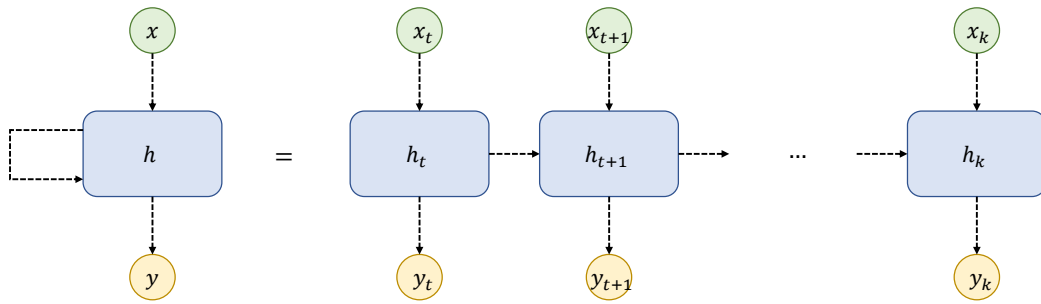


Figure 3.10.: Schematic representation of unrolled RNN processing an input of k timesteps. At each timestep t , the output of the network depends not only on the input x_t , but also on the state vector h_{t-1} produced after processing the previous timestep.

Recurrent neurons can be stacked into layers to create recurrent networks. These networks can contain, in addition to the recurrent neurons, fully connected (and non-recurrent) layers for additional processing, or convolutional layers for detecting short-term correlations in data. In particular, convolutional recurrent neural networks have been used[57] for tasks where one expects to find both spatial and temporal correlations in data; in this case, convolutional layers can extract and encode localized correlations, while the recurrent layers use those encodings to extract longer-term correlations (see Figure 3.11).

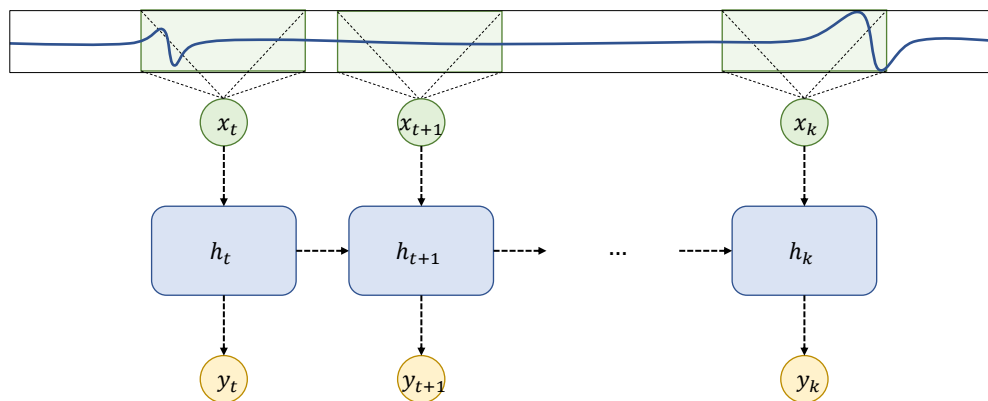


Figure 3.11.: Schematic representation of a convolutional recurrent network.

Unfortunately, in a simple recurrent neuron, the BPTT algorithm tends to produce vanishing or exploding gradients, especially when a network has to process

long sequences. If a single timestep has a small gradient, this value will be recursively propagated into the past timesteps, meaning that the network will have difficulty updating its weights; if on the other hand a step has a large gradient, it will recursively increase in size as it is backpropagated, generating large instabilities during training[46].

For this reason, more advanced architectures have been proposed for recurrent neurons. One of those variants is the Long Short-Term (LSTM) memory unit[58]. LSTMs add a gating mechanism to the recurrent neuron’s architecture, which essentially has the task of deciding, at each timestep, what information to drop, and what information to preserve for the future; this prevents the vanishing and exploding gradients problem. Any of the network architectures mentioned before can be modified to use LSTM units instead of simple recurrent neurons.

3.2.6. Sequence-to-sequence models

When mapping a sequence to another sequence — for example, when performing natural language translation — one can define the goal of a recurrent neural network as that of finding the most likely output sequence $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N)$ given an input sequence $x = (x_1, x_2, \dots, x_N)$ (where n is the sequences’ size). The output sequence \hat{y} is a sample taken from the joint probability distribution $P(y_1, y_2, \dots, y_n)$ conditioned on the input sequence:

$$(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N) = \arg \max_{y_{1:N}} P(y_1, y_2, \dots, y_N | x_{1:N}).$$

Vanilla RNN architectures, such as convolutional LSTMs, can do this task well[59], but they have several limitations. For example, their architectures mean that the input and output sequences must be of the same length (see Figure 3.12). In natural language translation, this can be problematic (different languages might require a different number of words to express the same meaning). It is for this task that sequence-to-sequence (seq2seq) models were developed.

Generically, seq2seq models are composed of two parts: an *encoder* and a *decoder*. This thesis focuses on models built with RNNs, but for example, encoders can also be built with CNNs.

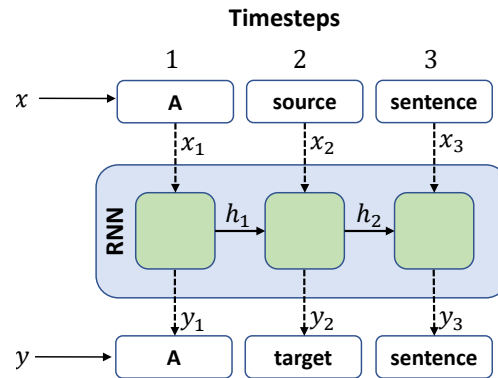


Figure 3.12.: RNN architecture for sequence-to-sequence mapping.

In seq2seq models, the encoder is responsible for receiving the input sequence, and mapping it into a latent-dimensional encoding; the dimensionality of this encoding is a model hyperparameter to be tuned according to the task being modeled. This encoding can be thought of as a representation of the underlying correlations present in the inputs. The decoder is then responsible for producing an appropriate output distribution, subject to the encoding (see Figure 3.14). When both the encoder and decoder are RNNs, they can be jointly trained as any other neural network. This separation into two components allows seq2seq models to produce misaligned outputs (i.e. target sequences with a length different from their inputs), which is particularly suited for language translation.

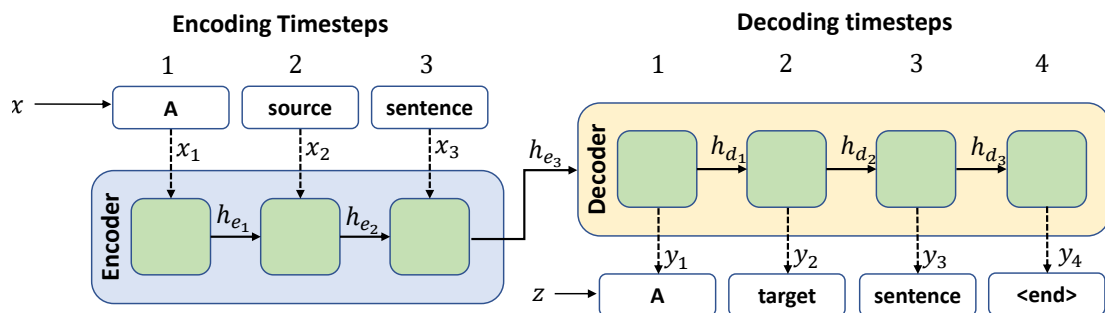


Figure 3.13.: Encoder-decoder architecture for sequence-to-sequence mapping. Encoder and decoder are here assumed to be recurrent neural networks. The elements of output vector y are the tokens for which the decoder gives highest probability at each decoding timestep.

When the encoder receives a new sequence x , with s source timesteps, it processes that sequence like any normal recurrent network; for example, the encoder can itself be a convolutional RNN. As the encoder reads the input sequence step by step, it updates its internal hidden state vector h . After the final input step has been processed, the final hidden state, h_s , is an encoding of all the information found in the source sequence.

The decoder is then initialized such that its first hidden state is equal to h_s ; from there on, at each timestep, the decoder updates its own internal state while generating an output sequence of target length t , a length which may or may not be equal to s .

One additional feature of vanilla RNNs is that they implicitly assume independence between past and future outputs: an output value y_k is conditioned *only* on the inputs (x_0, x_1, \dots, x_k) until timestep k . In practice, this means that a simple RNN architecture cannot reason on different possibilities for its past decisions: the output y_{k-1} has no bearing on y_k (see Figure 3.12). This is also the assumption made in the simplest class of seq2seq models.

In many settings, this independence assumption on the joint output probability distribution $P(y_1, y_2, \dots, y_n)$ is not necessarily problematic; nevertheless, in some problems, it is interesting to drop it. This way, an output y_k no longer depends only on the inputs up to that point, but *also* on the outputs (y_1, \dots, y_{k-1}) produced until that point. Seq2seq models allow for easily modelling this assumption, by making the decoder autoregressive, i.e., feeding it with its own past inputs.

Formally, in this case, the probability of a single timestep output y_k is no longer modelled as

$$P(y_k | x_{1:k}) \tag{3.3}$$

but rather, as

$$P(y_k | x_{1:k}, y_{1:k-1}). \tag{3.4}$$

For the joint distribution of possible outputs, this yields

$$P(y_1, y_2, \dots, y_N | x_{1:N}) = P(y_1 | x_1) P(y_2 | x_{1:2}, y_1) \dots P(y_N | x_{1:N}, y_{1:N-1})$$

$$= \prod_k P(y_k | x_{1:k}, y_{1:k-1}). \quad (3.5)$$

This distinction is important: with the independence assumption (expressed in Equation 3.3), one can only produce a single output sequence for a given input. Removing that assumption (as expressed in Equation 3.4), it is actually possible to generate an entire probability distribution of outputs for a single input; from that distribution, one can then collect samples of different output sequences, and explicitly choose the one with the highest probability. In addition, when sampling, it is possible to explicitly select only sequences that are known to be valid, subject to some expert domain knowledge; details about how this was done in this work can be found in chapter 7. In this case, a seq2seq model now directly computes the joint probability distribution of outputs, while an additional algorithm, such as for example a beam search algorithm, is responsible for drawing high-probability samples from that distribution.

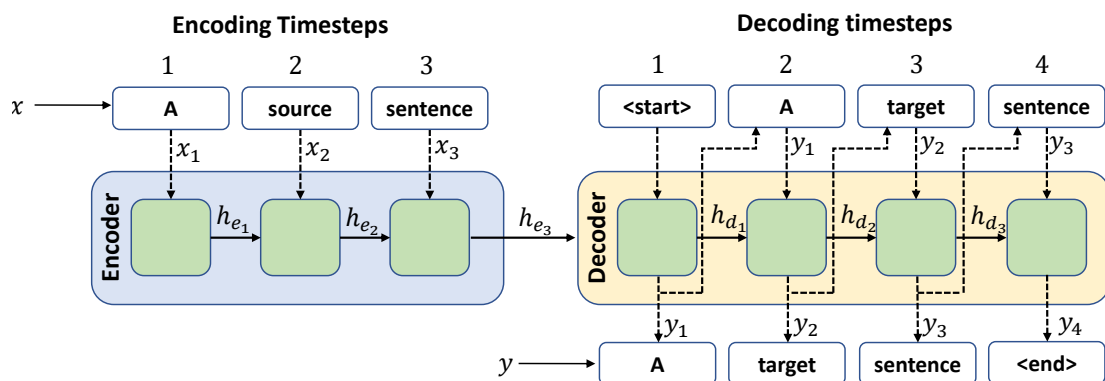


Figure 3.14.: Seq2seq architecture with an autoregressive decoder.

The feeding of past outputs as new inputs allows the decoder to reason over several possible sequences in parallel, and it is this that allows it to model the whole joint probability distribution defined in Equation 3.5. In the example of Figure 3.14, the encoder input at timestep t merely corresponds to the maximum probability output for time $t - 1$ (which is the token "A"). This is a valid way to generate a solution (i.e. a sample from the target joint distribution), but the generated sequence is not guaranteed to be optimal, because the past outputs are still fixed. To actually evaluate and compare different samples of the target joint distribution requires reasoning over several different past outputs

simultaneously. This can be done by, at each timestep, feeding *all* possible outputs back to the decoder, and at the next timestep, computing the new outputs by conditioning them to each individual input (i.e. as per Equation 3.4). This corresponds to evaluating different beams in a search tree: each beam is merely a different sequence drawn from the output distribution, and the goal is to find the beam with highest probability. The probability of each individual beam is computed at every timestep through Equation 3.5; for example, in Figure 3.15, the probability of beam (or sequence) "AA" will be $P(AA) = P(A|x_{0:1})P(A|x_{0:2}, A)$.

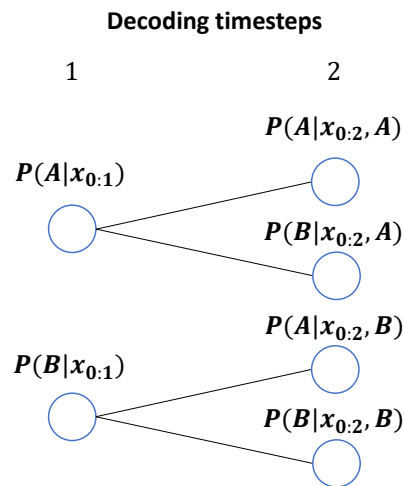


Figure 3.15.: Decoding with tree search in a task where 2 possible output tokens exist. x denotes the input to the seq2seq model, which has already been processed by the encoder. At timestep 2, the decoder has 4 possible output sequences (AA, AB, BA, BB) under evaluation.

In practice, this procedure can quickly become infeasible, especially when dealing with long outputs and a large number of possible output tokens, because the number of different sequences to evaluate grows exponentially. One way to get around this problem is by using a beam search algorithm to traverse the tree. In this case, at every timestep, only a pre-defined number of beams with the highest probability are expanded, with the rest being discarded. While the attained solution is then not guaranteed to be optimal, in many applications[60], it is usually close to it.

3.2.7. Attention mechanism

Although they are powerful, seq2seq encoder-decoder RNN models ultimately assume that the summary produced by the encoder, as its final hidden state, can fully express the information in the input data which is used to condition the decoder's output. More recent research[61] indicates that this assumption does not always hold, and it is for this reason that the attention mechanism was proposed. Attention is merely an extension to se2seq models, whereby the decoder no longer receives only the last encoder state, but rather, at every decoding timestep, the decoder has explicit access to all of the encoder's hidden states — in practice, the encoder can now access, albeit indirectly, all of the source (input) timesteps.

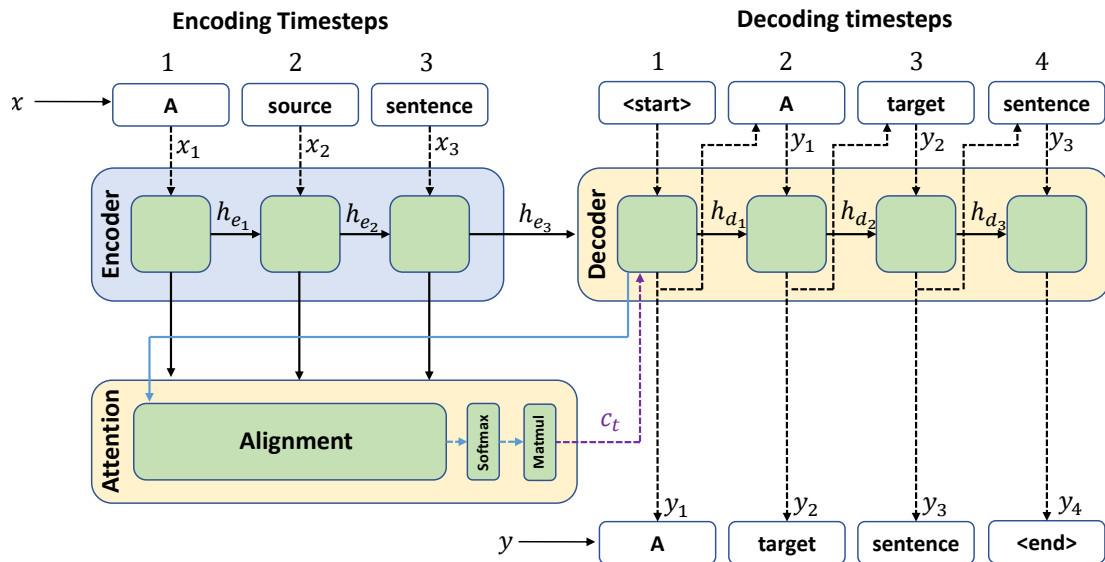


Figure 3.16.: Sequence-to-sequence model with attention. At each decoding timestep, the context vector c_t is recomputed by the attention layer.

At each decoding timestep, the attention layer receives all of the source (encoder) hidden states. It then computes an alignment vector, whose elements are individual weights to be assigned to different parts of the input hidden states, and whose values add up to 1. Conceptually, this alignment vector expresses how much attention should be paid to a particular encoder hidden state, at the current decoder timestep. This alignment vector is then used to compute a *context* vector which is fed back to the decoder. This way, the decoder has not only

access to the entire input sequence, but can also explicitly assign greater importance to some parts of the sequence than others, at each decoding timestep. This gives the decoder a much richer context, allowing it to produce better sequences. Furthermore, the alignment weights produced by the attention layer can actually be used to visualize, in inference time, which parts of the source sentence the decoder focused on when producing specific words. A thorough description of different types of attention mechanisms can be found in [61] and [62]. Further details about the implementation of seq2seq models and attention for this work can be found in Chapter 7.

3.3. Gaussian process regression

Generically, a Gaussian process is merely a generalization of the concept of a Gaussian probability distribution. Whereas a Gaussian distribution describes the values that a random variable (or variables) can take, a Gaussian process describes a family of functions, with some functions being more likely than others. Though they can also be used for classification, we will limit this section to discussing Gaussian processes for regression.

Just as with neural networks, when performing regression with the Gaussian process framework, one is generically interested in finding, or at least approximating, an underlying (but unknown) function. Several points (x, y) belonging to this function are previously known (they are equivalent to a neural network's training set), and the goal is to find an appropriate estimate of the function which takes the train points into account. To that end, a Gaussian process *model* must be specified. Generically, the term model can refer, for example, to parametric models for data, such as simple linear regression, or a different, complex, non-linear model. Such models are called parametric because one usually tries to discover a series of model weights, or parameters, that specify the model, which one hopes will match the underlying function being looked for; in fact, this is also the case with neural networks. In the Gaussian process framework, models are non-parametric: they can be thought of as compositions of an infinite amount of basis functions. This means that the underlying function's parameters are integrated out of the model, and gives Gaussian process models significant

flexibility.

A Gaussian process model has several components. Most important are the *prior* and the *posterior*. The prior represents one's beliefs about the underlying function of interest without taking any training data points into account, and is itself a distribution over possible functions. For example, one might assume that the function of interest is smooth, and encode a prior that favors smooth functions. For a certain model, the posterior is then the distribution of possible functions, subject to the model's prior, that also contain the training set. Although a Gaussian process model does not have function parameters *per se*, it does have *hyperparameters*, which specify the characteristics of the model's prior. Note that, if the observations in the training set are considered to be noisy (with a certain noise value σ), one can also compute the posterior on the same positions as the training data, and obtain an estimate of the real underlying values of the function in the points of the (noisy) training set.

Mathematically, the posterior is obtained through Bayes's rule, by multiplying the prior with a likelihood term and dividing by a marginal likelihood[63]:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \quad (3.6)$$

Because the terms in Equation 3.6 are probability distributions, for a fixed model, the bottom term of the equation is just a normalizing constant and can be ignored when computing the posterior. The prior is a probability distribution; in the multi-variate case, it is specified by a mean vector and a co-variance matrix. In most applications of Gaussian process regression the prior mean is specified to be 0 (as shown in Figure 3.17), which is the simplest possible assumption that one can make about the unknown function. The prior co-variance is computed through a co-variance function which is chosen to reflect what one assumes about the underlying and unknown function. For instance, a typically used co-variance function is the squared exponential: it encodes the belief that the underlying data distribution is smooth. The extent to which that smoothness holds across the function's domain, i.e., the extent to which different points are correlated, is usually called the model's *length scale*, and is an example of a Gaussian process model hyperparameter.

One of the differences between Gaussian process regression and typical dis-

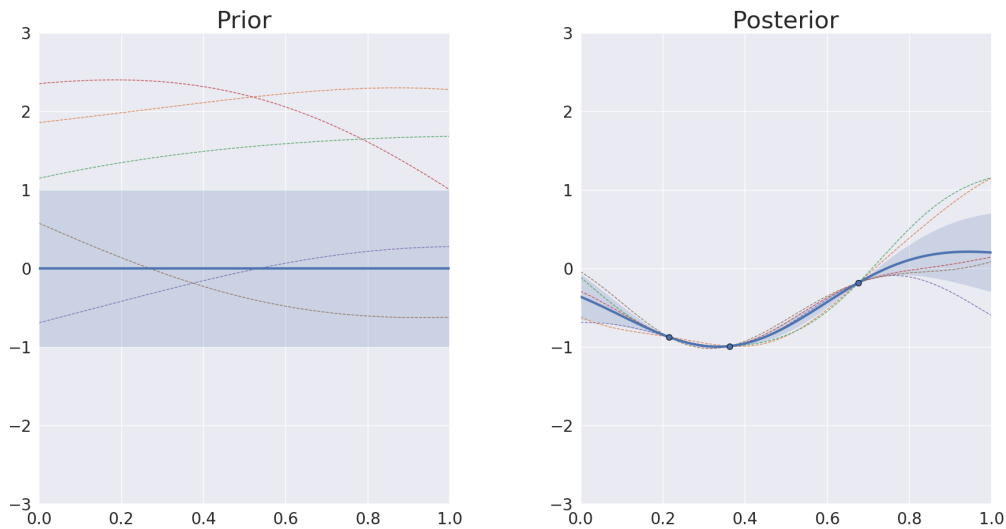


Figure 3.17.: Example of GP regression. The left figure shows the prior mean (blue) and standard deviation, as well as 5 functions drawn from the prior distribution. The right panel shows the posterior mean and standard deviation (blue), and 5 other functions drawn from the posterior, computed based on the shown points. In both cases, the functions are computed at 100 points in the interval $[0, 1]$.

criminative neural networks is that the former, unlike the latter, can yield uncertainty estimates regarding their own outputs; this can be done by considering the posterior standard deviation as a measure of the model’s uncertainty. Furthermore, Gaussian processes are highly flexible: in theory, one can always add more points, and the model (because it is non-parametric) will always be able to fit them. This stands in contrast to, for example, linear regression: in that case, one might try to fit non-linear data with a linear model, which becomes problematic.

So far, we have limited this discussion to the cases where one has already decided on a (Gaussian process) model, and then merely wishes to compute the posterior and get samples of the underlying function from it. However, this opens up the door to overfitting: how can one know that the chosen model isn’t too complex, and therefore fitting the data too tightly while being unable to generalize to unseen points? Fortunately, with Bayesian formalism, this can be solved, through the mechanism of model selection. Essentially, this means evaluating the evidence term of Equation 3.6 for different models, and then select-

ing the model whose evidence is highest. A general mathematical description of how this can be done can be found in [63], and chapter 5 of this thesis describes how it was done specifically in the context of this work. Nevertheless, the high-level idea is that the evidence term incorporates two different features of any model: its complexity (i.e. its capacity for generalization) and its capacity for fitting the data. Models that are either too complex (very good at fitting data, but poor at generalizing) or too simple (good at generalizing, but which fit the data poorly) have lower evidence scores. In practice, this means that when the evidence term is evaluated and compared for different models, the trade-off between models' complexity and data fit is automatic; given a series of models and their respective evidences, the one whose evidence is highest is guaranteed to be the simplest model that can explain the data. This may sound counter-intuitive at first, because it is easy to assume that one is interested in fitting the data as well as possible. And indeed in theory, one can always find models that fit the data ever more tightly; what the evidence term in Bayes's theorem tells us is that those models are not always, necessarily, the most likely ones, given the data.

The main drawback of performing model selection is its computational complexity. First, it requires evaluating an integral which in many cases is analytically intractable and has to be approximated by Markov chain methods. But, even in settings where the evidence term can be calculated analytically, its computational complexity means that model selection can become prohibitively expensive. Indeed, that is one of the motivations behind the work described in chapter 5.

3.4. Concluding remarks

This chapter gives an introduction to the algorithms and methods used over the course of this thesis. All of these methods have in common their ability to automatically extract information from data, with a view to discovering a function that correctly allows them to make make correct predictions on new points. With regards to neural networks, several topics are approached which build up on top of each other. In the simplest case, fully connected neurons can be composed into a multilayer perceptron for simple classification or regression

problems. More advanced units, such as convolutional and recurrent neurons, can be used to build network architectures that excel in looking for, respectively, localized and long term correlations in data with specific characteristics, namely images and sequences. Yet all these architectures are merely discriminative, in that they produce point-like estimates of outputs. A more advanced class of models, the sequence-to-sequence models, can be used to directly target probability distributions instead, which makes them much more powerful. Finally, Gaussian processes, which can be used for classification and regression, are very flexible models for machine learning, capable of fitting complex functions while at the same time estimating the uncertainty of their outputs. Furthermore, under certain conditions, one can explicitly search for Gaussian process models that reduce the likelihood of overfitting.

All of these algorithms have slight variations, as well as advantages and disadvantages. Sometimes, different models can be used for the same problem; for example, CNNs can be used, like RNNs, to process sequential data. However, CNNs require, by definition, inputs of fixed length, which might make it difficult to use them in settings where sequences of variable length exist (unless a strategy like padding is used). Furthermore, for long sequences, either very long convolutional windows would be necessary to keep track of long term correlations, or a CNN would need to have the entire sequence available to it, which in some settings might not be possible. Similarly, vanilla RNN architectures can generate sequences, but they cannot, for instance, be used for encoding explicit domain rules into the prediction process, which sequence-to-sequence models allow for. Gaussian processes, while interesting in many applications, have nothing like the neural network weights that are fixed and saved after training, and re-used for every new data point. This means that, when performing, for example, model selection for Gaussian processes, the process has to be repeated for every new point, whereas a neural network (which has a fixed model once trained) only requires a forward pass to estimate an output.

In either case, the goal of this chapter is to show that many algorithms exist that can learn from data, and that deciding which one to choose for a particular task is not always a trivial question to answer. For the nuclear fusion problems which were described in chapter 2, the following chapters will explain the rationale and assumptions behind why the models here described were used, and

how they were implemented.

Part II:

Publications

4. Summary

The goal of this work was to develop deep learning and artificial intelligence based approaches for modelling several problems in nuclear fusion experiments. That work is described in the three publications that constitute this thesis. At the time the thesis was delivered, two of them had been published, and the third one had been accepted for publication. At the time of the thesis defense, all papers had been published. The citations for the papers are:

-
- F. Matos, J. Svensson, A. Pavone, T. Odstrčil, and F. Jenko. “Deep learning for Gaussian process soft x-ray tomography model selection in the ASDEX Upgrade tokamak”. In: *Review of Scientific Instruments* 91.10 (2020), p. 103501.
 - F. Matos, V. Menkovski, F. Felici, A. Pau, F. Jenko, T. Team, E. M. Team, et al. “Classification of tokamak plasma confinement states with convolutional recurrent neural networks”. In: *Nuclear Fusion* 60.3 (2020), p. 036022 .
 - F. Matos, V. Menkovski, A. Pau, G. Marceca, F. Jenko, and the TCV Team. “Plasma confinement mode classification using a sequence-to-sequence neural network with attention”. In: *Nuclear Fusion* 61.4 (2021), p. 046019.

The first paper[64] presents a convolutional neural network (CNN) that automatically performs the Bayesian model selection procedure for Gaussian process regression, applied to tomography in fusion. Gaussian process tomography is an established method for performing tomographic inversion of certain physical properties of the plasma, given some data observation. The method first requires specifying a prior that encodes one’s beliefs about the underlying distribution before any data is seen. A posterior is then computed, subject to the

prior and the data, which can, under certain assumptions, readily yield the most likely values for the distribution of interest.

The Gaussian process regression framework still leaves the door open to overfitting, since a chosen model can be overly complex. To overcome this, Bayesian model selection can be used, whereby different models (individually specified by different priors) are compared with regards to their evidence. This evidence is a value that depends not only on how well a model fits the data, but also on how complex the model is. This term can, under certain assumptions, be calculated analytically, and one is interested in finding the model for which the evidence is highest. This will be the simplest model that can explain the data, and it solves the problem of overfitting. Yet even calculating this term analytically can be computationally intensive; this is especially true in settings such as nuclear fusion, where one can have dozens or hundreds of measurements, and is interested in discovering data distributions with possibly thousands of dimensions. It is this that motivates the use of a CNN that is trained to automatically determine, for individual measurements, their corresponding most likely Gaussian process model, without the need for the analytical step.

The second paper[65] presents two discriminative neural network models — one a CNN, the other a recurrent neural network (RNN). They are used to model the problem of finding a sequence of plasma confinement states, and of detecting edge localized modes (ELMs), given a series of signals collected from fusion experiments. ELMs are important for tokamak operation because they can be useful for removing impurities from the plasma. However, they can also, if uncontrolled, constitute unacceptable power loads on the reactor. The ability to understand and control ELMs is therefore essential for future fusion experiments. Similarly, it is important to understand and detect changes between different plasma confinement states, given that they yield different plasma behavior. However, the physical mechanisms behind the changes between these states are still not completely understood.

Nevertheless, changes between confinement modes, as well as ELMs, leave signatures in signal time traces. These can be used by experts to identify, after an experiment, where they occurred. The models described in the second paper replicate this process. They are trained on labeled samples to produce estimates of the probability of the occurrence of an ELM, or of changes between different

confinement states, as a function of time, given measurement data.

The third paper[66] takes the previous approach one step further, by using a generative sequence-to-sequence model that can produce the full output probability distribution of plasma confinement states, instead of point-estimates of the sequences. A separate beam search algorithm is then responsible for finding sequences of high probability from that distribution.

5. Deep learning for Gaussian process soft X-ray tomography model selection in the ASDEX Upgrade tokamak

Authors Francisco Matos¹
 Jakob Svensson²
 Andrea Pavone²
 Tomas Odstrcil³
 Frank Jenko¹

Affiliations: ¹Max Planck Institute for Plasma Physics, Garching, Germany
 ²Max Planck Institute for Plasma Physics, Greifswald, Germany
 ³Plasma Science and Fusion Center, Massachusetts Institute
 of Technology, Cambridge, MA, USA

Published in: Review of Scientific Instruments

DOI: <https://doi.org/10.1063/5.0020680>

Author contributions: The first author carried out the work of code implementation, algorithm development and model design. The remaining authors contributed significantly with discussions, suggestions, and reviewing the final paper. In addition, Tomas Odstrcil was responsible for getting the required data.

Abstract. Gaussian process tomography (GPT) is a method used for obtaining real-time tomographic reconstructions of the plasma emissivity profile in tokamaks, given some model for the underlying physical processes involved. GPT can also be used, thanks to Bayesian formalism, to perform model selection — i.e., comparing different models and choosing the one with maximum evidence. However, the computations involved in this particular step may become slow for data with high dimensionality, especially when comparing the evidence for many different models. Using measurements collected by the Soft X-ray (SXR) diagnostic in the ASDEX Upgrade tokamak, we train a convolutional neural network (CNN) to map SXR tomographic projections to the corresponding GPT model whose evidence is highest. We then compare the network’s results, and the time required to calculate them, with those obtained through analytical Bayesian formalism. In addition, we use the network’s classifications to produce tomographic reconstructions of the plasma emissivity profile.

5.1. Introduction

Computed tomography generally refers to the process of imaging the interior of a body through indirect measurements. In many applications, this is achieved by focusing penetrating radiation on an object of interest from several directions and measuring the resulting decrease in radiation intensity on the opposite side (due to absorption by the body itself). Use of this information, the so-called *projection* of the object, allows one to reconstruct its internal properties[14].

In the case of radiative bodies, an alternative way to determine their properties is to perform cross-sectional imaging by treating the emitted radiation itself as a projection[67]. In the field of nuclear fusion, this procedure is employed in many tokamaks for the reconstruction of plasma emissivity profiles[68]. More specifically, in the ASDEX Upgrade tokamak, such imaging can be done with information from the soft X-Ray (SXR) diagnostic, which measures the line-integrated radiation emitted by the plasma along several lines of sight; these can be used to perform tomographic reconstruction (or inversion) of the plasma emissivity profile. Knowledge of this is useful for exploring magnetohydrodynamic phe-

nomena, in addition to studying accumulation of impurities inside the plasma (particularly tungsten) due to their large contribution to the total amount of radiation[21].

Several techniques exist for solving the tomography problem[69]. One approach is to use regularization-based algorithms, namely Tikhonov[70, 71]- and minimum Fisher-based techniques[72]. More recently, work has also been done using machine learning methods, namely deep neural networks[24, 23, 73], that are trained to create new reconstructions based on existing ones.

Yet another method is Gaussian process tomography (GPT)[25]. GPT is an established method for performing tomographic inversion on many different types of physical distributions, that are modeled as posterior Gaussian distributions in a Bayesian setting. Computing a posterior first requires specifying a prior distribution, which encodes one's assumptions about the underlying physical process before any measurements of it are taken. The posterior can then be computed based on that prior, and on an observation (measurement) of the data generated by the physical process. The prior itself can either be a fixed distribution, or be drawn from a family of different models.

Knowing the posterior, GPT guarantees that one can obtain the most likely (maximum *a posteriori*, MAP) estimate for the tomographic reconstruction as well as its associated error values. More interestingly, however, through Bayesian inference, GPT prescribes a way to estimate the evidence for different models, through a process known as Bayesian model selection. This procedure can be of particular importance in cases where the choice of prior might have a strong effect on the results of the tomographic inversion.

Unfortunately, in a neural network, there are no guarantees[74] about whether the reconstructions obtained correspond to the MAP estimate of the underlying distribution, and there is no direct way, in standard Deep Neural Networks, such as convolutional neural networks, to obtain uncertainty estimates on the outputs. Bayesian neural networks[75, 76] and generative adversarial networks (GANs)[77] can generate probability distributions for their outputs; however, they can be computationally expensive and, in the case of GANs, difficult to train[78].

On the other hand, neural networks essentially store whatever function they have learned (through their training process) in their weights, making the in-

ference process for new data very fast. With GPT, computing the MAP estimate based on a fixed model is also sufficiently fast for real-time purposes. This does not necessarily hold true, however, when performing bayesian model selection, since the process requires a series of additional computational steps, namely matrix inversions or using non-linear optimizers, which can be time-consuming, especially for data with a high dimensionality.

Thus, we propose an approach where we train a convolutional neural network (CNN) to learn the GPT model selection procedure. To do this, we take SXR measurement samples from several ASDEX Upgrade shots and, through Bayesian model selection formalism, compute for each data point the corresponding model (out of a set of possible, pre-defined ones) with the highest evidence. We then train the CNN to reproduce this step, i.e., to map measurements to their highest evidence model. Finally, through the GPT framework, we compute the tomographic reconstruction of the plasma emissivity profile for each measurement, given the most likely models predicted by the CNN.

This paper is organized as follows. Section 5.2 gives an overview of the problem of tomography, in particular soft x-ray tomography ASDEX Upgrade tokamak, and the existing techniques to solve it, including a review of GPT with bayesian model selection. Section 5.3 details the data we collected, the formulation of our problem, and the model proposed to solve it. Section 5.4 details the direct results of the neural network classification, and the tomographic reconstructions obtained based on them; section 5.5 describes and discusses our conclusions.

5.2. Background

5.2.1. Computed Tomography

The purpose of tomography is to reconstruct the internal (either two- or three-dimensional) properties of a given body from non-local measurements. Radon showed[12] that a 2D distribution can be retrieved from an infinite set of line-integrated measurements. In practical applications, the number of available measurements is always finite, but it is nevertheless possible to produce accurate reconstructions from a discrete set of measurements[79]. Tomographic

algorithms can achieve this by taking many *projections* of the object of interest from different directions[14]. Mathematically, a projection is a function that computes the line-integrated absorbency (or, in the case of fusion plasmas, emissivity) of a body along several paths or lines of sight (LOSs) as

$$P_{\theta}(t) = \int_{L(\theta,t)} G(x,y) dL \quad (5.1)$$

where t is a point in the projection domain, $L(\theta,t)$ is the LOS crossing the body mapping to t (along a direction given by an angle θ), and $G(x,y)$ is the two-dimensional physical distribution of interest (see Figure 5.1).

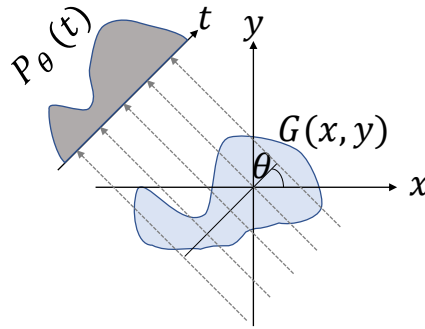


Figure 5.1.: An illustrated projection, P_{θ} , measured along an angle θ . The blue area $G(x,y)$ is the cross section of interest, and is being traversed by radiation. Because different rays traverse different areas of the object, the value at each point t in the projection space will be different. Figure reprinted with permission from F. Matos, "Deep Learning for Plasma Tomography" M.Sc. Thesis (University of Lisbon, 2016) [7].

By computing several tomographic projections with different directions (i.e. different values of θ), it is possible to reconstruct $G(x,y)$. For an exact reconstruction based only on the projections, an infinite number of them would need to be obtained. However, the problem is highly ill-posed[80], since small changes in projection space can translate into large changes in the tomographic reconstructions. Furthermore, in many settings such as nuclear fusion experiments, it is difficult, or impossible, to obtain more than a handful of such projections, making the problem under-determined — that is, the dimensionality of the reconstruction grid is much larger than that of the projection, ultimately re-

sulting in an infinite number of solutions (reconstructions) that can fit the data. For these reasons, in most tomography applications in fusion plasmas, some additional information, in the form of assumptions about the function $G(x, y)$, must be introduced in order to obtain a tomographic reconstruction.

5.2.2. SXR tomography at ASDEX Upgrade

In the ASDEX Upgrade Tokamak, the Soft X-ray (SXR) diagnostic[19] consists of eight pinhole cameras that measure the total radiation emitted by the plasma along 208 different volumes of sight (VOSs)[21]. We considered the extent of the VOSs in the toroidal and poloidal directions of the tokamak to be minimal, and treated them instead as lines of sight (LOSs). In addition, we also ignored the fact that the LOSs in the same camera array partially overlap. Based on this, the measurements collected by the individual cameras correspond to a single projection of the underlying plasma emissivity distribution, which is computed at 208 discrete positions, in a poloidal plane. In terms of the poloidal coordinates (R, z) of the 2D tokamak cross-section, the total brightness, b_i , incident on a single detector, i , is given by[81]

$$b_i = \int_r G(R, z) dr \quad (5.2)$$

where $G(R, z)$ is the plasma emissivity distribution (in W/m^3) and r is the LOS corresponding to b_i (Fig. 5.2).

By discretizing Equation 5.2, one obtains the plasma emissivity distribution at a finite number of positions (or pixels) along a tomographic reconstruction grid. In this case, the incident radiation on a single detector, assuming an associated noise, ξ , is[82]

$$b_i = \sum_{j=1}^n M_{i,j} g_j + \xi_i \quad i \in 1, \dots, 208. \quad (5.3)$$

From now on, we will denote the set of values b_i , that is, a set of 208 line-integrated SXR measurements of the plasma emissivity taken at a certain point in time, as the plasma's tomographic projection in that instant. We denote Equation 5.3 as the *forward model* of the problem. Here, n corresponds to the total

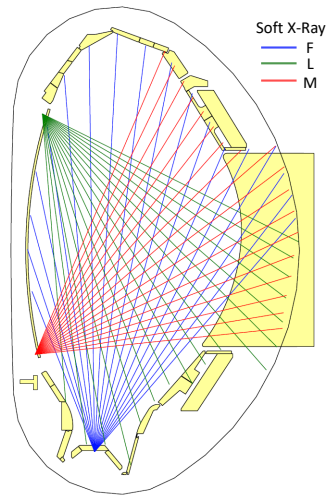


Figure 5.2.: ASDEX Upgrade cross-section, and partial schema of the SXR measurement system, with three cameras (F, L and M) shown (plot obtained with *diaggeom*)

number of pixels on a tomographic reconstruction grid, whereas $M_{i,j}$ is the discretization of the function $M(R,z)$ in Equation 5.2, mapping the relative contribution of pixel j of that grid to measurement i of the projection. The actual values of M were pre-defined and contingent on the geometry and configuration of the sensors inside the machine, which can vary between different shot campaigns. Consequently, we denote M as the *geometric matrix*. The goal of a tomographic reconstruction algorithm is then to solve the ill-posed problem by using the tomographic projection (i.e., the 208 measurements b_i) and some *a priori* knowledge about the plasma to find a suitable tomographic reconstruction g that satisfies Equation 5.3.

5.2.3. Regularization-based Methods

To solve the ill-posed problem, traditional tomographic algorithms use regularization techniques, usually based on assumptions regarding the smoothness of the plasma emissivity profile, that constrain the space of possible solutions. Such algorithms, however, are often computationally expensive and typically can only be used for post-experimental tomographic reconstruction, due to computational time constraints. In addition, the quality of the reconstructions is

highly dependent on assumptions made about the data[21]. Generally, those assumptions are encoded into the reconstructions through the use of Tikhonov regularization. In this case, computing the tomographic reconstruction of the plasma emissivity profile becomes a matter of finding a reconstruction \hat{g} such that

$$\hat{g} = \arg \min_g (\|Mg - b\|^2 + \Lambda O(g)) \quad (5.4)$$

where $O(g)$ is a penalty term that encodes information about expected properties of the target plasma distribution, multiplied by a regularizing parameter Λ that controls the regularization strength[83]. There are several options for the choice of the regularization term O ; typical choices are the Laplace operator, which favors smooth solutions, and minimum Fisher information[22], that favors solutions that are mostly flat in low-intensity regions, and peaked in high-intensity ones.

5.2.4. Deep Learning-based Methods

Recent work has applied deep learning algorithms to the tomographic problem, namely by using de-convolutional neural networks to produce tomographic reconstructions taking measurement data as input[23, 73, 84]. This is achieved by training the networks on reconstructions that have been previously computed using standard tomographic algorithms. Generically, in a deep learning setting, a Deep Neural Network is *trained* to learn a function mapping an input x into its target output y [46]; that is,

$$y = y(x, \theta) \quad (5.5)$$

where θ denotes the neural network's parameters, i.e. its weights and biases. The training process consists in finding an optimal value for θ that minimizes the mismatch between the network's outputs and their corresponding labels.

In our setup, training a deep neural network to produce tomographic reconstructions would have required training it with measurements from the SXR diagnostic and pre-computed reconstructions, produced by other algorithms

(namely, regularization-based ones). The expectation would then have been that the parameters θ computed during training would have converged to values such that if new, unseen data were fed into the network, it would be capable of *generalizing* to outside of its training set. However, even assuming good generalization capacity of a neural network, it is at most as good as whatever data it has been trained on. In other words, should existing tomograms have had errors, a neural network would have learned to reproduce them.

5.2.5. Gaussian Process Tomography

Another alternative is to use bayesian probability theory to produce tomographic reconstructions, by treating the underlying unknown plasma emissivity distribution as a *Gaussian process*. Evaluating that process along a discrete set of points (the tomographic reconstruction grid) yields a multi-dimensional Gaussian distribution.

By definition, in the Gaussian process framework, one assumes that multiple solutions for the tomographic reconstruction exist, in a Gaussian distribution of possible solutions. Treating the tomography problem with this framework allows using Bayesian formalism, which guarantees that the most likely solution for the tomographic reconstruction (i.e. the maximum *a posteriori*, or MAP, estimate), subject to some assumptions about the underlying physical and data distributions, can be computed through Bayes's formula[45],

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (5.6)$$

In the Gaussian process tomography (GPT) setting, the terms in the formula are multivariate probability distributions, which are assumed to be Gaussian. Each of those distributions is specified by a vector of means (which we denote μ), whose entries are the individual means of each random variable in the multivariate distribution, and a covariance matrix, which we denote Σ , where each entry denotes the pair-wise covariance between those same variables.

In Bayes's theorem, the term $P(A)$ is called the *prior*. In GPT, by denoting the underlying plasma emissivity as e , the prior distribution $P(e) \sim \mathcal{N}(\mu_{pr}, \Sigma_{pr})$ encodes existing assumptions about the physical emission process, without ob-

serving any data (SXR measurements); this is equivalent to the assumptions one might encode in the regularization parameter in traditional tomography (Equation 5.4). Each random variable in the prior distribution is also Gaussian, and corresponds to the prior plasma emissivity e at each point x in the tomographic reconstruction grid.

Here, the prior mean has a size equal to that of the tomographic reconstruction grid, n . Intuitively, the prior covariance matrix Σ_{pr} encodes information about the expected smoothness of the plasma emissivity. The entries in the covariance matrix are computed for all pairs of points in the reconstruction grid through a prior covariance function. One covariance function generally used in Gaussian process regression is the squared exponential[85]; in using this function, the prior covariance between a pair of points x_1 and x_2 in the tomographic reconstruction grid becomes

$$\text{cov}(x_1, x_2) = \theta_f^2 \exp\left(-\frac{d(x_1, x_2)}{2\theta_x^2}\right), \quad (5.7)$$

where $d(x_1, x_2)$ is a distance metric between points x_1 and x_2 . The prior covariance is only dependent on that distance and on $\theta = \{\theta_f, \theta_x\}$, which are the model's *hyperparameters* and are common to all points in the reconstruction grid. The parameter θ_f controls the prior variance of the plasma emissivity at a given location in the reconstruction grid, whereas the parameter θ_x , usually referred to as the *length scale*, controls the extent to which points at a certain distance from each other in the reconstruction grid are correlated. Models where the length scale is large yield high correlations even between grid points which are far apart, while smaller length scales yield covariance matrices where only points which are closer to each other are significantly correlated.

With these definitions, the prior becomes a probability distribution for the plasma emissivity, e , subject to the model's hyperparameters, i.e., $P(e|\theta)$, before any data, that is, a tomographic projection, has been observed. The prior can then be updated by multiplying it with the likelihood of the data d (as per Bayes' theorem), yielding the posterior distribution, $P(e|d, \theta)$:

$$P(e|d, \theta) = \frac{P(d|e, \theta)P(e|\theta)}{P(d|\theta)} \quad (5.8)$$

The denominator in Bayes's theorem is known as the model *evidence* or *marginal likelihood*; if one is merely computing the posterior $P(e|d, \theta)$, it can be ignored, as it is just a normalizing constant. Interestingly, however, one can use this term to compare several different models (each with their own prior), and choose the one which best fits the data[25]. In this case, one assumes a *hyper-prior*, from which different possible priors (individually specified by different hyperparameters) are sampled. The evidence can then be computed for different models — a process that is referred to as *marginalization* — and the model with the highest evidence can be selected[86]. Calculating this requires an evaluation of the integral[63] over e

$$P(d|\theta) = \int P(d|e, \theta)P(e|\theta) de \quad (5.9)$$

which in many cases is analytically intractable. However, in our case, the prior for an emissivity distribution is a multivariate Gaussian, defined as

$$P(e|\theta) = (2\pi)^{-\frac{k}{2}} |\Sigma_{pr}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mu_{pr} - e)^T \Sigma_{pr}^{-1} (\mu_{pr} - e)\right), \quad (5.10)$$

where k denotes the number of variables in the prior distribution (that is, the number of pixels in a reconstruction grid). In addition, we assume a data distribution which is also Gaussian, $P(d|\theta) \sim \mathcal{N}(\mu_d, K + \Sigma_d)$ [25]. The mean μ_d of the data distribution is merely the value of the measurements in a projection. The data co-variance has two components: matrix K denotes the (noise-free) co-variance values, and is a linear transformation of the prior covariance Σ_{pr} (imposed on the plasma emissivity) into measurement (data) space, given by $K = M\Sigma_{pr}M^T$, where M is the geometric matrix defined in Equation 5.3. The other component, Σ_d , is a diagonal matrix whose non-zero entries are the absolute noise values, ξ , of each measurement in a projection; we assume the noise values are independent from each other. By assuming that this noise is also Gaussian, the logarithm of the integral in Equation 5.9 can be analytically calculated as[63]

$$\log(P(d|\theta)) = -\frac{1}{2} \left(m \log(2\pi) + \log|K + \Sigma_d| + (\mu_d - f_L)^T (K + \Sigma_d)^{-1} (\mu_d - f_L) \right) \quad (5.11)$$

where m is the number of SXR measurements in a tomographic projection and f_L is the mapping of the prior mean, μ_{pr} , (imposed in reconstruction space) onto measurement space, given by $f_L = M \cdot \mu_{pr}$.

The marginalization procedure is particularly useful because the trade-off between model complexity and data fit is automatic — the model for which the evidence score is highest is always the simplest model that can explain the data, an embodiment of the Occam's Razor principle[87]. In addition, the model evidence is also a function of the variance σ^2 of the data (through matrix Σ_d), which means that it is possible to treat the expected projection error as an additional hyperparameter of the model to be tuned; this can be done, for example, by treating the data variance as a fraction of the measured value of SXR radiation in the tomographic projection, with the value of the fraction constituting an additional model hyperparameter. This means that, through the Gaussian process tomography framework, one can estimate not only the most likely model for the underlying plasma emissivity distribution, but also the most likely model for the error values of the data(though this is no replacement for a calibration of the detectors with a known source).

Once the most likely model is selected, and applying Bayes's formula, the posterior mean, μ_{post} , and posterior covariance, Σ_{post} , as a function of the prior mean μ_{pr} and prior covariance Σ_{pr} for that model, are respectively given by[16]

$$\mu_{post} = \mu_{pr} + \Sigma_{pr} M^T (K + \Sigma_d)^{-1} (d - f_L) \quad (5.12)$$

and

$$\Sigma_{post} = \Sigma_{pr} - \Sigma_{pr} M^T (K + \Sigma_d)^{-1} M \Sigma_{pr}. \quad (5.13)$$

By computing the posterior distribution $P(e|d, \theta) \sim \mathcal{N}(\mu_{post}, \Sigma_{post})$, one can then produce tomographic reconstructions either by sampling from $P(e|d, \theta)$, or simply by taking the mean of that distribution as the tomographic reconstruction (because the distribution is Gaussian, the mean corresponds to the maximum *a posteriori* estimate). In addition, one can directly obtain uncertainties for the tomographic reconstruction from the diagonal values of the posterior covariance matrix, which correspond to the individual posterior variances of each pixel in the reconstruction grid.

The drawback of the marginalization procedure, however, is its potential computational complexity. First, the calculation of the evidence term involves a series of matrix multiplications and an inversion, which can be cumbersome particularly in our setting because of the dimensionality of the data, which generates very large matrices. Matrix K in Equation 5.11 can be previously computed and kept in memory when performing model selection (which we do). However, in our setting, we treat the underlying error as a fraction of the data, and therefore, the values appearing in matrix Σ_d change with every new data point. As a result, the matrix determinant and inversion in Equation 5.11 must be recomputed for every new point, which is the main reason behind the high cost of the Bayesian optimization procedure. Furthermore, the evidence must be computed for all models that are taken into consideration. When each model has several hyperparameters, the number of possible models to evaluate can become very large, which means that finding the optimal one can be time-consuming. For practical purposes, this limits the number of models which can be evaluated and thus, potentially limits the quality of the results.

We therefore propose to bypass the need for analytical marginalization, by training a classifier (in this case, a convolutional neural network) to automatically choose the most likely model (out of several pre-defined ones) for the tomographic projection data collected by the ASDEX Upgrade SXR system.

This has potentially several advantages. On one hand, a Gaussian process model, while potentially having priors and posteriors with many dimensions, can be fully specified by its much smaller set of hyperparameters. In practice, this allows for parameterizing a distribution of high dimensionality with only a few variables. In the case of this work, this means that neural networks will learn to map tomographic projections to a lower-dimensional space (of dimensionality equal to the number of models under consideration). This should facilitate the network's learning process, allowing for easier generalization when compared with deep learning methods that attempt to map projections directly into a reconstruction space of larger dimensions. On the other hand, for potential real-time applications, this method potentially speeds up Gaussian process tomography, since it bypasses the marginalization procedure.

5.3. Methods

5.3.1. Soft X-Ray Data

For this work, we had at our disposal a collection of 112 ASDEX Upgrade shots, totalling 127528 data points (208-dimensional tomographic projections), with each dimension corresponding to a specific detector in the Soft X-ray (SXR) system. The projections come from the down-sampled signal of the SXR diagnostic, at a sampling rate of 250Hz. The dataset also contains an error model, which assigns every measurement in every projection an estimated error value; we develop this topic in Section 5.3.2. In many cases the SXR detectors can be damaged and yield completely erroneous measurements, such as for example negative brightness; in these cases, the measurement is simply considered to be faulty. A sample projection can be seen in Figure 5.3.

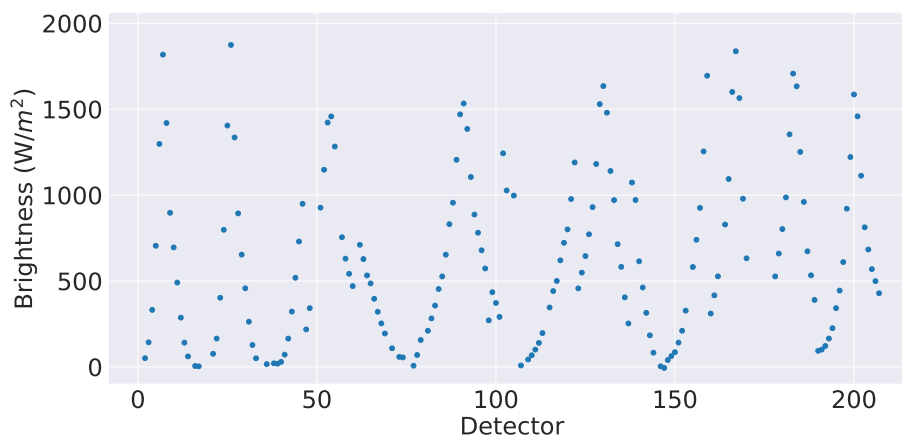


Figure 5.3.: Sample tomographic projection (SXR measurements) from the ASDEX Upgrade tokamak, taken from shot #30294 at $t = 5,8691s$. Faulty measurements have been removed.

We also possessed a geometric matrix M that maps the relative contribution of each pixel in a $60 \times 40(2400)$ -dimensional tomographic reconstruction grid to each of the 208 SXR measurements in a projection. Each pixel in the grid has a pair of poloidal coordinates (R, z) , based on the poloidal dimensions of ASDEX Upgrade; the tomographic reconstruction is computed on this grid. The geometric matrix itself was computed based on the physical layout of the SXR sensors in the ASDEX Upgrade vessel, and holds for all shots in our dataset.

5.3.2. Dataset Generation

Before training the neural network classifier, we generated its training and validation dataset by individually computing, for all the measured tomographic projections, the most likely model from which those projections were sampled. To that end, the first task was to define different models and their respective priors (individually specified by their specific set of hyperparameters) and then, through Equation 5.11, compare them based on their evidence.

As is typical in Gaussian process regression tasks [63], for all models, we defined the prior means, μ_{pr} , as vectors of zeros, of size 2400 (the size of the reconstruction grid). We computed the prior covariance matrices Σ_{pr} using a squared exponential function as defined in Equation 5.7; in essence, this covariance function encodes our belief that correlations between pixels on the tomographic reconstruction grid will decay exponentially as the distance between those points increases. We computed the distance between pairs of points in the reconstruction grid (expressed in terms of their poloidal coordinates) using the Euclidean definition, i.e., $d(x_1, x_2) = \sqrt{(R_1 - R_2)^2 + (z_1 - z_2)^2}$. The covariance function (Equation 5.7) has only two parameters: θ_f , the individual variance of single pixels, and θ_x , the length scale which controls the extent of the correlation between pixels in the reconstruction grid. Different models have priors specified by different values of these hyperparameters, but they all use the same definition of co-variance function and distance.

Finally, we defined, for each model, our assumptions regarding the data distribution associated with that model. For all models, we discarded measurements that had been previously labeled as faulty. In practice, this meant that, when evaluating the evidence for models, and when computing the maximum *a posteriori* (MAP) estimate for the plasma emissivity, some of the 208 measurements of each projection were not used. We treated the remaining (non-faulty) measurements in the projections as the mean values μ_d of the data distribution.

The individual variances, σ^2 , of the variables in the data distribution correspond to the entries in the diagonal of matrix Σ_d of Equation 5.11, and represent the uncertainties in the measurements. We computed the values σ^2 as fractions of the measurement values themselves; those fractions depend on a scaling factor θ_{err} which is multiplied by the measurements, and that constitutes an ad-

ditional hyperparameter for the models under consideration. We assumed this value to be global, i.e., for any given model, we assume that the scaling factor is the same for all measurements in a projection.

Formalizing, we iteratively computed, for each individual data point (i.e. projection), and from a set of pre-defined models for the plasma emissivity and data distributions that might have generated that projection, the highest-evidence model — that is, through Equation 5.11 we looked for $\hat{\theta} = (\hat{\theta}_f, \hat{\theta}_x, \hat{\theta}_{err})$ such that

$$\hat{\theta} = \arg \max_{\theta} \log P(d|\theta),$$

where $P(d|\theta)$ is the model evidence from Equation 5.8. We searched for the ideal hyperparameters (i.e. the hyperparameters that specify the highest-evidence model) in a grid by assuming a uniform hyper-prior (all models were considered to be equally likely), and computed the model evidences at several discrete positions in the hyper-prior space. The question was then, what positions in the hyper-prior space should one evaluate the models' evidences on? This required taking several factors into account.

The first requirement was the expected nature of the plasma emission process itself. A previous analysis of the measurement data, and of existing tomographic reconstructions from ASDEX Upgrade[21], showed that the plasma emissivity has a wide dynamic range for different regions of the plasma, with emissivity in the plasma core being up to 3 orders of magnitude higher than in the pedestal. Likewise, in some periods of some shots, the maximum radiation value in the reconstruction grid was in the order of magnitude of $10^2 W m^{-3}$, while in other phases, it could be as large as $10^5 W/m^{-3}$. Thus, we considered this range in emissivities a good region to explore possible values for the hyperparameter θ_f . In addition, ASDEX Upgrade has a minor radius $a = 0.5m$ (horizontally) and $b = 0.8m$ (vertically)[88]; given this and the size of our reconstruction grid, we assumed that a good region of the hyper-prior in which to evaluate the evidence for certain values of θ_x ranged, in the limit, from 0 (no correlation at all between pixels) to 1.6. For the hyperparameter θ_{err} we assumed that, in the limit, it could range from 0 (no noise in the tomographic projections) to 1 (all of the measured brightness corresponded to noise).

The second requirement related to the training process for neural networks.

For this work, we wanted to train a neural network to perform a classification task — to learn to map measurements to the the most likely model. Typically, in a machine learning classification setting, care should be taken such that training samples fed to a network are reasonably balanced with respect to their different classes; that is, a good training practice is that one class not be too over-represented in the data when compared to others. In our setting, achieving this balance required experimenting with different potential evaluation positions in the hyperparameter search grid. This comes at the cost of leaving out some grid positions for which some data points might have had higher evidence scores.

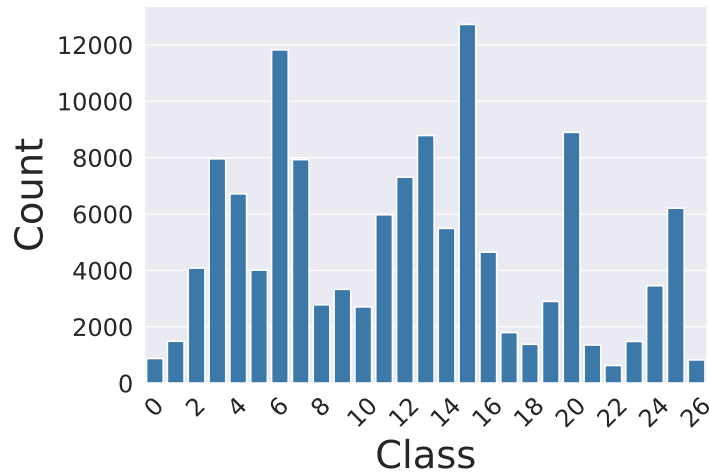


Figure 5.4.: Number of data samples (from all available shots) mapping to each class (set of hyperparameter values), after the marginalization procedure.

In practice, considering these requirements, we performed several evaluations through trial and error of the hyper-prior at different positions; we settled on a $3 \times 3 \times 3$ grid, where the points correspond to $\theta_f = \{500, 1000, 2500\}$, $\theta_x = \{.15, .175, .2\}$ and $\theta_{err} = \{.5, .75, .1\}$, which corresponds to 27 Gaussian process models. Performing the Bayesian model selection procedure on all projections in our dataset using the models parameterized by these values of $(\theta_f, \theta_x, \theta_{err})$ yielded a relative balance in terms of the amount of data samples mapping to each of the 27 possible classes (points on the hyperparameter grid); this can be seen in Figure 5.4, which shows the number of points mapping to each class.

Computing the evidence for different models for all tomographic projections took a total of 48h.

This dataset — i.e., the mappings between tomographic projections and the class to which their highest-evidence model belongs (out of 27 possible ones) — was then used to train and test the neural network classifier. The choice of modelling the task with a classifier, instead of treating it as a regression problem, has one motivation: the loss function to use, and how to model the network outputs.

If performing regression, one option would have been to have a separate model output for each hyperparameter and combine them with a mean squared error loss. However, it is not obvious that this would work correctly, because the evidence term depends on all hyperparameters together. For example, a target output $y_t = (\theta_f = j, \theta_x = k, \theta_{err} = l)$ could be approximated by the network as $y_n = (\theta_f = j, \theta_x = 0.9k, \theta_{err} = l)$. Computing the mean squared error between y_t and y_n would yield a potentially good score because on average because the hyperparameter values are similar in both cases; however, there is no such guarantee for the evidence score, which could be very different from one case to the other. In fact, it is for this very reason that when performing classification we use a single output with 27 possible categories, instead of a separate output (and loss function) for each hyperparameter, each with 3 possible categories. The alternative would have been to use a loss function based on the evidence score itself; however that would have been computationally infeasible because it would require the evaluation of Equation 5.11 for each network gradient update.

5.3.3. Deep Learning Model

Several possibilities exist when it comes to modelling deep neural network architectures. For our purposes (the learning of the Bayesian model selection procedure) we opted to use a convolutional neural network (CNN). CNNs are widely used for signal processing tasks, due to their ability to efficiently detect spatial correlations in data, which is what we expected to find in our SXR measurements. The model we used is, with regards to its architecture, inspired by the network for classification of images described in [52], popularly known as the VGG network. We designed the model using the Keras framework for deep

learning[89].

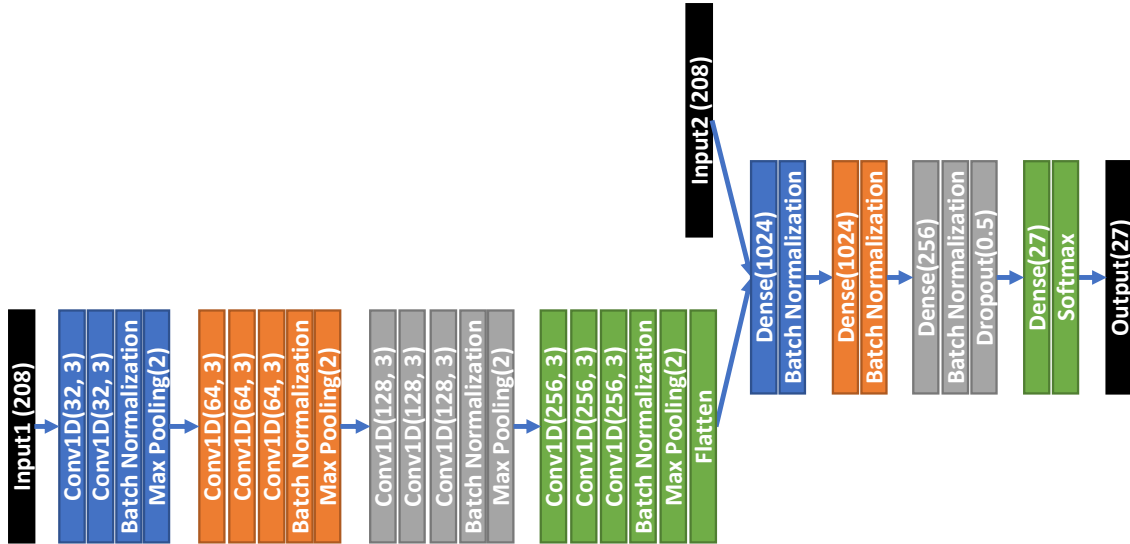


Figure 5.5.: Schematic of the deep learning model used for this work.

The network itself receives two inputs: a tomographic projection (208 SXR measurements, fed as input 1 in Figure 5.5), and a corresponding mask of ones and zeros (taken from the existing error model in our dataset) corresponding to input 2 in Figure 5.5, that gives information regarding which measurements in the projection are assumed to be faulty. The network uses a series of convolutional layers followed by max pooling layers to process high-level features in the measurement data. The output of those layers is then combined with the information in the error mask and processed in the last layers of the network, which are standard fully-connected layers. We also used batch normalization[90] layers to speed up training, and dropout in the final layer[91] to increase the network’s capacity for generalization outside of its training set. We used the rectified linear unit (ReLU) activation function throughout the entire network apart from the last layer, which uses a softmax function, because we modelled the network output as probabilities over 27 possible classes, which must add up to 1. For the same reason, we used categorical cross-entropy as loss function. We used the Adam optimizer[92], and left all optimizer hyperparameters at their default values.

5.4. Results

We here performed two separate assessments. First, we evaluated the accuracy of the neural network’s fit of the individual projections to their highest-evidence models. Then, based on the highest-probability class determined (by the network) for each data point, we computed the corresponding maximum *a posteriori* estimate of the tomographic profile, and measured the fit of those reconstructions to the data by projecting them back into measurement space (through the forward model in Equation 5.3), obtaining their *back-projections*. We then measured the deviation between those back-projections and the original tomographic projections.

5.4.1. Neural Network

To increase the robustness of our methods, we opted to train an ensemble of neural networks (of equal architectures), using the k-fold cross-validation strategy[93]. K-fold cross validation is useful to determine whether the choice of the train/test split has biased whatever results have been obtained, or whether the results can be assumed to hold independently of the data split. We opted to divide our data into $k=10$ folds — that is, we trained 10 networks with different overlapping splits of train data, and tested them on non-overlapping validation splits. We trained the networks for 50 epochs, and ran them on an NVIDIA Quadro RTX 5000 Graphics Processing Unit (GPU). The total training time for the whole ensemble was 1h, while total prediction time for the validation data was 41,62s.

As the networks performed a 27-way classification, we used top-k categorical accuracy as a metric for network classification quality. We now follow with a brief explanation of this metric.

Each data point x (corresponding to a tomographic projection) in our dataset was assigned a label, y_{label} , denoting for which of the 27 model classes the evidence was highest. A classifier learns, through the training process, to compute the probability of that point belonging to a certain class $P(C(x) = c)$, where c can take one out of 27 possible values; we denote the vector containing the probabilities of belonging to each of those classes y_{pred} . We further define y_{pred_k} as

the k -th most likely class given by a classifier for x ; for example, for y_{pred_1} , one would get

$$y_{pred_1} = \arg \max_c P(C(x) = c) = \arg \max_c y_{pred}$$

whereas for c_{27} one would have

$$y_{pred_{27}} = \arg \min_c P(C(x) = c) = \arg \min_c y_{pred}.$$

Based on this, the top-k accuracy metric then calculates for each data point:

$$acc_k(x) = \begin{cases} 1 & \text{if } y_{label} \in \{y_{pred_1}, \dots, y_{pred_k}\}. \\ 0 & \text{otherwise.} \end{cases}$$

We then computed the categorical accuracy metric on the validation data for the 27 different values of k . Because we opted for a cross-validation train and test strategy (with an ensemble of 10 classifiers), we show the mean value and standard deviation of the top-k accuracy across all members of the ensemble. The results of the metric can be seen in Figure 5.6 (up to $k=27$) and Table 5.1 (up to $k=5$). In Table 5.1 we show the results only up to $k=5$ for ease of comprehension.

	K				
	1	2	3	4	5
mean	0.509	0.783	0.903	0.955	0.977
st. dev.	0.041	0.038	0.033	0.016	0.01

Table 5.1.: Accuracy mean and standard deviation across ensemble of 10 neural networks, for validation data, up to top-5 accuracy.

An analysis of Table 5.1 and Figure 5.6 shows that the ensemble of 10 neural networks achieves very good results on the classification task, with a mean top-5 accuracy score of 0.976 (out of a maximum score of 1) for validation data. This means that for any data point, the correct prior is found within the top-5 most likely outputs predicted by the network in 97,7% of cases. In practice, if one is exclusively interested in finding the single, most likely, prior, this result reduces

the search space for the right hyperparameters from 27 classes to 5. Should one be interested only in comparing different models for certain physical distributions, this result also allows for quickly estimating which priors are more or less likely. Furthermore, the standard deviation of the accuracy score demonstrates consistently low values, indicating that the choice of train/test split for our data did not significantly bias the achieved results; all neural networks in the ensemble behave similarly, even if tested on different data.

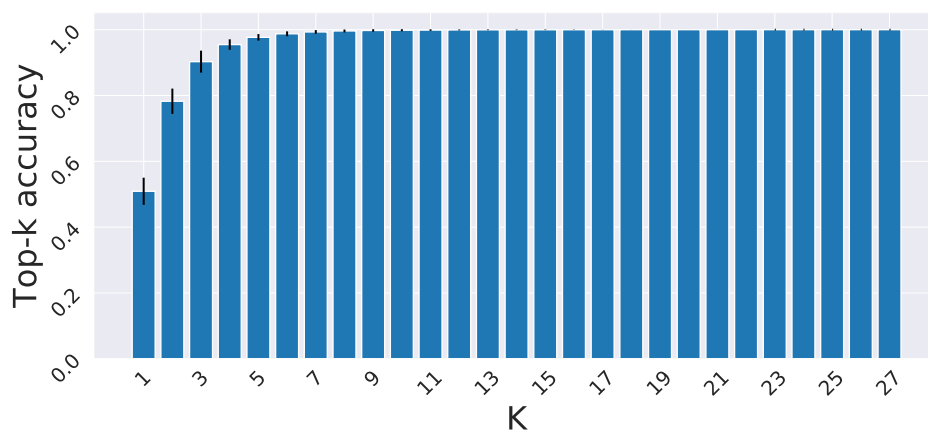


Figure 5.6.: Top-k accuracy (up to $k=27$) for validation data. The blue bars indicate the mean accuracy across the ensemble of 10 networks, while the smaller black bars indicate the accuracy’s standard deviation across the ensemble (for each k)

5.4.2. Sample Reconstructions

In addition to evaluating the neural network’s capacity for classification purposes, we also produced and evaluated tomographic reconstructions. To that end we took, for each data point in the validation dataset, the most likely class prediction given by the neural network; this class prediction maps to one of the models we have previously defined. We then computed, based on the class prediction and on Equations 5.12 and 5.13, the posterior mean and covariance for each data point. We took the posterior means (i.e. the maximum *a posteriori* estimates) as the tomographic reconstructions of the plasma emissivity - each mean was a 2400-dimensional vector, where each entry μ_j denotes the most

likely value for the plasma emissivity in a point j in the reconstruction grid. The posterior covariances allowed us to determine the error of the tomographic reconstruction, by taking the diagonal of the covariance matrix, which corresponds to the individual variance σ_{post}^2 of each pixel in the reconstruction grid; we converted the value of that variance into a percentage error by once again taking advantage of the $3-\sigma$ rule, and computing said percentage, for pixel j , as

$$\%err_j = 3 \frac{\sqrt{\sigma_{postj}^2}}{\mu_j} \times 100\%$$

Two sample results can be seen in Figures 5.7 and 5.8. For the reconstruction error, we show only points where the percentage error was found to be below 100%. Notice how in Figure 5.8, despite the value of σ_f being 2500, a reconstruction with a much larger maximum intensity can still be produced. Furthermore, in both cases, the reconstruction error values are noticeably lower in the center of the grid. We explain this with two factors: the larger number of LOS covering that region, which lower the uncertainty in the reconstructed values, and the higher intensity of the plasma emissivity, which lowers the relative error.

5.4.3. Model complexity and data fit

To evaluate the quality of the models we performed, for each maximum *a posteriori* (MAP) tomographic reconstruction, a pass through the forward model defined in Equation 5.3 to obtain the corresponding back-projection, i.e., the projection of the reconstruction back into measurement space. The marginalization procedure guarantees that, from the ensemble of models that is evaluated for a data point, the simplest model that can fit the data will be chosen, and we have shown that the proposed convolutional neural network can in most cases do this as well. However, a problem can arise if the ensemble of models from which we sample is itself mostly composed of overly simple or overly complex models.

Computing the backprojections allowed us to see how the obtained MAP estimates fit the original SXR data. Our expectation was that, if the models we defined were too complex, we would observe very tight fits of the data, with very low deviations between it and the backprojections. Conversely, if the models were too simple, we would tend to see large differences between the backprojec-

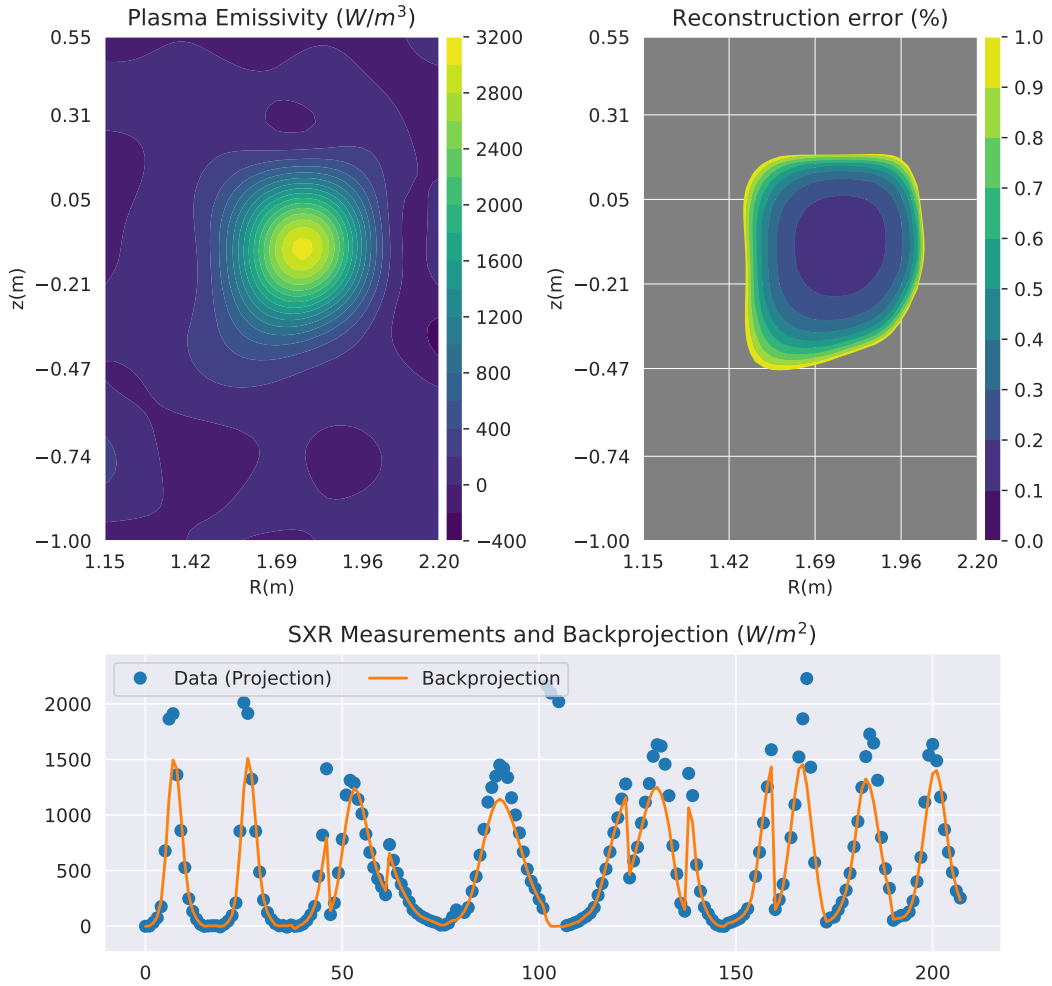


Figure 5.7.: Sample tomographic reconstruction and error, and comparison between the SXR measurement and the back-projected reconstruction, for ASDEX Upgrade shot #30857, $t=4,0441s$. The determined model hyperparameters by the classifier were $\theta_{err} = 0,75, \theta_f = 500, \theta_x = 0,175$; 200 measurements (out of 208) were used for this reconstruction.

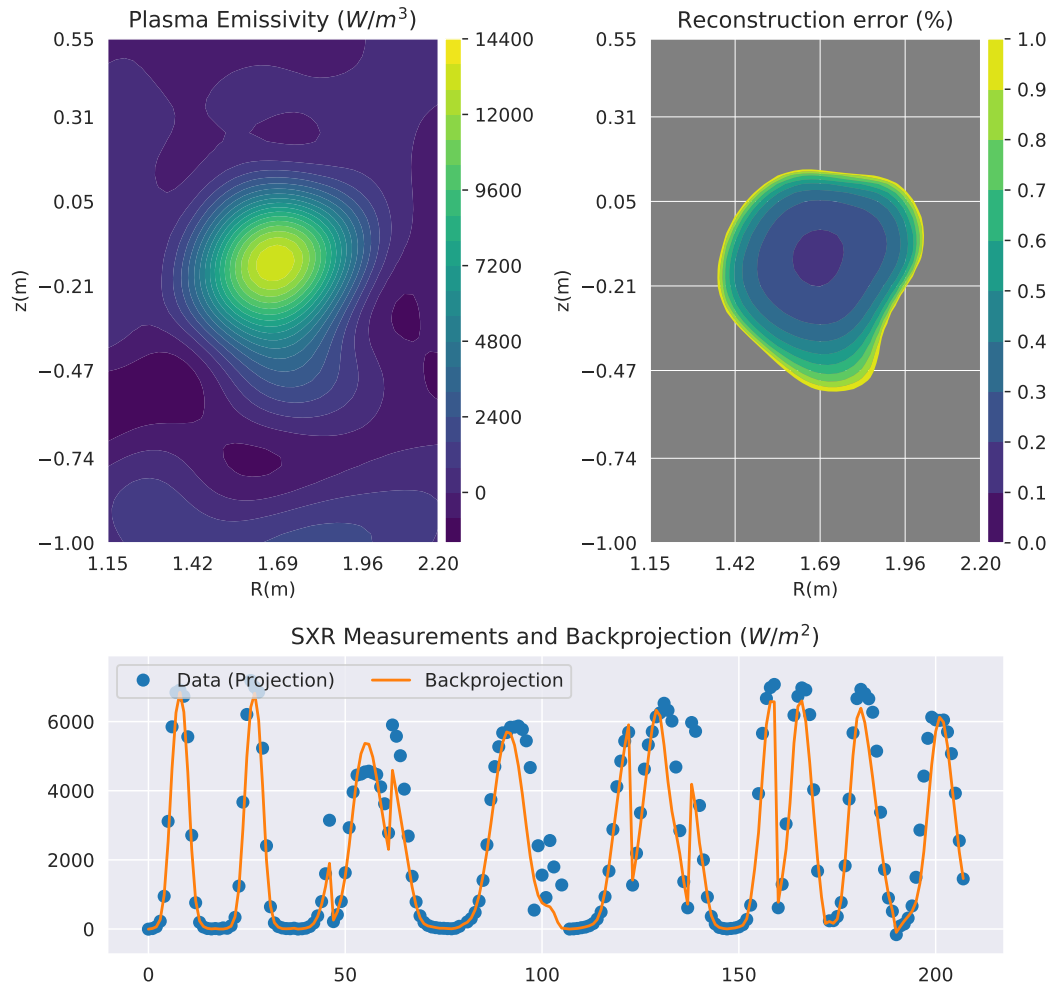


Figure 5.8.: Sample tomographic reconstruction and error, and comparison between the SXR measurement and the back-projected reconstruction, from ASDEX Upgrade shot #31238, $t = 3,2641$. The determined model hyperparameters by the classifier were $\theta_{err} = 1, 0, \theta_f = 2500, \theta_x = 0, 175$. 199 measurements (out of 208) were used for this reconstruction.

tions and the data.

To check this, we computed the percent deviation between the back-projections and the original tomographic projections, and did this for every measurement in every data point. We computed this value as

$$err = \frac{|backprojection - measurement|}{measurement} \times 100\%.$$

The histograms in Figures 5.9a and 5.9b show the results of this evaluation, up to a deviation of 100%, a threshold which covers 99,38% of the validation data. Note that the deviation was computed by comparing the back-projection only with valid (non-faulty) measurements. On average, 91.25% of the 208 measurements in each data point were used to compute the tomographic reconstructions.

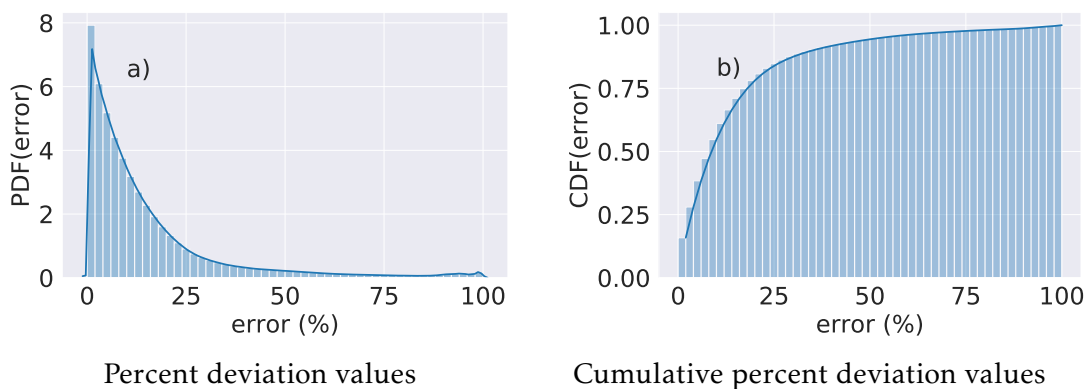


Figure 5.9.: Cumulative distribution of the deviations between the tomographic reconstructions' back-projections into measurement space, and the data. 54,4% of the individual back-projected measurements have a relative deviation lower than 10%; 93.83% have a deviation lower than 50%; and 99,38% have a deviation lower than 100%.

5.4.4. Discussion

Looking at the distribution of the deviations between backprojections and measurement data in Figure 5.9, one can see that most backprojected values have relatively low deviations from the data — 90% have a deviation lower than 50%. On the other hand, one can also see that some backprojected values have larger

deviations, and in fact, a few had an error greater than 100% (though they are not represented in the figure for ease of visualization; they represent only 0.6% of cases). We interpret this as a good result: on one hand, the low backprojection deviations indicate that the models chosen for the tomographic reconstructions have the necessary rigidity to constrain the solutions to mostly match the data, while at the same time being flexible enough to allow for large deviations, when not doing so would render overly complex models. We would like to point out the fact that many other types of models (other than the ones we used) can be chosen. For example, we used Euclidean distance and a single length scale for the covariance function. Nevertheless, it's possible to use more complex covariance functions with different length scales in the R and z directions, or with a distance metric that uses the radial and angular coordinates of pixels in the reconstruction grid, to account for the fact that points in the same flux surface are considered to be highly correlated. In principle, it's always possible to define more complex models that fit the data better, and reduce the deviation between projection and backprojection; nevertheless, those models will not necessarily have the highest evidence when compared with simpler ones.

5.5. Conclusions

Gaussian process tomography makes it possible to obtain the most likely estimate for an unknown, potentially infinite-dimensional, quantity, given some assumptions about the underlying physical distribution and about the data generated by that distribution. The tomography problem, based on SXR measurement data from the ASDEX Upgrade tokamak, lends itself to investigation under this framework. If one assumes a fixed model for the behavior of the underlying physical distribution (i.e. the plasma emissivity) and for the data, for example by specifying the length-scales involved in the emission process and the expected fraction of noise in the measurements, Gaussian process tomography (GPT) inversion techniques readily yield the corresponding maximum *a posteriori* estimate of the plasma SXR emissivity in the two-dimensional tokamak cross-section.

Nevertheless, this raises the issue of what models one would like to assume

in the first place. Through the Bayesian Occam's Razor principle, GPT answers this question by computing the evidence for different possible models, out of which the one with the highest score can then be selected. This can be useful if one wishes to test different assumptions regarding the data distribution: for example, what fraction of noise can be expected in the observations (measurements)? However, in a setting such as SXR tomography with ASDEX Upgrade data, this task can become cumbersome due to the dimensionality of the tomographic projections. This is further compounded when the number of models under evaluation is large.

For these reasons, we developed a novel method for automatic selection of the best model (out of 27 pre-defined ones) for the plasma SXR emissivity distribution and the corresponding data, for measurements from the ASDEX Upgrade tokamak. The individual models had different assumptions regarding the noise level in the collected data, the correlations between variables in the tomographic reconstruction grid, and the individual variances of those same variables. The method then consisted in training a convolutional neural network to perform the bayesian model selection (marginalization) procedure, and bypass the need to perform that task analytically. Our results show that the neural network achieved good classification results when compared to the analytical bayesian marginalization step, with top-5 accuracy (out of 27 possible classes) reaching a value of 0.976 (out of a maximum of 1). Furthermore, while the marginalization procedure across the entire dataset (of 127528 tomographic projections), through analytical methods, took approximately 48h, the same computation, performed by the neural network, took only 43s. Thus, the neural network approach can be particularly useful for high-dimensional data settings such as ours, as well as problems where the number of models under consideration is large, which would otherwise render the model comparison problem too slow through analytical methods. This can be particularly useful for settings where not only real-time inversion of tomographic profiles, but also real-time comparison of different models for certain physical distributions is a necessity.

Acknowledgements

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Data Availability

The data that support the findings of this study are available from the corresponding author upon request.

6. Classification of tokamak plasma confinement states with convolutional recurrent neural networks

Authors Francisco Matos¹
 Vlado Menkovski²
 Federico Felici³
 Alessandro Pau³
 Frank Jenko¹
Affiliations: ¹Max Planck Institute for Plasma Physics, Garching, Germany
 ²Eindhoven University of Technology, Eindhoven, Netherlands
 ³Swiss Plasma Center, Lausanne, Switzerland
Published in: Nuclear Fusion
DOI: <https://doi.org/10.1088/1741-4326/ab6c7a>

Author contributions: The first author carried out the work of code implementation and algorithm development. The remaining authors contributed significantly with discussions and suggestions, as well as reviewing and making additions and corrections to the paper. Federico Felici was responsible for getting the required data, and Alessandro Pau developed a tool for easing the data labeling process.

Abstract. During a tokamak discharge, the plasma can vary between different confinement regimes: Low (L), High (H) and, in some cases, a temporary (intermediate state), called Dithering (D). In addition, while the plasma is in H mode, Edge Localized Modes (ELMs) can occur. The automatic detection of changes between these states, and of ELMs, is important for tokamak operation. Motivated by this, and by recent developments in deep learning (DL), we developed and compared two methods for automatic detection of the occurrence of L-D-H transitions and ELMs, applied on data from the TCV tokamak. These methods consist in a Convolutional Neural Network (CNN) and a Convolutional Long Short Term Memory Neural Network (Conv-LSTM). We measured our results with regards to ELMs using ROC curves and Youden's score index, and regarding state detection using Cohen's Kappa Index.

6.1. Introduction

In a fusion experiment, plasma can typically be described as being in one of two different confinement regimes or modes: Low (L) and High (H). Furthermore, the plasma can also sometimes be described as being in a third, additional, mode, called the Intermediate or Dithering (D)[94] phase. In addition, when the plasma is in H mode, Edge Localized Modes (ELMs) can periodically occur.

Current tokamaks regularly run in H mode, which motivates the necessity for some measure of control (and therefore, detection) of ELMs and transitions between plasma modes. Furthermore, it is expected that future machines will also run in the same operating conditions[95]. Thus, the development of automated, data-based approaches to automatically detect the occurrence of certain events would be useful for both existing and future tokamak experiments and operation[96]. A detector would not only simplify and speed-up the post-experimental, offline analysis of shots, but also (ideally) detect ELMs and plasma state rapidly enough to allow for its usage in the real-time control systems of a fusion experiment, for purposes of plasma control and real-time discharge monitoring and supervision[97].

Due to uncertainties in the scaling laws, it is difficult to determine, *a pri-*

ori, when, during a discharge, a switch between different plasma modes will occur[98]. Nevertheless, physicists can usually pinpoint, through a post-experimental visual analysis of several diagnostic signal time-traces, at what point in time any transitions between different modes did take place. Similarly to transitions between plasma modes, the occurrence of an ELM can usually be pinpointed by looking at the time-traces of several diagnostics from a plasma discharge post-shot. Yet through an analysis of signals, some types of ELMs can be easily confused with dithers; a distinction between the two phenomena can not always be clearly made[99].

Although the identification by an expert, through post experimental visual analysis of signal time-traces, of a single ELM, or a single transition between plasma modes, is relatively straightforward for a typical shot, it becomes much more cumbersome to carry out that analysis effectively for many shots, especially when the associated time-series data is long, and when a shot has many transitions between different modes.

Recent advances in the ML field with the introduction of deep learning (DL) approaches deal with exactly such challenges. In the past years, the field of deep learning has brought about significant advances in Computer Vision and Sequential Data Processing. Convolutional Neural Networks (CNNs) have proven adept at localization, recognition and detection tasks in both 2-dimensional[100, 101, 102, 103, 104] and 1-dimensional[105, 106, 107, 108, 109, 110] data (i.e. signal analysis) in many different fields of science. In addition, Long Short-Term Memory (LSTM) Networks, which are one type of Recurrent Neural Network, have been successfully used for processing of sequential data where one expects correlations to exist across time, namely, automatic translation, natural language modelling[111], traffic analysis[112], and automated video description[113]. These tasks are much akin to what one can expect to find in terms of processing fusion shot data.

Given this, a deep learning approach is well motivated to address this challenge. Specifically, deep neural network models offer particular advantages when modeling high-dimensional data as given in this setting. In this work, we develop an approach for automatic classification of L-D-H plasma states and detection of ELMs based on two deep neural network models. The first model is based on a sliding-window feed-forward neural network, specifically a convolu-

tional neural network (CNN). The second model is based on a recurrent neural network (RNN), specifically a long short-term memory network (LSTM) with convolutional layers. The first model captures the local correlations within the windows to classify the transitions between plasma states from the shape of the signals. The second model extends this to capturing longer-term dependencies in the evolution of the states with the recurrent neural network layers.

We empirically demonstrate the approach on data collected from the TCV tokamak experiment, labelled by an ensemble of experts. The presented results demonstrate the effectiveness of the proposed model to detect the state and events of the plasma. We further discuss the trade-offs between increased precision and increased complexity of both models.

This paper is organized as follows: Section 6.2 discusses related work and Section 6.3 describes the physical phenomena being analyzed. Section 6.4 formalizes our problem, details the data we have available, and explains our decisions regarding how we model the data and design and train the neural networks. Section 6.5 gives an overview of the metrics we used to evaluate our results and our rationale behind using those metrics. Section 6.6 gives an overview of the results achieved, and we wrap up with a discussion in Section 6.7.

6.2. Previous work

Several different approaches for automated detection of events in plasma experiments exist. One such approach is to use threshold-based detectors. This corresponds to defining a point or series of points (in time) at which a signal surpasses a certain amplitude as corresponding to a detection[36, 37, 38], with additional constraints such as an increasing probability of the occurrence of an ELM as time passes since the last one. These approaches are limited to simple thresholding and cannot compute complex patterns in the data. Other work builds upon methods such as Kalman Filters to model the expected characteristics of the signal over a period of time, whilst also keeping track (in each time point) of the current plasma mode, according to a pre-defined model. In both of these cases, a detection algorithm's performance depends on the extent to which the theoretical assumptions and mathematical descriptions as to how

the signals should behave are correct, whether those assumptions are exhaustive (i.e., whether there may be additional causes which are unaccounted for), and whether some of those assumptions are more important than others; in other words, it is difficult to design an exhaustive rule-based system to detect the occurrence of transitions between plasma modes, as well as to detect ELMs.

The alternative is to use a purely data-based, supervised, Machine Learning (ML), approach, whereby a set of data, previously manually labeled by an expert (for example, through visual analysis), is used to train a detector. In this case, one does not specify which characteristics or correlations in the data are thought to correspond to the occurrence of an event; rather, it is expected that the algorithm can automatically learn what those correlations are, based on the labels, and then use the learned data features to make correct classifications on new data. Examples of such work are the usage of Support Vector Machines (SVMs)[39, 40, 41, 42] and Multi-Layer Perceptron (MLP) Neural Networks[114] on data from several tokamaks for detection of L-H transitions, classification of L and H modes, and detection of ELMs.

This type of scenario is, indeed, well suited for application of ML methods towards enabling automation. However, traditional ML methods such as SVMs and MLPs typically have limitations when faced with data with complex dynamics, such as the long sequences (i.e., signal time-series) present in this environment. SVMs typically depend on expert-defined feature engineering, which, while being superior to simple threshold-based detectors, is nevertheless insufficient when considering the complex data correlations which are observed in this setting. On the other hand, MLPs, while not requiring that sort of expert-defined input, are very inefficient when compared to modern deep learning models such as CNNs and RNNs, requiring much larger numbers of neurons and layers to perform the same task. These limitations are what motivate us to use deep learning approaches instead.

6.3. Background

6.3.1. Low, dither and high plasma confinement modes

When a discharge starts, the plasma is considered to be in Low (L) confinement mode. Once a certain threshold of input heating power to the plasma is reached[8], the plasma can spontaneously transition into High (H) confinement mode. Originally discovered at the ASDEX-Upgrade Tokamak[115], High (H) mode is nowadays regularly observed in almost all other machines[116]. H mode is characterized by the appearance, in the plasma edge, of a steep gradient in the electron density and the electron/ion temperatures, and a reduction in the transport of particles and energy. As a consequence of this edge transport barrier, the temperature and energy in the plasma core increase. When compared to L-mode, H mode allows for a larger amount of stored plasma energy per input power, thus rendering the fusion process more efficient. Yet the actual input power threshold that triggers the transition between the two modes is dependent on many factors, such as, for example, the configuration of the magnetic field, plasma density, and plasma size [98]. Furthermore, when the input heating power passes the aforementioned threshold but a change from L to H mode does not immediately occur, the plasma can be considered to be in a dithering (D)[94] phase. In this case, a temporary, weak, edge transport barrier starts to develop at the plasma edge, only to collapse and reappear in rapid succession[8]. These oscillations then repeat themselves until the plasma transitions into L or H mode. The localization of transitions into, and out of, D mode can, however, be difficult to identify, and there are often disagreements between experts as to which periods of a shot are in a Dithering phase [117].

6.3.2. Edge Localized Modes

When the plasma enters H mode, the corresponding accumulation of energy and the large pressure gradient at the plasma edge can trigger the occurrence of Edge Localized Modes (ELMs). These consist of periodic bursts of particles and energy which, if a long amount of time passes between successive ELMs, can impose a significant power load on the divertor, potentially damaging it. However, ELMs also allow for the periodic removal of accumulated impurities

from the plasma, and for a relaxation of the plasma density, which can otherwise increase as the H mode progresses, eventually triggering a disruption[118]. On the other hand, frequent, less energetic, ELMs lower the power load on the divertor, at the cost of reduced plasma confinement. Thus, tokamak operation requires knowledge of the occurrence of ELMs, in particular for larger machines where ELMs may cause deterioration of in-vessel components. Although several different types of ELMs exist, for the purposes of this work, we did not make any distinctions between them — we train the models to detect all occurring ELMs equally, regardless of their subclass.

6.4. Methods

6.4.1. Problem formulation and approach

To develop a model for this task, we formulate the problem as follows:

We observe a sequence of measurements x_t for $0 < t \leq N$ from the sensors for each shot. These observations are conditioned on the state of the plasma z_t at corresponding time t , where $z_t \in Z$ and $Z : \{ 'Low', 'Dither', 'High' \}$. Our goal is to find the most likely sequences $\hat{z}_{1:N}$ and $\hat{e}_{1:N}$ that explain the observations $x_{0:N}$. We define $\hat{z}_{1:N}$ as

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} p(z_1, z_2, \dots, z_N).$$

We choose to represent the joint probability $p(z_1, z_2, \dots, z_N)$ as

$$p(z_1, z_2, \dots, z_N) = \log(p(z_1|x_{0:1})p(z_2|x_{0:2}, z_1)\dots p(z_N|x_{0:N}, z_{N-1}))$$

where $p(z_t|x_{0:t}, z_{t-1})$ denotes the probability of observing state z at time t , given the sequence of observed signals x from time 0 to time t and the previous state z_{t-1} . This yields

$$\begin{aligned} \hat{z}_{1:N} &= \arg \max_{z_{1:N}} \log\left(\prod_t p(z_t|x_{0:t}, z_{t-1})\right) \\ &= \arg \max_{z_{1:N}} \sum_t \log p(z_t|x_{0:t}, z_{t-1}). \end{aligned}$$

Similarly, we define $\hat{e}_{1:N}$ as

$$\hat{e}_{1:N} = \arg \max_{e_{1:N}} p(e_1, e_2, \dots, e_N)$$

while representing $p(e_1, e_2, \dots, e_N)$ as

$$p(e_1, e_2, \dots, e_N) = (p(e_1|x_{0:1})p(e_2|x_{0:2})\dots p(e_N|x_{0:N}))$$

where $p(e_t|x_{0:t})$ denotes the probability of an ELM occurring at time t given the observations x from time 0 to time t . This then yields

$$\begin{aligned} \hat{e}_{1:N} &= \arg \max_{e_{1:N}} \log\left(\prod_t p(e_t|x_{0:t})\right) \\ &= \arg \max_{e_{1:N}} \sum_t \log p(e_t|x_{0:t}). \end{aligned}$$

To find $\hat{z}_{1:N}$ and $\hat{e}_{1:N}$ we develop two models. The first model is trained to detect the transitions between the different states of the plasma defined as $q_t \in Q$ where $Q : \{ 'Low \rightarrow Dither', 'Dither \rightarrow Low', 'Low \rightarrow High', 'High \rightarrow Low', 'Dither \rightarrow High', 'High \rightarrow Dither', 'NoTransition' \}$ and to detect the ELM events as $e_t \in E$ where $E : \{ 'ELM', 'NoELM' \}$.

We implement this model with a feed-forward CNN that processes a window of observations $x_{t-w}, \dots, x_t, \dots, x_{t+w}$ and produces a probability distribution over the transitions $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ and over the presence of an ELM $p(ELM_t | x_{t-w:t+w})$ at t .

We now define the probability of transitioning to z_t after being in z_{t-1} ($p(z_t|x_{0:t}, z_{t-1})$) with our model $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ where w is the number of observations around t , therefore:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} \sum_t \log p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$$

Practically, we implement the $\arg \max$ given above as a state evolution of a final state machine $S_t(z^{(a)} \rightarrow z^{(b)})$ where $z^{(a)}$ and $z^{(b)}$ are elements in Z and the transition probabilities are given by $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ at time t (see Figure 6.1). The evolution of the state machine produces several possible sequences of states,

and the one most likely to have generated the observed sequence of transitions can be found through an implementation of the Viterbi algorithm[119].

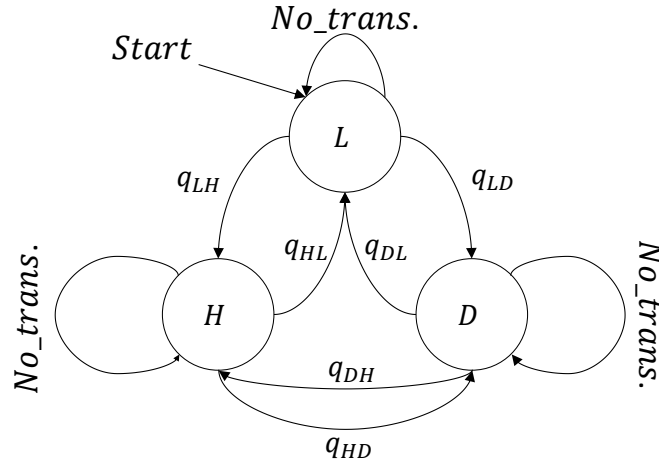


Figure 6.1.: State machine for processing of the CNN outputs

The first model can capture the localized correlations in the signals that indicate the transition of the state of plasma well. However, it is incapable of capturing the longer distance correlations that may be present in the signal. To generalize the approach further, we introduce a sequence model that models the full sequence of observations up to time t and produce a probability distribution $p(z_t|x_{0:t})$ for $0 < t \leq N$, as well as a distribution over the presence of the *ELMs* ($p(ELM_t|x_{0:t})$). This model is implemented by extending the CNN with a recurrent (LSTM) neural network. In this case, the model now observes a sequence of sliding windows $x_{t-w}, \dots, x_t, \dots, x_{t+w}$ for each t in the range $\{0, \dots, t\}$.

The first model has a lower computational complexity and can be trained more efficiently, as we only need windows of the signal with or without the different transitions, but it is limited to the information only present in the given window (see Figure 6.2). Increasing the size of this window that forms the context, increases the complexity both of the model and of dealing with multiple transitions appearing.

The second model addresses these challenges by modeling the sequence rather than a fixed window (see Figure 6.3). As a sequential model, it has an internal representation of the past observations x_0, \dots, x_t , that enable it to weigh-in the

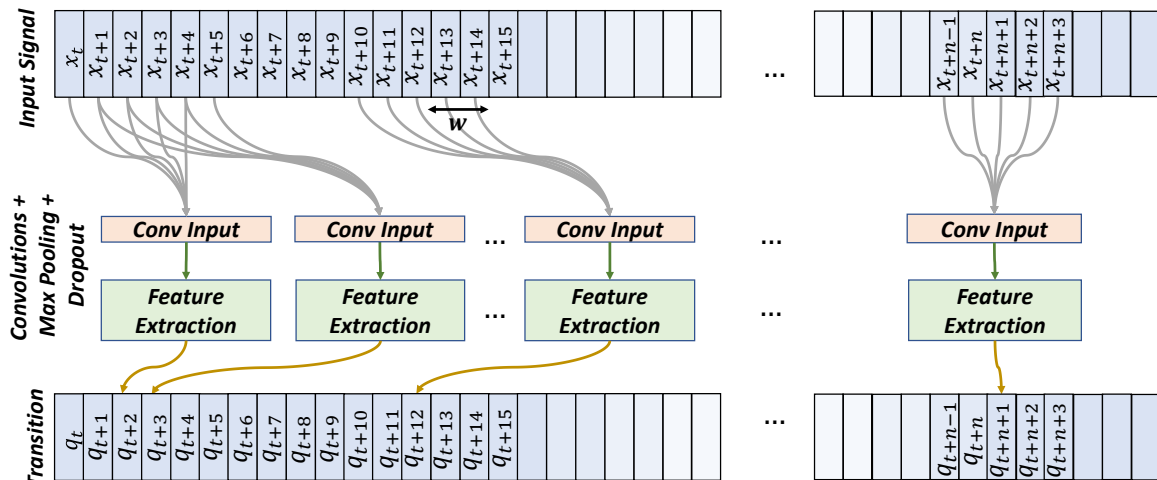


Figure 6.2.: Representation of how a CNN can be used to model the transitions between different plasma modes. The network’s output prediction for a time slice t depends only on the data features in a defined region immediately surrounding t .

likelihood of transition based on information in the more distant past[120]. The LSTM effectively assumes the role of the finite state machine and so the model can directly model the state of the plasma rather than the transitions. However, it introduces higher level of complexity, particularly for training, as we need to train on sequences rather than fixed-length windows.

6.4.2. Data and event features

For the purposes of this work, we have assembled a dataset based on the time-traces of four signals originating in the TCV tokamak[121, 122]. We opted, for the purposes of this work, to use the same, limited set of diagnostic signals that experimentalists use to determine, in post-shot analysis, the state of the plasma (Figure 6.4).

1. *Photodiode (PD) signal.* Corresponds to the measurements given by the photodiode diagnostic at TCV along a vertical chord, measuring the line-integrated emitted visible radiation; the photodiode has an H_α filter which measures radiation at 653.3 nm.

Transitions between different plasma states, as well as ELMs, can be most easily observed through analysis of the photodiode (PD) signal (Figure 6.5).

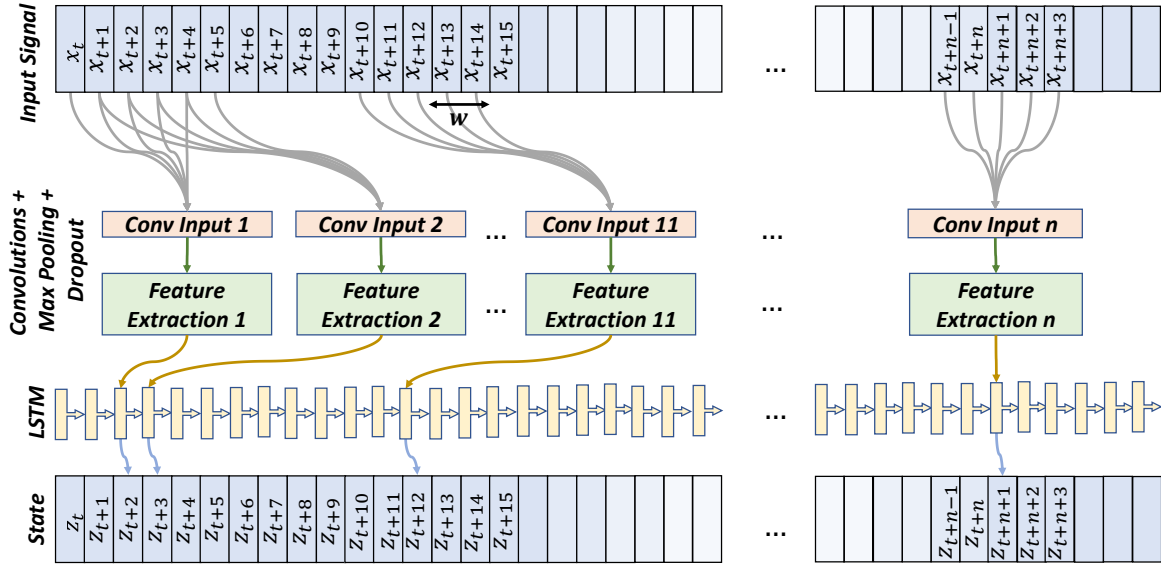


Figure 6.3.: Schematic representation of the flow of data inside a convolutional LSTM Neural Network. The network’s prediction (i.e. output probability) at any time t of a shot depends not only on whatever features the convolutional layers have extracted from the points immediately around t , but also on features extracted in the past.

Transitions from L to H mode are characterized by a sudden drop in the baseline value of the signal, whereas transitions back into L mode have the opposite trace, i.e., the baseline PD signal suddenly increases and remains at a steady level. ELMs are characterized by a sudden spike in the PD signal, followed by a relaxation that takes at most 2ms. D modes generate rapid fluctuations in the signal (see Figure 6.6); they do not necessarily correspond to a change in the baseline signal value, unless they are followed by a transition into a different state from the one at the point where they started.

2. *Interferometer (FIR) signal.* The interferometers at TCV measure the line-integrated electron density in the plasma along 14 parallel, vertical lines of sight. Of these, we take the mean value, per time instant, of the 12 inner-most detectors.

In the interferometer signal, the transition between L and H mode can most easily be seen as a sudden increase in the time derivative of the signal,

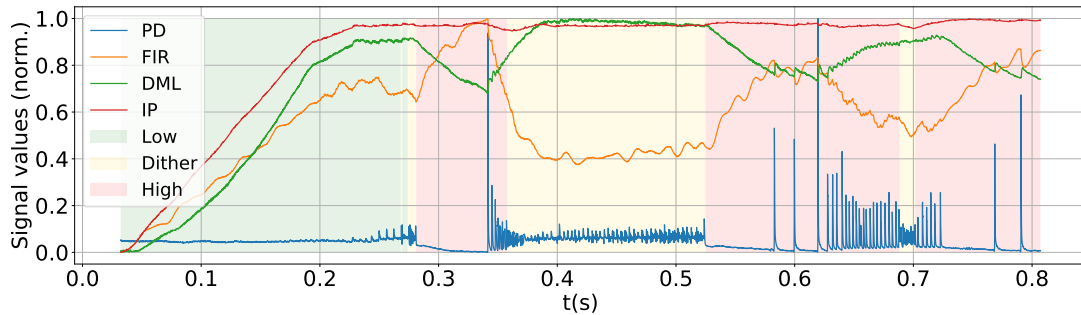


Figure 6.4.: Switches between different plasma modes(Low, Dither and High), and time-traces of the collected signals, TCV shot #32195

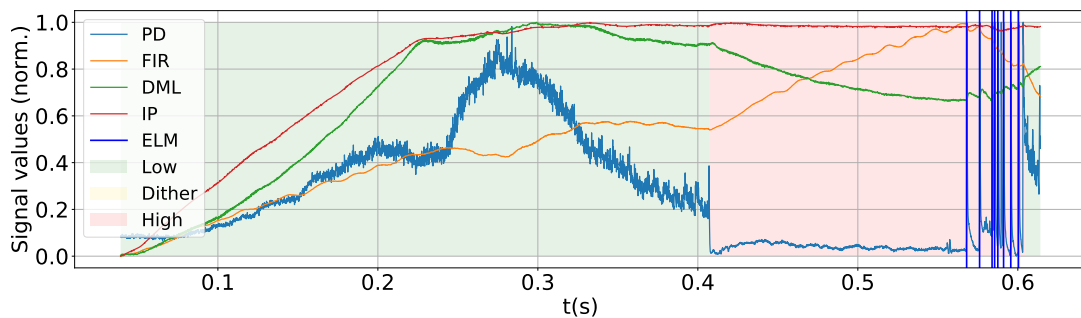


Figure 6.5.: ELMs and L and H plasma modes, TCV shot #33446

while transitions back into L mode correspond to a decrease in the derivative. Similarly to what happens with the photodiode signal, ELMs may provoke short (albeit less pronounced) spikes in the FIR signal.

3. *Diamagnetic Loop (DML) signal.* Refers to the measurement of the total toroidal magnetic flux of the plasma[123]. The derivative of the DML signal frequently switches signs when a transition occurs between L and H mode, as well as when an ELM occurs (Figure 6.7). Furthermore, the sign of this signal's derivative changes depending on the sign of the plasma current.
4. *Plasma Current (IP) signal.* Refers to the total plasma electric current. For this work, we use the current value to determine when the actual classification of plasma states should begin. Specifically, we ignore, for classification purposes, time points where the absolute value of the current is lower than

50 kA.

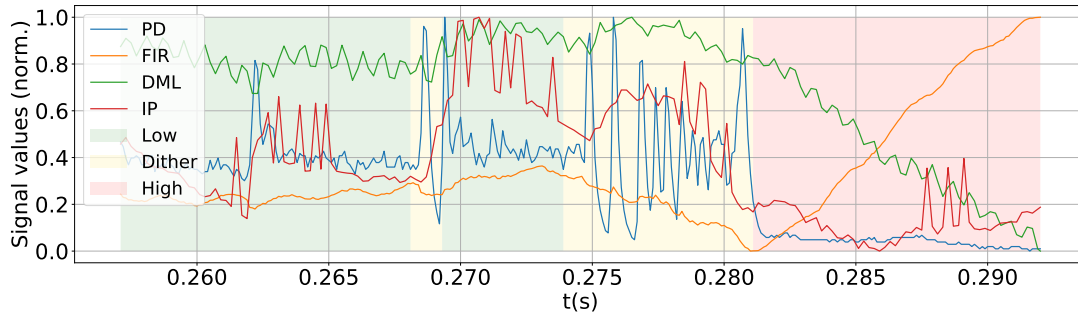


Figure 6.6.: L, D and H modes from a section of TCV shot #32195.

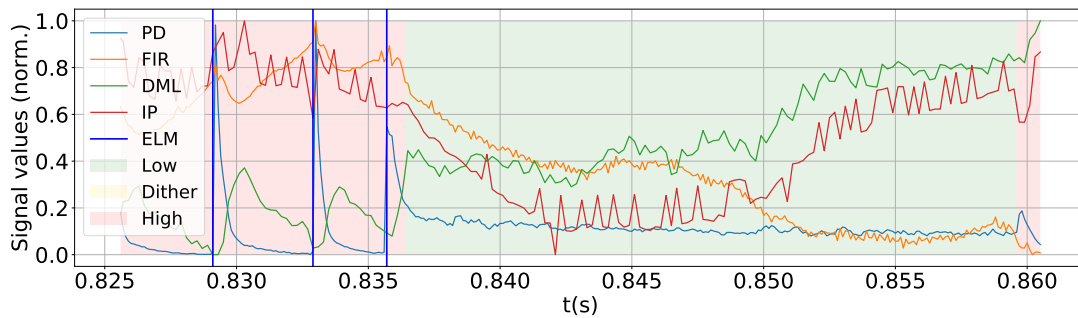


Figure 6.7.: ELMs, and L and H modes from a section of TCV shot #31650.

The 4 different signals used for this work have different sampling rates. As a first step, we resampled all of them to the same frequency of 10kHz. Since each TCV shot is usually up to 2 seconds long, this means that our shot signal data consists of time-series of up to about 20.000 time slices.

We want to train a classifier to recognize features in the data which allow for detecting the occurrence of ELMs and transitions between different plasma modes — i.e., a supervised learning task. As such, the first step was to collect labels for each shot time series, through visual analysis taking into account the data features described above. The collected data was visually labeled by 3 different experts for the same shots. This means for some shots, the same regions might have different labels (namely, the experts might disagree on whether a certain part of a shot is dithering). Training the network with labels which are

different in some regions has several potential advantages. For example, it compensates for any possible discrepancies in labeling originating from human error. It also allows us to incorporate the uncertainty in the labels into the network training process itself, that is, it acts as a form of regularization: if there is no majority agreement between experts regarding a section of a shot, then it is to be expected that the network will also learn not to strongly favor any class in that region. Conversely, if the three experts agree, then the network will learn that the features in that region most certainly correspond to a certain class, which renders the classification more robust. Finally, getting labels from different experts allows us to increase the size of our training dataset.

6.4.3. Model training

The two proposed models develop different maps. The first model is a map between a fixed window of observations and a distribution over transitions, while the second models a sequence of observations and produces a sequence of states (see Figure 6.8).

Accordingly, the training data has different arrangements. For transition classification, we need to prepare a dataset $D_1, \{(x, q, e)\}$, where a training point consists of a section of the recorded signal $(x_{t-w}, \dots, x_t, \dots, x_{t+w})$, the corresponding label of one of the transitions q_t in Q and the matching label e_t indicating the presence (or not) of an ELM. Figure 6.9 illustrates this in detail.

For the second model $D_2, \{(x, z, e)\}$, a training point consists of a sequence of windows of observations drawn from x_t to x_{t+l+w} (where l is a defined sequence length, and w is the window length), a sequence of state labels z_t in Z of length l , with each label corresponding to the state of the plasma at times t , and a sequence of labels e_t of length l corresponding to the presence of an ELM at times t . Figure 6.9 illustrates this in detail.

There is an inherent uncertainty in the labeling of the ELMs and plasma states, particularly when it comes to transitions into and out of dithers. The raw data only has hard, binary, one-hot encodings[124] — that is, a transition between two states, for example, is labeled as a sudden switch (from one time slice to the next) from one state to another. This means that it is easy to mistakenly label an event or transition in a slightly shifted time slice. This type of hard threshold

also makes it difficult for a neural network to generalize to outside of its training set[125].

Therefore, for the first model (CNN), we process the target time-series such that the probability of an ELM, or of a given state transition, is a continuous value, starting at zero and peaking at one, with several intermediate probabilities. In practical terms, we apply on each event a gaussian smoothing such that, if an ELM or state transition occurs at time t , its probability at that point is 1, and we define an interval Δt — before and after t — where the probability, respectively, smoothly increases and decreases. We defined these smoothing intervals as corresponding to 2ms, which, at the defined sampling rate, translates to 20 time slices. We do the same with the states z_t for the second model (ConvLSTM), such that a switch between two different states, from z_1 to z_2 , does not happen immediately from one time slice to the next, but rather, the probability of z_1 decreases, while that of z_2 increases, over a span of 20 time slices.

This procedure not only models the uncertainty in the labeling process, but also acts as an automatic regularization for the neural network training process, i.e., it makes it easier for a neural network to generalize what it learns to unseen data[126].

The choice of the size of the temporal windows with which the CNN is trained is a trade-off between the assumptions made about the data, and computational feasibility. Larger windows contain more spatial information and thus, intuitively, should make the classification at a particular time slice more precise, but also make the training and inference process by the network slower. Smaller windows contain arguably less information, but can be processed faster. We opted to train the CNN with temporal windows with a length of 20ms, which we judged to be a good compromise between those two requirements. At our sampling rate, these windows are 200 time slices long. This is illustrated in Figure 6.9: the green region represents a window of signals (in this case, only the PD signal) which is fed to the neural network, and its associated target, which is the probability of an ELM occurring at $t = 0.304s$. There is an offset between the time at the window's rightmost edge, and the time for which the probability is computed; in the example of 6.9, the offset is of 2ms, which means that to detect the ELM occurring at $t = 0.304s$, the window would have information on the signals from $t = 0.286s$ to $t = 0.306s$. In formal terms, the windows compute in

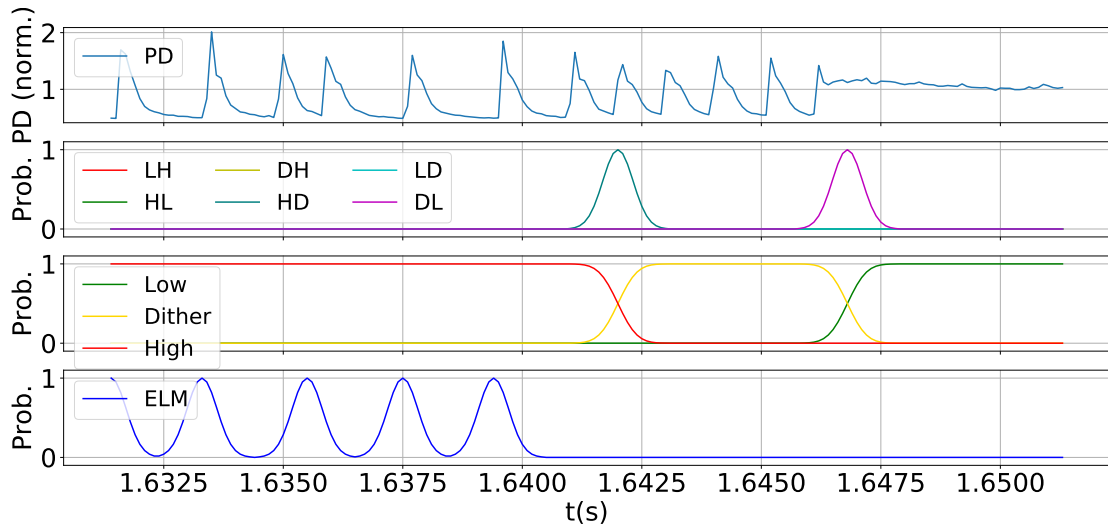


Figure 6.8.: Representation of the different types of encoding of the target “smooth” data distributions, to be learned by the two classifiers, from TCV shot #30262. Here, we show only the labels produced by a single expert, though the networks are trained with labels from all of them. The second plot from the top illustrates the transitions to be learned by the CNN, while the bottom-most plot illustrates the states to be learned by the Conv-LSTM.

that case $p(e_t|x_{t-w_1:t+w_2})$ and $p(q_{z_{t-1} \rightarrow z_t}|x_{t-w_1:t+w_2})$, where $w_1 = 180$ and $w_2 = 20$. In practice, in a real-time setting, that offset would constitute a minimum delay between the occurrence of an event in a machine, and a detection by the classifier. Once again, the size of this offset is a trade-off: a smaller offset is ideal for real-time applications because it gives more time for feedback control mechanisms, but it also contains less information for the network to accurately classify an event.

We train the Conv-LSTM not with windows, but with sequences of windows. The distinction is an important one, for it implies different assumptions about the data. In the case of the windows fed to the CNN, it is assumed that each window is independent of each other. In the data fed to the Conv-LSTM, each sequence itself is composed of several windows, with future windows depending on past ones. We defined each of those sequences to consist of 200 windows (since that was also the length of the windows fed to the CNN). In this case, each of the individual windows has a length of 4ms (40 time slices), with an

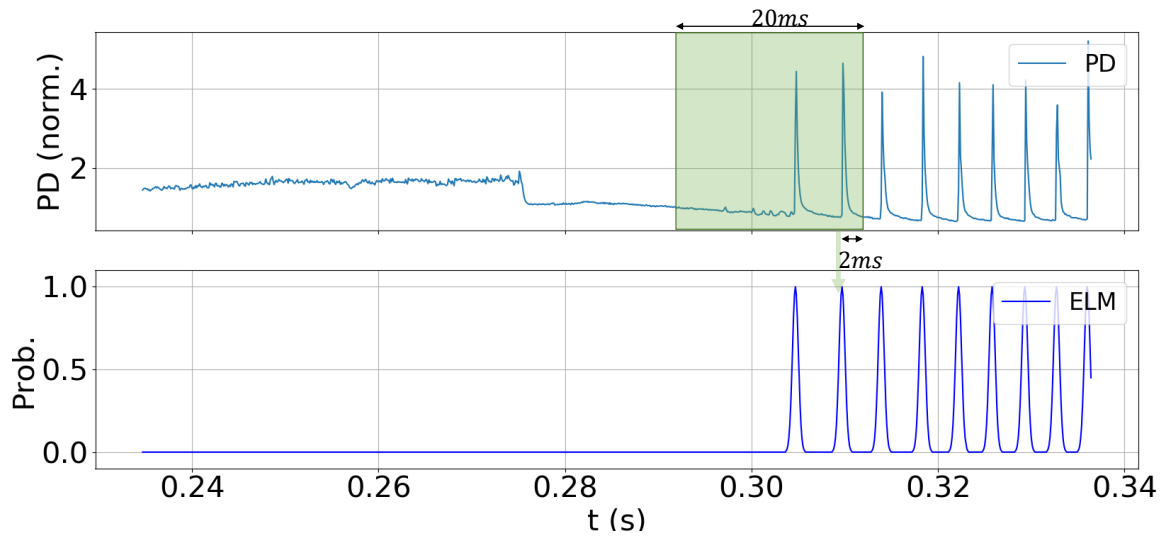


Figure 6.9.: Representation of the sliding temporal windows fed to the CNN on top of the PD signal, and their corresponding ELM probability output. At inference time, these windows slide over the 4 signals across the whole shot, each of them rendering an output probability for a single time slice.

offset of 2ms, as in the data for the CNN (see Figure 6.10). The sequences have a stride[127] of 1: each window starts and ends exactly 1 time slice after the previous one finishes. Each of these sequences is randomly subsampled from the whole shots, and the corresponding targets for them are chosen randomly from one of the three labelers.

Although not all of these subsamples start in L mode, our expectation is that the network would learn by itself that an actual shot always begins in that state. There are several reasons for this. First, the network will learn to recognize any features in the subsequences that are consistent with the beginning of a shot, and learn that those features correlate to L mode. Second, even if some training sequences start in D or H mode, the network will statistically learn that these modes are more frequently the result of a transition from a previous mode.

6.4.4. Model design

The architecture of the neural networks used for the transition detection starts with a 1-D convolutions with four channels, each of which receives the values

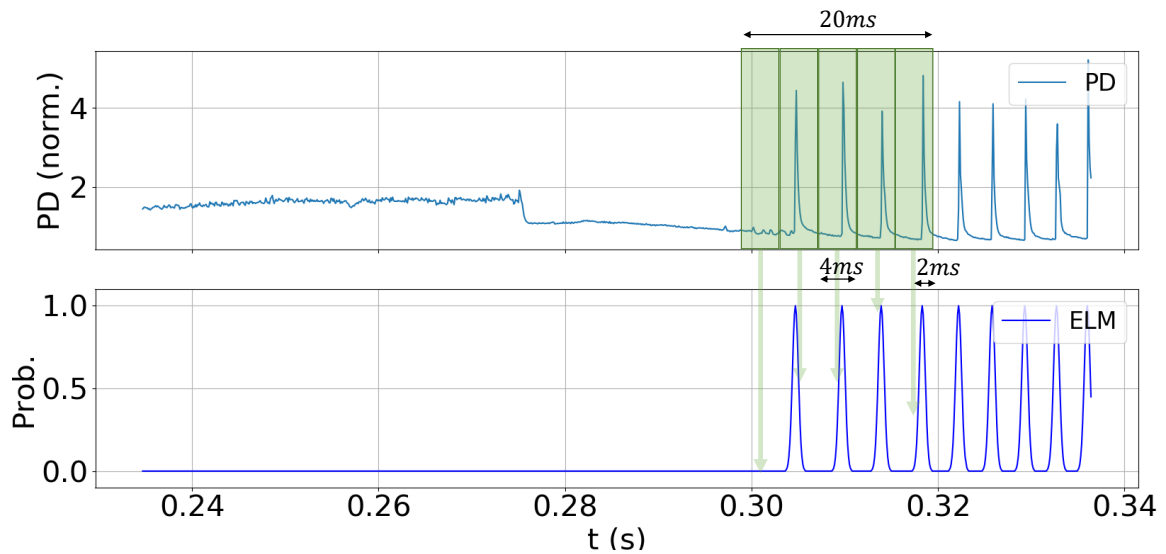


Figure 6.10.: Example of a sequence fed to the LSTM. At a 10kHz sampling rate, it consists of 200 overlapping temporal windows of length 40. The output probability for a given window depends not only on what data features are present in that window, but also on the past windows in the sequence.

from the PD, FIR, IP and DML signals. These are followed by several convolutional layers, interspersed with pooling and dropout layers, which are trained for feature extraction, with deeper layers extracting higher-level data features (Figure 6.11). The last layers of the network are fully-connected, and are responsible for receiving the pre-processed high-level features and producing an appropriate output for them, i.e., the desired classification. This model is loosely inspired by the VGG architecture for classification of images where fixed sized filters are used[52].

Our convolutional LSTM network builds on top of CNN model that showed the best performance on the transition detection task. We add a recurrent layer that processes the output of the CNN to capture the longer-distance correlations in the data (Figure 6.12).

We designed the networks using the Keras framework for deep learning[89]. Both networks used a categorical cross-entropy loss function, and were trained with the Adam optimizer[92] using the default learning rate value provided by Keras.

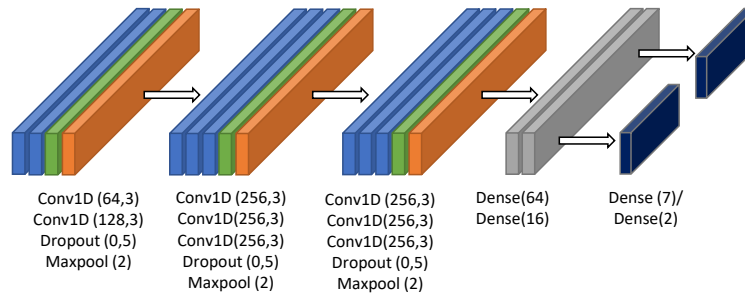


Figure 6.11.: Architecture of the convolutional NN

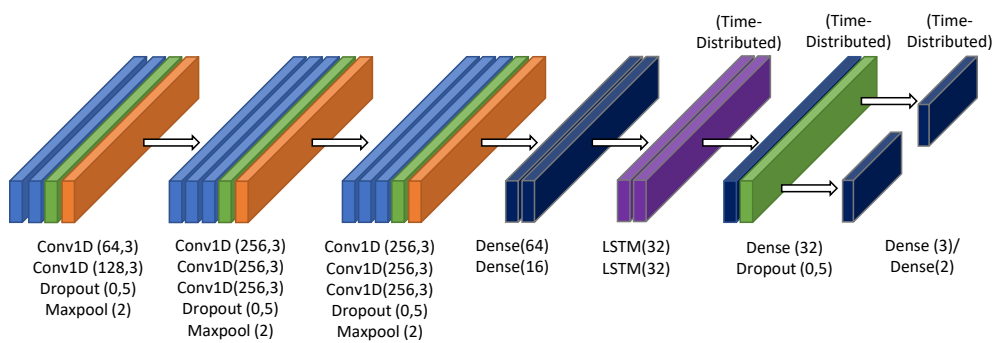


Figure 6.12.: Architecture of the convolutional LSTM. All layers and nodes use ReLU activation functions, apart from the final output layer, which uses Softmax activation.

6.4.5. Data split

In total, we possessed 54 shots fully labeled by the three experts. In a typical deep learning setting, some sort of normalization[128] is usually applied on the available data. The most common procedure would have been to normalize across the entire dataset. However, because of the different calibrations of the PD signals and the subsequent large variance and multimodal distribution associated with it, we decided, at this stage, to normalize each shot separately dividing each signal in each shot by its own mean across the whole shot. For potential real-time applications, as any new shots could fall outside the normal-

ization range, the procedure would require grouping and normalizing the shots with respect to different signal gains and calibrations.

From these normalized full sequences, we draw batches of smaller temporal windows and subsequences to train the neural networks. There are several reasons for this subsampling. First, the full shot time-series are up to about 20,000 time slices long, but the actual length of a shot can vary significantly. Yet for purposes of training the networks, we require batches of data of fixed length, which can be achieved by subsampling from the full sequences.

Second, this method allows us to automatically perform data augmentation for training, since one long sequence will contain many shorter subsequences and windows.

Third, feeding very large temporal windows to a CNN would be computationally difficult, as the number of network parameters requiring training would grow considerably.

Finally, the distribution of the data in the full sequences is highly unbalanced: in most shots, dithering phases are significantly shorter than L and H phases; only a few dozen transitions happen at most per shot; and, some transitions tend to be more frequent than others. Training with whole sequences would significantly bias the networks towards the events and transitions that occur more frequently in the labeled data. Drawing subsequences allows us to control the data fed to the network such that this inherent bias is mitigated. To do this, the training data batches must be balanced, i.e., generated such that they contain roughly equal fractions of the different types of events and/or transitions of interest. In the CNN, there are 8 possible events of interest — LH, HL, HD, DH, LD, DL, ELM, and no transition. Generating batches for the CNN means that, for a batch containing n data samples, $n/8$ of those samples will correspond to each of those different types of transitions. Similarly, for the Conv-LSTM, the batches are generated such that the three target distributions (L, D and H) correspond to approximately $1/3$ of the data samples each.

6.5. Evaluation metrics

6.5.1. ROC curve

We consider the detection of single, discrete ELMs by the networks as corresponding to a point in time (in a shot) where the direct network outputs for ELM probability $\hat{e}_{1:N}$ reach a maximum value. This is not necessarily a point where the output network probability for ELM is 1, but rather, a point t where the output probability $P(ELM_t)$ follows a series of strictly increasing probability values, and precedes a series of strictly decreasing ones. Because we defined the length of the gaussian smoothing of the probabilities as 20, here we consider a local maximum for $P(ELM_t)$ within a 20-wide interval to correspond to the detection of a single ELM — which we denote as a positive. The remaining points are considered non-detections, i.e., negatives. In addition, we defined different probability thresholds for what can be considered a detection of an ELM by the network. For example, defining a threshold of 50% implies that only ELM probability maxima above that threshold are considered positives.

Positives and negatives must then be compared to the labeled ELMs. To that end, we build the ELM Confusion Matrix, which defines several variables: negatives that match their label at the same point in time are True Negatives (TN), while those that do not are False Negatives (FN). Similarly, positives that match their label are True Positives (TP) and those that do not are False Positives (FP).

Using this method to determine the points in which the network detects individual ELMs, one can then compute the True Positive Rate (TPR) and False Positive Rate (FPR) for different detection thresholds:

$$TPR = \frac{TP}{TP + FN} \quad (6.1)$$

$$FPR = \frac{FP}{FP + TN} \quad (6.2)$$

Plotting the TPR versus FPR for a series of different detection thresholds yields the classifier's ROC curve[129], which illustrates the network's capacity for discrimination given different detection thresholds. There are several ways to com-

pute the ideal detection threshold based on the ROC curve, depending on the task in question. In our case, we use the Youden index[130], whereby the best threshold is the value which maximizes the difference $TPR - FPR$, the maximum value being 1.

6.5.2. Kappa statistic

To compare the models' accuracy with that of the human labelers, we use Cohen's Kappa-statistic coefficient, which measures agreement between two sets of categorical data[131], defined as

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \quad (6.3)$$

where p_0 denotes the actual relative agreement between the two sets, and p_e denotes the probability of the two sets randomly agreeing with each other. Generically, the κ coefficient's values oscillate between 0 and 1, the former indicating poor performance, and the latter indicating perfect performance. In our case, given two sequences z_1 and z_2 of plasma states, Kohen's Kappa measures the overlap between them. If $z_{1_t} = z_{2_t}$ for all time instants t , the metric will yield a score of 1; if there are mismatches between the two sequences, the score will go down.

The κ -statistic can be interpreted differently based on the sections of the data for which it is computed. For that reason, we will now define several variables that allow us to interpret the κ -statistic scores.

Remember that we possess labels drawn from three different experts; as such, generically, labeled shot states at each point in time t of a shot can be in one of three possible categories:

- No majority agreement, i.e., all labelers disagree as to what state the plasma is in, which we denote as category C_1 .
- Majority agreement, i.e., two labelers agree on the state of the plasma, while one disagrees, which we denote as category C_2 .
- Consensual agreement — all labelers agree as to what state the plasma is in, which we denote as category C_3 .

We define the union of C_2 and C_3 as *ground truth* (C_4), i.e., they are sections of shots where there is at least a majority opinion as to what state the plasma is in. We also have, for each shot, the most likely sequences $\hat{z}_{1:N}$ of states (given the observed data) produced by the neural networks, which we will now denote as C_5 .

Computing the κ -statistic score, κ_l , between sets C_2 and C_4 gives an indication of the probability that a single labeler disagrees with the ground truth: a κ_l -score of 1 would indicate that there is agreement between all the labelers all the time, while a lower score would indicate that at least some of the time, one labeler disagrees with the others. Simultaneously, computing the κ -statistic score between sets C_5 and C_4 (κ_n) gives an indication of the networks' performance given the ground truth. But, in addition, we can directly compare κ_l and κ_n . This comparison allows to test how a network and a single labeler compare against each other, on average, given the ground truth.

The κ -coefficient is calculated separately for each of the three possible labels for the plasma state (L, D and H), and as a weighted mean across all three states. The weights of that mean are taken to be the relative frequencies of each individual state in the dataset, based on the ground truth (C_4) labels.

6.6. Results

We performed several training runs using the data labeled by the three experts; we carried out experiments where we trained both models (CNN and ConvLSTM) three times, each time randomizing the training and test shots, to test whether differences in the data could lead to different results. In a typical deep learning setting, the data is usually split so that approximately 80 – 90% is used for training, and 20 – 10% is used for validation of the results, i.e., testing the network's capability to accurately predict on data that was not used for training. In our case, we opted for a training/test data split of 50%, i.e., of the 54 shots, we used 27 for training and 27 for testing. The results that follow are the best results of those three experiments, for each model. We also experimented with varying offsets (see Figure 6.9) for the convolutional windows to see what effect that factor could have on the results; we settled for an offset value of $2ms$

(20 time slices), as smaller offsets degraded results, while larger ones did not improve them. We computed the metric scores on the training and test data at several points during training to control for overfitting[44], and present the results from the epoch where the state detection results on test data were the highest. We ran the neural networks on an NVIDIA Quadro RTX 5000 GPU.

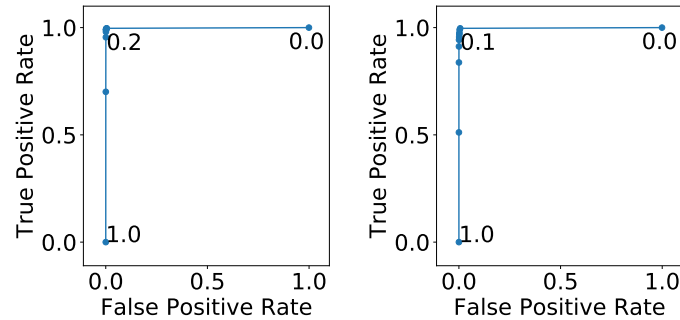
6.6.1. CNN

We computed the κ -statistic based on the regions defined in Subsection 6.5.2 — that is, we compute scores based on the network output versus the ground truth (κ_n), and based on labeler disagreement versus the ground truth (κ_l). We computed the scores on a per-state (L, D and H) basis, and also computed a mean of the values obtained for each state.

We trained the CNN for 250 epochs, allowing for the loss function to stabilize; each epoch consisted in 32 batches, with each batch containing 64 data samples. Upon completion of training, we tested the CNN’s accuracy on both the training and test data. The model’s results on ELM classification (ROC curve) can be seen in Figure 6.13. Table 6.1 shows the scores κ_n and κ_l for the entire dataset, while Figure 6.14 contains histograms showing the κ_n s distribution on a per-shot basis.

		L	D	H	Mean
κ_n	Train	0.691	0.358	0.657	0.649
	Test	0.219	0.115	0.157	0.182
κ_l	Train	0.937	0.896	0.987	0.958
	Test	0.941	0.848	0.986	0.962

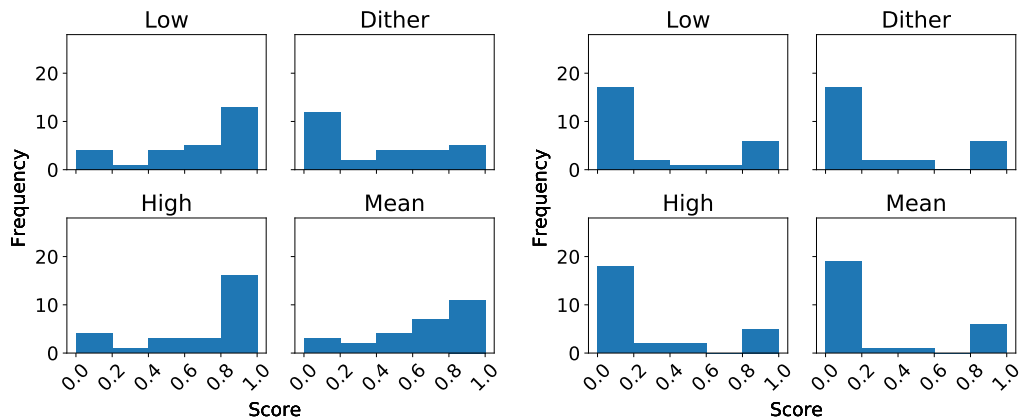
Table 6.1.: κ -statistic scores (κ_n and κ_l) for each plasma mode and as a mean, on training and test data (values across all shots), for the CNN



(a) Training data.

(b) Test data.

Figure 6.13.: ROC curves for ELM detection for the CNN model. The detection thresholds that maximize the Youden index are 0.2 and 0.1 for training and test data, respectively yielding index values of 0.993 and 0.99. Using the ideal threshold for the training data (0.2) on the test data gives a slightly lower Youden index of 0.986.



(a) Training data.

(b) Test data.

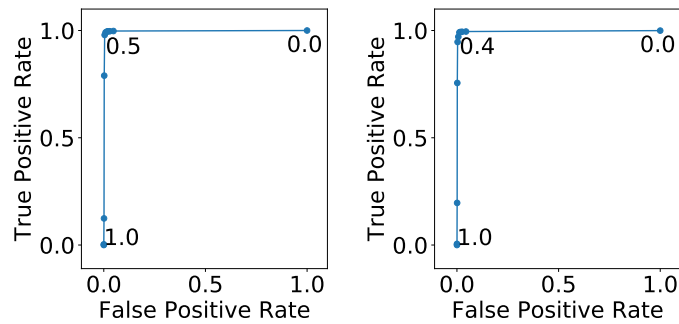
Figure 6.14.: Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the CNN.

6.6.2. Conv-LSTM

We trained the convolutional LSTM for 400 epochs, allowing the loss function to stabilize. Each epoch consisted of 64 batches, with each batch containing 64 data samples. The results of computing scores κ_l and κ_n , using the same definitions as for the CNN, can be seen in Table 6.2. The ROC curves detailing the results on ELM detection can be seen in Figure 6.15. Figure 6.16 contains histograms showing the score K_n values on a per-shot basis.

		L	D	H	Mean
K_n	Train	0.96	0.889	0.967	0.96
	Test	0.82	0.766	0.85	0.832
K_l	Train	0.96	0.94	0.992	0.98
	Test	0.901	0.808	0.98	0.935

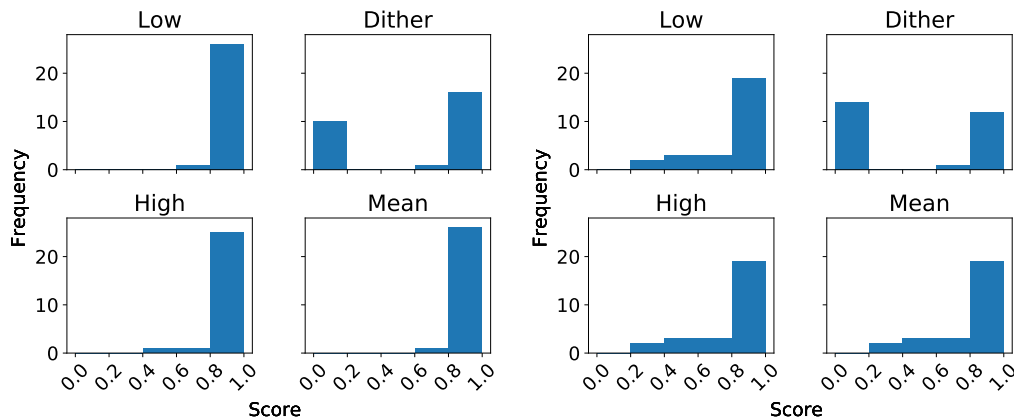
Table 6.2.: κ -statistic scores (κ_n and κ_l) for each plasma mode on training and test data, for the Conv-LSTM



(a) Training data.

(b) Test data.

Figure 6.15.: ROC curves for ELM detection for the Conv-LSTM model. The detection threshold which maximizes the Youden index is 0.5 for training and 0.4 for test data; this yields index values of 0.977 and 0.969 for each set respectively. Using the ideal threshold for the training data (0.5) on the test data gives a slightly lower Youden index of 0.95.



(a) Training data.

(b) Test data.

Figure 6.16.: Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the Conv-LSTM

6.6.3. Discussion

A comparison of the κ_n scores on training and test data for each classifier shows that the convolutional LSTM performs better than the CNN for all three plasma states. Furthermore, looking at the distribution of the mean κ_n scores on a per-shot basis through the histograms, one can see that the worst Conv-LSTM classifications do not have a score lower than 0.6 on training data, while for the CNN alone, even on training data, mean κ_n scores lower than 0.2 exist. For both classifiers, the performance on training data surpasses that on test data, both on a state-by-state basis, and as a mean across all states, which indicates the occurrence of overfitting.

For both networks, an analysis of the κ_l scores of their training and test data indicates that human labeler disagreement is highest for dithers — the scores for that particular state are consistently lower. Interestingly, both networks also score their lowest results for dithers.

Comparing the Conv-LSTM's κ_l and κ_n scores shows that, at least on training data, the network behaves, on average, similarly to a single human labeler, making errors (or disagreeing with the ground truth) at approximately the same rate — the mean κ_l score for training data is 0.98, while the mean κ_n score for training data is 0.96. On test data, the Conv-LSTM performs slightly worse than a single human labeler, as seen by the fact that the network's mean K-index score on test data κ_n is 0.832, while κ_l is 0.935.

As measured by the Youden index, we achieve excellent performance in detection of ELMs on both training and test data using both models; the ideal detection thresholds generate true positive detection rates very close to 1, while bringing false positive detection rates essentially to 0. The Youden indexes for test data are only slightly lower than for training data, which suggests that overfitting is minimal. Furthermore, for both models, on both training and test data, the ROC curves' points are mostly concentrated close to True Positive Rates of 1 and False Positive Rates of 0, which indicates that the choice of ELM detection threshold does not significantly change the behavior of the classifiers.

Finally, the scores for ELMs being essentially the same for both models indicates that the features in the data which allow for identification of ELMs are mostly local: the CNN, even without knowledge of long-term temporal correla-

tions, performs excellent classification.

Because the Conv-LSTM has highest κ_n scores, we made a case-by-case analysis of that network's classification of all our available shots. Broadly, the Conv-LSTM's results on state detection, on a per-shot basis, can be placed into six different categories:

1. A (sometimes very) short detection, of a dither that is not labeled in the data. Due to the way the K-score κ_n is computed, a mistaken dither classification by the network of a single time point (in a whole sequence), in a shot which has no regions where the ground truth (C_4) is dithering, will bring the score for that state down to 0, even if the remainder of the shot is correctly classified (17 shots).
2. A clearly incorrect classification, of a temporal region of a shot as being in a dithering state (4 shots);
3. A missed detection of an L-H transition (1 shot);
4. A missed detection of an H-L transition (2 shots);
5. An overall bad detection across an entire shot (7 shots);
6. An overall good detection across an entire shot (23 shots).

Table 6.3 lists 6 shots which are representative of each of the types of results listed above. The table shows the computed κ_n scores for each of those shots on a per-state basis, as well as the score's mean value, and the fraction of time, for the ground truth of each shot, that a particular state is labeled. The table also lists which of the 6 cases above the shot is representative of. Figures 6.17, 6.18, 6.19, 6.20, 6.21, and 6.22 are plots of those same shots, where the background color in the top plot denotes the state detected by the Conv-LSTM, and in the bottom plot, denotes the ground truth label. Small gray areas in the bottom plot denote regions where ground truth is not defined, i.e., there is no majority agreement between labelers.

Case	Shot ID	L		D		H		Mean
		Fraction	Score	Fraction	Score	Fraction	Score	
1	57751	0.756	0.97	0	0	0.243	0.97	0.97
2	34010	0.679	0.856	0.073	0.232	0.248	0.602	0.748
3	58182	0.22	0.912	0.095	0.969	0.685	0.927	0.928
4	30197	0.951	0.384	0	1	0.049	0.384	0.384
5	33459	0.811	0.662	0	0	0.189	0.846	0.697
6	33942	0.455	0.953	0.183	0.884	0.412	0.997	0.962

Table 6.3.: κ -statistic (κ_n) scores for each plasma mode on training and test data for selected shots representative of each of the six result categories

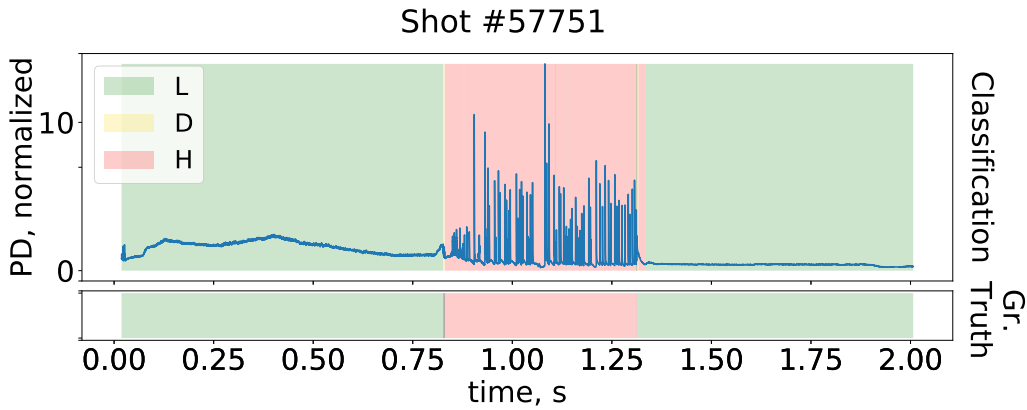


Figure 6.17.: TCV shot #57751 (PD signal) and the Conv-LSTM's classification of state as the shot evolves. Notice the (very short) detected dithering phase shortly after $t = 0.75$: no dithers are present in the labels, so the score for D is 0.

6.7. Conclusions

We have developed two deep learning-based classifiers to perform automatic detection of ELMs and classification of plasma modes. The task was two-fold: on one hand, to perform a binary classification, for each time slice of a plasma shot, on whether an ELM is occurring or not; and, to automatically determine which plasma mode (or alternatively, whether a transition between plasma modes) is occurring. One approach is to use a convolutional Neural Network (CNN), which uses only local correlations in data to perform classification. The second approach uses a Convolutional LSTM (Conv-LSTM) neural network, which also takes advantages of long-term temporal correlations in data.

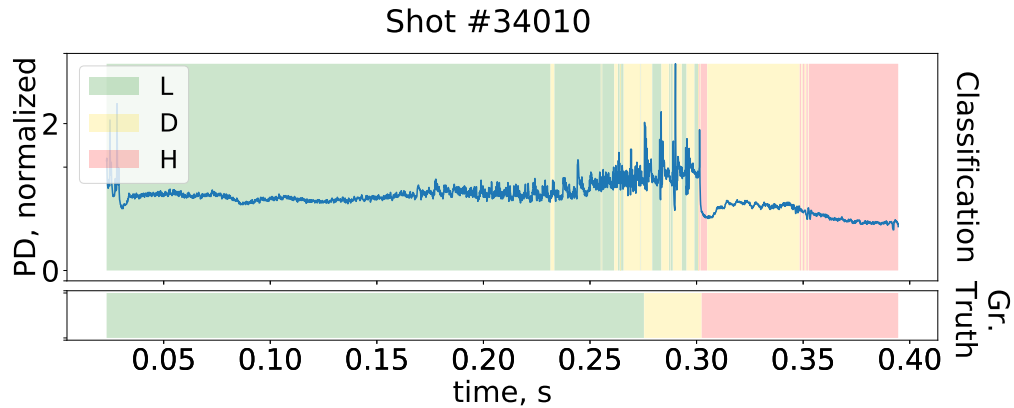


Figure 6.18.: In TCV shot # 34010, the network correctly identifies the transition into H mode at $t = 0.3s$, but it shortly thereafter (incorrectly) switches back to dithering.

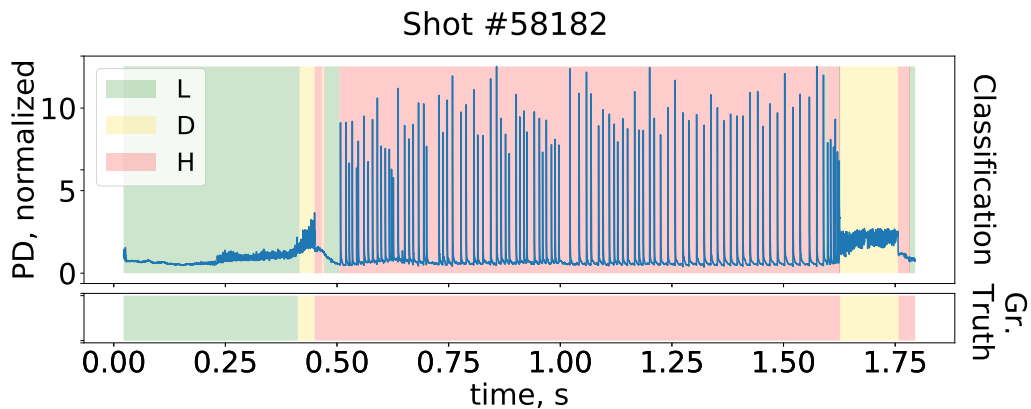


Figure 6.19.: In TCV shot #58182, the network correctly identifies a transition into H mode (shortly before $t = 0.5s$) but then incorrectly switches back to L mode and remains there until the first ELMs (spikes in the PD signal) appear.

On ELM detection, the two networks can achieve essentially equal results. On the plasma state classification, a clear difference can be seen between the results obtained with the CNN, and those obtained with the Conv-LSTM. Comparing the κ -index (κ_n) scores of each network shows that the LSTM's scores are clearly higher, which suggests that, at least when it comes to detection of plasma modes, the processing of long-term correlations in data facilitates accurate classification. There is some indication that overfitting occurred. However,

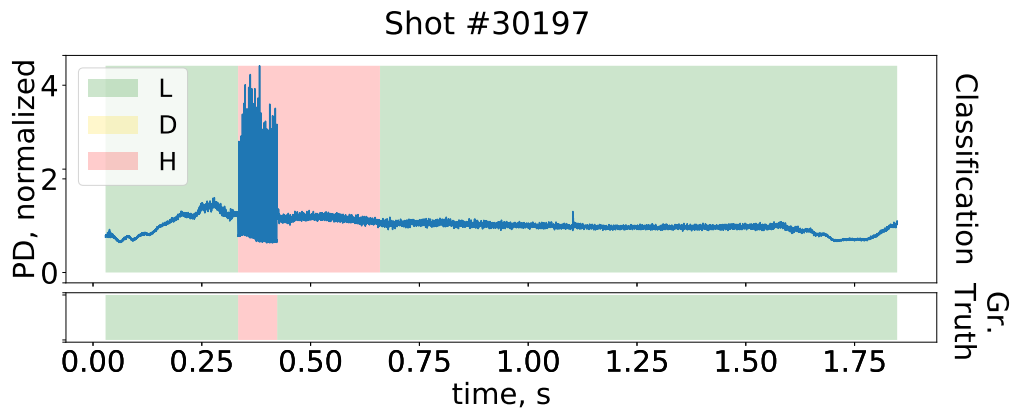


Figure 6.20.: In shot #30197, the network misses the transition from H to L mode, which happens immediately after the series of spikes in the PD signal, and only makes the switch after $t = 0.5s$.

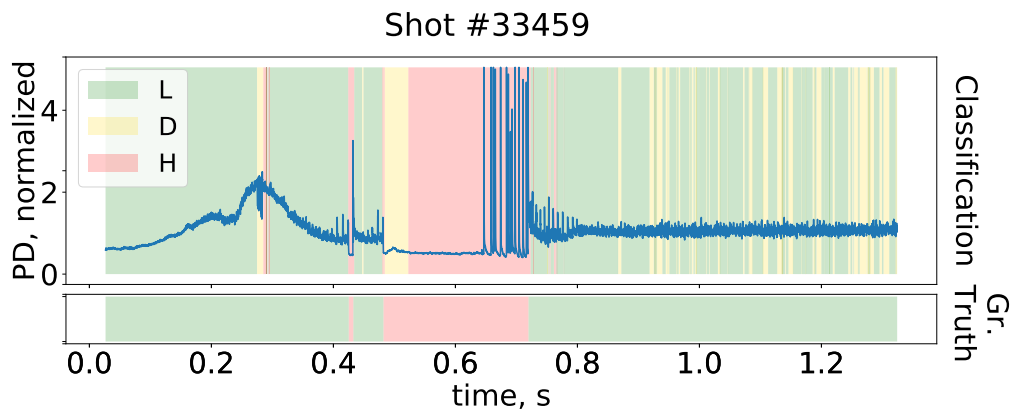


Figure 6.21.: Shot #33459 represents an overall bad classification by the network; many dithers are incorrectly classified, while the transition from L to H mode is missed. Around $t = 0.3s$, immediately after classifying a D mode, the network oscillates between L and H in quick succession for about 0.01s, which to the naked eye might appear in this plot as a gray area; in reality, it is an artifact of the plot, with alternating red and green regions.

our monitoring of the training progression indicated that, while the metric values for test data are always lower, they did, nevertheless, become better as training progressed. Thus, an overfitting-avoidance strategy such as early stopping would, in this case, not have helped achieve better test accuracy.

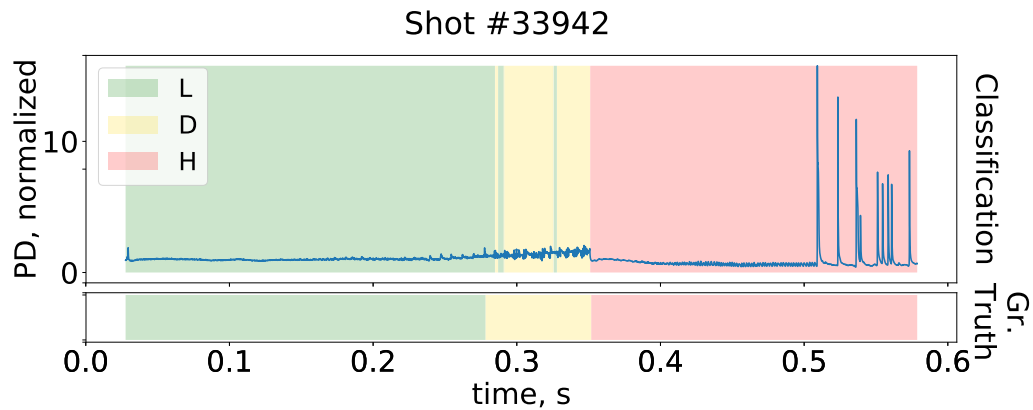


Figure 6.22.: Shot #33942 is an example of an overall good detection.

While the results from the Conv-LSTM are better, that network is also more complex with both network training and inference taking longer.

Although this work used data from the TCV tokamak, it should also be possible to adapt it to other machines; as a matter of fact, the data sources used exist on most tokamaks. As long as the data fed to the neural networks is from those same sources, this model could in principle be used for automatic labeling of shots from a number of different machines.

Acknowledgements

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. We would like express our gratitude to B. Labit, R. Maurizio and O. Sauter at SPC/EPFL for taking the time to manually label the data used for training. This work was supported in part by the Swiss National Science Foundation.

7. Plasma Confinement Mode Classification Using a Sequence-to-Sequence Neural Network With Attention

Authors Francisco Matos¹
 Vlado Menkovski²
 Alessandro Pau³
 Gino Marceca³
 Frank Jenko¹
Affiliations: ¹Max Planck Institute for Plasma Physics, Garching, Germany
 ²Eindhoven University of Technology, Eindhoven, Netherlands
 ³Swiss Plasma Center, Lausanne, Switzerland
Published in: Nuclear Fusion
DOI: <https://doi.org/10.1088/1741-4326/abe370>

Author contributions: The first author carried out the work of code implementation and algorithm development. The remaining authors contributed significantly with discussions and suggestions, as well as reviewing and making additions and corrections to the paper. Alessandro Pau and Gino Marceca were responsible for collecting the new data.

Abstract. In a typical fusion experiment, the plasma can have several possible confinement modes. At the TCV tokamak, aside from the Low (L) and High (H) confinement modes, an additional mode, dithering (D), is frequently observed. Developing methods that automatically detect these modes is considered to be important for future tokamak operation. Previous work[65] with deep learning methods, particularly convolutional long short-term memory networks (conv-LSTMs), indicates that they are a suitable approach. Nevertheless, those models are sensitive to noise in the temporal alignment of labels, and that model in particular is limited to making individual decisions taking into account only the input data at a given timestep and the past data, represented in its hidden state. In this work, we propose an architecture for a sequence-to-sequence neural network model with attention which solves both of those issues. Using a carefully calibrated dataset, we compare the performance of a conv-LSTM with that of our proposed sequence-to-sequence model, and show two results: one, that the conv-LSTM can be improved upon with new data; two, that the sequence-to-sequence model can improve the results even further, achieving excellent scores on both train and test data.

7.1. Introduction

During nuclear fusion experiments, the plasma can be described as being in one of several possible confinement states. At the TCV tokamak, it is typically classified as being in either Low(L), Dithering(D) or High(H) confinement mode. All shots, during the ramp-up phase of the plasma, begin in L mode. By applying sufficient heating power, the plasma spontaneously transitions into H mode[8] (typically at TCV this process lasts approximately 1 ms). This mode is termed High confinement because, once it is reached, one can observe significantly reduced transport of particles and energy from the plasma to the surrounding vessel walls. This allows for a larger energy confinement per input power; for this reason, most current designs for future tokamaks assume that they will regularly run in H-mode. In some cases the transition from L to H mode does not happen directly, but rather the plasma oscillates rapidly between the two con-

finement regimes. In this case, the plasma is considered to be in a Dithering[27] mode.

Many studies have been done on the physical factors behind the transition between L and H mode, but the phenomenon is still not completely understood[98]. Furthermore, there is no simple set of rules that can be used to determine the plasma mode given the values of the signals during a fusion experiment. Nevertheless, most of the time, there are highly salient patterns in these measured signals that can be used by domain experts to determine the plasma mode with high confidence. For example, a transition from L to H mode can typically be identified by observing a sudden drop in the emitted H- α radiation. However, these data patterns can be rather complicated and ambiguous; for example, Dithers leave signatures in the emission of photons similar to that of type III Edge Localized Modes[99], which are events that occur during H-mode.

This process of manually labeling the experimental data can be quite cumbersome in many cases, particularly when one wishes to conduct large studies and analyze many shots. For that reason, work has been put into developing tools capable of automating the task of detecting different confinement modes. In particular, in the past few years, research has been done on using machine learning[39, 40, 41, 42, 114] and, more recently, deep learning[65] for this task. These algorithms are particularly suitable for dealing with such challenges of extracting patterns from high-dimensional data collected during these experiments.

For example, when analyzing transitions between L and H modes, the type of correlations one expects to find in the data — localized, spatial correlations as well as long-term temporal ones — can be, respectively, efficiently discovered using Convolutional[132, 133] and Recurrent Neural Networks (RNNs)[134, 56]. Previous work with models of this sort, in particular with Long Short-Term Memory (LSTM) networks, indicates that they can be very accurate in this task[65]. One of the main challenges of these models, however, is that they have to produce a decision about the plasma mode at each timestep by looking only at a given context of the signals and their own past states, which represent past data values. Bi-directional LSTMs can look at future information, but they still only have access to the data when making decisions. In contrast, when a human expert faces a difficult decision, they not only look at the data, but also rea-

son through several possible sequences of plasma confinement mode evolutions. They go back and forward through the input signal, consider the consequences of labeling a mode with a given value for all consecutive modes, and in doing so, frequently revise decisions until the most likely sequence of plasma confinement modes can be determined.

Conceptually, the tasks of automated language translation and the automated labeling of plasma confinement modes are closely related: one wishes to translate a sentence in a source language to a different sentence with the same meaning in a target language. In the case of automated labeling of plasma confinement modes, one can consider signal time traces to constitute the sentence in the source language, while the corresponding confinement modes can be thought of as the “translated” sentence in the target language.

For this reason, in this paper, we propose an approach that builds upon previous work with deep learning applied to automated detection of plasma confinement modes, by using recent developments in the field of neural machine translation (NMT). Broadly, in that field, one can talk of two main types of algorithms in use: sequence-to-sequence models based on RNNs, and transformers. Transformers are ideal when a large amount of data is available, since they parallelize and scale well with large amounts of data when compared to RNNs, and have achieved tremendous success in processing of short sequences[135].

However, those are not the conditions we face in our setting: the amount of data we have is (compared to typical NMT tasks) rather limited, and furthermore, the sequences we deal with (i.e., plasma shots) are quite long. This suggests that RNN-based models are a more suitable approach, given their capacity to process long sequences with their hidden states. A general, vanilla RNN, such as a conv-LSTM, is, by itself, incapable of producing decisions over sequences of outputs, and is limited to making a sequence of independent decisions based only on the observed data. It is also susceptible to noise produced by misaligned labels. However, sequence-to-sequence models can solve both of these problems. These models, as well as associated mechanisms such as attention[61, 62, 136], have considerably advanced the field of neural machine translation and transduction in the past few years.

This paper is organized as follows. Section 7.2 provides an overview of the field of neural machine translation, in particular by explaining the function-

ing of sequence-to-sequence models, and how we can adapt them to suit our task. Section 7.3 details our considerations regarding the data and the problem formulation, our preprocessing steps and the proposed model architecture. Section 7.4 shows some of the obtained results and scores, and in particular, we compare the results of this model with those obtained in[65]. We then wrap up with a discussion in section 7.5.

7.2. Background

7.2.1. Sequence-to-sequence models

Sequence-to-sequence models have achieved tremendous success in the field of neural machine translation[59, 137]. These models are characterized by two separate components performing different tasks: an *encoder* that reads a sentence in the source language and produces an encoded representation of that sentence, and a *decoder* that, based on the *encoding*, produces an appropriate translation into the target language. The encoder and decoder can technically be any type of algorithm, though in most applications, they are built with RNNs[55] that are jointly trained.

In neural machine translation, the encoder typically maps a word or sequence of words in the source language to a numerical representation of said words, as a function of a pre-defined size of the source language vocabulary. This is then followed by a recurrent layer, typically a long short-term memory (LSTM) layer, or a gated recurrent unit (GRU) that is trained to find sequential correlations in the embedded input sentences. These models can keep track of correlations between points that are far apart in time because they have internal *hidden states*, though their capacity deteriorates if the time interval is large enough. At every source timestep j , with $0 < j < k$ (where k is the length of the source sequence), the encoder computes a new hidden state, h_j ; each vector h_j in the sequence of hidden states (h_0, \dots, h_k) constitutes a summary of the information that the encoder has processed until timestep j , and the final hidden state of the encoder's recurrent layer (h_k) can thus be considered to be an encoded representation of the entire input sequence.

The decoder must, subject to the encoding produced by the encoder (h_k), pro-

duce an appropriate corresponding sequence of words in the target language. When using an RNN decoder, this is done by setting its initial hidden state to h_k . Therefore, unlike a simple RNN model, which only receives a part of the source data as input at each timestep, in the sequence-to-sequence model, the decoder can work with a representation of the entire source sequence. The decoder then outputs, at each decoding timestep, a probability distribution over the discrete set of possible outputs, conditioned on the source sequence. Furthermore, in the general formulation of the sequence modeling problem, where the input sequence is not aligned to the output (e.g., different sampling rates of the input and output), the model needs not only to determine the output sequence, but also to align the symbols of the output sequence with the input. Lastly, the decoder can be made autoregressive, by feeding it a selected output at the previous timestep as an input in the next.

Figure 7.1 illustrates this mechanism: the encoder produces an encoding of the input sequence; the decoder, using that encoding as a starting state, produces a translation into a target sentence. At timestep 1 of the decoding process, a $\langle \text{start} \rangle$ character is fed to the decoder; in subsequent steps, the selected output from the previous timestep is fed as input.

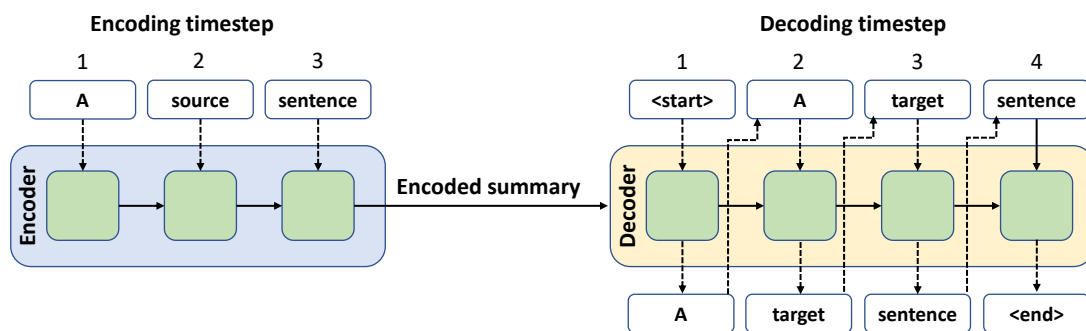


Figure 7.1.: Representation of the flow of information in a sequence-to-sequence encoder-decoder model. Dashed lines denote model inputs and outputs, while solid lines denote hidden states. At decoding timestep 1, the output distribution gives highest probability to the word A , which is then fed as input at timestep 2.

An autoregressive decoder can, at inference time (i.e., after training), evaluate several output sequences, by conditioning its prediction at each timestep on different past outputs. Therefore, it is not limited to outputting a single solution,

but rather, it can produce a probability distribution of possible solutions.

In practice, sampling from the output joint probability distribution can be done by treating the distribution as a tree data structure, where each path in the tree represents a sample from the distribution, i.e., a different possible output sequence. Expanding a new node in that tree corresponds to sampling a different output from the previous timestep to condition future outputs; searching this tree for solutions (paths) of high probability can be done efficiently with a beam search algorithm[138]. In addition, the tree search can be done (and simplified) by explicitly incorporating domain knowledge, which can allow for pruning paths that are known to be impossible (for example, in our case, by discarding paths that start in H-mode). All of this contrasts with using a simple RNN for translation. In that case, one expects the model to output a (single) solution of high likelihood, there is no way (in inference time) to explicitly encode knowledge that rules out certain solutions, and the source and target must be of the same length.

7.2.2. Attention

While encoder-decoder models achieve very good results, they can nevertheless still lose performance when translating long input sequences[61]. There are several reasons for this, but the main one is that the encoder is expected to be able to encode all the information of the source sentence in a single vector; in practice, especially for long input sentences, training such models with algorithms that back-propagate gradient updates can be challenging. For this reason, the *attention* mechanism was developed. The main idea behind it is to extend the decoder with an attention layer that can access the entire sequence of encoder hidden states $h = (h_0, \dots, h_k)$. The attention layer can then, at every decoding timestep i , compute an attention vector α_i , whose values are normalized to add up to 1, and which constitute a series of weights that are used to compute the decoder's *context vector* c_i , defined as:

$$c_i = \sum_{j=0}^k \alpha_{ij} h_j.$$

Intuitively, at every decoding timestep i , the corresponding context vector c_i

will change to reflect the greater (or lesser) relevance of some components of h for computing the decoder output at that timestep. While the final encoder state h_k is still fed into the decoder as an initial hidden state, the decoder is no longer wholly dependent on it, and has access to a much richer context thanks to the attention layer, which is trained with the rest of the model. Moreover, the existence of the attention layer can give additional insight into the inner working of the model. At evaluation time, the attention vectors can be collected and used to visualize what parts of an input have been focused on by the model when generating a certain output. Attention significantly improved the results obtained by sequence-to-sequence models, and is also the main mechanism behind the functioning of transformer networks[135].

7.3. Methods

7.3.1. Problem Formulation

The data used for this work comes from 4 different signals from the TCV tokamak: the photodiode (PD), measuring H- α radiation; the plasma current (IP); the diamagnetic loop (DML) measuring the plasma toroidal flux; and the interferometer (FIR), measuring the line-integrated electron density. All of the signals are re-sampled to a frequency of 10 kHz; a more thorough description can be found in [65]. As in that work, we are generically interested in finding, for a given temporal sequence of measurements x_t , with $0 < t \leq N$ (which constitute a single shot), the most likely sequence of plasma confinement mode $\hat{z}_{1:N}$ that explain the observations $x_{0:N}$.

The approach proposed in this paper does this in two parts. On the one hand, a model is trained to estimate the joint probability distribution of the sequence of plasma modes $p(z_1, z_2, \dots, z_N | x_{0:N})$ for a given shot. Second, an algorithm finds a sequence $\hat{z}_{1:N}$, drawn from the joint distribution, with high probability. Formally, the task is to find:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} p(z_1, z_2, \dots, z_N | x_{0:N}).$$

with $z_t \in Z$ and $Z : \{ 'Low', 'Dither', 'High' \}$, and $z_0 = \{ 'Low' \}$, since any shot is

assumed to begin in L-mode.

In practice, in a real-time environment, we would not possess the entire sequence of measurements (the whole shot), but rather, only the signal values up to a certain point in time t . Thus, one of our requirements is to find a sequence of high probability up until t while looking only at past measurements. For this task, a simple recurrent neural network (LSTM) model can be used[65]. However, such models, when making a decision, rely only on the input data and their own internal state. They cannot take their own past outputs into account when making a decision, and therefore, are limited to producing a single point-like estimate for the output sequence of confinement states. In contrast, sequence-to-sequence models can explicitly take their own past outputs into account when making a decision. This way, at time t , a sequence-to-sequence model does not decide on a single output for t , but rather, it decides on the entire sequence of plasma mode evolutions up to that point in time; that is, the model computes the distribution $p(z|x)$ as:

$$\begin{aligned} & p(z_1, z_2, \dots, z_N | x_{0:N}) \\ &= p(z_1 | x_{0:1}, z_0) p(z_2 | x_{0:2}, z_{0:1}) \dots p(z_N | x_{0:N}, z_{0:N-1}) \\ &= \prod_t p(z_t | x_{0:t}, z_{0:t-1}), \end{aligned}$$

where $p(z_t | x_{0:t}, z_{0:t-1})$ denotes the probability of observing mode z at time t , given the sequence of observed signals x from time 0 to time t , and the sequence of outputs until t . It is the additional conditioning on past outputs that allows the sequence-to-sequence model to approximate the full joint distribution $p(z|x)$.

One caveat of the sequence-to-sequence model architecture is that it requires that the model observe windows, or subsequences, of the input data (up until time t) of fixed size. This means that in a real-time environment, for most values of t , a sequence-to-sequence model would have a delay when computing $p(z_t | x_{0:t})$. This delay corresponds to a pre-defined size of the signal windows that the model receives, which therefore must be minimal.

With a sequence-to-sequence model, using the notation above, finding a sequence of high probability means finding:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} \prod_t p(z_t | x_{0:t}, z_{0:t-1}). \quad (7.1)$$

The task of finding samples of high probability from the distribution is done, in the case of this work, with a beam search algorithm[138]. Because in our setting the sequences have potentially thousands of timesteps, computing their individual likelihoods using products as in Equation 7.1 would yield numerically unstable results; thus, the beam search uses the logarithm of the probabilities:

$$\begin{aligned} \hat{z}_{1:N} &= \arg \max_{z_{1:N}} \log\left(\prod_t p(z_t | x_{0:t}, z_{0:t-1})\right) \\ &= \arg \max_{z_{1:N}} \sum_t \log p(z_t | x_{0:t}, z_{0:t-1}), \end{aligned} \quad (7.2)$$

which allows it to use sums instead, and to look for the sequence whose log-probability is greatest.

7.3.2. Data engineering

Events such as, for example, the LH transition can be roughly pinpointed in a signal time trace. However, it is difficult to specify precisely, on a consistent basis, at which exact point in time the transition happens; for example, in some shots, the transition might be quite sudden, whereas in others, the transition signatures in the data can be more spread out over time. The typical time that a TCV shot takes to make an HL transition is on the order of 1 ms. Considering also that the sampling frequency of our signals is 10 kHz, this translates to an intrinsic uncertainty for the label determination of at least 10 timesteps. So, for example, one expert might determine, for a shot, the start of the H mode at the point in time where the PD signal starts dropping, whereas another expert might claim that the H mode actually only starts at the point where the signal has already stabilized (perhaps 1 ms later). While the difference may sound trivial, this can become problematic in a supervised learning task such as the one we face in this work. In principle, if the amount of training data is sufficiently large, small inconsistencies in labels (such as variations of 1 ms in the localization of transitions) will tend to be averaged out by a classifier. Nevertheless,

these mismatches can produce instabilities during training and can ultimately degrade performance.

For that reason, one of the steps we took in this work is to reduce the temporal resolution of the sequence-to-sequence model’s outputs. This can be done thanks to the model’s architecture, which allows for a mismatch between the size of the input and output sequences. We do this by grouping the existing labels (i.e., sequences of shot classifications) in our dataset into *blocks* of a fixed size. In the pre-processing stage, each of those blocks is mapped to a certain plasma confinement mode (L, D or H); this mapping is done by computing the source label with the highest number of occurrences within that block (see Figure 7.2). Our expectation is that this decrease in temporal resolution will yield better performance in both training and inference time, at a minimal cost to the physical validity of the results.



Figure 7.2.: Representation of the computation of a block of target labels, where each bottom block corresponds to 4 source timesteps. The green color indicates a label of L (Low confinement) while red represents H mode. A majority of the labels in the source timesteps 4 - 7 are in H mode, so the corresponding block at target timestep 1 is labeled as H.

7.3.3. Model architecture

Our model’s architecture is based on existing architectures for neural machine translation, in particular, the one proposed in [62]. The architecture consists of an encoder-decoder model with attention, which we detail in the following paragraphs.

Unlike most work with language translation, where one receives discrete units (words) as inputs, in our case, the inputs to the model are the continuous signal time-traces from TCv shots. In those time-traces, one expects to find not only localized, spatial correlations in the data — for example, a sudden drop in the PD signal — but also long-term contextual correlations, namely, which modes

a shot may have been in in the past. For that reason, the encoder in our model consists of a convolutional recurrent neural network, much like the conv-LSTM used in [65]. We made slight adjustments, namely in the number of convolutions used, but otherwise preserved that model’s architecture, and used long short-term memory (LSTM) units for the recurrent layers. The inputs fed to the encoder are sequences of overlapping windows, which slide across the signal time-series. These windows were defined to have a size of 40 source timesteps, with a stride between windows of 10 timesteps. This last detail, in particular, means that the full sequence-to-sequence model has approximately 10 times fewer network parameters than the convolutional LSTM in previous work.

The convolutional layers are trained to find local correlations in the windows, while the recurrent layers, based on the output of the convolutions, keep track of long-term correlations in the input sequences. For example, the convolutional layers can detect a local shape in a signal window indicating a possible L-H transition, and the recurrent layers then use that, as well as their stored information about the current plasma mode (i.e. the long-term dependency), to determine whether a transition indeed has occurred.

In [65], the conv-LSTM was used to directly map the input sequence of measurements, x_t , into a sequence of outputs, z_t , indicating a plasma confinement mode at a particular point in time (see Equation 7.1). In this work, the task of this particular submodel is instead to produce an encoded summary of x_t , which is stored in its internal hidden state vector h . During both training and inference time, we feed the encoder not with entire shots, but rather, subsequences of signals drawn from the shots. There are several reasons for this. On one hand, in recurrent neural networks, gradients can vanish when being backpropagated through time, which can be particularly problematic when working with long sequences; training with smaller subsequences mitigates this problem. On the other hand, the existing data is imbalanced with regard to the labels; for example, Dithers tend to be much less frequent than L and H modes. Using subsequences allows us to feed the entire model, during the training process, with a more balanced number of samples for each class (by oversampling the dithers), thus preventing a potential source of biased results. Finally, any future usage of the methods proposed in this paper for real-time plasma data analysis implies, by definition, that only information until a particular point in time, and not the

entire shot, is available for classification.

In training time, the subsequences are drawn uniformly (with respect to the three classes) from our existing data ensemble. In inference time, for any given shot, the subsequences are drawn and fed to the encoder consecutively. In both cases, the encoder’s state is reset each time a new subsequence is fed to it, which means that the context vectors only hold information about the current subsequence under consideration.

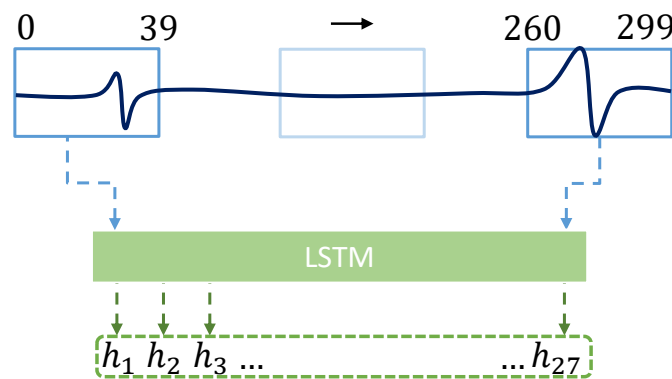


Figure 7.3.: Illustration of the encoder architecture and how a single subsequence is processed and encoded in the encoder hidden state vector h . The dark blue line represents a subsequence of a signal timeseries. The sliding windows are in light blue.

In practice, we defined the subsequences drawn for training and inference to have a size of 300 source timesteps (this contrasts with a typical shot size of, at least, 10000 timesteps at a 10kHz sampling rate). This value also corresponds to the delay that the model would have in a real-time setting. With a window size of 40 and a window stride of 10, these 300 timesteps are fed to the convolutional layers as sequences of 27 convolutional windows (further shrinking the total size of the sequence fed to the LSTM), with window 1 observing x_t from $t = 0$ to $t = 39$, and window 27 observing x_t from $t = 260$ to $t = 299$. The convolutional LSTM processes these windows to produce the hidden state vector h , which has a length of 27 elements and is fed to the decoder. An illustration of this can be found in Figure 7.3.

The task of the decoder is to approximate a probability distribution $p(z|x)$ of plasma confinement modes, subject to the summary given to it by the encoder.

Our decoder is composed of a recurrent neural network (specifically, an LSTM layer), an attention layer, and a series of dense layers.

Each individual element of the output sequence is processed in a single timestep, with each target timestep corresponding to a single block of classifications. We defined the blocks as having a size of 10 — that is, each target timestep yields a single classification for 10 source timesteps.

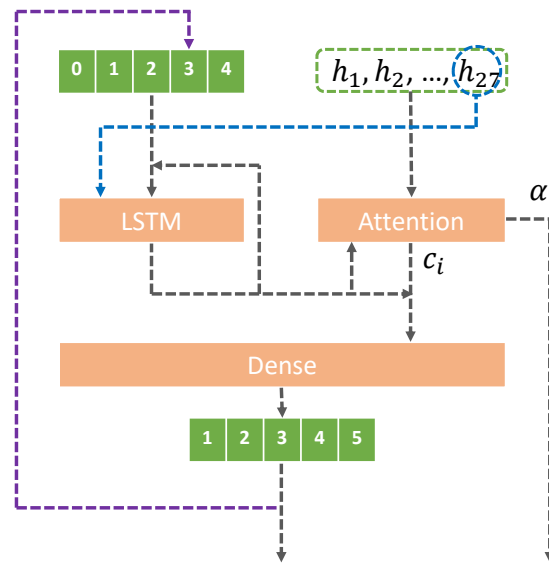


Figure 7.4.: Schematic representation of the decoder’s architecture. Represented here is the sequence of operations carried out by the decoder to produce the output distribution of plasma confinement modes at decoding timestep $t = 2$. Gray arrows denote data flows at t . The purple arrow denotes the autoregressive feeding of the output at $t + 1$. The blue arrow denotes the initial setting of the decoder state to the last encoder state. Joining arrows denote concatenation, α_i and c_i are the attention weights and the context vector, respectively.

In the first decoding timestep of each new subsequence, the decoder’s initial hidden state is set to the decoder’s final hidden state, h_k , for that subsequence. At each decoding timestep, the decoder receives as input its own last processed output (plasma confinement mode z_{t-1}), and the last output from its own LSTM cell; these are concatenated as suggested in [62] and fed to the decoder’s LSTM, which also updates its own internal state. The output of the LSTM is then concatenated to the output of the attention layer (which receives the entire encoder

hidden state vector h), and fed through a series of dense layers to produce the final decoder output, which is a vector whose entries add up to 1 and which, individually, represent the computed probability of a given plasma confinement mode z_t (see Figure 7.4). The decoder’s LSTM is built with a latent dimensionality of 32 units, that is, we process each target timestep with 32 LSTM cells. In terms of the attention layer, in [62], several different mechanisms for computing the alignment scores are proposed; we opted for the general form described in that paper.

One of our considerations when designing the decoder was to take into account how a human expert would label the signals. Our intuition was that, when looking at a time point of a shot, a labeler always takes into consideration information around that point — namely, the events happening immediately before and after. For that reason, we designed the decoder such that, for each incoming context vector (which encodes information regarding 300 source timesteps), the decoder produces a sequence of 18 blocks of labels (with a block size of 10, this corresponds to 180 source timesteps), which are the classifications for source subsequence steps [60 : 240]. The idea is that the extra information present in the remaining source timesteps would help improve the classification results. In evaluation time, this setup requires that the consecutive subsequences drawn from a shot overlap with each other, so that an output sequence can be produced for the entire shot. This also leads to the loss of the initial and final 6 target classification blocks of a shot (see Figure 7.5), but we consider this to have no bearing on our results.

In evaluation time, the decoder always produces a distribution of possible plasma confinement states whose probabilities add up to 1. One possibility would be to use a greedy approach and simply take, at each target timestep, the plasma state for which the output probability is highest, and feed that state at the next decoding timestep. This would yield a possible solution (i.e., a single sequence of plasma confinement states), but there would be no guarantee of it being optimal. For that reason, we use a beam search algorithm [138] to traverse the tree structure of possible solutions (different sequences of z), which allows for obtaining samples closer to the optimal \hat{z} . This is done by, at each target timestep, expanding the search tree for all previous outputs, and not just for the output of highest probability. Then, for each previous output z_{t-1} under consid-

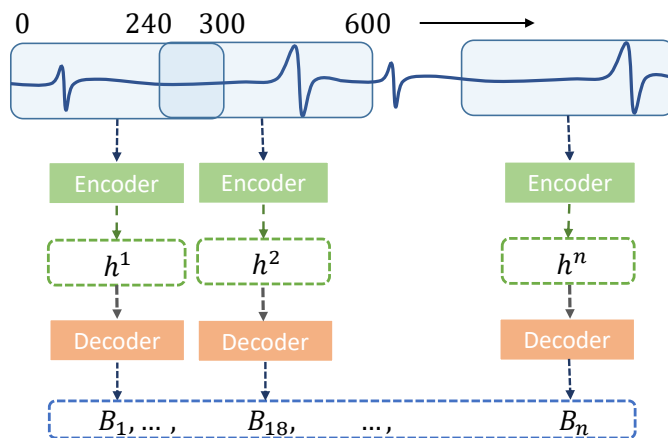


Figure 7.5.: Representation of the full sequence-to-sequence model’s architecture. Notice how the subsequences (translucid blue) overlap with each other. The encoder’s state vectors, h^k (where k is the current subsequence), are fed to the decoder, which produces blocks of outputs.

eration, the conditional probability, and the log-likelihood, defined in Equation 7.2 are computed for the current timestep t . Once all target timesteps have been processed, the value of z for which the likelihood is highest is returned.

Naturally, expanding the full tree at every timestep would quickly become unwieldy, owing to the large number of different paths to be processed. On the other hand, expanding a single beam at each timestep would be equivalent to a greedy search. For these reasons, we defined the beam search to have a maximum width of 20, i.e., at every step, only the 20 paths with the highest log-probability are expanded, while the rest are discarded. In addition, we encoded in the beam search a rule for expanding only those paths that start in L-mode, which simplifies the search; an illustration of this can be seen in Figure 7.6.

7.3.4. Dataset Preparation

For the work in [65], a total of 54 shots were used for training and testing the proposed models. In this work, we carried out a more careful treatment of the dataset preparation with respect to the previous publication. In the first place, the selection of the discharges for training and testing was done in order to cover as exhaustively as possible the space of the plasma confinement modes in TCv,

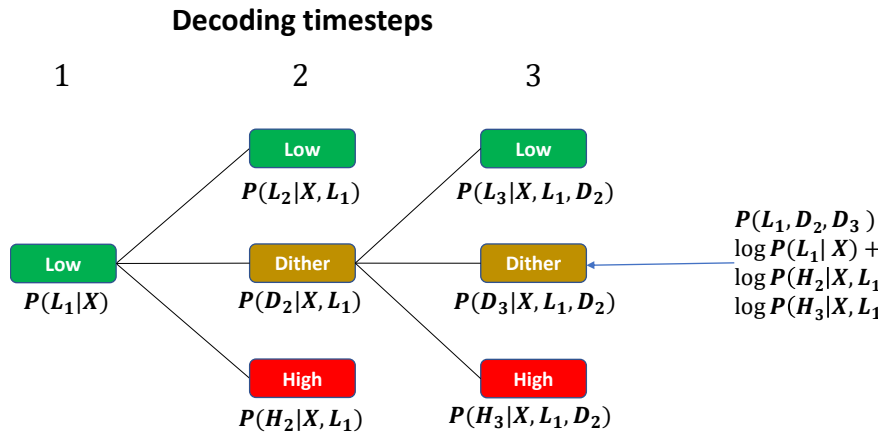


Figure 7.6.: Illustration of the beam search algorithm in the first 3 target timesteps. X denotes the context vector and the final encoder state which condition the output. In timestep 1, the decoder computes the probability of a path starting in L mode, and we manually set that to be the only path to be expanded. In timestep 2, the conditional probabilities of three paths ($\{L_1 L_2\}$, $\{L_1 D_2\}$, $\{L_1 H_2\}$) are computed; the paths are then expanded (for simplicity, only one expansion, that of $\{L_1 D_2\}$, is shown here, but 9 paths would be evaluated in timestep 3).

accounting for the different temporal evolutions of the plasma. Using a Dynamic Time Warping (DTW)[139] algorithm, we measured the similarity between pairs of temporal sequences and assigned them to a given group, based on a similarity measure. The desired number of groups was obtained by applying a hierarchical clustering algorithm to univariate time sequences corresponding to the entire plasma discharges. A total of 293 discharges were selected and processed through the DTW, setting the number of clusters to 100. From each of the 100 clusters, shots were extracted and classified as an interesting (or not) shot from the physics point of view, as far as our problem (i.e., the presence of L/D/H transitions) is concerned. Some clusters were discarded since they consisted of disruptions without even achieving an H mode confinement state, while others presented technical issues in the ramp-up phase before reaching the stationary phase. Limiting ourselves to the interesting shots and maximizing the number of clusters where a given shot arose from, 88 discharges were selected for ground truth determination. For the latter, instead of having different annotations by

different labelers for a particular shot, a consensus on a common convention between two experts was established to determine the label of each timestep for all shots. A detailed revision of the different transitions was performed with particular attention to the presence of short transitions on the order of 2 ms. The outcome was a unique, consistent, ground truth per shot. A test set to evaluate the final results of the model was carefully determined and fixed during all the experiments. A total of 27 shots were selected, each extracted from a different cluster. Out of 27 shots, 17 shots were “unpolluted” cases (without the presence of type III ELMs), while the others 10 were special “noisy” discharges with type III ELMs. The proportion between the noisy and “unpolluted” discharges in the test set followed approximately the same proportion as in the complete dataset.

7.4. Results

We begin this section with a direct comparison between this sequence-to-sequence model, and the conv-LSTM used in [65]. To that end, we trained and tested the old model, preserving all the original architecture and hyperparameters, with the new train and test data (shots and labels) compiled for this work. Table 7.1 shows the results. As in the previous work, we performed the evaluation using Cohen’s Kappa-statistic coefficient [131], which gives an indication of the match between two sets of categorical data (with a score of 1 for a perfect match and 0 for no match). In our case, it reflects the match between the models’ outputs, and the labeled data. We computed the score on a per-class basis and also on a weighted mean basis, in order to indicate whether the classifications produced by the sequence-to-sequence model match the data’s labels. We designed the model with Tensorflow [140], and ran it on an NVIDIA Quadro RTX 5000 Graphics Processing Unit (GPU).

A comparison between the results in Table 7.1 and those described in [65] shows that the current dataset already improved the capacity of the old model. Nevertheless, it was still underperforming, particularly on dithers; even on training data, the mean dither score was 0.9.

We then ran the new sequence-to-sequence model. The results on both the

		L	D	H	Mean
κ scores	Train	0.98	0.91	0.98	0.98
	Test	0.92	0.78	0.91	0.9

Table 7.1.: κ -statistic scores for each plasma mode and as a mean, on training and test data, for the conv-LSTM model from[65] on the data used for this work.

train and test sets can be seen in Table 7.2. They were obtained by training the network for 150 epochs, with each epoch consisting of 128 batches of data, and each batch consisting of 128 data samples, i.e., uniformly sampled subsequences of each of the existing classes drawn from the training shots. We downsampled the source labels to the same temporal resolution as the model’s output blocks to compute the scores. Figures 7.7a and 7.7b show the distribution of the scores on a per-class basis, for train and test data, as well as a weighted mean value, taking into account the relative frequencies of each class in the labels.

		L	D	H	Mean
κ scores	Train	0.99	0.99	0.99	0.99
	Test	0.94	0.86	0.96	0.94

Table 7.2.: κ -statistic scores for each plasma mode and as a mean, on training and test data, for the sequence-to-sequence model.

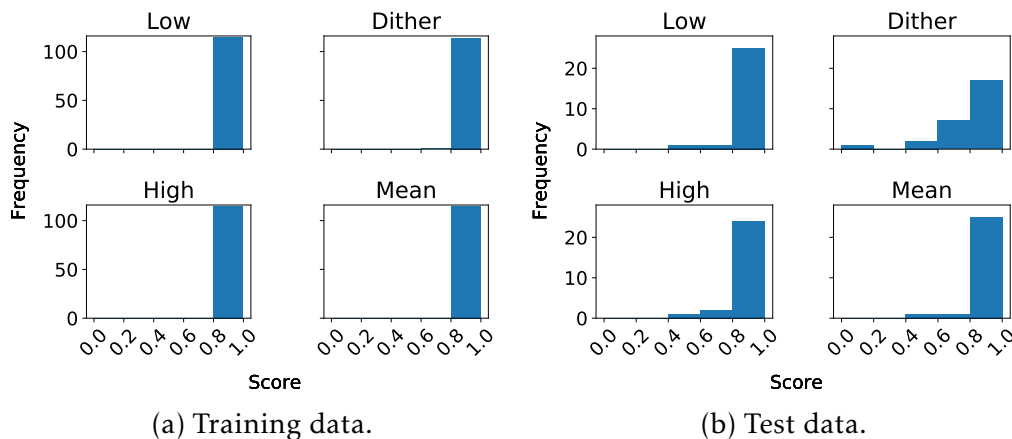


Figure 7.7.: Distribution of the κ -statistic score on a per-shot basis

7.5. Discussion

In this section, we discuss the results in further detail, in particular, the cases where the sequence-to-sequence model’s performance was poorer.

With regards to the training data, the classification was excellent in all cases; all shots achieved scores, both on a per-mode basis and as a mean, above 0.8. The lowest mean train score for a shot was 0.965, while the lowest scores for L, D and H modes were, respectively, 0.96, 0.8 and 0.98. Nevertheless, for all modes, the mean score was 0.99, a result that indicates that the sequence-to-sequence model has the capacity to learn the underlying correlations in the data to make accurate predictions.

Shot ID	L		D		H		Mean
	Fraction	Score	Fraction	Score	Fraction	Score	
42197	0.57	0.5	0.35	0.66	0.08	0.46	0.55
61057	0.5	0.72	0.04	0.68	0.46	0.73	0.72
61274	0.69	0.84	0.12	0.7	0.19	0.96	0.84
32911	0.52	0.84	0.28	0.79	0.20	0.97	0.85
61043	0.78	0.92	0.13	0.69	0.1	0.76	0.87
45105	0.42	0.91	0.02	0.52	0.56	0.94	0.92
34309	0.14	1	.03	0.8	0.83	0.96	0.96
33459	0.8	0.98	0.01	0.5	0.19	1	0.98
64376	0.92	0.9	.01	0.73	0.08	1	0.98
30268	0.24	1	0	0	0.76	1	1

Table 7.3.: κ -statistic scores for shots with at least one mode whose score was lower than 0.8. The “fraction” column indicates what percentage of the labels were labeled in a particular state for that shot.

With regard to the test data, the scores are slightly lower. Table 7.3 shows the detailed breakdown of the scores for all shots where at least one mode had a score lower than 0.8. Some of those shots have low scores on more than one class: for example, #42197 had a score lower than 0.8 for both L and D modes. Notice how, even though there are more shots with lower dither scores, the overall mean values are in most cases above 0.8; this is due to the fact that dithers are rather less frequent in the data than L and H modes.

Figure 7.8 shows the classification results for shot #42197, together with the ground truth (label), shown in the lower panel of the Figure (we show only the

photodiode signal values for ease of comprehension). The classification has two major errors: the non-detection of a switch to L mode at the end of the shot, around $t = 1.2s$, and the rapid oscillation between L and D modes from approximately $t = 0.8s$ to $t = 1.1s$. This oscillation in particular is questionable because this dithering phase presented particularly odd fluctuations that were not a common behavior in our dataset.

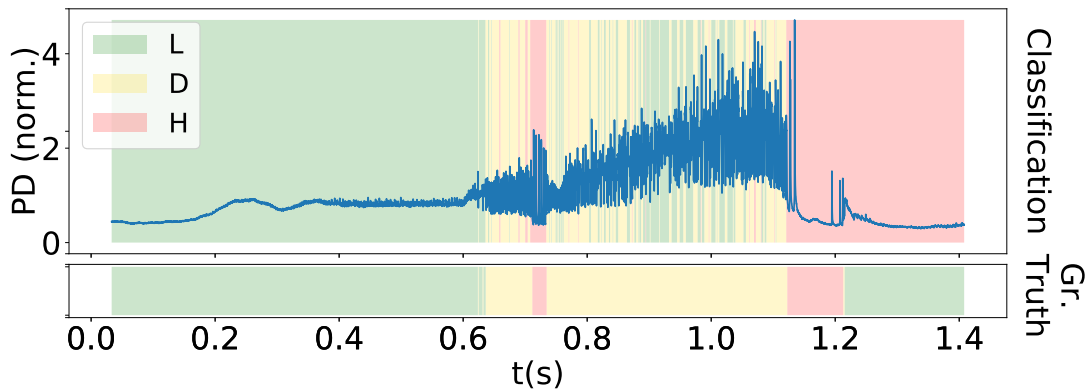


Figure 7.8.: Detection results for shot #42197. The blue line denotes the photodiode signal, while the solid background color denotes the confinement states.

For shot #61057, two short dither bursts near $t = 1.5s$ are missed, as well as the final transition back into H mode near $t = 1.6s$ (see Figure 7.9).

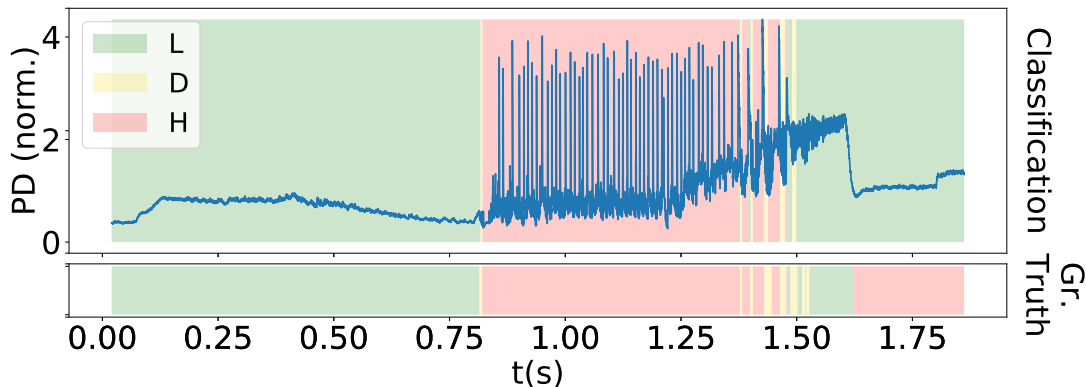


Figure 7.9.: Detection results for shot #61057.

For shot #61274, the overall classification score is already above 0.8; the model's largest mistake is in incorrectly classifying a region around $t = 1.5s$ as

L mode, which explains both the lower score for that mode and for dither (see Figure 7.10).

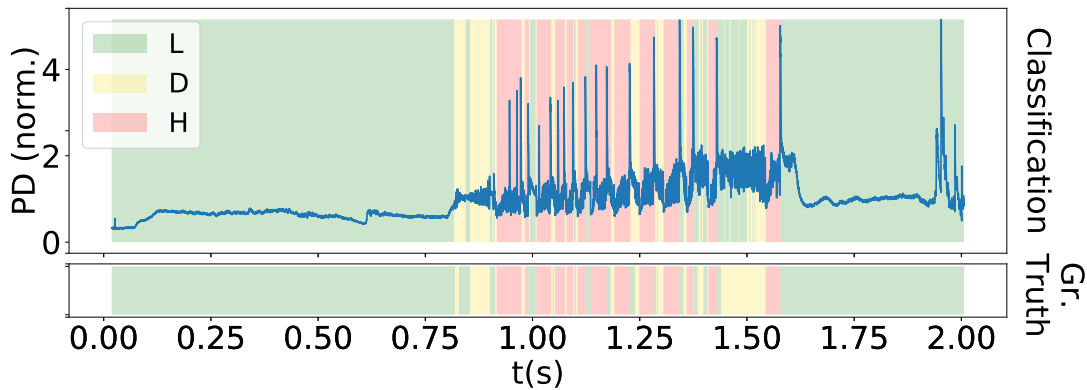


Figure 7.10.: Detection results for shot #61274.

In shot #32911, the lower scores for dither and L mode are also due to rapid fluctuations between the two modes; we think that a big factor behind the lower score is the accumulation of many small mismatches between the labels and the classifications (see Figure 7.11).

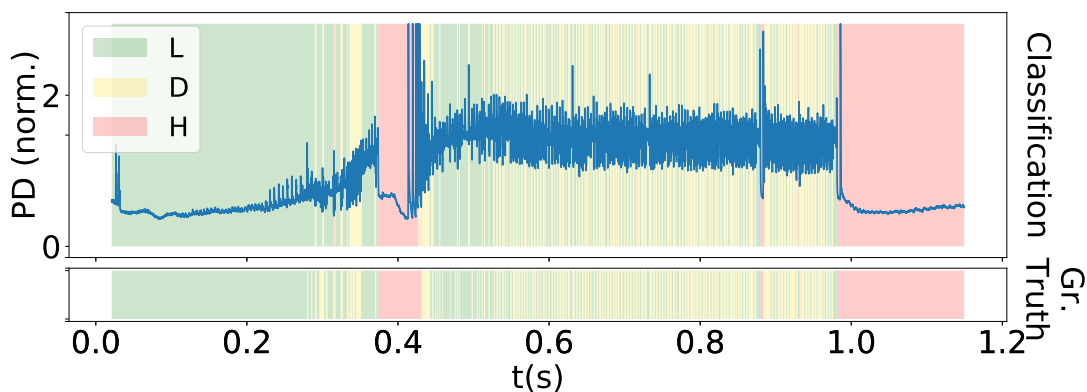


Figure 7.11.: Detection results for shot #32911.

Shot #61043 contains several switches between H mode and dithering; the model incorrectly classifies some of these (see Figure 7.12).

We consider the remaining test shots to be good classifications overall. We explain the occasionally lower dither scores by the fact that the labeled dither phases are very short (at most 3% of any given shot); any mismatch, even if small, between the label and dither classification produces a low score.

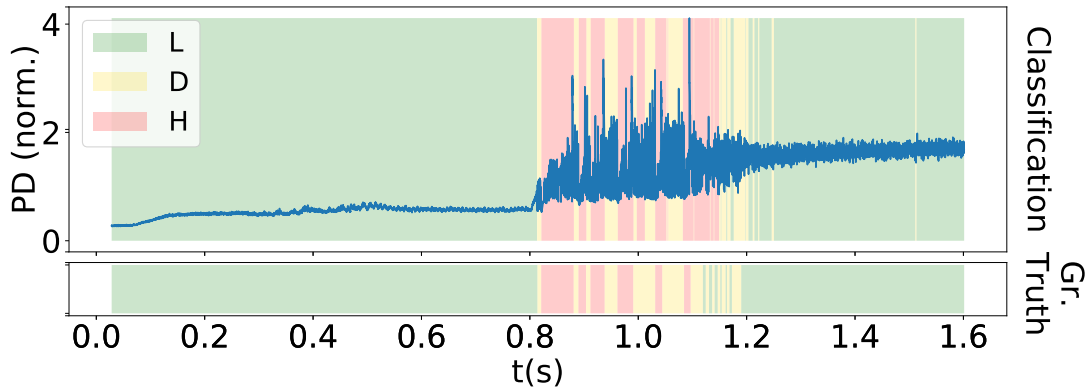


Figure 7.12.: Detection results for shot #61043.

7.6. Conclusions

This work developed a sequence-to-sequence neural network model with attention for automated classification of plasma confinement modes at the TCV tokamak. Taking previous work [65] with a convolutional long short-term memory network (conv-LSTM) for the same task as a baseline, our intuition was that one of the factors holding previous models back from achieving even better scores was the fact that vanilla LSTMs cannot reason over different past outputs, and are limited to the signal observations. This contrasts with the way a human labeler would perform the same task; typically, a labeler will reason over different possibilities for past confinement modes before deciding on a label for a given timestep, which is a process that a sequence-to-sequence model can emulate. In addition, we theorized that the varying length of transitions between plasma confinement modes, and resulting misalignment between data and labels, was producing instabilities during the training of the conv-LSTM. Finally, we also hypothesized that performance was limited by the lack of sufficient train data. Therefore, we extended the dataset used in previous work with more shots and chose a train/test split that thoroughly represented the operational shot space of TCV. In addition, we took particular care with the labeling process, to ensure a high quality of the data.

Testing the conv-LSTM from [65] on the new dataset increased its scores on both test and train data, suggesting that indeed the size and quality of the dataset was having an impact on the obtained results. Nevertheless, even with the new

dataset, the conv-LSTM still failed to completely fit the training set, particularly with regards to dithers.

On the other hand, running the sequence-to-sequence model with the new data, we achieved excellent results on the train set (with a mean score of 0.99 out of 1) and likewise, very good results on the test set (with a mean score of 0.94 out of 1). Both of these scores are important. On the one hand, the training score shows that the sequence-to-sequence model can correctly fit all the training data. This is something that the previous model (the conv-LSTM) could not do, and suggests that that model, in spite of achieving good results, had indeed some limitations compared to the new one. This is particularly interesting given that the sequence-to-sequence model has a lower number of network parameters: approximately 100 000, compared to the conv-LSTM's 1 000 000 (i.e., an order of magnitude less). We explain the fact that the results are nevertheless better with three factors, which coincide with our initial assumptions: firstly, the larger context from the data available to the sequence-to-sequence model; secondly, the model's ability to deal with misaligned labels and with transitions of varying length; and finally, the autoregressive decoder which can reason over several sequences instead of independent outputs.

On the other hand, the test scores indicate that the sequence-to-sequence model can also generalize well. They are slightly lower, but we explain that with the fact that several test shots were particularly challenging to classify, even for a human labeler. Nevertheless, the results obtained indicate that the task of plasma confinement mode classification can be best carried out by the sequence-to-sequence model presented here, given that it achieves a significant improvement over the conv-LSTM, reaching almost domain expert accuracy.

Finally, we would like to make a comment regarding a real-time (RT) implementation of this model, which is outside the scope of this publication, but should be possible. As we mentioned in section 7.3, the only fixed constraint, as far as an RT implementation would be concerned, is the size of the subsequences fed to the encoder, which induce a small minimum delay (30 ms) into any model prediction. Ideally, an RT implementation would make extensive use of parallelization, in particular in the beam search; that step could otherwise become a potential performance bottleneck in a non-parallel implementation, given that the time of the search scales with the number of beams. Also, an RT

implementation would ideally require multiple CPUs or GPUs, such as what we used for developing the method, since deep learning frameworks can efficiently run convolutional layers in parallel. Nevertheless, the model shown here, as it stands, can already be used for offline labeling of data. In addition, in light of the current research, within the field of Neural Language Processing, on transfer learning with encoder-decoder networks, we believe that this model could also be suited for tasks of domain adaptation, for example with a view to creating useful databases for future fusion devices such as ITER and DEMO. It would also be interesting to investigate, in the future, applications of transformer networks to this and similar problems in fusion.

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Part III:
Conclusions and outlook

8. Conclusions

In this work, deep learning methods and algorithms were used for modelling several outstanding problems in nuclear fusion from a data-centric point of view. One problem — that of detecting plasma confinement states and edge localized modes (ELMs) — is difficult to model with a rules-based system due to the inherent complexity of the phenomena of interest, which motivates the use of an approach with artificial intelligence that discovers those rules automatically. The other problem — that of model selection in Gaussian process tomography — while analytically solvable, is computationally expensive, and would benefit from an approach that can more easily estimate the parameters of interest.

For the detection of ELMs and plasma confinement states (Low, Dither and High), several models were proposed, which are detailed in chapters 6 and 7. They all treat the problem as that of producing a sequence of classifications, where each classification is the probability of being in a certain plasma state, or of the occurrence of an ELM. The models differ from each other, however, with regards to their architectures, which reflect different assumptions about the existing data.

In the simplest case, only localized, short-term correlations in signal time-series are assumed, and concurrently, a convolutional neural network model, with a Viterbi decoding algorithm for post processing, are proposed. To also account for long-term correlations in the temporal data, a second approach proposes also using recurrent neural networks. Both of these approaches use discriminative neural networks that are trained to produce single-point outputs (sequences of events and plasma confinement states) based on the signals fed to them as input. Finally, a third approach uses a sequence-to-sequence model with an encoder-decoder architecture with an attention layer. This model can generate a full output probability distribution, instead of a single point estimate,

making it much more powerful. Pairing it with a beam search algorithm to find samples of high probability from the output distribution, this model achieves excellent results, being capable of fully fitting the training data, achieving only slightly lower scores only for Dither modes. We explain this mostly with the fact that dithers are often quite short, which, given the rather harsh metric used, tends to bring down their score, though other factors cannot be excluded.

For the Bayesian model selection of Gaussian process models for tomographic reconstruction, a convolutional neural network model was proposed, which is detailed in chapter 5. As a first step, the target values which were to be approximated by the network had to be analytically computed for the entire set of measurement data. For each measurement, that target was the set of Gaussian process hyperparameter values that parameterized the most likely corresponding model for the distribution which generated the data. Those parameter values were computed in a grid search, and this was a motivating factor for designing a neural network for classification. A k-fold cross-validation strategy was used to make sure that the choice of train and test sets did not bias the results. The neural network is based on existing architectures for image detection, and achieves very good results on validation data.

Regarding this problem, several avenues of research are still open. For example, it should be possible to take into account the temporal evolution of the SXR measurement data, by using a recurrent neural network model, in parallel with the proposed CNN architecture. In addition, the Gaussian process priors could be chosen to have poloidal coordinate systems and correlations (as opposed to the Euclidean correlations used in this work). Such an approach, however, would open up the door to searching for ideal hyperparameters in an even larger space, making the generation of a labeled training set more difficult.

Nevertheless, as they stand, the algorithms described in this thesis already show the potential for using deep learning to aid in fusion research. The sequence-to-sequence model from Chapter 7 achieved an accuracy comparable to a human expert, and therefore could already be used, for example, for the labeling of data; this can be useful if one is interested in creating large databases for further analysis. The same model could also be used for real-time monitoring of fusion experiments; here, we deal with the evolution in the plasma's confinement states, but it would also be interesting to see whether the algorithm can be

used for detecting other physical events and phenomena. One other avenue of research which was not explored is the potential for transfer learning and domain adaptation, specifically with regards to using the proposed methods for other tokamaks; it would be interesting to verify whether one can adapt the models for use on data from other machines. With regards to the model used for Bayesian estimation of tomographic priors, it should be noted that it can also be used for exploring the effect of the choice of the prior on the posterior solution. For example, one might not select the highest-output, but the k-highest probability classes given by the network, and compute the respective posteriors. In this case, one might be interested not in the highest-evidence model, but rather, in studying the effect of the choice of the prior on the estimated data values.

In short, the described models can learn the correlations existing in data from fusion experiments, and can be used, or readily adapted, for several different tasks. However, supervised deep learning models are always dependent on the existence of good quality labels for data. And, while a lot of data is generated by nuclear fusion experiments, the quality and amount of labels existing for supervised learning techniques is often rather limited, which diminishes the expressiveness, and potential for extrapolation, of the results.

Therefore, for both projects, it would be interesting to use unsupervised, or semi-supervised, learning algorithms to explore the data. For example, with regards to plasma state classification, one could attempt to explore why dithers are consistently more difficult to correctly classify than the other states. To this end, one possible next step would be to use fully generative, unsupervised models based on variational auto-encoders (VAEs)[141], built with the same building blocks as the models proposed in this work. General-purpose VAEs require only data observations, eliminating the need for expert labeling. In addition, they can also offer certain guarantees regarding interpretability and quantification of the uncertainty of their results. Such models could be used for simulation and synthetic data generation: neural networks excel in learning in high-dimensional data spaces, and in situations where one wishes to generate a lot of data from such complex distributions, generative models can provide an efficient alternative to currently existing simulation software, which is often much more computationally intensive. However, certain types of generative models, namely domain invariant VAEs[142], can go even further, and could (in theory)

be useful for facilitating scientific discovery, for example by disentangling factors that explain the behavior of data. Those models do require some labels, as their learning approach is semi-supervised. However, they are highly promising, since they would (in theory) allow for knowing the full joint probability distributions of data observations and the latent factors behind them, which could permit the development of models for design of experiments. For example, if the relationship between a shot's operational space and the confinement states of the plasma were to be fully known, it should be possible to develop tools capable of designing an experiment such that a transition from L to H mode happens at a specified point in time.

Finally, it is also important to mention that most of the supervised learning algorithms presented here, while achieving very good results, do not take any explicit physical or theoretical constraints into account when computing their outputs. The only exception is the sequence-to-sequence architecture, where the beam search step allows for the use of such an approach. It would be interesting to further explore models (whether supervised or unsupervised) that can take physical laws into account, as such laws provide additional information and context.

In conclusion, the research into the use of deep learning for nuclear fusion is still very much an open and recent field, and as such, much work remains to be done with regards to exploring the use of these algorithms for this scientific domain. Nevertheless, the methods presented here already give a glimpse into these models' potential. Future work with applying these methods to fusion, together with continued developments in the capabilities of AI algorithms, means that deep learning can become an important tool for aiding scientists and engineers in the future design of nuclear fusion reactors.

List of Figures

2.1. Schematic drawing of a tokamak and its 2D cross section.	10
2.2. Bolometer system installed at JET	14
2.3. Existing tomographic reconstruction at ASDEX Upgrade	16
3.1. Classification versus regression.	24
3.2. Schema of an individual neuron	25
3.3. Schema of a fully connected neural net	26
3.4. Examples of overfitting, underfitting and a good fit	30
3.5. Sample images from the MNIST digit database	31
3.6. Example of the operation performed by a CNN layer.	32
3.7. Example of a max pooling layer operating on a 1-channel 2×2 input.	33
3.8. Schema of the VGG network architecture.	33
3.9. Example of sequential data.	35
3.10. Schematic representation of unrolled RNN	36
3.11. Schematic representation of a convolutional recurrent network . .	36
3.12. RNN architecture for sequence-to-sequence mapping.	38
3.13. Encoder-decoder architecture for sequence-to-sequence mapping.	38
3.14. Seq2seq architecture with an autoregressive decoder.	40
3.15. Decoding with tree search.	41
3.16. Sequence-to-sequence model with attention.	42
3.17. Example of Gaussian process regression.	45
5.1. An illustrated tomographic projection	59
5.2. ASDEX Upgrade cross-section	61
5.3. Sample tomographic projection from the ASDEX Upgrade tokamak	68
5.4. Number of data samples mapping to each class	71
5.5. CNN used for the SXR tomography project	73

List of Figures

5.6. Top-k accuracy for validation data	76
5.7. Sample tomographic reconstruction and error 1	78
5.8. Sample tomographic reconstruction and error 2	79
5.9. Distribution of the deviations between back-projections and the data	80
6.1. State machine for processing of the CNN outputs	93
6.2. Representation of a CNN	94
6.3. Representation of a convolutional LSTM	95
6.4. Switches between different plasma modes, TCV shot #32195	96
6.5. ELMs and L and H plasma modes, TCV shot #33446	96
6.6. L, D and H modes from a section of TCV shot #32195.	97
6.7. ELMs, and L and H modes from a section of TCV shot #31650. . .	97
6.8. Different types of encoding of the target data distributions, from TCV shot #30262	100
6.9. Sliding temporal windows fed to the CNN	101
6.10. Example of a sequence fed to the LSTM	102
6.11. Architecture of the convolutional NN	103
6.12. Architecture of the convolutional LSTM	103
6.13. ROC curves for ELM detection for the CNN model	109
6.14. Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the CNN.	109
6.15. ROC curves for ELM detection for the Conv-LSTM model	110
6.16. Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the Conv-LSTM	110
6.17. TCV shot #57751 (PD signal) and the Conv-LSTM's classification .	113
6.18. TCV shot #34010 (PD signal) and the Conv-LSTM's classification .	114
6.19. TCV shot #58182 (PD signal) and the Conv-LSTM's classification .	114
6.20. TCV shot #30197 (PD signal) and the Conv-LSTM's classification .	115
6.21. TCV shot #33459 (PD signal) and the Conv-LSTM's classification .	115
6.22. TCV shot #33942 (PD signal) and the Conv-LSTM's classification .	116
7.1. Flow of information in a sequence-to-sequence encoder-decoder model	122

7.2. Representation of the computation of a block of target labels for the sequence-to-sequence model	127
7.3. Illustration of the encoder architecture	129
7.4. Schematic representation of the decoder’s architecture	130
7.5. Representation of the full sequence-to-sequence model’s architecture.	132
7.6. Illustration of the beam search algorithm in the first 3 target timesteps.	133
7.7. Distribution of the κ -statistic score on a per-shot basis	135
7.8. Detection results for shot #42197.	137
7.9. Detection results for shot #61057.	137
7.10. Detection results for shot #61274.	138
7.11. Detection results for shot #32911.	138
7.12. Detection results for shot #61043.	139

List of Tables

5.1. Accuracy mean and standard deviation for validation data across ensemble	75
6.1. κ -statistic scores for the CNN	108
6.2. κ -statistic scores for the Conv-LSTM	110
6.3. κ -statistic scores for selected shots	113
7.1. κ -statistic scores for the conv-LSTM model on the new data	135
7.2. κ -statistic scores for each plasma mode and as a mean, on training and test data	135
7.3. κ -statistic scores for shots with at least one mode whose score was lower than 0.8	136

Bibliography

- [1] G. Merlo. “Flux-tube and global grid-based gyrokinetic simulations of plasma microturbulence and comparisons with experimental TCV measurements”. PhD thesis. École Polytechnique Fédérale de Lausanne, 2016.
- [2] J. D. Lawson. “Some Criteria for a Power Producing Thermonuclear Reactor”. In: *Proceedings of the Physical Society. Section B* 70.1 (Jan. 1957), pp. 6–10.
- [3] J. Wesson. *Tokamaks*. 4th. Oxford University Press, 2011.
- [4] Y. Xu. “A general comparison between tokamak and stellarator plasmas”. In: *Matter and Radiation at Extremes* 1.4 (2016), pp. 192–200.
- [5] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. “GPU computing”. In: *Proceedings of the IEEE* 96.5 (2008), pp. 879–899.
- [6] K. Finken. “Divertor concepts I: Edge physics, divertors, pump limiters”. In: *Fusion science and technology* 41.2T (2002), pp. 330–336.
- [7] F. Matos. “Deep Learning for Plasma Tomography”. MA thesis. Técnico Lisboa, 2016.
- [8] G. Xu, H. Wang, M. Xu, B. Wan, H. Guo, P. Diamond, G. Tynan, R. Chen, N. Yan, D. Kong, et al. “Dynamics of L–H transition and I-phase in EAST”. In: *Nuclear Fusion* 54.10 (2014), p. 103002.
- [9] A. Team et al. “The H-mode of ASDEX”. In: *Nuclear Fusion* 29.11 (1989), p. 1959.
- [10] T. Pütterich, R. Neu, R. Dux, A. Whiteford, M. O’Mullane, H. Summers, et al. “Calculation and experimental test of the cooling factor of tungsten”. In: *Nuclear Fusion* 50.2 (2010), p. 025012.

- [11] T. Odstrčil. “On the origin, properties, and implications of asymmetries in the tungsten impurity density in tokamak plasmas”. PhD thesis. Technische Universität München, 2017.
- [12] J. Radon. “Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten”. In: *Berichte über die Verhandlungen der Königlich-Sächsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse* 69 (1917), p. 262.
- [13] W. A. Kalender. “X-ray computed tomography”. In: *Physics in Medicine & Biology* 51.13 (2006), R29.
- [14] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.
- [15] P. J. Carvalho. “Tomography Algorithms for Real-Time Control in IST-TOK”. PhD thesis. IST, Apr. 2010.
- [16] J. Svensson, A. Werner, and JET-EFDA Contributors. “Current tomography for axisymmetric plasmas”. In: *Plasma Physics and Controlled Fusion* 50.8 (2008), p. 085002.
- [17] D. Vezinet, D. Mazon, D. Clayton, R. Guirlet, M. O’Mullane, and D. Villegas. “Fast nickel and iron density estimation using soft X-ray measurements in Tore Supra: preliminary study”. In: *Fusion Science and Technology* 63.1 (2013), pp. 9–19.
- [18] A. Huber, K. McCormick, P. Andrew, P. Beaumont, S. Dalley, J. Fink, J. Fuchs, K. Fullard, W. Fundamenski, L. Ingesson, et al. “Upgraded bolometer system on JET for improved radiation measurements”. In: *Fusion Engineering and Design* 82.5 (2007), pp. 1327–1334.
- [19] V. Igochine, A. Gude, M. Maraschek, and the ASDEX Upgrade Team. *Hotlink based Soft X-ray Diagnostic on ASDEX Upgrade*. Tech. rep. IPP 1/338. Max-Planck-Institut für Plasmaphysik, 2010.
- [20] A. Bock, H. Doerk, R. Fischer, D. Rittich, J. Stober, A. Burckhart, E. Fable, B. Geiger, A. Mlynek, M. Reich, et al. “Advanced tokamak investigations in full-tungsten ASDEX Upgrade”. In: *Physics of Plasmas* 25.5 (2018), p. 056115.

-
- [21] T. Odstrčil, T. Pütterich, M. Odstrčil, A. Gude, V. Igochine, U. Stroth, and the ASDEX Upgrade Team. “Optimized tomography methods for plasma emissivity reconstruction at the ASDEX Upgrade tokamak”. In: *Review of Scientific Instruments* 87.12 (2016), p. 123505.
- [22] M. Anton, H. Weisen, M. Dutch, W. Von der Linden, F. Buhlmann, R. Chavan, B. Marletaz, P. Marmillod, and P. Paris. “X-ray tomography on the TCV tokamak”. In: *Plasma physics and controlled fusion* 38.11 (1996), p. 1849.
- [23] F. A. Matos, D. R. Ferreira, P. J. Carvalho, and JET Contributors. “Deep learning for plasma tomography using the bolometer system at JET”. In: *Fusion Engineering and Design* 114 (2017), pp. 18–25.
- [24] D. R. Ferreira, P. J. Carvalho, and H. Fernandes. “Deep learning for plasma tomography and disruption prediction from bolometer data”. In: *IEEE Transactions on Plasma Science* 48.1 (2019), pp. 36–45.
- [25] J. Svensson. *Non-parametric tomography using Gaussian processes*. Tech. rep. EFDA-JET-PR(11)24, 2011.
- [26] A. Malinin and M. Gales. “Predictive uncertainty estimation via prior networks”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 7047–7058.
- [27] A. H. Nielsen, G. Xu, J. Madsen, V. Naulin, J. J. Rasmussen, and B. Wan. “Simulation of transition dynamics to high confinement in fusion plasmas”. In: *Physics Letters A* 379.47-48 (2015), pp. 3097–3101.
- [28] J. Lohr, B. Stallard, R. Prater, R. Snider, K. Burrell, R. Groebner, D. Hill, K. Matsuda, C. Moeller, T. Petrie, et al. “Observation of H-mode confinement in the DIII-D tokamak with electron cyclotron heating”. In: *Physical review letters* 60.25 (1988), p. 2630.
- [29] J. Connor and H. Wilson. “A review of theories of the LH transition”. In: *Plasma Physics and Controlled Fusion* 42.1 (2000), R1.
- [30] P. H. Diamond, S. Itoh, K. Itoh, and T. Hahm. “Zonal flows in plasma—a review”. In: *Plasma Physics and Controlled Fusion* 47.5 (2005), R35.
-

- [31] C. Bourdelle. “Staged approach towards physics-based LH transition models”. In: *Nuclear Fusion* 60.10 (2020), p. 102002.
- [32] H. Zohm. “Edge localized modes (ELMs)”. In: *Plasma Physics and Controlled Fusion* 38.2 (1996), p. 105.
- [33] H. Zohm. “Dynamic behavior of the L-H transition”. In: *Physical review letters* 72.2 (1994), p. 222.
- [34] J. Vega, R. Moreno, A. Pereira, G. Rattá, A. Murari, S. Dormido-Canto, S. Esquembri, E. Barrera, and M. Ruiz. “Review of disruption predictors in nuclear fusion: Classical, from scratch and anomaly detection approaches”. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 6375–6379.
- [35] S. Coda, J. Ahn, R. Albanese, S. Alberti, E. Alessi, S. Allan, H. Anand, G. Anastassiou, Y. Andrèbe, C. Angioni, et al. “Overview of the TCV tokamak program: scientific progress and facility upgrades”. In: *Nuclear Fusion* 57.10 (2017), p. 102011.
- [36] A. Webster and R. Dendy. “Statistical characterization and classification of edge-localized plasma instabilities”. In: *Physical review letters* 110.15 (2013), p. 155004.
- [37] J. Greenhough, S. Chapman, R. Dendy, and D. Ward. “Probability distribution functions for ELM bursts in a series of JET tokamak discharges”. In: *Plasma physics and controlled fusion* 45.5 (2003), p. 747.
- [38] A. Shabbir, G. Verdoolaege, O. J. Karadaun, A. J. Webster, R. O. Dendy, and J.-M. Noterdaeme. “Discrimination and visualization of ELM types based on a probabilistic description of inter-ELM waiting times”. In: *41st European Physical Society Conference on Plasma Physics* (Berlin, Germany). 2014.
- [39] J. Vega, A. Murari, G. Vagliasindi, G. Rattá, J.-E. Contributors, et al. “Automated estimation of L/H transition times at JET by combining Bayesian statistics and support vector machines”. In: *Nuclear Fusion* 49.8 (2009), p. 085023.

-
- [40] S. Gonzalez, J. Vega, A. Murari, A. Pereira, S. Dormido-Canto, J. Ramirez, J.-E. contributors, et al. “Automatic location of L/H transition times for physical studies with a large statistical basis”. In: *Plasma Physics and Controlled Fusion* 54.6 (2012), p. 065009.
- [41] A. Murari, G. Vagliasindi, M. K. Zedda, R. Felton, C. Sammon, L. Fortuna, and P. Arena. “Fuzzy logic and support vector machine approaches to regime identification in JET”. In: *IEEE Transactions on Plasma Science* 34.3 (2006), pp. 1013–1020.
- [42] A. Lukianitsa, F. Zhdanov, and F. Zaitsev. “Analyses of ITER operation mode using the support vector machine technique for plasma discharge classification”. In: *Plasma Physics and Controlled Fusion* 50.6 (2008), p. 065013.
- [43] G. R. A. Akkermans. *Real-time, Model-based Event Detection in Tokamaks*. Master’s Thesis. Jan. 2017.
- [44] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [45] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O’Reilly Media, Incorporated, 2019.
- [46] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [47] K. Hornik, M. Stinchcombe, H. White, et al. “Multilayer feedforward networks are universal approximators.” In: *Neural networks* 2.5 (1989), pp. 359–366.
- [48] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. “Handwritten digit recognition: Applications of neural network chips and automatic learning”. In: *IEEE Communications Magazine* 27.11 (1989), pp. 41–46.
- [49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

- [50] Y. LeCun, C. Cortes, and C. Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010).
- [51] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [52] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *International Conference on Learning Representations* (San Diego, USA). 2015. URL: <https://iclr.cc/archive/www/2015.html>.
- [53] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. “Temporal convolutional networks: A unified approach to action segmentation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 47–54.
- [54] D. Zhang and D. Wang. “Relation classification: Cnn or rnn?” In: *Natural Language Understanding and Intelligent Applications*. Springer, 2016, pp. 665–675.
- [55] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *The 2014 Conference on Empirical Methods In Natural Language Processing* (Doha, Qatar). Oct. 2014. URL: <https://emnlp2014.org/>.
- [56] J. L. Elman. “Finding structure in time”. In: *Cognitive science* 14.2 (1990), pp. 179–211.
- [57] A. Hassan and A. Mahmood. “Convolutional recurrent deep learning model for sentence classification”. In: *Ieee Access* 6 (2018), pp. 13949–13957.
- [58] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [59] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

- [60] P. Koehn, F. J. Och, and D. Marcu. *Statistical phrase-based translation*. Tech. rep. UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2003.
- [61] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: *International Conference on Learning Representations* (San Diego, USA). 2015. URL: <https://iclr.cc/archive/www/2015.html>.
- [62] M.-T. Luong, H. Pham, and C. D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Conference on Empirical Methods in Natural Language Processing* (Lisbon, Portugal). 2015. URL: <http://www.emnlp2015.org/>.
- [63] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. Cambridge, MA: MIT press, 2006.
- [64] F. Matos, J. Svensson, A. Pavone, T. Odstrčil, and F. Jenko. “Deep learning for Gaussian process soft x-ray tomography model selection in the ASDEX Upgrade tokamak”. In: *Review of Scientific Instruments* 91.10 (2020), p. 103501.
- [65] F. Matos, V. Menkovski, F. Felici, A. Pau, F. Jenko, T. Team, E. M. Team, et al. “Classification of tokamak plasma confinement states with convolutional recurrent neural networks”. In: *Nuclear Fusion* 60.3 (2020), p. 036022.
- [66] F. Matos, V. Menkovski, A. Pau, G. Marceca, F. Jenko, and the TCV Team. “Plasma confinement mode classification using a sequence-to-sequence neural network with attention”. In: *Nuclear Fusion* 61.4 (2021), p. 046019.
- [67] L. Ingesson, B. Alper, B. Peterson, and J.-C. Vallet. “Chapter 7: Tomography diagnostics: bolometry and soft-x-ray detection”. In: *Fusion science and technology* 53.2 (2008), pp. 528–576.
- [68] J. Mlynar, V. Weinzettl, G. Bonheure, A. Murari, and JET-EFDA contributors. “Inversion techniques in the soft-X-ray tomography of fusion plasmas: toward real-time applications”. In: *Fusion Science and Technology* 58.3 (2010), pp. 733–741.

- [69] J. Mlynar, T. Craciunescu, D. R. Ferreira, P. Carvalho, O. Ficker, O. Grover, M. Imrisek, J. Svoboda, and Jet Contributors. “Current research into applications of tomography for fusion diagnostics”. In: *Journal of Fusion Energy* 38.3-4 (2019), pp. 458–466.
- [70] A. N. Tikhonov. “Regularization of incorrectly posed problems”. In: *Dokl. Akad. Nauk. SSSR*. Vol. 153. 1963, p. 49.
- [71] A. N. Tikhonov. “Solution of incorrectly formulated problems and the regularization method”. In: *Dokl. Akad. Nauk. SSSR*. Vol. 151. 1963, p. 501.
- [72] V. Loffelmann, J. Mlynar, M. Imrisek, D. Mazon, A. Jardin, V. Weinzettl, and M. Hron. “Minimum Fisher Tikhonov regularization adapted to real-time tomography”. In: *Fusion Science and Technology* 69.2 (2016), pp. 505–513.
- [73] A. Jardin, J. Bielecki, D. Mazon, J. Dankowski, K. Król, Y. Peysson, and M. Scholz. “Neural networks: from image recognition to tokamak plasma tomography”. In: *Laser and Particle Beams* 37.2 (2019), pp. 171–175.
- [74] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight Uncertainty in Neural Network”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1613–1622. URL: <https://icml.cc/2015/>.
- [75] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1050–1059. URL: <http://proceedings.mlr.press/v48/gal16.html>.
- [76] A. Pavone, J. Svensson, A. Langenberg, N. Pablant, U. Hoefel, S. Kwak, R. Wolf, and the Wendelstein 7-X Team. “Bayesian uncertainty calculation in neural network inference of ion and electron temperature profiles at W7-X”. In: *Review of Scientific Instruments* 89.10 (2018), 10K102.

-
- [77] A. Radford, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [78] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved techniques for training gans”. In: *Advances in neural information processing systems* (Barcelona, Spain). 2016, pp. 2234–2242. URL: <http://papers.nips.cc/book/advances-in-neural-information-processing-systems-29-2016>.
- [79] R. M. Lewitt. “Reconstruction algorithms: transform methods”. In: *Proceedings of the IEEE* 71.3 (1983), pp. 390–408.
- [80] A. Murari, E. Joffrin, R. Felton, D. Mazon, L. Zabeo, R. Albanese, P. Arena, G. Ambrosino, M. Ariola, O. Barana, et al. “Development of real-time diagnostics and feedback algorithms for JET in view of the next step”. In: *Plasma physics and controlled fusion* 47.3 (2005), p. 395.
- [81] A. Jardin, J. Bielecki, D. Mazon, J. Dankowski, K. Król, Y. Peysson, and M. Scholz. “Synthetic X-ray Tomography Diagnostics for Tokamak Plasmas”. In: *Journal of Fusion Energy* (2020), pp. 1–11.
- [82] L. Ingesson, P. Böcker, R. Reichle, M. Romanelli, and P. Smeulders. “Projection-space methods to take into account finite beam-width effects in two-dimensional tomography algorithms”. In: *JOSA A* 16.1 (1999), pp. 17–27.
- [83] M. Odstrcil, J. Mlynar, T. Odstrcil, B. Alper, A. Murari, and JET contributors. “Modern numerical methods for plasma tomography optimisation”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 686 (2012), pp. 156–161.
- [84] D. Carvalho, D. Ferreira, P. Carvalho, M. Imrisek, J. Mlynar, H. Fernandes, and J. Contributors. “Deep neural networks for plasma tomography with applications to JET and COMPASS”. In: *Journal of Instrumentation* 14.09 (2019), p. C09011.
-

- [85] D. Li, J. Svensson, H. Thomsen, F. Medina, A. Werner, and R. Wolf. “Bayesian soft X-ray tomography using non-stationary Gaussian Processes”. In: *Review of Scientific Instruments* 84.8 (2013), p. 083506.
- [86] D. J. MacKay. “Comparison of approximate methods for handling hyperparameters”. In: *Neural computation* 11.5 (1999), pp. 1035–1068.
- [87] D. MacKay. “Bayesian Methods for Adaptive Models”. PhD thesis. Caltech, 1991.
- [88] C. Lechte, G. Conway, T. Görler, C. Tröster-Schmid, and the ASDEX Upgrade Team. “X mode Doppler reflectometry k-spectral measurements in ASDEX Upgrade: experiments and simulations”. In: *Plasma Physics and Controlled Fusion* 59.7 (2017), p. 075006.
- [89] F. Chollet et al. *Keras*. <https://github.com/fchollet/keras>. 2015.
- [90] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: <http://proceedings.mlr.press/v37/ioffe15.html>.
- [91] X. Li, S. Chen, X. Hu, and J. Yang. “Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, USA). June 2019, pp. 2677–2685. DOI: [10.1109/CVPR.2019.00279](https://doi.org/10.1109/CVPR.2019.00279). URL: <http://cvpr2019.thecvf.com/>.
- [92] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [93] G. James. *An introduction to statistical learning: with applications in R*. New York, NY: Springer, 2013. ISBN: 978-1-4614-7138-7.
- [94] T. Zhang, X. Gao, S. Zhang, Y. Wang, X. Han, Z. Liu, B. Ling, E. Team, et al. “Characteristics of dithering cycles during the L–I–H transition on Experimental Advanced Superconducting Tokamak (EAST)”. In: *Physics Letters A* 377.28-30 (2013), pp. 1725–1735.

-
- [95] A. Loarte, G. Huijsmans, S. Futatani, L. Baylor, T. Evans, D. Orlov, O. Schmitz, M. Becoulet, P. Cahyna, Y. Gribov, et al. “Progress on the application of ELM control schemes to ITER scenarios from the non-active phase to DT operation”. In: *Nuclear Fusion* 54.3 (2014), p. 033007.
- [96] D. Rastovic. “Fuzzy scaling and stability of tokamaks”. In: *Journal of fusion energy* 28.1 (2009), pp. 101–106.
- [97] D. Humphreys, G. Ambrosino, P. de Vries, F. Felici, S. H. Kim, G. Jackson, A. Kallenbach, E. Kolemen, J. Lister, D. Moreau, et al. “Novel aspects of plasma control in ITER”. In: *Physics of Plasmas* 22.2 (2015), p. 021806.
- [98] Y. R. Martin, T. Takizuka, and the ITPA CDBM H-mode Threshold Data Group. “Power requirement for accessing the H-mode in ITER”. In: *Journal of Physics: Conference Series* 123 (July 2008), p. 012033. doi: [10.1088/1742-6596/123/1/012033](https://doi.org/10.1088/1742-6596/123/1/012033).
- [99] F. Ryter, K. Buchl, C. Fuchs, O. Gehre, O. Gruber, A. Herrmann, A. Kallenbach, M. Kaufmann, W. Koppendorfer, F. Mast, et al. “H-mode results in ASDEX Upgrade”. In: *Plasma physics and controlled fusion* 36.7A (1994), A99.
- [100] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going Deeper With Convolutions”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, USA). June 2015. URL: <http://www.pamitc.org/cvpr15/>.
- [101] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj. “Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks.” In: *IEEE Trans. Industrial Electronics* 63.11 (2016), pp. 7067–7075.
- [102] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Red Hook, USA: Curran Associates, Inc., 2012, pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
-

- [103] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. “Efficient Object Localization Using Convolutional Networks”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, USA). June 2015. URL: <http://www.pamitc.org/cvpr15/>.
- [104] T. Lähivaara, L. Kärkkäinen, J. M. Huttunen, and J. S. Hesthaven. “Deep convolutional neural networks for estimating porous material parameters with ultrasound tomography”. In: *The Journal of the Acoustical Society of America* 143.2 (2018), pp. 1148–1158.
- [105] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, and H. Adeli. “Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals”. In: *Computers in biology and medicine* 100 (2018), pp. 270–278.
- [106] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman. “Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks”. In: *Journal of Sound and Vibration* 388 (2017), pp. 154–170.
- [107] S. Malek, F. Melgani, and Y. Bazi. “One-dimensional convolutional neural networks for spectroscopic signal regression”. In: *Journal of Chemometrics* 32.5 (2018), e2977.
- [108] P. Golik, Z. Tüske, R. Schlüter, and H. Ney. “Convolutional neural networks for acoustic modeling of raw time signal in LVCSR”. In: *Sixteenth annual conference of the international speech communication association* (Dresden, Germany). 2015. URL: <http://interspeech2015.org/>.
- [109] S. Kiranyaz, T. Ince, and M. Gabbouj. “Real-time patient-specific ECG classification by 1-D convolutional neural networks”. In: *IEEE Transactions on Biomedical Engineering* 63.3 (2015), pp. 664–675.
- [110] C. A. Ronao and S.-B. Cho. “Human activity recognition with smartphone sensors using deep learning neural networks”. In: *Expert systems with applications* 59 (2016), pp. 235–244.
- [111] M. Sundermeyer, R. Schlüter, and H. Ney. “LSTM neural networks for language modeling”. In: *Thirteenth annual conference of the international speech communication association*. 2012.

-
- [112] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang. “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data”. In: *Transportation Research Part C: Emerging Technologies* 54 (2015), pp. 187–197.
- [113] S. Venugopalan, L. A. Hendricks, R. Mooney, and K. Saenko. “Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1961–1966. doi: [10 . 18653 / v1 / D16 - 1204](https://doi.org/10.18653/v1/D16-1204). URL: <https://www.aclweb.org/anthology/D16-1204>.
- [114] A. Meakins, D. McDonald, et al. “The application of classification methods in a data driven investigation of the JET L–H transition”. In: *Plasma Physics and Controlled Fusion* 52.7 (2010), p. 075005.
- [115] F. 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”. In: *Physical Review Letters* 49.19 (1982), p. 1408.
- [116] S.-I. Itoh and K. Itoh. “Model of L to H-Mode Transition in Tokamak”. In: *Phys. Rev. Lett.* 60 (22 May 1988), pp. 2276–2279. doi: [10 . 1103 / PhysRevLett . 60 . 2276](https://doi.org/10.1103/PhysRevLett.60.2276).
- [117] N. Basse, S. Zoletnik, G. Antar, J. Baldzuhn, A. Werner, et al. “Characterization of turbulence in L-and ELM-free H-mode Wendelstein 7-AS plasmas”. In: *Plasma physics and controlled fusion* 45.4 (2003), p. 439.
- [118] Y. Martin. *ELMing H-mode accessibility in shaped TCV plasmas*. Tech. rep. TCV Team, 2001.
- [119] D. Jurafsky and J. H. Martin. *Speech and Language Processing, second edition*. USA: Prentice Hall, 2014. ISBN: 9780131873216.
- [120] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. “High-dimensional sequence transduction”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (Vancouver, Canada). IEEE.
-

- 2013, pp. 3178–3182. URL: <https://www2.securecms.com/ICASSP2013/default.asp>.
- [121] F. Hofmann, J. Lister, W. Anton, S. Barry, R. Behn, S. Bernel, G. Besson, F. Buhlmann, R. Chavan, M. Corboz, et al. “Creation and control of variably shaped plasmas in TCV”. In: *Plasma Physics and Controlled Fusion* 36.12B (1994), B277.
- [122] S. Coda et al. “PHYSICS RESEARCH ON THE TCV TOKAMAK FACILITY: FROM CONVENTIONAL TO ALTERNATIVE SCENARIOS AND BEYOND”. In: *Nuclear Fusion* 59 112023 (2019).
- [123] J.-M. Moret, F. Buhlmann, and G. Tonetti. “Fast single loop diamagnetic measurements on the TCV tokamak”. In: *Review of Scientific Instruments* 74.11 (2003), pp. 4634–4643.
- [124] D. Harris and S. Harris. *Digital Design and Computer Architecture*. Computer organization bundle, VHDL Bundle. Elsevier Science, 2010. ISBN: 9780080547060.
- [125] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.
- [126] Q. Zheng, M. Yang, J. Yang, Q. Zhang, and X. Zhang. “Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process”. In: *IEEE Access* 6 (2018), pp. 15844–15869.
- [127] V. Dumoulin and F. Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).
- [128] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
- [129] T. Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.
- [130] X. Liu. “Classification accuracy and cut point selection”. In: *Statistics in medicine* 31.23 (2012), pp. 2676–2686.

-
- [131] J. R. Landis and G. G. Koch. “The measurement of observer agreement for categorical data”. In: *biometrics* (1977), pp. 159–174.
- [132] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* (Lake Tahoe, Nevada). Dec. 2012, pp. 1097–1105. URL: <https://nips.cc/Conferences/2012>.
- [133] D. Ciregan, U. Meier, and J. Schmidhuber. “Multi-column deep neural networks for image classification”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.
- [134] A. Graves, A.-r. Mohamed, and G. Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.
- [135] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.
- [136] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning* (Lille, France). July 2015, pp. 2048–2057. URL: <https://icml.cc/Conferences/2015/>.
- [137] F. Stahlberg. “Neural machine translation: A review”. In: *Journal of Artificial Intelligence Research* 69 (2020), pp. 343–418.
- [138] A. Graves. “Sequence transduction with recurrent neural networks”. In: *arXiv preprint arXiv:1211.3711* (2012).
- [139] M. Łuczak. “Hierarchical clustering of time series data with parametric derivative dynamic time warping”. In: *Expert Systems with Applications* 62 (2016), pp. 116–130.
-

Bibliography

- [140] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [141] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *ICLR Conference Proceedings* (Banff, Canada). Apr. 2014. URL: <https://iclr.cc/archive/2014/old-site/>.
- [142] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. “Diva: Domain invariant variational autoencoders”. In: *Medical Imaging with Deep Learning*. PMLR. 2020, pp. 322–348.

Part IV:
Appendices

A. Original Publications

PAPER

Classification of tokamak plasma confinement states with convolutional recurrent neural networks

To cite this article: F. Matos *et al* 2020 *Nucl. Fusion* **60** 036022

View the [article online](#) for updates and enhancements.



IOP | ebooks™

Bringing together innovative digital publishing with leading authors from the global scientific community.

Start exploring the collection—download the first chapter of every title for free.

Classification of tokamak plasma confinement states with convolutional recurrent neural networks

F. Matos¹, V. Menkovski², F. Felici³, A. Pau³, F. Jenko¹, the TCV Team^{3,a} and the EUROfusion MST1 Team^b

¹ Max Planck Institute for Plasma Physics, Boltzmannstraße 2, 85748 Garching, Germany

² Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands

³ École Polytechnique Fédérale de Lausanne (EPFL), Swiss Plasma Center (SPC), CH-1015 Lausanne, Switzerland

E-mail: francisco.matos@ipp.mpg.de

Received 19 November 2019, revised 10 January 2020

Accepted for publication 16 January 2020

Published 20 February 2020



CrossMark

Abstract

During a tokamak discharge, the plasma can vary between different confinement regimes: low (L), high (H) and, in some cases, a temporary (intermediate state), called dithering (D). In addition, while the plasma is in H mode, edge localized modes (ELMs) can occur. The automatic detection of changes between these states, and of ELMs, is important for tokamak operation. Motivated by this, and by recent developments in deep learning, we developed and compared two methods for automatic detection of the occurrence of L-D-H transitions and ELMs, applied on data from the TCV tokamak. These methods consist in a convolutional neural network and a convolutional long short term memory neural network. We measured our results with regards to ELMs using ROC curves and Youden's score index, and regarding state detection using Cohen's Kappa index.

Keywords: CNN, LSTM, deep learning, ELM, H mode, L mode, dither

(Some figures may appear in colour only in the online journal)

1. Introduction

In a fusion experiment, plasma can typically be described as being in one of two different confinement regimes or modes: low (L) and high (H). Furthermore, the plasma can also sometimes be described as being in a third, additional, mode, called the intermediate or dithering (D) [1] phase. In addition, when the plasma is in H mode, edge localized modes (ELMs) can periodically occur.

Current tokamaks regularly run in H mode, which motivates the necessity for some measure of control (and therefore, detection) of ELMs and transitions between plasma modes. Furthermore, it is expected that future machines will

also run in the same operating conditions [2]. Thus, the development of automated, data-based approaches to automatically detect the occurrence of certain events would be useful for both existing and future tokamak experiments and operation [3]. A detector would not only simplify and speed-up the post-experimental, offline analysis of shots, but also (ideally) detect ELMs and plasma state rapidly enough to allow for its usage in the real-time control systems of a fusion experiment, for purposes of plasma control and real-time discharge monitoring and supervision [4].

Due to uncertainties in the scaling laws, it is difficult to determine, *a priori*, when, during a discharge, a switch between different plasma modes will occur [5]. Nevertheless, physicists can usually pinpoint, through a post-experimental visual analysis of several diagnostic signal time-traces, at what point in time any transitions between different modes did take place. Similarly to transitions between plasma modes, the occurrence of an ELM can usually be pinpointed by looking

^a See Coda *et al* (<https://doi.org/10.1088/1741-4326/ab25cb>) for the TCV team.

^b See Labit *et al* (<https://doi.org/10.1088/1741-4326/ab2211>) for the EUROfusion MST1 team.

at the time-traces of several diagnostics from a plasma discharge post-shot. Yet through analysis of signals, some types of ELMs can be easily confused with dithers; a distinction between the two phenomena can not always be clearly made [6].

Although the identification by an expert, through post experimental visual analysis of signal time-traces, of a single ELM, or a single transition between plasma modes, is relatively straightforward for a typical shot, it becomes much more cumbersome to carry out that analysis effectively for many shots, especially when the associated time-series data is long, and when a shot has many transitions between different modes.

Recent advances in the ML field with the introduction of deep learning (DL) approaches deal with exactly such challenges. In the past years, the field of deep learning has brought about significant advances in computer vision and sequential data processing. Convolutional neural networks (CNNs) have proven adept at localization, recognition and detection tasks in both two-dimensional [7–11] and one-dimensional [12–17] data (i.e. signal analysis) in many different fields of science. In addition, long short-term memory (LSTM) networks, which are one type of recurrent neural network, have been successfully used for processing of sequential data where one expects correlations to exist across time, namely, automatic translation, natural language modelling [18], traffic analysis [19], and automated video description [20]. These tasks are much akin to what one can expect to find in terms of processing fusion shot data.

Given this, a deep learning approach is well motivated to address this challenge. Specifically, deep neural network models offer particular advantages when modeling high-dimensional data as given in this setting. In this work, we develop an approach for automatic classification of L-D-H plasma states and detection of ELMs based on two deep neural network models. The first model is based on a sliding-window feed-forward neural network, specifically a CNN. The second model is based on a recurrent neural network (RNN), specifically a long short-term memory network (LSTM) with convolutional layers. The first model captures the local correlations within the windows to classify the transitions between plasma states from the shape of the signals. The second model extends this to capturing longer-term dependencies in the evolution of the states with the recurrent neural network layers.

We empirically demonstrate the approach on data collected from the TCV tokamak experiment, labelled by an ensemble of experts. The presented results demonstrate the effectiveness of the proposed model to detect the state and events of the plasma. We further discuss the trade-offs between increased precision and increased complexity of both models.

This paper is organized as follows: section 2 discusses related work and section 3 describes the physical phenomena being analyzed. Section 4 formalizes our problem, details the data we have available, and explains our decisions regarding how we model the data and design and train the neural networks. Section 5 gives an overview of the metrics we used to evaluate our results and our rationale behind using those

metrics. Section 6 gives an overview of the results achieved, and we wrap up with a discussion in section 7.

2. Previous work

Several different approaches for automated detection of events in plasma experiments exist. One such approach is to use threshold-based detectors. This corresponds to defining a point or series of points (in time) at which a signal surpasses a certain amplitude as corresponding to a detection [21–23], with additional constraints such as an increasing probability of the occurrence of an ELM as time passes since the last one. These approaches are limited to simple thresholding and cannot compute complex patterns in the data. Other work builds upon methods such as Kalman Filters to model the expected characteristics of the signal over a period of time, whilst also keeping track (in each time point) of the current plasma mode, according to a pre-defined model. In both of these cases, a detection algorithm's performance depends on the extent to which the theoretical assumptions and mathematical descriptions as to how the signals should behave are correct, whether those assumptions are exhaustive (i.e. whether there may be additional causes which are unaccounted for), and whether some of those assumptions are more important than others; in other words, it is difficult to design an exhaustive rule-based system to detect the occurrence of transitions between plasma modes, as well as to detect ELMs.

The alternative is to use a purely data-based, supervised, machine learning (ML), approach, whereby a set of data, previously manually labeled by an expert (for example, through visual analysis), is used to train a detector. In this case, one does not specify which characteristics or correlations in the data are thought to correspond to the occurrence of an event; rather, it is expected that the algorithm can automatically learn what those correlations are, based on the labels, and then use the learned data features to make correct classifications on new data. Examples of such work are the usage of support vector machines (SVMs) [24–27] and multi-layer perceptron (MLP) neural networks [28] on data from several tokamaks for detection of L-H transitions, classification of L and H modes, and detection of ELMs.

This type of scenario is, indeed, well suited for application of ML methods towards enabling automation. However, traditional ML methods such as SVMs and MLPs typically have limitations when faced with data with complex dynamics, such as the long sequences (i.e. signal time-series) present in this environment. SVMs typically depend on expert-defined feature engineering, which, while being superior to simple threshold-based detectors, is nevertheless insufficient when considering the complex data correlations which are observed in this setting. On the other hand, MLPs, while not requiring that sort of expert-defined input, are very inefficient when compared to modern deep learning models such as CNNs and RNNs, requiring much larger numbers of neurons and layers to perform the same task. These limitations are what motivate us to use deep learning approaches instead.

3. Background

3.1. Low, dither and high plasma confinement modes

When a discharge starts, the plasma is considered to be in L confinement mode. Once a certain threshold of input heating power to the plasma is reached [29], the plasma can spontaneously transition into H confinement mode. Originally discovered at the ASDEX-Upgrade Tokamak [30], H mode is nowadays regularly observed in almost all other machines [31]. H mode is characterized by the appearance, in the plasma edge, of a steep gradient in the electron density and the electron/ion temperatures, and a reduction in the transport of particles and energy. As a consequence of this edge transport barrier, the temperature and energy in the plasma core increase. When compared to L mode, H mode allows for a larger amount of stored plasma energy per input power, thus rendering the fusion process more efficient. Yet the actual input power threshold that triggers the transition between the two modes is dependent on many factors, such as, for example, the configuration of the magnetic field, plasma density, and plasma size [5]. Furthermore, when the input heating power passes the aforementioned threshold but a change from L to H mode does not immediately occur, the plasma can be considered to be in a D [1] phase. In this case, a temporary, weak, edge transport barrier starts to develop at the plasma edge, only to collapse and reappear in rapid succession [29]. These oscillations then repeat themselves until the plasma transitions into L or H mode. The localization of transitions into, and out of, D mode can, however, be difficult to identify, and there are often disagreements between experts as to which periods of a shot are in a Dithering phase [32].

3.2. Edge localized modes

When the plasma enters H mode, the corresponding accumulation of energy and the large pressure gradient at the plasma edge can trigger the occurrence of edge localized modes (ELMs). These consist of periodic bursts of particles and energy which, if a long amount of time passes between successive ELMs, can impose a significant power load on the divertor, potentially damaging it. However, ELMs also allow for the periodic removal of accumulated impurities from the plasma, and for a relaxation of the plasma density, which can otherwise increase as the H mode progresses, eventually triggering a disruption [33]. On the other hand, frequent, less energetic, ELMs lower the power load on the divertor, at the cost of reduced plasma confinement. Thus, tokamak operation requires knowledge of the occurrence of ELMs, in particular for larger machines where ELMs may cause deterioration of in-vessel components. Although several different types of ELMs exist, for the purposes of this work, we did not make any distinctions between them—we train the models to detect all occurring ELMs equally, regardless of their subclass.

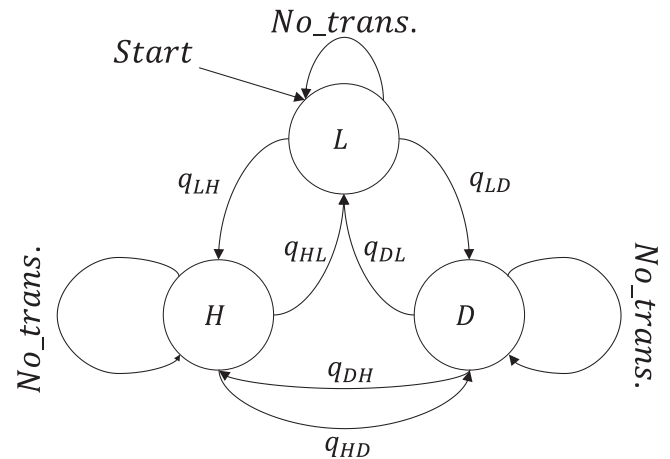


Figure 1. State machine for processing of the CNN outputs.

4. Methods

4.1. Problem formulation and approach

To develop a model for this task, we formulate the problem as follows: we observe a sequence of measurements x_t for $0 < t \leq N$ from the sensors for each shot. These observations are conditioned on the state of the plasma z_t at corresponding time t , where $z_t \in Z$ and $Z : \{ \text{'Low'}, \text{'Dither'}, \text{'High'} \}$. Our goal is to find the most likely sequences $\hat{z}_{1:N}$ and $\hat{e}_{1:N}$ that explain the observations $x_{0:N}$. We define $\hat{z}_{1:N}$ as

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} p(z_1, z_2, \dots, z_N).$$

We choose to represent the joint probability $p(z_1, z_2, \dots, z_N)$ as

$$p(z_1, z_2, \dots, z_N) = \log(p(z_1|x_{0:1})p(z_2|x_{0:2}, z_1) \dots p(z_N|x_{0:N}, z_{N-1}))$$

where $p(z_t|x_{0:t}, z_{t-1})$ denotes the probability of observing state z at time t , given the sequence of observed signals x from time 0 to time t and the previous state z_{t-1} . This yields

$$\begin{aligned} \hat{z}_{1:N} &= \arg \max_{z_{1:N}} \log\left(\prod_t p(z_t|x_{0:t}, z_{t-1})\right) \\ &= \arg \max_{z_{1:N}} \sum_t \log p(z_t|x_{0:t}, z_{t-1}). \end{aligned}$$

Similarly, we define $\hat{e}_{1:N}$ as

$$\hat{e}_{1:N} = \arg \max_{e_{1:N}} p(e_1, e_2, \dots, e_N)$$

while representing $p(e_1, e_2, \dots, e_N)$ as

$$p(e_1, e_2, \dots, e_N) = (p(e_1|x_{0:1})p(e_2|x_{0:2}) \dots p(e_N|x_{0:N}))$$

where $p(e_t|x_{0:t})$ denotes the probability of an ELM occurring at time t given the observations x from time 0 to time t . This then yields

$$\begin{aligned} \hat{e}_{1:N} &= \arg \max_{e_{1:N}} \log\left(\prod_t p(e_t|x_{0:t})\right) \\ &= \arg \max_{e_{1:N}} \sum_t \log p(e_t|x_{0:t}). \end{aligned}$$

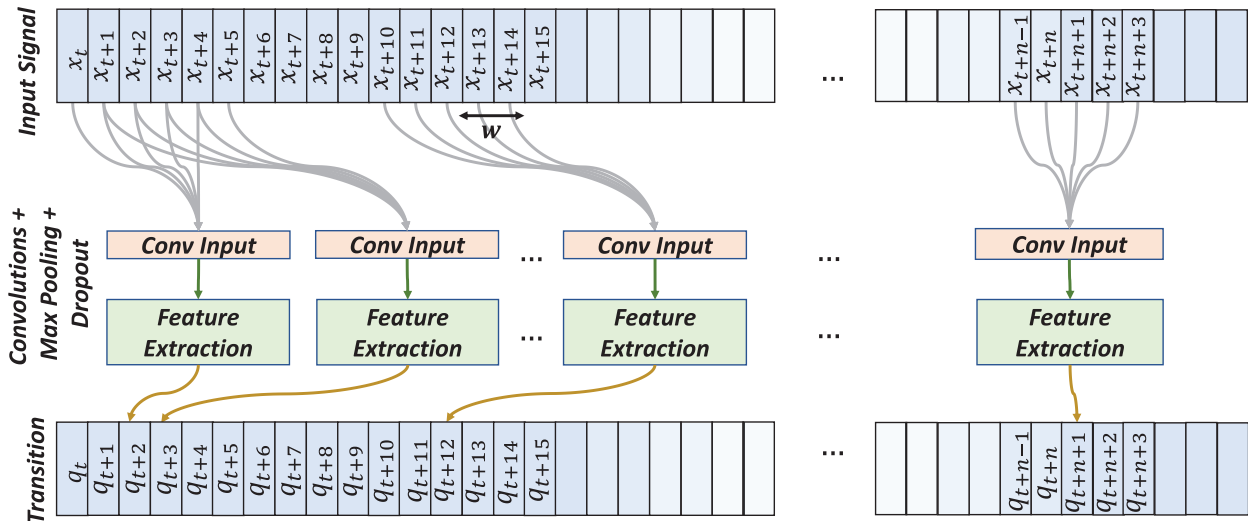


Figure 2. Representation of how a CNN can be used to model the transitions between different plasma modes. The network's output prediction for a time slice t depends only on the data features in a defined region immediately surrounding t .

To find $\hat{z}_{1:N}$ and $\hat{e}_{1:N}$ we develop two models. The first model is trained to detect the transitions between the different states of the plasma defined as $q_t \in Q$ where $Q : \{ \text{'Low} \rightarrow \text{Dither}' , \text{'Dither} \rightarrow \text{Low}' , \text{'Low} \rightarrow \text{High}' , \text{'High} \rightarrow \text{Low}' , \text{'Dither} \rightarrow \text{High}' , \text{'High} \rightarrow \text{Dither}' , \text{'NoTransition'} \}$ and to detect the ELM events as $e_t \in E$ where $E : \{ \text{'ELM}' , \text{'NoELM'} \}$.

We implement this model with a feed-forward CNN that processes a window of observations $x_{t-w}, \dots, x_t, \dots, x_{t+w}$ and produces a probability distribution over the transitions $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ and over the presence of an ELM $p(\text{ELM}_t | x_{t-w:t+w})$ at t .

We now define the probability of transitioning to z_t after being in z_{t-1} ($p(z_t | x_{0:t}, z_{t-1})$) with our model $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ where w is the number of observations around t , therefore:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} \sum_t \log p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w}).$$

Practically, we implement the $\arg \max$ given above as a state evolution of a final state machine $S_t(z^{(a)} \rightarrow z^{(b)})$ where $z^{(a)}$ and $z^{(b)}$ are elements in Z and the transition probabilities are given by $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w:t+w})$ at time t (see figure 1). The evolution of the state machine produces several possible sequences of states, and the one most likely to have generated the observed sequence of transitions can be found through an implementation of the Viterbi algorithm [34].

The first model can capture the localized correlations in the signals that indicate the transition of the state of plasma well. However, it is incapable of capturing the longer distance correlations that may be present in the signal. To generalize the approach further, we introduce a sequence model that models the full sequence of observations up to time t and produce a probability distribution $p(z_t | x_{0:t})$ for $0 < t \leq N$, as well as a distribution over the presence of the ELMs ($p(\text{ELM}_t | x_{0:t})$). This model is implemented by extending the CNN with a recurrent (LSTM) neural network. In this case, the model now observes a sequence of sliding windows $x_{t-w}, \dots, x_t, \dots, x_{t+w}$ for each t in the range $\{0, \dots, t\}$.

The first model has a lower computational complexity and can be trained more efficiently, as we only need windows of the signal with or without the different transitions, but it is limited to the information only present in the given window (see figure 2). Increasing the size of this window that forms the context, increases the complexity both of the model and of dealing with multiple transitions appearing.

The second model addresses these challenges by modeling the sequence rather than a fixed window (see figure 3). As a sequential model, it has an internal representation of the past observations x_0, \dots, x_t , that enable it to weigh-in the likelihood of transition based on information in the more distant past [35]. The LSTM effectively assumes the role of the finite state machine and so the model can directly model the state of the plasma rather than the transitions. However, it introduces higher level of complexity, particularly for training, as we need to train on sequences rather than fixed-length windows.

4.2. Data and event features

For the purposes of this work, we have assembled a dataset based on the time-traces of four signals originating in the TCV tokamak [36, 37]. We opted, for the purposes of this work, to use the same, limited set of diagnostic signals that experimentalists use to determine, in post-shot analysis, the state of the plasma (figure 4).

- (i) *Photodiode (PD) signal.* Corresponds to the measurements given by the photodiode diagnostic at TCV along a vertical chord, measuring the line-integrated emitted visible radiation; the photodiode has an H_α filter which measures radiation at 653.3 nm.

Transitions between different plasma states, as well as ELMs, can be most easily observed through analysis of the photodiode (PD) signal (figure 5). Transitions from L to H mode are characterized by a sudden drop in the baseline value of the signal, whereas transitions back

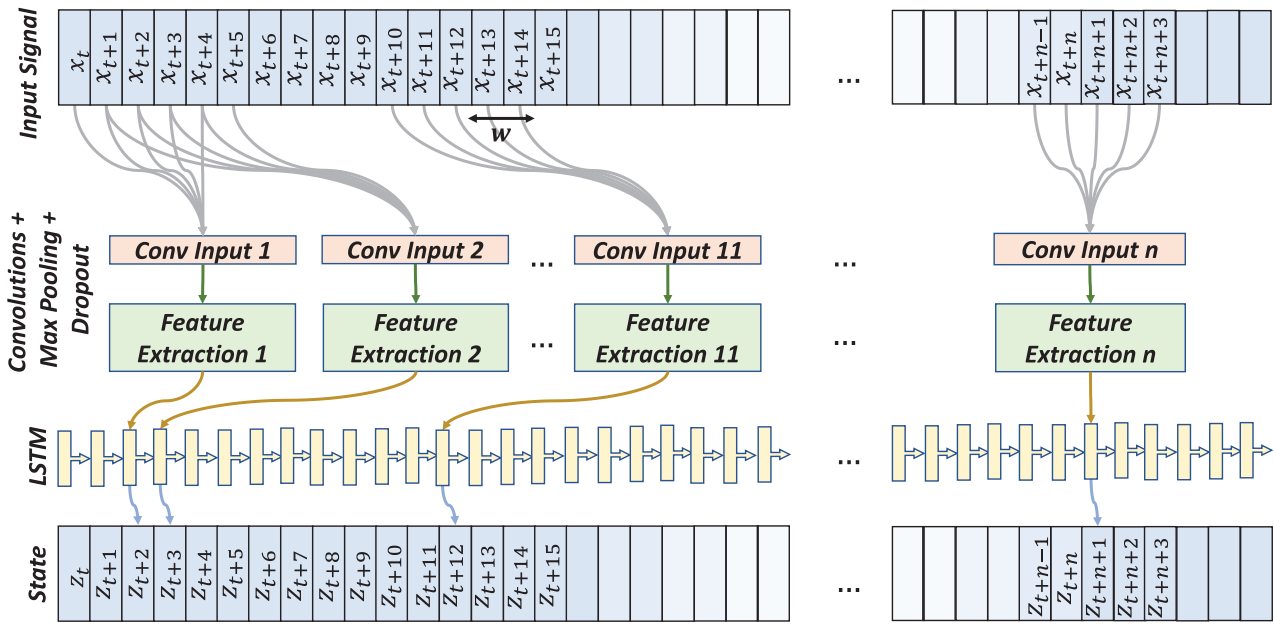


Figure 3. Schematic representation of the flow of data inside a convolutional LSTM neural network. The network’s prediction (i.e. output probability) at any time t of a shot depends not only on whatever features the convolutional layers have extracted from the points immediately around t , but also on features extracted in the past.

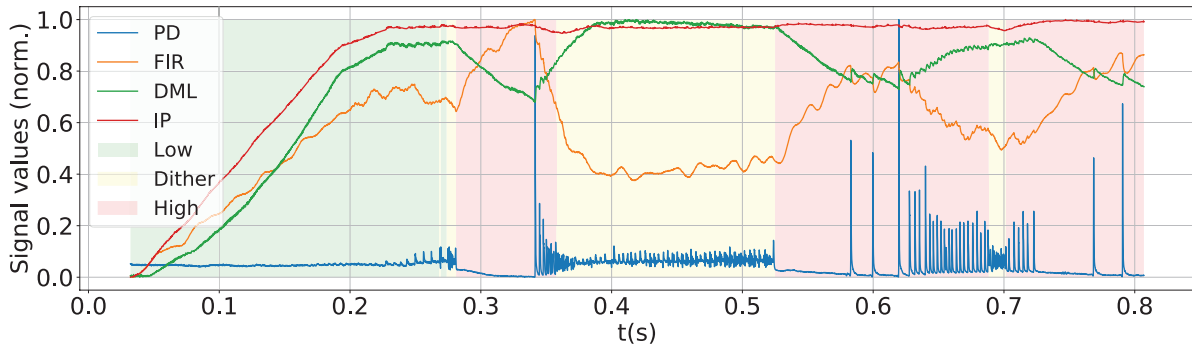


Figure 4. Switches between different plasma modes (low, dither and high), and time-traces of the collected signals, TCV shot #32195.

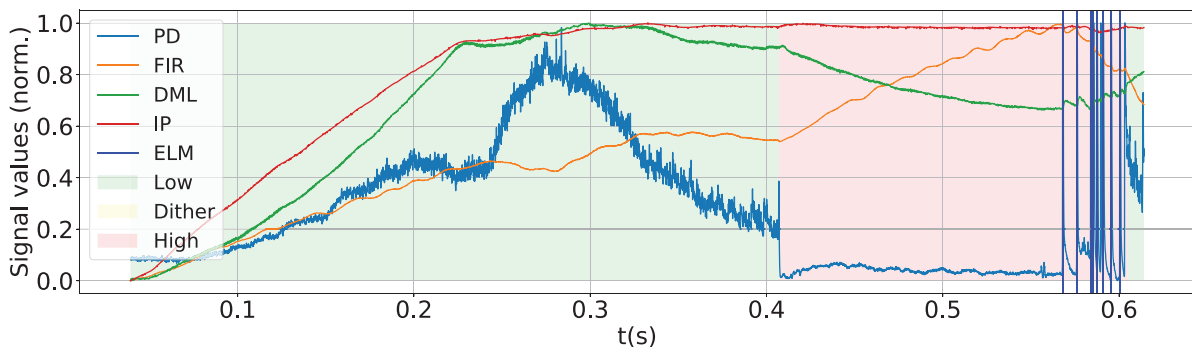


Figure 5. ELMs and L and H plasma modes, TCV shot #33446.

into L mode have the opposite trace, i.e. the baseline PD signal suddenly increases and remains at a steady level. ELMs are characterized by a sudden spike in the PD signal, followed by a relaxation that takes at most 2ms. D modes generate rapid fluctuations in the signal (see figure 6); they do not necessarily correspond to a change in the baseline signal value, unless they are followed by

a transition into a different state from the one at the point where they started.

(ii) *Interferometer (FIR) signal.* The interferometers at TCV measure the line-integrated electron density in the plasma along 14 parallel, vertical lines of sight. Of these, we take the mean value, per time instant, of the 12 inner-most detectors.

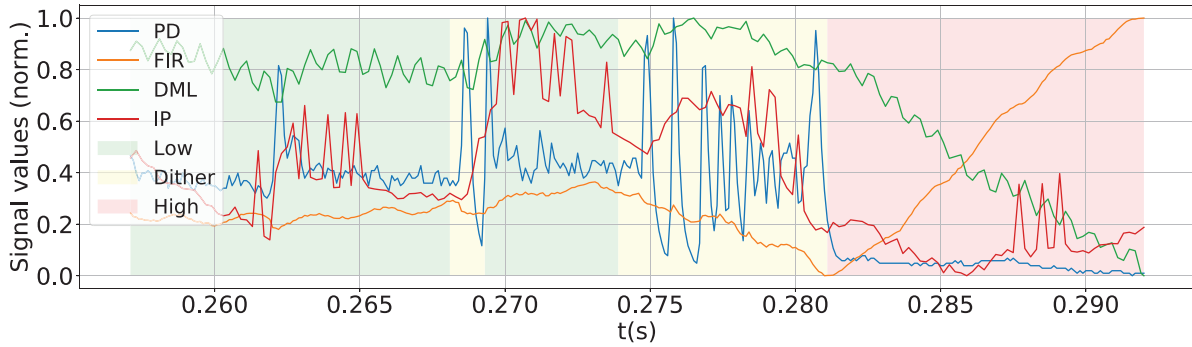


Figure 6. L, D and H modes from a section of TCV shot #32195.

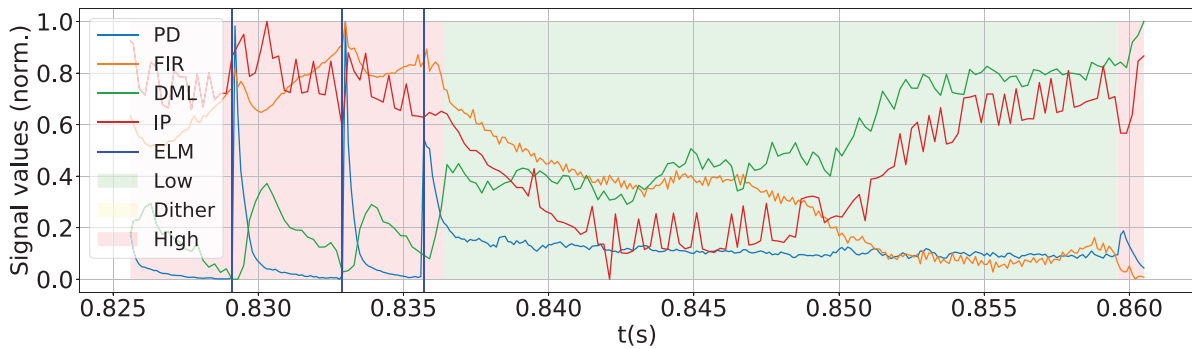


Figure 7. ELMs, and L and H modes from a section of TCV shot #31650.

In the interferometer signal, the transition between L and H mode can most easily be seen as a sudden increase in the time derivative of the signal, while transitions back into L mode correspond to a decrease in the derivative. Similarly to what happens with the photodiode signal, ELMs may provoke short (albeit less pronounced) spikes in the FIR signal.

- (iii) *Diamagnetic Loop (DML) signal*. Refers to the measurement of the total toroidal magnetic flux of the plasma [38]. The derivative of the DML signal frequently switches signs when a transition occurs between L and H mode, as well as when an ELM occurs (figure 7). Furthermore, the sign of this signal's derivative changes depending on the sign of the plasma current.
- (iv) *Plasma Current (IP) signal*. Refers to the total plasma electric current. For this work, we use the current value to determine when the actual classification of plasma states should begin. Specifically, we ignore, for classification purposes, time points where the absolute value of the current is lower than 50 kA.

The 4 different signals used for this work have different sampling rates. As a first step, we resampled all of them to the same frequency of 10kHz. Since each TCV shot is usually up to 2 s long, this means that our shot signal data consists of time-series of up to about 20000 time slices.

We want to train a classifier to recognize features in the data which allow for detecting the occurrence of ELMs and transitions between different plasma modes—i.e. a supervised learning task. As such, the first step was to collect labels for each shot time series, through visual analysis taking into account the data features described above. The collected data

was visually labeled by three different experts for the same shots. This means for some shots, the same regions might have different labels (namely, the experts might disagree on whether a certain part of a shot is dithering). Training the network with labels which are different in some regions has several potential advantages. For example, it compensates for any possible discrepancies in labeling originating from human error. It also allows us to incorporate the uncertainty in the labels into the network training process itself, that is, it acts as a form of regularization: if there is no majority agreement between experts regarding a section of a shot, then it is to be expected that the network will also learn not to strongly favor any class in that region. Conversely, if the three experts agree, then the network will learn that the features in that region most certainly correspond to a certain class, which renders the classification more robust. Finally, getting labels from different experts allows us to increase the size of our training dataset.

4.3. Model training

The two proposed models develop different maps. The first model is a map between a fixed window of observations and a distribution over transitions, while the second models a sequence of observations and produces a sequence of states (see figure 8).

Accordingly, the training data has different arrangements. For transition classification, we need to prepare a dataset $D_1, \{(x, q, e)\}$, where a training point consists of a section of the recorded signal $(x_{t-w}, \dots, x_t, \dots, x_{t+w})$, the corresponding label of one of the transitions q_t in Q and the matching label e_t indicating the presence (or not) of an ELM. Figure 9 illustrates this in detail.

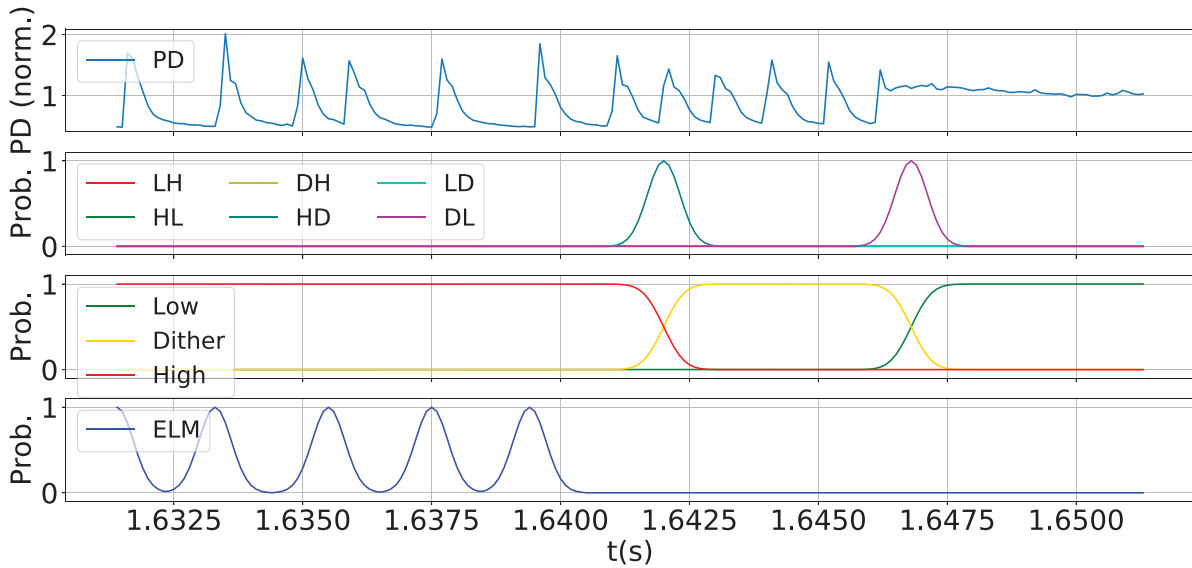


Figure 8. Representation of the different types of encoding of the target ‘smooth’ data distributions, to be learned by the two classifiers, from TCV shot #30262. Here, we show only the labels produced by a single expert, though the networks are trained with labels from all of them. The second plot from the top illustrates the transitions to be learned by the CNN, while the bottom-most plot illustrates the states to be learned by the Conv-LSTM.

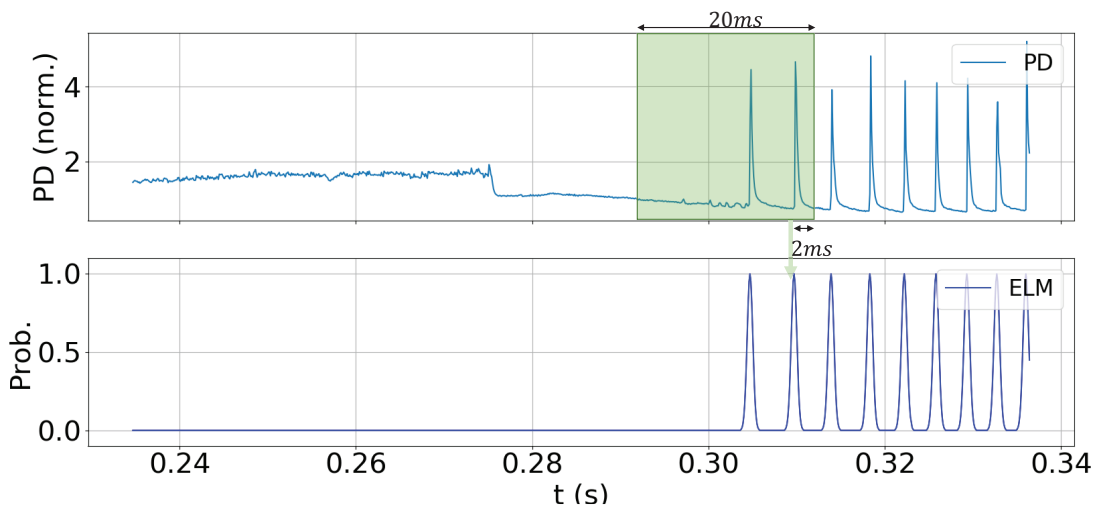


Figure 9. Representation of the sliding temporal windows fed to the CNN on top of the PD signal, and their corresponding ELM probability output. At inference time, these windows slide over the 4 signals across the whole shot, each of them rendering an output probability for a single time slice.

For the second model $D_2, \{(x, z, e)\}$, a training point consists of a sequence of windows of observations drawn from x_t to x_{t+l+w} (where l is a defined sequence length, and w is the window length), a sequence of state labels z_t in Z of length l , with each label corresponding to the state of the plasma at times t , and a sequence of labels e_t of length l corresponding to the presence of an ELM at times t . Figure 9 illustrates this in detail.

There is an inherent uncertainty in the labeling of the ELMs and plasma states, particularly when it comes to transitions into and out of dithers. The raw data only has hard, binary, one-hot encodings [39]—that is, a transition between two states, for example, is labeled as a sudden switch (from one time slice to the next) from one state to another. This means that it is easy to mistakenly label an event or transition in a

slightly shifted time slice. This type of hard threshold also makes it difficult for a neural network to generalize to outside of its training set [40].

Therefore, for the first model (CNN), we process the target time-series such that the probability of an ELM, or of a given state transition, is a continuous value, starting at zero and peaking at one, with several intermediate probabilities. In practical terms, we apply on each event a gaussian smoothing such that, if an ELM or state transition occurs at time t , its probability at that point is 1, and we define an interval Δt —before and after t —where the probability, respectively, smoothly increases and decreases. We defined these smoothing intervals as corresponding to 2ms, which, at the defined sampling rate, translates to 20 time slices. We do the same with the states z_t for the second model (Conv-LSTM), such that a switch

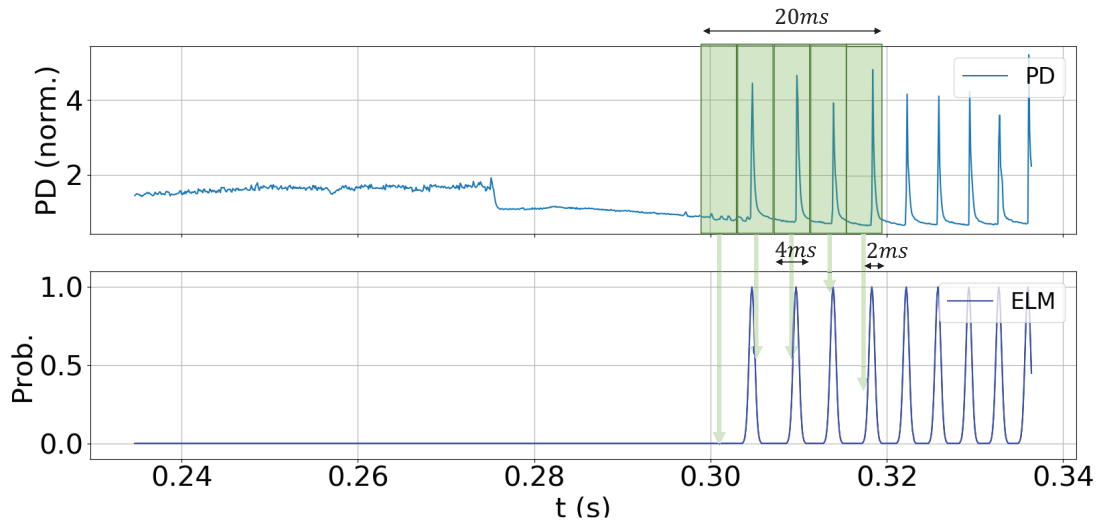


Figure 10. Example of a sequence fed to the LSTM. At a 10kHz sampling rate, it consists of 200 overlapping temporal windows of length 40. The output probability for a given window depends not only on what data features are present in that window, but also on the past windows in the sequence.

between two different states, from z_1 to z_2 , does not happen immediately from one time slice to the next, but rather, the probability of z_1 decreases, while that of z_2 increases, over a span of 20 time slices.

This procedure not only models the uncertainty in the labeling process, but also acts as an automatic regularization for the neural network training process, i.e. it makes it easier for a neural network to generalize what it learns to unseen data [41].

The choice of the size of the temporal windows with which the CNN is trained is a trade-off between the assumptions made about the data, and computational feasibility. Larger windows contain more spatial information and thus, intuitively, should make the classification at a particular time slice more precise, but also make the training and inference process by the network slower. Smaller windows contain arguably less information, but can be processed faster. We opted to train the CNN with temporal windows with a length of 20 ms, which we judged to be a good compromise between those two requirements. At our sampling rate, these windows are 200 time slices long. This is illustrated in figure 9: the green region represents a window of signals (in this case, only the PD signal) which is fed to the neural network, and its associated target, which is the probability of an ELM occurring at $t = 0.304$ s. There is an offset between the time at the window's rightmost edge, and the time for which the probability is computed; in the example of figure 9, the offset is of 2 ms, which means that to detect the ELM occurring at $t = 0.304$ s, the window would have information on the signals from $t = 0.286$ s to $t = 0.306$ s. In formal terms, the windows compute in that case $p(e_t | x_{t-w_1:t+w_2})$ and $p(q_{z_{t-1} \rightarrow z_t} | x_{t-w_1:t+w_2})$, where $w_1 = 180$ and $w_2 = 20$. In practice, in a real-time setting, that offset would constitute a minimum delay between the occurrence of an event in a machine, and a detection by the classifier. Once again, the size of this offset is a trade-off: a smaller offset is ideal for real-time applications because it gives more time for feedback control mechanisms, but it also

contains less information for the network to accurately classify an event.

We train the Conv-LSTM not with windows, but with sequences of windows. The distinction is an important one, for it implies different assumptions about the data. In the case of the windows fed to the CNN, it is assumed that each window is independent of each other. In the data fed to the Conv-LSTM, each sequence itself is composed of several windows, with future windows depending on past ones. We defined each of those sequences to consist of 200 windows (since that was also the length of the windows fed to the CNN). In this case, each of the individual windows has a length of 4 ms (40 time slices), with an offset of 2 ms, as in the data for the CNN (see figure 10). The sequences have a stride [42] of 1: each window starts and ends exactly 1 time slice after the previous one finishes. Each of these sequences is randomly subsampled from the whole shots, and the corresponding targets for them are chosen randomly from one of the three labelers.

Although not all of these subsamples start in L mode, our expectation is that the network would learn by itself that an actual shot always begins in that state. There are several reasons for this. First, the network will learn to recognize any features in the subsequences that are consistent with the beginning of a shot, and learn that those features correlate to L mode. Second, even if some training sequences start in D or H mode, the network will statistically learn that these modes are more frequently the result of a transition from a previous mode.

4.4. Model design

The architecture of the neural networks used for the transition detection starts with 1-D convolutions with four channels, each of which receives the values from the PD, FIR, IP and DML signals. These are followed by several convolutional layers, interspersed with pooling and dropout layers, which are trained for feature extraction, with deeper layers extracting

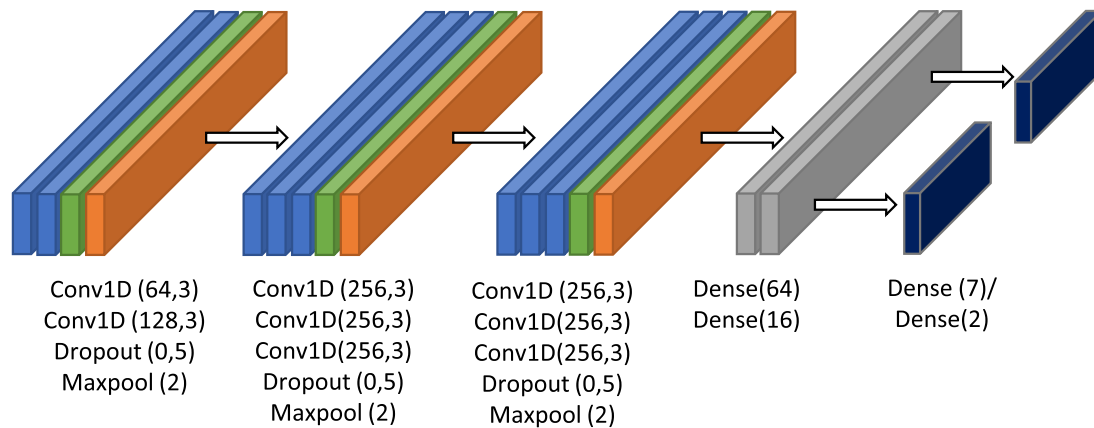


Figure 11. Architecture of the convolutional NN.

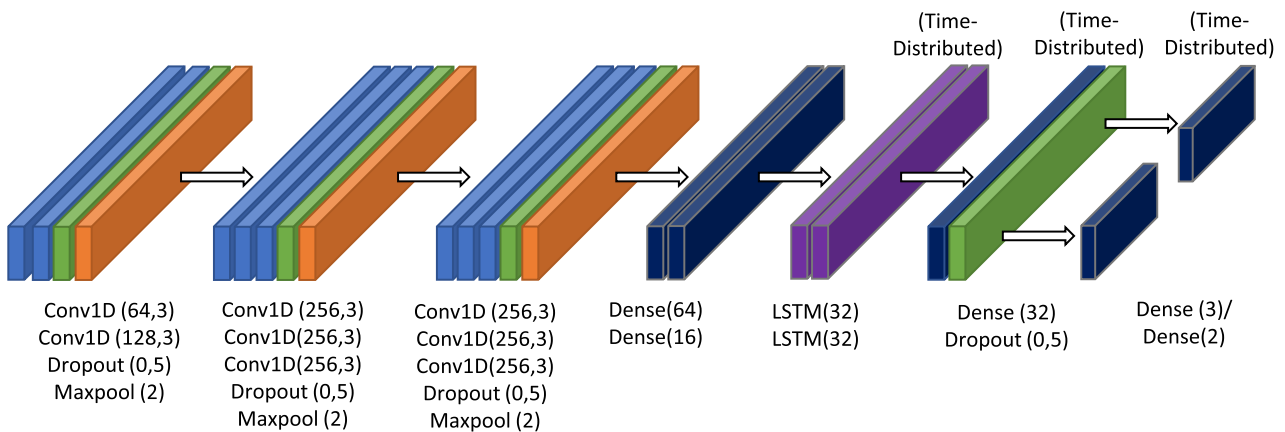


Figure 12. Architecture of the convolutional LSTM. All layers and nodes use ReLU activation functions, apart from the final output layer, which uses Softmax activation.

higher-level data features (figure 11). The last layers of the network are fully-connected, and are responsible for receiving the pre-processed high-level features and producing an appropriate output for them, i.e. the desired classification. This model is loosely inspired by the VGG architecture for classification of images where fixed sized filters are used [43].

Our convolutional LSTM network builds on top of CNN model that showed the best performance on the transition detection task. We add a recurrent layer that processes the output of the CNN to capture the longer-distance correlations in the data (figure 12).

We designed the networks using the Keras framework for deep learning [44]. Both networks used a categorical cross-entropy loss function, and were trained with the Adam optimizer [45] using the default learning rate value provided by Keras.

4.5. Data split

In total, we possessed 54 shots fully labeled by the three experts. In a typical deep learning setting, some sort of normalization [46] is usually applied on the available data. The most common procedure would have been to normalize across the entire dataset. However, because of the different calibrations of the PD signals and the subsequent large variance and multimodal distribution associated with it, we decided,

at this stage, to normalize each shot separately dividing each signal in each shot by its own mean across the whole shot. For potential real-time applications, as any new shots could fall outside the normalization range, the procedure would require grouping and normalizing the shots with respect to different signal gains and calibrations.

From these normalized full sequences, we draw batches of smaller temporal windows and subsequences to train the neural networks. There are several reasons for this subsampling. First, the full shot time-series are up to about 20000 time slices long, but the actual length of a shot can vary significantly. Yet for purposes of training the networks, we require batches of data of fixed length, which can be achieved by subsampling from the full sequences.

Second, this method allows us to automatically perform data augmentation for training, since one long sequence will contain many shorter subsequences and windows.

Third, feeding very large temporal windows to a CNN would be computationally difficult, as the number of network parameters requiring training would grow considerably.

Finally, the distribution of the data in the full sequences is highly unbalanced: in most shots, dithering phases are significantly shorter than L and H phases; only a few dozen transitions happen at most per shot; and, some transitions tend to be more frequent than others. Training with whole sequences

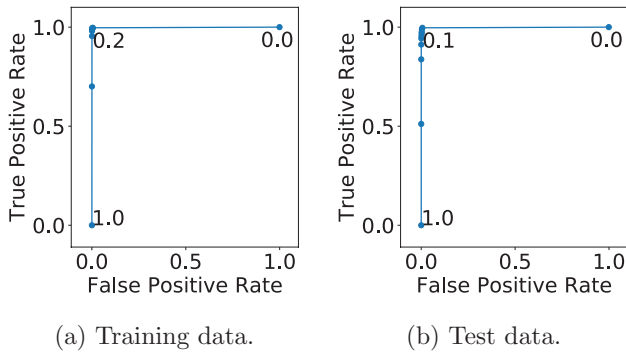


Figure 13. ROC curves for ELM detection for the CNN model. The detection thresholds that maximize the Youden index are 0.2 and 0.1 for training and test data, respectively yielding index values of 0.993 and 0.99. Using the ideal threshold for the training data (0.2) on the test data gives a slightly lower Youden index of 0.986. (a) Training data. (b) Test data.

would significantly bias the networks towards the events and transitions that occur more frequently in the labeled data. Drawing subsequences allows us to control the data fed to the network such that this inherent bias is mitigated. To do this, the training data batches must be balanced, i.e. generated such that they contain roughly equal fractions of the different types of events and/or transitions of interest. In the CNN, there are eight possible events of interest—LH, HL, HD, DH, LD, DL, ELM, and no transition. Generating batches for the CNN means that, for a batch containing n data samples, $n/8$ of those samples will correspond to each of those different types of transitions. Similarly, for the Conv-LSTM, the batches are generated such that the three target distributions (L, D and H) correspond to approximately $1/3$ of the data samples each.

5. Evaluation metrics

5.1. ROC curve

We consider the detection of single, discrete ELMs by the networks as corresponding to a point in time (in a shot) where the direct network outputs for ELM probability $\hat{e}_{1:N}$ reach a maximum value. This is not necessarily a point where the output network probability for ELM is 1, but rather, a point t where the output probability $P(ELM_t)$ follows a series of strictly increasing probability values, and precedes a series of strictly decreasing ones. Because we defined the length of the gaussian smoothing of the probabilities as 20, here we consider a local maximum for $P(ELM_t)$ within a 20-wide interval to correspond to the detection of a single ELM—which we denote as a positive. The remaining points are considered non-detections, i.e. negatives. In addition, we defined different probability thresholds for what can be considered a detection of an ELM by the network. For example, defining a threshold of 50% implies that only ELM probability maxima above that threshold are considered positives.

Positives and negatives must then be compared to the labeled ELMs. To that end, we build the ELM Confusion Matrix, which defines several variables: negatives that match their label at the same point in time are true negatives (TN),

Table 1. κ -statistic scores (κ_n and κ_l) for each plasma mode and as a mean, on training and test data (values across all shots), for the CNN.

		L	D	H	Mean
κ_n	Train	0.691	0.358	0.657	0.649
	Test	0.219	0.115	0.157	0.182
κ_l	Train	0.937	0.896	0.987	0.958
	Test	0.941	0.848	0.986	0.962

while those that do not are false negatives (FN). Similarly, positives that match their label are true positives (TP) and those that do not are false positives (FP).

Using this method to determine the points in which the network detects individual ELMs, one can then compute the true positive rate (TPR) and false positive rate (FPR) for different detection thresholds:

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

$$FPR = \frac{FP}{FP + TN} \quad (2)$$

Plotting the TPR versus FPR for a series of different detection thresholds yields the classifier's ROC curve [47], which illustrates the network's capacity for discrimination given different detection thresholds. There are several ways to compute the ideal detection threshold based on the ROC curve, depending on the task in question. In our case, we use the Youden index [48], whereby the best threshold is the value which maximizes the difference $TPR - FPR$, the maximum value being 1.

5.2. Kappa statistic

To compare the models' accuracy with that of the human labelers, we use Cohen's Kappa-statistic coefficient, which measures agreement between two sets of categorical data [49], defined as

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \quad (3)$$

where p_0 denotes the actual relative agreement between the two sets, and p_e denotes the probability of the two sets randomly agreeing with each other. Generically, the κ coefficient's values oscillate between 0 and 1, the former indicating poor performance, and the latter indicating perfect performance. In our case, given two sequences z_1 and z_2 of plasma states, Cohen's Kappa measures the overlap between them. If $z_{1,t} = z_{2,t}$ for all time instants t , the metric will yield a score of 1; if there are mismatches between the two sequences, the score will go down.

The κ -statistic can be interpreted differently based on the sections of the data for which it is computed. For that reason, we will now define several variables that allow us to interpret the κ -statistic scores.

Remember that we possess labels drawn from three different experts; as such, generically, labeled shot states at

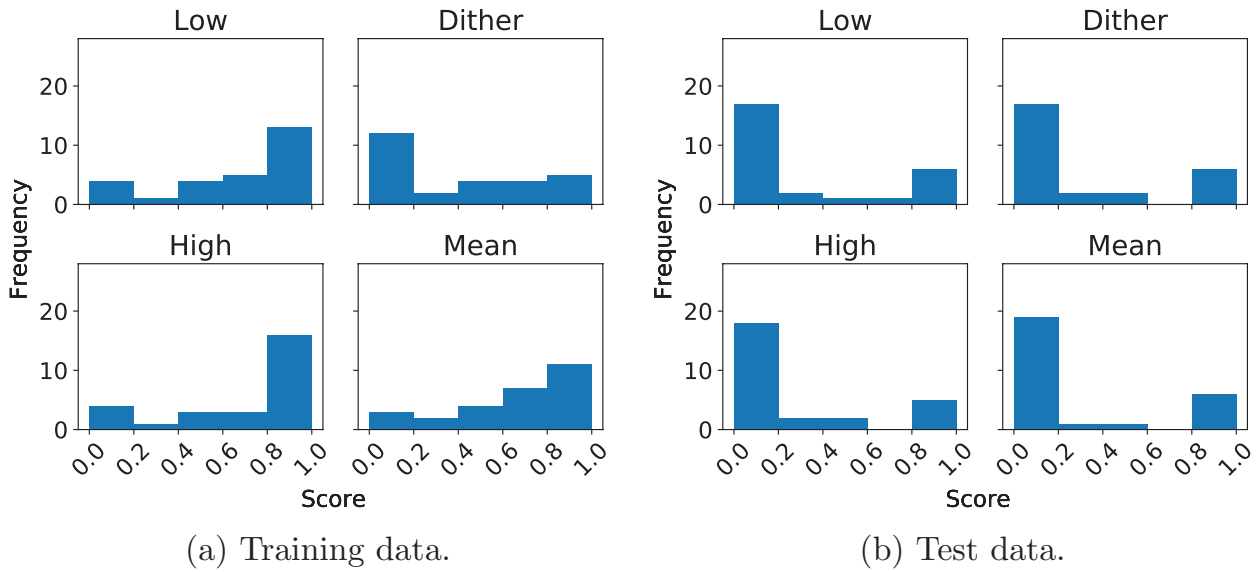


Figure 14. Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the CNN. (a) Training data. (b) Test data.

Table 2. κ -statistic scores (κ_n and κ_l) for each plasma mode on training and test data, for the Conv-LSTM.

		L	D	H	Mean
κ_n	Train	0.96	0.889	0.967	0.96
	Test	0.82	0.766	0.85	0.832
κ_l	Train	0.96	0.94	0.992	0.98
	Test	0.901	0.808	0.98	0.935

each point in time t of a shot can be in one of three possible categories:

- no majority agreement, i.e. all labelers disagree as to what state the plasma is in, which we denote as category C_1 ;
- majority agreement, i.e. two labelers agree on the state of the plasma, while one disagrees, which we denote as category C_2 ; and
- consensual agreement—all labelers agree as to what state the plasma is in, which we denote as category C_3 .

We define the union of C_2 and C_3 as *ground truth* (C_4), i.e. they are sections of shots where there is at least a majority opinion as to what state the plasma is in. We also have, for each shot, the most likely sequences $\hat{z}_{1:N}$ of states (given the observed data) produced by the neural networks, which we will now denote as C_5 .

Computing the κ -statistic score, κ_l , between sets C_2 and C_4 gives an indication of the probability that a single labeler disagrees with the ground truth: a κ_l -score of 1 would indicate that there is agreement between all the labelers all the time, while a lower score would indicate that at least some of the time, one labeler disagrees with the others. Simultaneously, computing the κ -statistic score between sets C_5 and C_4 (κ_n) gives an indication of the networks' performance given the ground truth. But, in addition, we can directly compare κ_l and κ_n . This comparison allows to test how a network and a single labeler compare against each other, on average, given the ground truth.

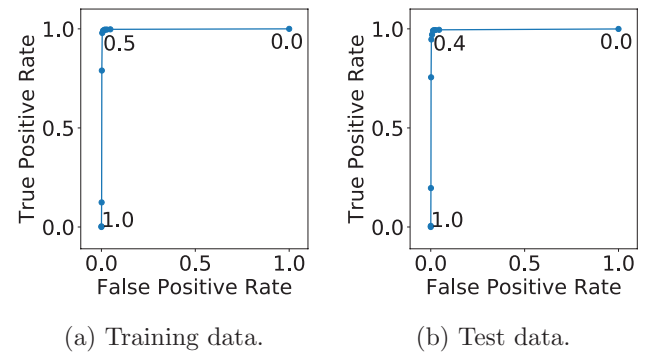


Figure 15. ROC curves for ELM detection for the Conv-LSTM model. The detection threshold which maximizes the Youden index is 0.5 for training and 0.4 for test data; this yields index values of 0.977 and 0.969 for each set, respectively. Using the ideal threshold for the training data (0.5) on the test data gives a slightly lower Youden index of 0.95. (a) Training data. (b) Test data.

The κ -coefficient is calculated separately for each of the three possible labels for the plasma state (L, D and H), and as a weighted mean across all three states. The weights of that mean are taken to be the relative frequencies of each individual state in the dataset, based on the ground truth (C_4) labels.

6. Results

We performed several training runs using the data labeled by the three experts; we carried out experiments where we trained both models (CNN and Conv-LSTM) three times, each time randomizing the training and test shots, to test whether differences in the data could lead to different results. In a typical deep learning setting, the data is usually split so that approximately 80%–90% is used for training, and 20%–10% is used for validation of the results, i.e. testing the network's capability to accurately predict on data that was not used for training. In our case, we opted for a training/test data split of 50%, i.e. of the 54 shots, we used 27 for training and 27 for testing. The results that follow are the best results of those

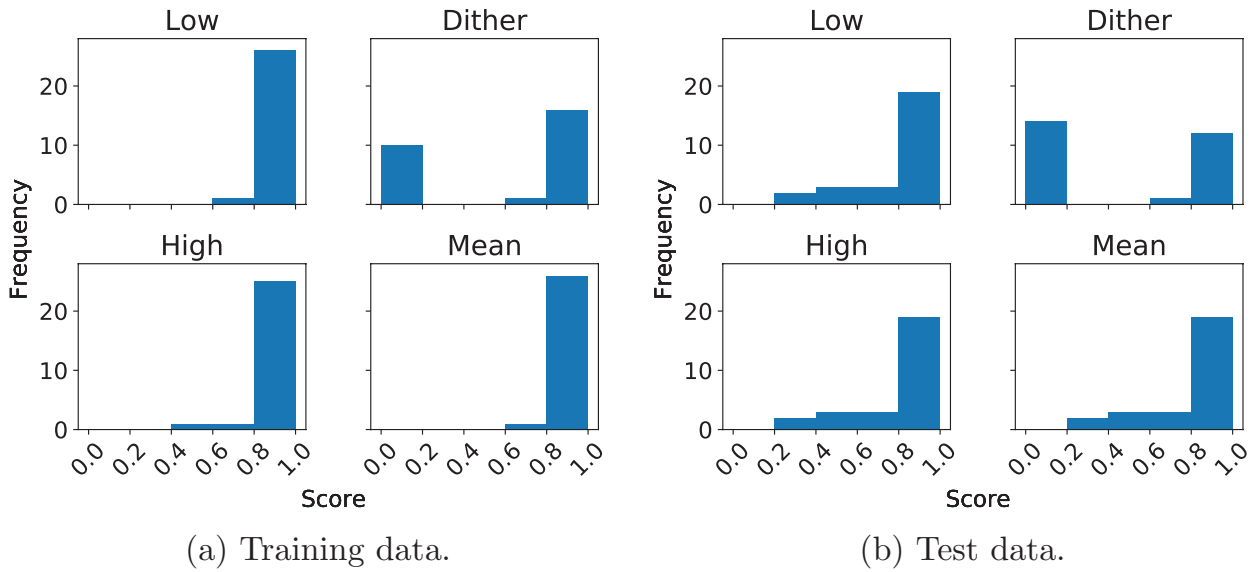


Figure 16. Distribution of the κ -statistic score (κ_n) on a per-shot basis, for the Conv-LSTM. (a) Training data. (b) Test data.

Table 3. Kappa statistic (κ_n) scores for each plasma mode on training and test data for selected shots representative of each of the six result categories.

Case	Shot ID	L		D		H		Mean
		Fraction	Score	Fraction	Score	Fraction	Score	
1	57751	0.756	0.97	0	0	0.243	0.97	0.97
2	34010	0.679	0.856	0.073	0.232	0.248	0.602	0.748
3	58182	0.22	0.912	0.095	0.969	0.685	0.927	0.928
4	30197	0.951	0.384	0	1	0.049	0.384	0.384
5	33459	0.811	0.662	0	0	0.189	0.846	0.697
6	33942	0.455	0.953	0.183	0.884	0.412	0.997	0.962

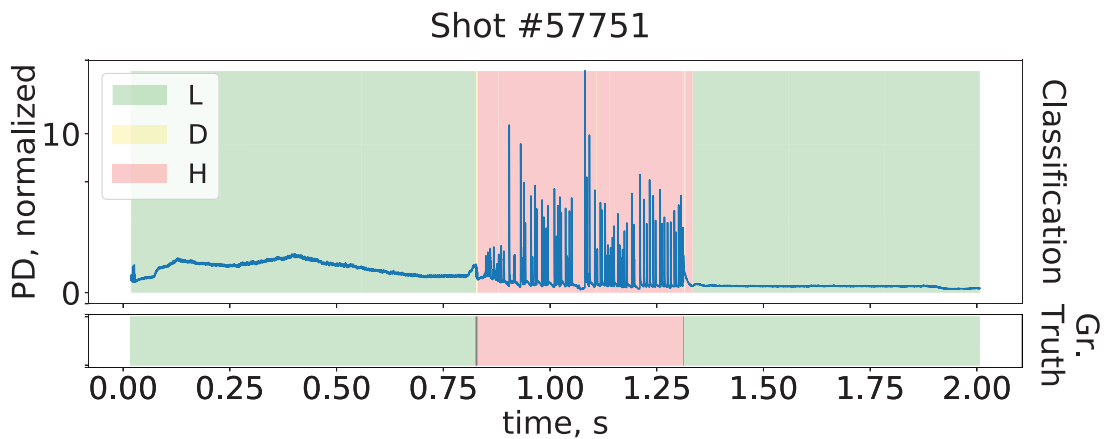


Figure 17. TCV shot #57751 (PD signal) and the Conv-LSTM’s classification of state as the shot evolves. Notice the (very short) detected dithering phase shortly after $t = 0.75$: no dithers are present in the labels, so the score for D is 0.

three experiments, for each model. We also experimented with varying offsets (see figure 9) for the convolutional windows to see what effect that factor could have on the results; we settled for an offset value of 2 ms (20 time slices), as smaller offsets degraded results, while larger ones did not improve them. We computed the metric scores on the training and test data at several points during training to control for overfitting [50], and present the results from the epoch where the state detection results on test data were the highest. We ran the neural networks on an NVIDIA Quadro RTX 5000 GPU.

6.1. CNN

We computed the κ -statistic based on the regions defined in section 5.2—that is, we compute scores based on the network output versus the ground truth (κ_n), and based on labeler disagreement versus the ground truth (κ_l). We computed the scores on a per-state (L, D and H) basis, and also computed a mean of the values obtained for each state.

We trained the CNN for 250 epochs, allowing for the loss function to stabilize; each epoch consisted in 32 batches,

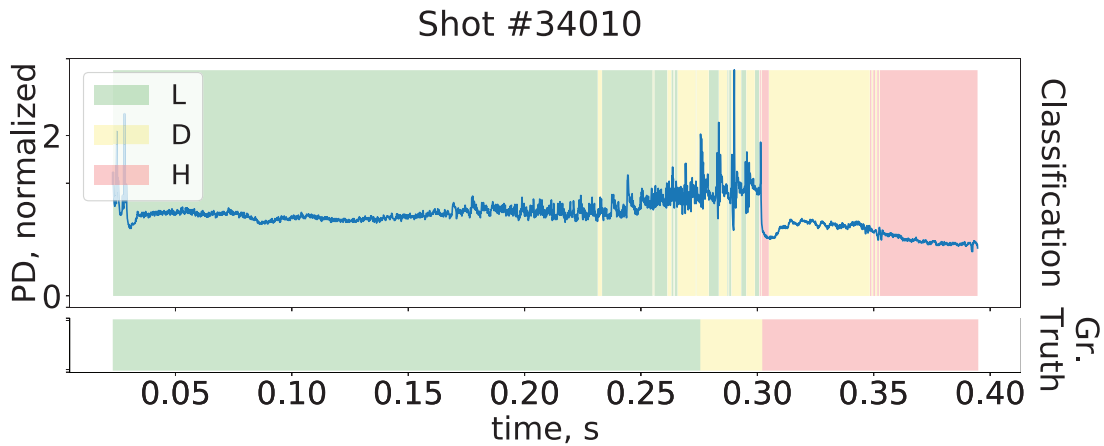


Figure 18. In TCV shot # 34010, the network correctly identifies the transition into H mode at $t = 0.3$ s, but it shortly thereafter (incorrectly) switches back to dithering.

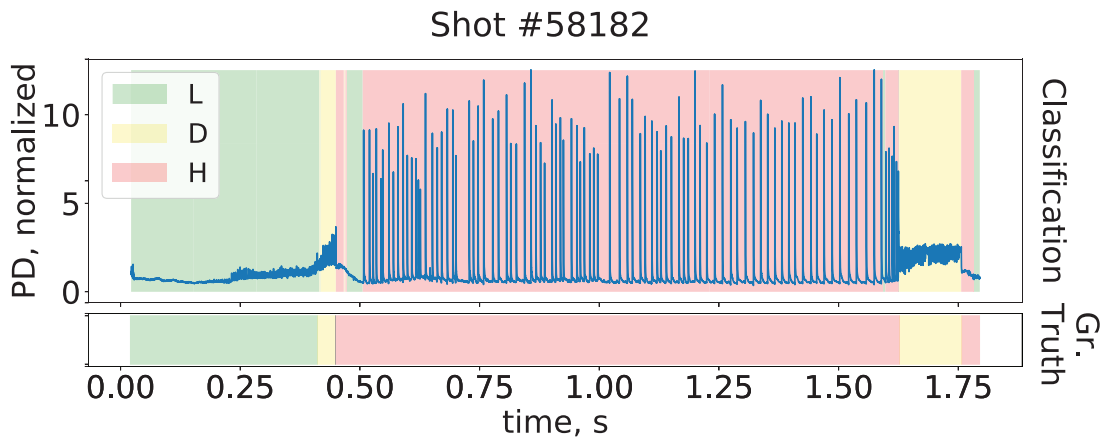


Figure 19. In TCV shot #58182, the network correctly identifies a transition into H mode (shortly before $t = 0.5$ s) but then incorrectly switches back to L mode and remains there until the first ELMs (spikes in the PD signal) appear.

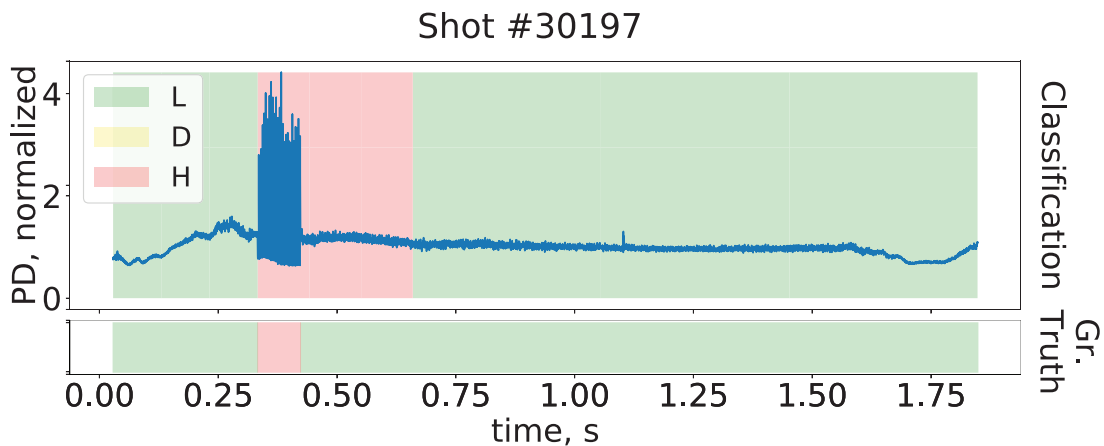


Figure 20. In shot #30197, the network misses the transition from H to L mode, which happens immediately after the series of spikes in the PD signal, and only makes the switch after $t = 0.5$ s.

with each batch containing 64 data samples. Upon completion of training, we tested the CNN’s accuracy on both the training and test data. The model’s results on ELM classification (ROC curve) can be seen in figure 13. Table 1 shows the scores κ_n and κ_l for the entire dataset, while figure 14 contains histograms showing the κ_n s distribution on a per-shot basis.

6.2. Conv-LSTM

We trained the convolutional LSTM for 400 epochs, allowing the loss function to stabilize. Each epoch consisted of 64 batches, with each batch containing 64 data samples. The results of computing scores κ_l and κ_n , using the same definitions as for the CNN can be seen in table 2. The ROC

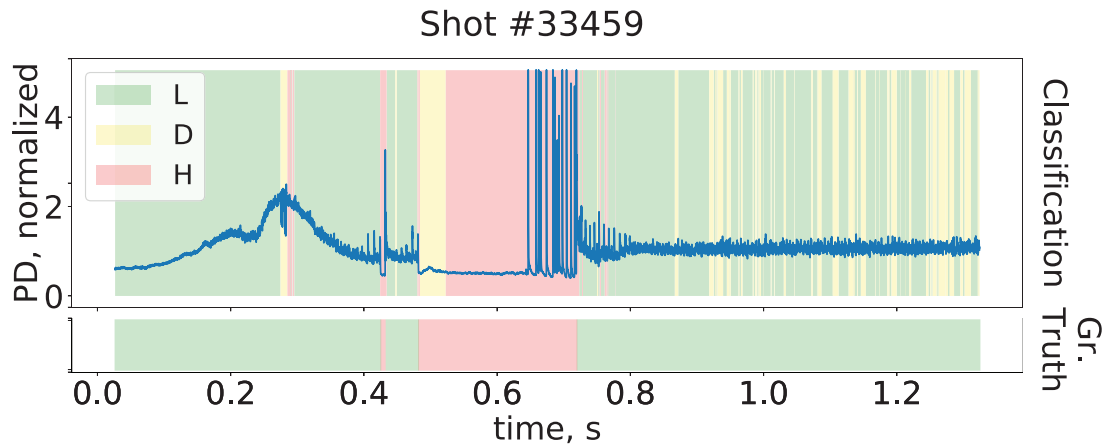


Figure 21. Shot #33459 represents an overall bad classification by the network; many dithers are incorrectly classified, while the transition from L to H mode is missed. Around $t = 0.3$ s, immediately after classifying a D mode, the network oscillates between L and H in quick succession for about 0.01 s, which to the naked eye might appear in this plot as a gray area; in reality, it is an artifact of the plot, with alternating red and green regions.

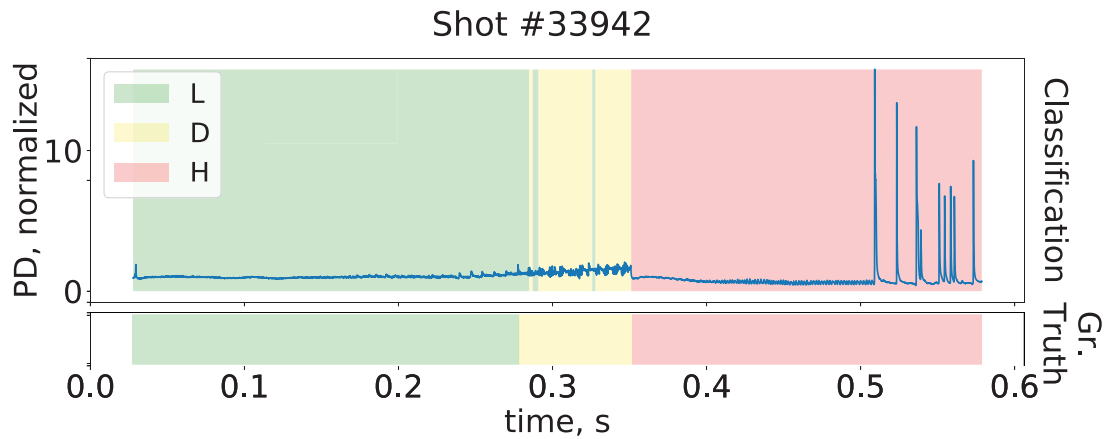


Figure 22. Shot #33942 is an example of an overall good detection.

curves detailing the results on ELM detection can be seen in figure 15. Figure 16 contains histograms showing the score K_n values on a per-shot basis.

6.3. Discussion

A comparison of the κ_n scores on training and test data for each classifier shows that the convolutional LSTM performs better than the CNN for all three plasma states. Furthermore, looking at the distribution of the mean κ_n scores on a per-shot basis through the histograms, one can see that the worst Conv-LSTM classifications do not have a score lower than 0.6 on training data, while for the CNN alone, even on training data, mean κ_n scores lower than 0.2 exist. For both classifiers, the performance on training data surpasses that on test data, both on a state-by-state basis, and as a mean across all states, which indicates the occurrence of overfitting.

For both networks, an analysis of the κ_l scores of their training and test data indicates that human labeler disagreement is highest for dithers—the scores for that particular state are consistently lower. Interestingly, both networks also score their lowest results for dithers.

Comparing the Conv-LSTM's κ_l and κ_n scores shows that, at least on training data, the network behaves, on average, similarly to a single human labeler, making errors (or disagreeing with the ground truth) at approximately the same rate—the mean κ_l score for training data is 0.98, while the mean κ_n score for training data is 0.96. On test data, the Conv-LSTM performs slightly worse than a single human labeler, as seen by the fact that the network's mean K-index score on test data κ_n is 0.832, while κ_l is 0.935.

As measured by the Youden index, we achieve excellent performance in detection of ELMs on both training and test data using both models; the ideal detection thresholds generate true positive detection rates very close to 1, while bringing false positive detection rates essentially to 0. The Youden indexes for test data are only slightly lower than for training data, which suggests that overfitting is minimal. Furthermore, for both models, on both training and test data, the ROC curves' points are mostly concentrated close to true positive rates of 1 and false positive rates of 0, which indicates that the choice of ELM detection threshold does not significantly change the behavior of the classifiers.

Finally, the scores for ELMs being essentially the same for both models indicates that the features in the data which allow

for identification of ELMs are mostly local: the CNN, even without knowledge of long-term temporal correlations, performs excellent classification.

Because the Conv-LSTM has highest κ_n scores, we made a case-by-case analysis of that network's classification of all our available shots. Broadly, the Conv-LSTM's results on state detection, on a per-shot basis, can be placed into six different categories:

- (i) a (sometimes very) short detection, of a dither that is not labeled in the data. Due to the way the K-score κ_n is computed, a mistaken dither classification by the network of a single time point (in a whole sequence), in a shot which has no regions where the ground truth (C_4) is dithering, will bring the score for that state down to 0, even if the remainder of the shot is correctly classified (17 shots);
- (ii) a clearly incorrect classification, of a temporal region of a shot as being in a dithering state (4 shots);
- (iii) a missed detection of an L-H transition (1 shot);
- (iv) a missed detection of an H-L transition (2 shots);
- (v) an overall bad detection across an entire shot (7 shots); and
- (vi) an overall good detection across an entire shot (23 shots).

Table 3 lists six shots which are representative of each of the types of results listed above. The table shows the computed κ_n scores for each of those shots on a per-state basis, as well as the score's mean value, and the fraction of time, for the ground truth of each shot, that a particular state is labeled. Table 3 also lists which of the six cases above the shot is representative of. Figures 17–22 are plots of those same shots, where the background color in the top plot denotes the state detected by the Conv-LSTM, and in the bottom plot, denotes the ground truth label. Small gray areas in the bottom plot denote regions where ground truth is not defined, i.e. there is no majority agreement between labelers.

7. Conclusions

We have developed two deep learning-based classifiers to perform automatic detection of ELMs and classification of plasma modes. The task was two-fold: on one hand, to perform a binary classification, for each time slice of a plasma shot, on whether an ELM is occurring or not; and, to automatically determine which plasma mode (or alternatively, whether a transition between plasma modes) is occurring. One approach is to use a convolutional neural network (CNN), which uses only local correlations in data to perform classification. The second approach uses a Convolutional LSTM (Conv-LSTM) neural network, which also takes advantages of long-term temporal correlations in data.

On ELM detection, the two networks can achieve essentially equal results. On the plasma state classification, a clear difference can be seen between the results obtained with the CNN, and those obtained with the Conv-LSTM. Comparing the κ -index (κ_n) scores of each network shows that the LSTM's scores are clearly higher, which suggests that, at least when it comes to detection of plasma modes, the processing of long-term correlations in data facilitates

accurate classification. There is some indication that overfitting occurred. However, our monitoring of the training progression indicated that, while the metric values for test data are always lower, they did, nevertheless, become better as training progressed. Thus, an overfitting-avoidance strategy such as early stopping would, in this case, not have helped achieve better test accuracy.




While the results from the Conv-LSTM are better, that network is also more complex with both network training and inference taking longer.

Although this work used data from the TCX tokamak, it should also be possible to adapt it to other machines; as a matter of fact, the data sources used exist on most tokamaks. As long as the data fed to the neural networks is from those same sources, this model could in principle be used for automatic labeling of shots from a number of different machines.

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under Grant Agreement No. 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. We would like express our gratitude to B. Labit, R. Maurizio and O. Sauter at SPC/EPFL for taking the time to manually label the data used for training. This work was supported in part by the Swiss National Science Foundation.

ORCID iDs

F. Matos  <https://orcid.org/0000-0002-3110-6639>
 F. Felici  <https://orcid.org/0000-0001-7585-376X>
 A. Pau  <https://orcid.org/0000-0002-7122-3346>

References

- [1] Zhang T. *et al* 2013 *Phys. Lett. A* **377** 1725–35
- [2] Loarte A. *et al* 2014 *Nucl. Fusion* **54** 033007
- [3] Rastovic D. 2009 *J. Fusion Energy* **28** 101–6
- [4] Humphreys D. *et al* 2015 *Phys. Plasmas* **22** 021806
- [5] Martin Y. *et al* 2008 *J. Phys.: Conf. Ser.* **123** 012033
- [6] Ryter F. *et al* 1994 *Plasma Phys. Control. Fusion* **36** A99
- [7] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. and Rabinovich A. 2015 Going deeper with convolutions *The IEEE Conf. on Computer Vision and Pattern Recognition* 1–9
- [8] Ince T., Kiranyaz S., Eren L., Askar M. and Gabbouj M. 2016 *IEEE Trans. Ind. Electron.* **63** 7067–75
- [9] Krizhevsky A., Sutskever I. and Hinton G.E. 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* 25 ed Pereira F *et al* (Curran Associates, Inc.) pp 1097–105
- [10] Tompson J., Goroshin R., Jain A., LeCun Y. and Bregler C. 2015 Efficient object localization using convolutional networks *The IEEE Conf. on Computer Vision and Pattern Recognition*
- [11] Lähivaara T., Kärkkäinen L., Huttunen J.M. and Hesthaven J.S. 2018 *J. Acoust. Soc. Am.* **143** 1148–58

- [12] Acharya U.R., Oh S.L., Hagiwara Y., Tan J.H. and Adeli H. 2018 *Comput. Biol. Med.* **100** 270–8
- [13] Abdeljaber O., Avci O., Kiranyaz S., Gabbouj M. and Inman D.J. 2017 *J. Sound Vib.* **388** 154–70
- [14] Malek S., Melgani F. and Bazi Y. 2018 *J. Chemometr.* **32** e2977
- [15] Golik P., Tüske Z., Schlüter R. and Ney H. 2015 Convolutional neural networks for acoustic modeling of raw time signal in LVCSR *16th Annual Conf. of the Int. Speech Communication Association (Dresden, Germany, 6–10 September 2015)* (<https://www-i6.informatik.rwth-aachen.de/publications/download/974/Golik--2015.pdf>)
- [16] Kiranyaz S., Ince T. and Gabbouj M. 2015 *IEEE Trans. Biomed. Eng.* **63** 664–75
- [17] Ronao C.A. and Cho S.B. 2016 *Expert Syst. Appl.* **59** 235–44
- [18] Sundermeyer M., Schlüter R. and Ney H. 2012 LSTM neural networks for language modeling *13th Annual Conf. of the Int. Speech Communication Association*
- [19] Ma X., Tao Z., Wang Y., Yu H. and Wang Y. 2015 *Transp. Res. C* **54** 187–97
- [20] Venugopalan S., Hendricks L.A., Mooney R. and Saenko K. 2016 *Proc. 2016 Conf. Empirical Methods Natural Language Processing* 1961–6
- [21] Webster A. and Dendy R. 2013 *Phys. Rev. Lett.* **110** 155004
- [22] Greenhough J., Chapman S., Dendy R. and Ward D. 2003 *Plasma Phys. Control. Fusion* **45** 747
- [23] Shabbir A., Verdoolaege G., Karadaun O.J., Webster A.J., Dendy R.O. and Noterdaeme J.M. 2014 Discrimination and visualization of elm types based on a probabilistic description of inter-elm waiting times *41st European Physical Society Conf. on Plasma Physics (Berlin, Germany, 23–27 June 2014)* (<http://ocs.ciemat.es/EPS2014PAP/html/>)
- [24] Vega J. et al 2009 *Nucl. Fusion* **49** 085023
- [25] González S. et al 2012 *Plasma Phys. Control. Fusion* **54** 065009
- [26] Murari A., Vagliasindi G., Zedda M.K., Felton R., Sammon C., Fortuna L. and Arena P. 2006 *IEEE Trans. Plasma Sci.* **34** 1013–20
- [27] Lukianitsa A., Zhdanov F. and Zaitsev F. 2008 *Plasma Phys. Control. Fusion* **50** 065013
- [28] Meakins A. et al 2010 *Plasma Phys. Control. Fusion* **52** 075005
- [29] Xu G. et al 2014 *Nucl. Fusion* **54** 103002
- [30] Wagner F. et al 1982 *Phys. Rev. Lett.* **49** 1408
- [31] Itoh S.I. and Itoh K. 1988 *Phys. Rev. Lett.* **60** 2276–9
- [32] Basse N. et al 2003 *Plasma Phys. Control. Fusion* **45** 439
- [33] Martin Y. 2001 Elming h-mode accessibility in shaped TCV plasmas *Technical Report TCV Team*
- [34] Jurafsky D. and Martin J.H. 2014 *Speech and Language Processing* 2nd edn (Englewood Cliffs, NJ: Prentice Hall)
- [35] Boulanger-Lewandowski N., Bengio Y. and Vincent P. 2013 High-dimensional sequence transduction *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (IEEE) (Vancouver, Canada, 26–31 May 2013)* 3178–82 (<https://www2.securecms.com/ICASSP2013/default.asp>)
- [36] Hofmann F. et al 1994 *Plasma Phys. Control. Fusion* **36** B277
- [37] Coda S. et al 2019 *Nucl. Fusion* **59** 112023
- [38] Moret J.M., Buhlmann F. and Tonetti G. 2003 *Rev. Sci. Instrum.* **74** 4634–43
- [39] Harris D. and Harris S. 2010 *Digital Design and Computer Architecture (Computer Organization Bundle, VHDL Bundle)* (Amsterdam: Elsevier)
- [40] Szegedy C., Vanhoucke V., Ioffe S., Shlens J. and Wojna Z. 2016 Rethinking the inception architecture for computer vision *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 2818–26
- [41] Zheng Q., Yang M., Yang J., Zhang Q. and Zhang X. 2018 *IEEE Access* **6** 15844–69
- [42] Dumoulin V. and Visin F. 2018 A guide to convolution arithmetic for deep learning (arXiv:1603.07285)
- [43] Simonyan K. and Zisserman A. 2014 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
- [44] Chollet F. et al 2015 Keras (<https://github.com/fchollet/keras>)
- [45] Kingma D.P. and Ba J. 2014 (arXiv:1412.6980)
- [46] Han J., Pei J. and Kamber M. 2011 *Data Mining: Concepts and Techniques* (Amsterdam: Elsevier)
- [47] Fawcett T. 2006 *Pattern Recognit. Lett.* **27** 861–74
- [48] Liu X. 2012 *Stat. Med.* **31** 2676–86
- [49] Landis J.R. and Koch G.G. 1977 *Biometrics* 159–74
- [50] Bishop C.M. 2006 *Pattern Recognition and Machine Learning* (Berlin: Springer)
- [51] Labit B. et al 2019 *Nucl. Fusion* **59** 086020

Deep learning for Gaussian process soft x-ray tomography model selection in the ASDEX Upgrade tokamak

Cite as: Rev. Sci. Instrum. **91**, 103501 (2020); <https://doi.org/10.1063/5.0020680>

Submitted: 02 July 2020 . Accepted: 29 September 2020 . Published Online: 19 October 2020

 F. Matos, J. Svensson,  A. Pavone,  T. Odstrčil, and  F. Jenko



View Online



Export Citation



CrossMark

ARTICLES YOU MAY BE INTERESTED IN

[Apparatus to investigate liquid oxygen droplet combustion in hydrogen under microgravity conditions](#)



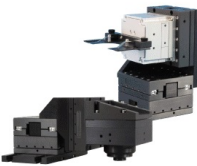
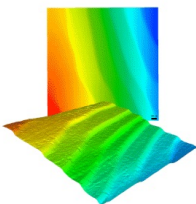
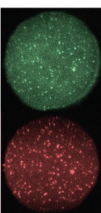
Review of Scientific Instruments **91**, 105110 (2020); <https://doi.org/10.1063/5.0020988>

[A low-pass filtering Fresnel zone plate for soft x-ray microscopic analysis down to the lithium K-edge region](#)

Review of Scientific Instruments **91**, 103110 (2020); <https://doi.org/10.1063/5.0020956>

[A radiofrequency voltage-controlled current source for quantum spin manipulation](#)

Review of Scientific Instruments **91**, 104708 (2020); <https://doi.org/10.1063/5.0011813>

 <p>MCL MAD CITY LABS INC. www.madcitylabs.com</p>	<p>Nanopositioning Systems</p> 	<p>Modular Motion Control</p> 	<p>AFM and NSOM Instruments</p> 	<p>Single Molecule Microscopes</p> 
--	--	--	---	--

Deep learning for Gaussian process soft x-ray tomography model selection in the ASDEX Upgrade tokamak

Cite as: Rev. Sci. Instrum. 91, 1 03501 (2020); doi: 10.1063/5.0020680

Submitted: 2 July 2020 • Accepted: 29 September 2020 •

Published Online: 19 October 2020



F. Matos,^{1,a)}  J. Svensson,² A. Pavone,²  T. Odstrčil,³  and F. Jenko¹ 

AFFILIATIONS

¹Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany

²Max Planck Institute for Plasma Physics, Wendelsteinstr. 1, 17491 Greifswald, Germany

³Plasma Science and Fusion Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

^{a)}Author to whom correspondence should be addressed: francisco.matos@ipp.mpg.de

ABSTRACT

Gaussian process tomography (GPT) is a method used for obtaining real-time tomographic reconstructions of the plasma emissivity profile in tokamaks, given some model for the underlying physical processes involved. GPT can also be used, thanks to Bayesian formalism, to perform model selection, i.e., comparing different models and choosing the one with maximum evidence. However, the computations involved in this particular step may become slow for data with high dimensionality, especially when comparing the evidence for many different models. Using measurements collected by the Soft X-Ray (SXR) diagnostic in the ASDEX Upgrade tokamak, we train a convolutional neural network to map SXR tomographic projections to the corresponding GPT model whose evidence is highest. We then compare the network's results, and the time required to calculate them, with those obtained through analytical Bayesian formalism. In addition, we use the network's classifications to produce tomographic reconstructions of the plasma emissivity profile.

Published under license by AIP Publishing. <https://doi.org/10.1063/5.0020680>

I. INTRODUCTION

Computed tomography generally refers to the process of imaging the interior of a body through indirect measurements. In many applications, this is achieved by focusing penetrating radiation on an object of interest from several directions and measuring the resulting decrease in radiation intensity on the opposite side (due to absorption by the body itself). The use of this information, the so-called *projection* of the object, allows one to reconstruct its internal properties.¹

In the case of radiative bodies, an alternative way to determine their properties is to perform cross-sectional imaging by treating the emitted radiation itself as a projection.² In the field of nuclear fusion, this procedure is employed in many tokamaks for the reconstruction of plasma emissivity profiles.³ More specifically, in the ASDEX Upgrade tokamak, such imaging can be done with information from the Soft X-Ray (SXR) diagnostic, which measures the line-integrated radiation emitted by the plasma along several lines of sight (LOSs);

these can be used to perform the tomographic reconstruction (or inversion) of the plasma emissivity profile. Knowledge of this is useful for exploring magnetohydrodynamic phenomena in addition to studying the accumulation of impurities inside the plasma (particularly, tungsten) due to their large contribution to the total amount of radiation.⁴

Several techniques exist for solving the tomography problem.⁵ One approach is to use regularization-based algorithms, namely, Tikhonov-based^{6,7} and minimum Fisher-based techniques.⁸ More recently, work has also been done using machine learning methods, namely, deep neural networks,^{9–11} that are trained to create new reconstructions based on existing ones.

Yet another method is Gaussian process tomography (GPT).¹² GPT is an established method for performing tomographic inversion on many different types of physical distributions that are modeled as posterior Gaussian distributions in a Bayesian setting. Computing a posterior first requires specifying a prior distribution, which encodes one's assumptions about the underlying physical process before any

measurements of it are taken. The posterior can then be computed based on that prior and on an observation (measurement) of the data generated by the physical process. The prior itself can either be a fixed distribution or be drawn from a family of different models.

Knowing the posterior, GPT guarantees that one can obtain the most likely [maximum *a posteriori* (MAP)] estimate for the tomographic reconstruction as well as its associated error values. More interestingly, however, through Bayesian inference, GPT prescribes a way to estimate the evidence for different models through a process known as Bayesian model selection. This procedure can be of particular importance in cases where the choice of prior might have a strong effect on the results of the tomographic inversion.

Unfortunately, in a neural network, there are no guarantees¹³ about whether the reconstructions obtained correspond to the MAP estimate of the underlying distribution, and there is no direct way, in standard deep neural networks, such as convolutional neural networks (CNNs), to obtain uncertainty estimates on the outputs. Bayesian neural networks^{14,15} and generative adversarial networks (GANs)¹⁶ can generate probability distributions for their outputs; however, they can be computationally expensive and, in the case of GANs, difficult to train.¹⁷

On the other hand, neural networks essentially store whatever function they have learned (through their training process) in their weights, making the inference process for new data very fast. With GPT, computing the MAP estimate based on a fixed model is also sufficiently fast for real-time purposes. This does not necessarily hold true, however, when performing Bayesian model selection, since the process requires a series of additional computational steps, namely, matrix inversions or using non-linear optimizers, which can be time-consuming, especially for data with a high dimensionality.

Thus, we propose an approach where we train a convolutional neural network (CNN) to learn the GPT model selection procedure. To do this, we take SXR measurement samples from several ASDEX Upgrade shots and, through Bayesian model selection formalism, compute for each data point the corresponding model (out of a set of possible, pre-defined ones) with the highest evidence. We then train the CNN to reproduce this step, i.e., to map measurements to their highest evidence model. Finally, through the GPT framework, we compute the tomographic reconstruction of the plasma emissivity profile for each measurement, given the most likely models predicted by the CNN.

This paper is organized as follows. Section II gives an overview of the problem of tomography, in particular, the soft x-ray tomography ASDEX Upgrade tokamak, and the existing techniques to solve it, including a review of GPT with Bayesian model selection. Section III details the data we collected, the formulation of our problem, and the model proposed to solve it. Section IV details the direct results of the neural network classification and the tomographic reconstructions obtained based on them; Sec. V describes and discusses our conclusions.

II. BACKGROUND

A. Computed tomography

The purpose of tomography is to reconstruct the internal (either two- or three-dimensional) properties of a given body from

non-local measurements. Radon showed¹⁸ that a 2D distribution can be retrieved from an infinite set of line-integrated measurements. In practical applications, the number of available measurements is always finite, but it is, nevertheless, possible to produce accurate reconstructions from a discrete set of measurements.¹⁹ Tomographic algorithms can achieve this by taking many *projections* of the object of interest from different directions.¹ Mathematically, a projection is a function that computes the line-integrated absorbency (or, in the case of fusion plasmas, emissivity) of a body along several paths or lines of sight (LOSs) as

$$P_{\theta}(t) = \int_{L(\theta,t)} G(x,y) dL, \quad (1)$$

where t is a point in the projection domain, $L(\theta, t)$ is the LOS crossing the body mapping to t (along a direction given by an angle θ), and $G(x, y)$ is the two-dimensional physical distribution of interest (see Fig. 1).

By computing several tomographic projections with different directions (i.e., different values of θ), it is possible to reconstruct $G(x, y)$. For an exact reconstruction based only on the projections, an infinite number of them would need to be obtained. However, the problem is highly ill-posed²¹ since small changes in the projection space can translate into large changes in the tomographic reconstructions. Furthermore, in many settings such as nuclear fusion experiments, it is difficult, or impossible, to obtain more than a handful of such projections, making the problem under-determined, that is, the dimensionality of the reconstruction grid is much larger than that of the projection, ultimately resulting in an infinite number of solutions (reconstructions) that can fit the data. For these reasons, in most tomography applications in fusion plasmas, some additional information, in the form of assumptions about the function $G(x, y)$, must be introduced in order to obtain a tomographic reconstruction.

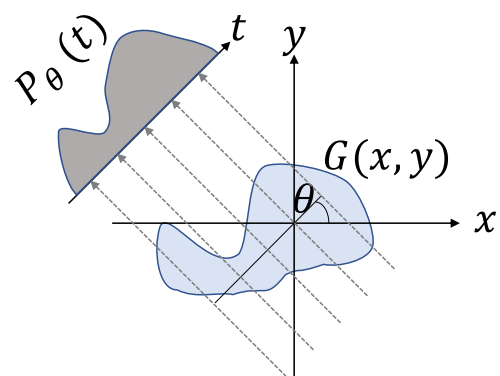


FIG. 1. An illustrated projection, P_{θ} , measured along an angle θ . The blue area $G(x, y)$ is the cross section of interest and is being traversed by radiation. Because different rays traverse different areas of the object, the value at each point t in the projection space will be different. Figure reprinted with permission from F. Matos, "Deep learning for plasma tomography," M.Sc. thesis, Técnico Lisboa, 2016.²⁰

B. SXR tomography at ASDEX Upgrade

In the ASDEX Upgrade tokamak, the Soft X-Ray (SXR) diagnostic²² consists of eight pinhole cameras that measure the total radiation emitted by the plasma along 208 different volumes of sight (VOSs).⁴ We considered the extent of the VOSs in the toroidal and poloidal directions of the tokamak to be minimal and treated them instead as lines of sight (LOSs). In addition, we also ignored the fact that the LOSs in the same camera array partially overlap. Based on this, the measurements collected by the individual cameras correspond to a single projection of the underlying plasma emissivity distribution, which is computed at 208 discrete positions, in a poloidal plane. In terms of the poloidal coordinates (R, z) of the 2D tokamak cross section, the total brightness, b_i , incident on a single detector, i , is given by²³

$$b_i = \int_r G(R, z) dr, \quad (2)$$

where $G(R, z)$ is the plasma emissivity distribution (in W/m^3) and r is the LOS corresponding to b_i (Fig. 2).

By discretizing Eq. (2), one obtains the plasma emissivity distribution at a finite number of positions (or pixels) along a

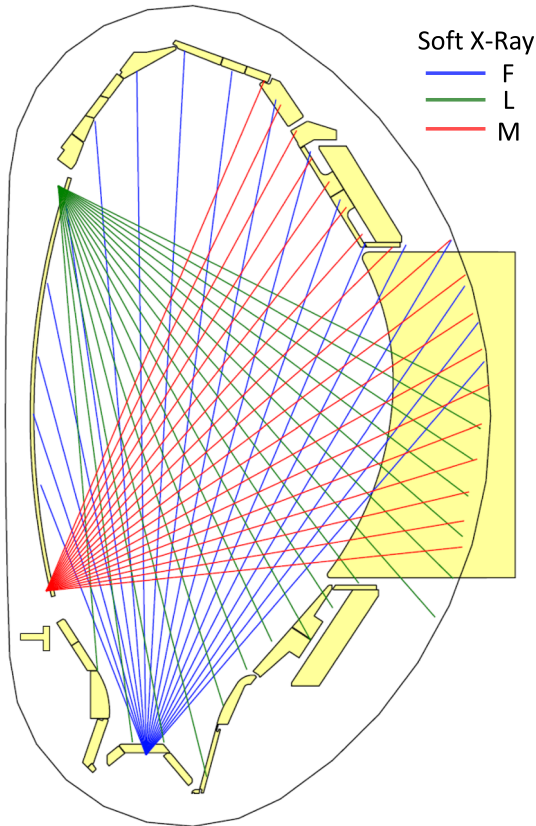


FIG. 2. ASDEX Upgrade cross section, and partial schema of the SXR measurement system, with three cameras (F, L, and M) shown (plot obtained with *diaggeom*).

tomographic reconstruction grid. In this case, the incident radiation on a single detector, assuming an associated noise, ξ , is²⁴

$$b_i = \sum_{j=1}^n M_{ij} g_j + \xi_i \quad i = 1, \dots, 208. \quad (3)$$

From now on, we will denote the set of values b_i , that is, a set of 208 line-integrated SXR measurements of the plasma emissivity taken at a certain point in time, as the plasma's tomographic projection in that instant. We denote Eq. (3) as the *forward model* of the problem. Here, n corresponds to the total number of pixels on a tomographic reconstruction grid, whereas $M_{i,j}$ is the discretization of the function $M(R, z)$ in Eq. (2), mapping the relative contribution of pixel j of that grid to the measurement i of the projection. The actual values of M were pre-defined and contingent on the geometry and configuration of the sensors inside the machine, which can vary between different shot campaigns. Consequently, we denote M as the *geometric matrix*. The goal of a tomographic reconstruction algorithm is then to solve the ill-posed problem by using the tomographic projection (i.e., the 208 measurements b_i) and some *a priori* knowledge about the plasma to find a suitable tomographic reconstruction g that satisfies Eq. (3).

C. Regularization-based methods

To solve the ill-posed problem, traditional tomographic algorithms use regularization techniques, usually based on assumptions regarding the smoothness of the plasma emissivity profile, that constrain the space of possible solutions. Such algorithms, however, are often computationally expensive and typically can only be used for post-experimental tomographic reconstruction due to computational time constraints. In addition, the quality of the reconstructions is highly dependent on assumptions made about the data.⁴ Generally, those assumptions are encoded into the reconstructions through the use of Tikhonov regularization. In this case, computing the tomographic reconstruction of the plasma emissivity profile becomes a matter of finding a reconstruction \hat{g} such that

$$\hat{g} = \arg \min_g (\|Mg - b\|^2 + \Lambda O(g)), \quad (4)$$

where $O(g)$ is a penalty term that encodes information about expected properties of the target plasma distribution, multiplied by a regularizing parameter Λ that controls the regularization strength.²⁵ There are several options for the choice of the regularization term O ; typical choices are the Laplace operator, which favors smooth solutions, and minimum Fisher information,²⁶ which favors solutions that are mostly flat in low-intensity regions and peaked in high-intensity ones.

D. Deep learning-based methods

Recent work has applied deep learning algorithms to the tomographic problem, namely, by using de-convolutional neural networks to produce tomographic reconstructions taking measurement data as input.^{10,27} This is achieved by training the networks on reconstructions that have been previously computed using standard tomographic algorithms. Generically, in a deep learning setting, a deep neural network is *trained* to learn a function mapping an input

x into its target output y ,²⁸ that is,

$$y = y(x, \theta), \quad (5)$$

where θ denotes the neural network's parameters, i.e., its weights and biases. The training process consists in finding an optimal value for θ that minimizes the mismatch between the network's outputs and their corresponding labels.

In our setup, training a deep neural network to produce tomographic reconstructions would have required training it with measurements from the SXR diagnostic and pre-computed reconstructions, produced by other algorithms (namely, regularization-based ones). The expectation would then have been that the parameters θ computed during training would have converged to values such that if new unseen data were fed into the network, it would be capable of *generalizing* outside of its training set. However, even assuming good generalization capacity of a neural network, it is at most as good as whatever data it has been trained on. In other words, should existing tomograms have had errors, a neural network would have learned to reproduce them.

E. Gaussian process tomography

Another alternative is to use Bayesian probability theory to produce tomographic reconstructions, by treating the underlying unknown plasma emissivity distribution as a *Gaussian process*. Evaluating that process along a discrete set of points (the tomographic reconstruction grid) yields a multi-dimensional Gaussian distribution.

By definition, in the Gaussian process framework, one assumes that multiple solutions for the tomographic reconstruction exist in a Gaussian distribution of possible solutions. Treating the tomography problem with this framework allows using Bayesian formalism, which guarantees that the most likely solution for the tomographic reconstruction (i.e., the maximum *a posteriori*, or MAP, estimate), subject to some assumptions about the underlying physical and data distributions, can be computed through Bayes's formula,²⁹

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (6)$$

In the Gaussian process tomography (GPT) setting, the terms in the formula are multivariate probability distributions, which are assumed to be Gaussian. Each of those distributions is specified by a vector of means (which we denote μ), whose entries are the individual means of each random variable in the multivariate distribution, and a covariance matrix, which we denote Σ , where each entry denotes the pair-wise covariance between those same variables.

In Bayes's theorem, the term $P(A)$ is called the *prior*. In GPT, by denoting the underlying plasma emissivity as e , the prior distribution $P(e) \sim \mathcal{N}(\mu_{pr}, \Sigma_{pr})$ encodes existing assumptions about the physical emission process, without observing any data (SXR measurements); this is equivalent to the assumptions one might encode in the regularization parameter in traditional tomography [Eq. (4)]. Each random variable in the prior distribution is also Gaussian and corresponds to the prior plasma emissivity e at each point x in the tomographic reconstruction grid.

Here, the prior mean has a size equal to that of the tomographic reconstruction grid, n . Intuitively, the prior covariance matrix Σ_{pr}

encodes information about the expected smoothness of the plasma emissivity. The entries in the covariance matrix are computed for all pairs of points in the reconstruction grid through a prior covariance function. One covariance function generally used in Gaussian process regression is the squared exponential;³⁰ in using this function, the prior covariance between a pair of points x_1 and x_2 in the tomographic reconstruction grid becomes

$$\text{cov}(x_1, x_2) = \theta_f^2 \exp\left(-\frac{d(x_1, x_2)}{2\theta_x}\right), \quad (7)$$

where $d(x_1, x_2)$ is a distance metric between points x_1 and x_2 . The prior covariance is only dependent on that distance and on $\theta = \{\theta_f, \theta_x\}$, which are the model's *hyperparameters* and are common to all points in the reconstruction grid. The parameter θ_f controls the prior variance of the plasma emissivity at a given location in the reconstruction grid, whereas the parameter θ_x , usually referred to as the *length scale*, controls the extent to which points at a certain distance from each other in the reconstruction grid are correlated. Models where the length scale is large yield high correlations even between grid points that are far apart, while smaller length scales yield covariance matrices where only points that are closer to each other are significantly correlated.

With these definitions, the prior becomes a probability distribution for the plasma emissivity, e , subject to the model's hyperparameters, i.e., $P(e|\theta)$, before any data, that is, a tomographic projection, has been observed. The prior can then be updated by multiplying it with the likelihood of the data d (as per Bayes's theorem), yielding the posterior distribution, $P(e|d, \theta)$,

$$P(e|d, \theta) = \frac{P(d|e, \theta)P(e|\theta)}{P(d|\theta)}. \quad (8)$$

The denominator in Bayes's theorem is known as the model *evidence* or *marginal likelihood*; if one merely computes the posterior $P(e|d, \theta)$, it can be ignored, as it is just a normalizing constant. Interestingly, however, one can use this term to compare several different models (each with their own prior) and choose the one that best fits the data.¹² In this case, one assumes a *hyper-prior*, from which different possible priors (individually specified by different hyperparameters) are sampled. The evidence can then be computed for different models—a process that is referred to as *marginalization*—and the model with the highest evidence can be selected.³¹ Calculating this requires an evaluation of the integral³² over e ,

$$P(d|\theta) = \int P(d|e, \theta)P(e|\theta) de, \quad (9)$$

which in many cases is analytically intractable. However, in our case, the prior for an emissivity distribution is a multivariate Gaussian, defined as

$$P(e|\theta) = (2\pi)^{-\frac{k}{2}} |\Sigma_{pr}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mu_{pr} - e)^T \Sigma_{pr}^{-1} (\mu_{pr} - e)\right), \quad (10)$$

where k denotes the number of variables in the prior distribution (that is, the number of pixels in a reconstruction grid). In addition, we assume a data distribution that is also Gaussian, $P(d|\theta) \sim \mathcal{N}(\mu_d, K + \Sigma_d)$.¹² The mean μ_d of the data distribution is merely the value of the measurements in a projection. The data co-variance has two components: matrix K denotes the (noise-free) co-variance

values and is a linear transformation of the prior covariance Σ_{pr} (imposed on the plasma emissivity) into the measurement (data) space, given by $K = M\Sigma_{pr}M^T$, where M is the geometric matrix defined in Eq. (3). The other component, Σ_d , is a diagonal matrix whose non-zero entries are the absolute noise values, ξ , of each measurement in a projection; we assume the noise values are independent from each other. By assuming that this noise is also Gaussian, the logarithm of the integral in Eq. (9) can be analytically calculated as³²

$$\log(P(d|\theta)) = -\frac{1}{2} \left(m \log(2\pi) + \log|K + \Sigma_d| + (\mu_d - f_L)^T \times (K + \Sigma_d)^{-1} (\mu_d - f_L) \right), \quad (11)$$

where m is the number of SXR measurements in a tomographic projection and f_L is the mapping of the prior mean, μ_{pr} (imposed in the reconstruction space), onto the measurement space, given by $f_L = M \cdot \mu_{pr}$.

The marginalization procedure is particularly useful because the trade-off between the model complexity and data fit is automatic—the model for which the evidence score is highest is always the simplest model that can explain the data, an embodiment of Occam's razor principle.³³ In addition, the model evidence is also a function of the variance σ^2 of the data (through matrix Σ_d), which means that it is possible to treat the expected projection error as an additional hyperparameter of the model to be tuned; this can be done, for example, by treating the data variance as a fraction of the measured value of SXR radiation in the tomographic projection, with the value of the fraction constituting an additional model hyperparameter. This means that through the Gaussian process tomography framework, one can estimate not only the most likely model for the underlying plasma emissivity distribution but also the most likely model for the error values of the data (though this is no replacement for a calibration of the detectors with a known source).

Once the most likely model is selected, and applying Bayes's formula, the posterior mean, μ_{post} , and posterior covariance, Σ_{post} , as functions of the prior mean μ_{pr} and prior covariance Σ_{pr} for that model are, respectively, given by³⁴

$$\mu_{post} = \mu_{pr} + \Sigma_{pr}M^T(K + \Sigma_d)^{-1}(d - f_L) \quad (12)$$

and

$$\Sigma_{post} = \Sigma_{pr} - \Sigma_{pr}M^T(K + \Sigma_d)^{-1}M\Sigma_{pr}. \quad (13)$$

By computing the posterior distribution $P(e|d, \theta) \sim \mathcal{N}(\mu_{post}, \Sigma_{post})$, one can then produce tomographic reconstructions either by sampling from $P(e|d, \theta)$ or simply by taking the mean of that distribution as the tomographic reconstruction (because the distribution is Gaussian, the mean corresponds to the maximum *a posteriori* estimate). In addition, one can directly obtain uncertainties for the tomographic reconstruction from the diagonal values of the posterior covariance matrix, which correspond to the individual posterior variances of each pixel in the reconstruction grid.

The drawback of the marginalization procedure, however, is its potential computational complexity. First, the calculation of the evidence term involves a series of matrix multiplications and an inversion, which can be cumbersome particularly in our setting because of

the dimensionality of the data, which generates very large matrices. Matrix K in Eq. (11) can be previously computed and kept in memory when performing model selection (which we do). However, in our setting, we treat the underlying error as a fraction of the data, and therefore, the values appearing in matrix Σ_d change with every new data point. As a result, the matrix determinant and inversion in Eq. (11) must be re-computed for every new point, which is the main reason behind the high cost of the Bayesian optimization procedure. Furthermore, the evidence must be computed for all models that are taken into consideration. When each model has several hyperparameters, the number of possible models to evaluate can become very large, which means that finding the optimal one can be time-consuming. For practical purposes, this limits the number of models that can be evaluated and, thus, potentially limits the quality of the results.

We, therefore, propose to bypass the need for analytical marginalization, by training a classifier (in this case, a convolutional neural network) to automatically choose the most likely model (out of several pre-defined ones) for the tomographic projection data collected by the ASDEX Upgrade SXR system.

This has potentially several advantages. On one hand, a Gaussian process model, while potentially having priors and posteriors with many dimensions, can be fully specified by its much smaller set of hyperparameters. In practice, this allows for parameterizing a distribution of high dimensionality with only a few variables. In the case of this work, this means that neural networks will learn to map tomographic projections to a lower-dimensional space (of dimensionality equal to the number of models under consideration). This should facilitate the network's learning process, allowing for easier generalization when compared with deep learning methods that attempt to map projections directly into a reconstruction space of larger dimensions. On the other hand, for potential real-time applications, this method potentially speeds up Gaussian process tomography since it bypasses the marginalization procedure.

III. METHODS

A. Soft x-ray data

For this work, we had at our disposal a collection of 112 ASDEX Upgrade shots, totaling 127 528 data points (208-dimensional tomographic projections), with each dimension corresponding to a specific detector in the Soft X-Ray (SXR) system. The projections come from the down-sampled signal of the SXR diagnostic, at a sampling rate of 250 Hz. The dataset also contains an error model, which assigns every measurement in every projection an estimated error value; we develop this topic in Sec. III B. In many cases, the SXR detectors can be damaged and yield completely erroneous measurements, such as negative brightness; in these cases, the measurement is simply considered to be faulty. A sample projection can be seen in Fig. 3.

We also possessed a geometric matrix M that maps the relative contribution of each pixel in a 60×40 (2400)-dimensional tomographic reconstruction grid to each of the 208 SXR measurements in a projection. Each pixel in the grid has a pair of poloidal coordinates (R, z) based on the poloidal dimensions of ASDEX Upgrade; the tomographic reconstruction is computed on this grid. The

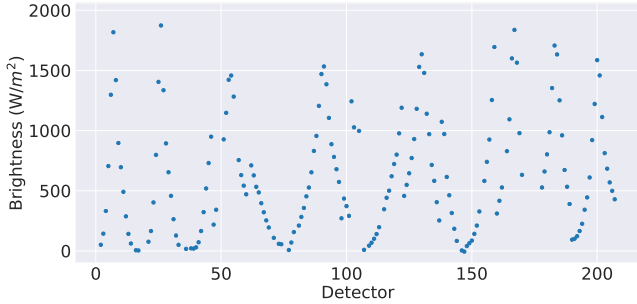


FIG. 3. Sample tomographic projection (SXR measurements) from the ASDEX Upgrade tokamak, taken from shot No. 30294 at $t = 58\,691$ s. Faulty measurements have been removed.

geometric matrix itself was computed based on the physical layout of the SXR sensors in the ASDEX Upgrade vessel and holds for all shots in our dataset.

B. Dataset generation

Before training the neural network classifier, we generated its training and validation dataset by individually computing, for all the measured tomographic projections, the most likely model from which those projections were sampled. To that end, the first task was to define different models and their respective priors (individually specified by their specific set of hyperparameters) and then, through Eq. (11), compare them based on their evidence.

As is typical in Gaussian process regression tasks,³² for all models, we defined the prior means, μ_{pr} , as vectors of zeros, of size 2400 (the size of the reconstruction grid). We computed the prior covariance matrices Σ_{pr} using a squared exponential function, as defined in Eq. (7); in essence, this covariance function encodes our belief that correlations between pixels on the tomographic reconstruction grid will decay exponentially as the distance between those points increases. We computed the distance between pairs of points in the reconstruction grid (expressed in terms of their poloidal coordinates) using the Euclidean definition, i.e., $d(x_1, x_2) = \sqrt{(R_1 - R_2)^2 + (z_1 - z_2)^2}$. The covariance function [Eq. (7)] has only two parameters: θ_f , the individual variance of single pixels, and θ_x , the length scale that control the extent of the correlation between pixels in the reconstruction grid. Different models have priors specified by different values of these hyperparameters, but they all use the same definition of the co-variance function and distance.

Finally, we defined, for each model, our assumptions regarding the data distribution associated with that model. For all models, we discarded measurements that had been previously labeled as faulty. In practice, this meant that, when evaluating the evidence for models, and when computing the maximum *a posteriori* (MAP) estimate for the plasma emissivity, some of the 208 measurements of each projection were not used. We treated the remaining (non-faulty) measurements in the projections as the mean values μ_d of the data distribution.

The individual variances, σ^2 , of the variables in the data distribution correspond to the entries in the diagonal of matrix Σ_d

of Eq. (11) and represent the uncertainties in the measurements. We computed the values σ^2 as fractions of the measurement values themselves; those fractions depend on a scaling factor θ_{err} that is multiplied by the measurements and constitutes an additional hyperparameter for the models under consideration. We assumed this value to be global, i.e., for any given model, we assume that the scaling factor is the same for all measurements in a projection.

Formalizing, we iteratively computed, for each individual data point (i.e., projection), and from a set of pre-defined models for the plasma emissivity and data distributions that might have generated that projection, the highest-evidence model, that is, through Eq. (11), we looked for $\hat{\theta} = (\hat{\theta}_f, \hat{\theta}_x, \hat{\theta}_{err})$ such that

$$\hat{\theta} = \arg \max_{\theta} \log P(d|\theta),$$

where $P(d|\theta)$ is the model evidence from Eq. (8). We searched for the ideal hyperparameters (i.e., the hyperparameters that specify the highest-evidence model) in a grid by assuming a uniform hyper-prior (all models were considered to be equally likely) and computed the model evidence at several discrete positions in the hyper-prior space. The question was then, what positions in the hyper-prior space should one evaluate the models' evidence on. This required taking several factors into account.

The first requirement was the expected nature of the plasma emission process itself. A previous analysis of the measurement data, and of existing tomographic reconstructions from ASDEX Upgrade,⁴ showed that the plasma emissivity has a wide dynamic range for different regions of the plasma, with emissivity in the plasma core being up to 3 orders of magnitude higher than in the pedestal. Likewise, in some periods of some shots, the maximum radiation value in the reconstruction grid was in the order of magnitude of 10^2 W m^{-3} , while in other phases, it could be as large as 10^5 W m^{-3} . Thus, we considered this range in emissivities a good region to explore possible values for the hyperparameter θ_f . In addition, ASDEX Upgrade has minor radii $a = 0.5$ m (horizontally) and $b = 0.8$ m (vertically),³⁵ given this and the size of our reconstruction grid, we assumed that a good region of the hyper-prior in which to evaluate the evidence for certain values of θ_x ranged, in the limit, from 0 (no correlation at all between pixels) to 1.6. For the hyperparameter θ_{err} , we assumed that, in the limit, it could range from 0 (no noise in the tomographic projections) to 1 (all the measured brightness corresponded to noise).

The second requirement related to the training process for neural networks. For this work, we wanted to train a neural network to perform a classification task—to learn to map measurements to the most likely model. Typically, in a machine learning classification setting, care should be taken such that training samples fed to a network are reasonably balanced with respect to their different classes, that is, a good training practice is that one class not be too over-represented in the data when compared to others. In our setting, achieving this balance required experimenting with different potential evaluation positions in the hyperparameter search grid. This comes at the cost of leaving out some grid positions for which some data points might have had higher evidence scores.

In practice, considering these requirements, we performed several evaluations through trial and error of the hyper-prior at different positions; we settled on a $3 \times 3 \times 3$ grid, where the points

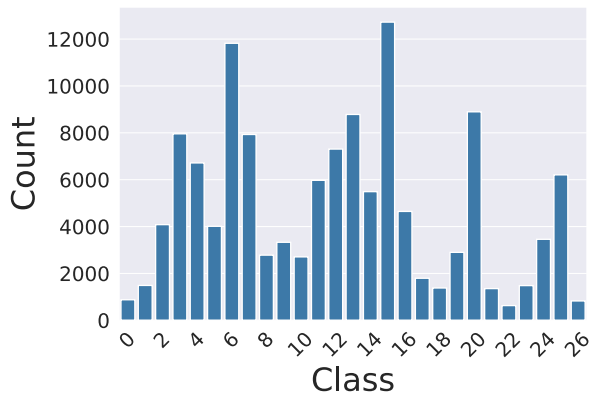


FIG. 4. Number of data samples (from all available shots) mapping to each class (set of hyperparameter values), after the marginalization procedure.

correspond to $\theta_f = \{500, 1000, 2500\}$, $\theta_x = \{0.15, 0.175, 0.2\}$, and $\theta_{err} = \{0.5, 0.75, 1\}$, which corresponds to 27 Gaussian process models. Performing the Bayesian model selection procedure on all projections in our dataset using the models parameterized by these values of $(\theta_f, \theta_x, \theta_{err})$ yielded a relative balance in terms of the amount of data samples mapping to each of the 27 possible classes (points on the hyperparameter grid); this can be seen in Fig. 4 that shows the number of points mapping to each class. Computing the evidence for different models for all tomographic projections took a total of 48 h.

This dataset—i.e., the mappings between tomographic projections and the class to which their highest-evidence model belongs (out of 27 possible ones)—was then used to train and test the neural network classifier. The choice of modeling the task with a classifier, instead of treating it as a regression problem, has one

motivation: the loss function to use, and how to model the network outputs.

If performing regression, one option would have been to have a separate model output for each hyperparameter and combine them with a mean squared error loss. However, it is not obvious that this would work correctly because the evidence term depends on all hyperparameters together. For example, a target output $y_t = (\theta_f = j, \theta_x = k, \theta_{err} = l)$ could be approximated by the network as $y_n = (\theta_f = j, \theta_x = 0.9 k, \theta_{err} = l)$. Computing the mean squared error between y_t and y_n would yield a potentially good score because on average because the hyperparameter values are similar in both cases; however, there is no such guarantee for the evidence score, which could be very different from one case to the other. In fact, it is for this very reason that when performing classification, we use a single output with 27 possible categories, instead of a separate output (and loss function) for each hyperparameter, each with three possible categories. The alternative would have been to use a loss function based on the evidence score itself; however, it would have been computationally infeasible because it would require the evaluation of Eq. (11) for each network gradient update.

C. Deep learning model

Several possibilities exist when it comes to modeling deep neural network architectures. For our purposes (the learning of the Bayesian model selection procedure), we opted to use a convolutional neural network (CNN). CNNs are widely used for signal processing tasks due to their ability to efficiently detect spatial correlations in data, which is what we expected to find in our SXR measurements. The model we used is, with regards to its architecture, inspired by the network for the classification of images described in Ref. 36, popularly known as the VGG network. We designed the model using the Keras framework for deep learning.³⁷

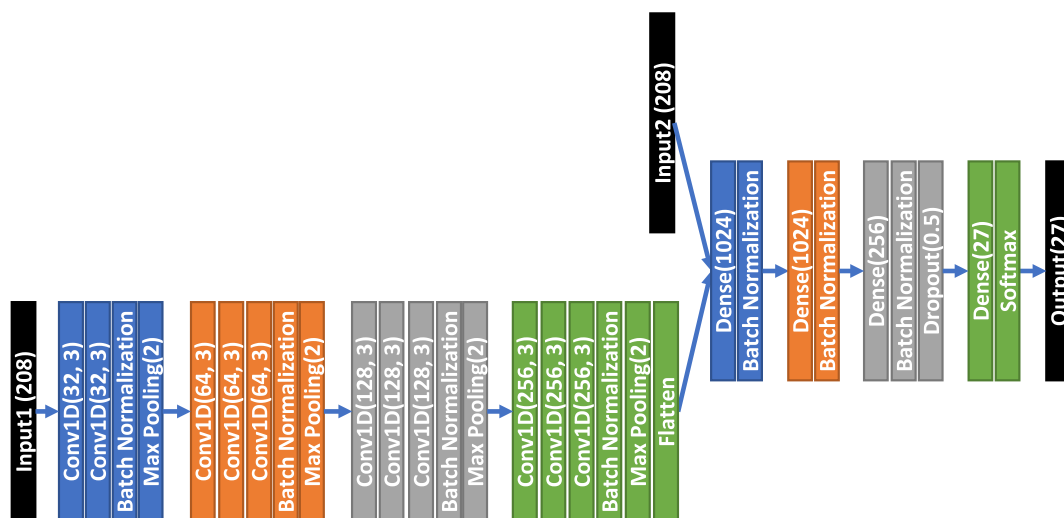


FIG. 5. Schematic of the deep learning model used for this work.

The network itself receives two inputs: a tomographic projection (208 SXR measurements, fed as input 1 in Fig. 5) and a corresponding mask of ones and zeros (taken from the existing error model in our dataset) corresponding to input 2 in Fig. 5, which gives information regarding which measurements in the projection are assumed to be faulty. The network uses a series of convolutional layers followed by max pooling layers to process high-level features in the measurement data. The output of those layers is then combined with the information in the error mask and processed in the last layers of the network, which are standard fully connected layers. We also used batch normalization³⁸ layers to speed up training and dropout in the final layer³⁹ to increase the network's capacity for generalization outside of its training set. We used the rectified linear unit (ReLU) activation function throughout the entire network apart from the last layer, which uses a softmax function, because we modeled the network output as probabilities over 27 possible classes, which must add up to 1. For the same reason, we used categorical cross-entropy as the loss function. We used the Adam optimizer⁴⁰ and left all optimizer hyperparameters at their default values.

IV. RESULTS

We here performed two separate assessments. First, we evaluated the accuracy of the neural network's fit of the individual projections to their highest-evidence models. Then, based on the highest-probability class determined (by the network) for each data point, we computed the corresponding maximum *a posteriori* estimate of the tomographic profile and measured the fit of those reconstructions to the data by projecting them back into the measurement space [through the forward model in Eq. (3)], obtaining their *back-projections*. We then measured the deviation between those back-projections and the original tomographic projections.

A. Neural network

To increase the robustness of our methods, we opted to train an ensemble of neural networks (of equal architectures), using the *k*-fold cross-validation strategy.⁴¹ The *k*-fold cross-validation is useful to determine whether the choice of the train/test split has biased whatever results have been obtained or whether the results can be assumed to hold independently of the data split. We opted to divide our data into *k* = 10 folds, that is, we trained ten networks with different overlapping splits of train data and tested them on non-overlapping validation splits. We trained the networks for 50 epochs and ran them on an NVIDIA Quadro RTX 5000 graphics processing unit (GPU). The total training time for the whole ensemble was 1 h, while the total prediction time for the validation data was 41.62 s.

As the networks performed a 27-way classification, we used top-*k* categorical accuracy as a metric for network classification quality. We now follow with a brief explanation of this metric.

Each data point *x* (corresponding to a tomographic projection) in our dataset was assigned a label, y_{label} , denoting for which of the 27 model classes, the evidence was highest. A classifier learns, through the training process, to compute the probability of that point belonging to a certain class $P(C(x) = c)$, where *c* can take one out of 27 possible values; we denote the vector containing the probabilities of belonging to each of those classes y_{pred} . We further define y_{pred_k} as

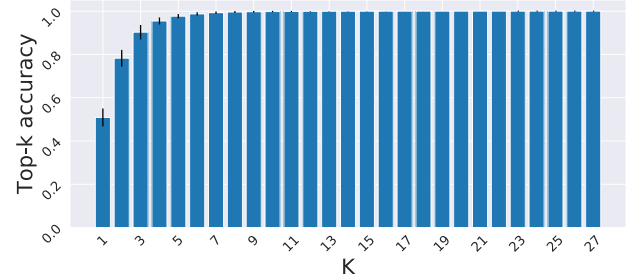


FIG. 6. Top-*k* accuracy (up to *k* = 27) for validation data. The blue bars indicate the mean accuracy across the ensemble of ten networks, while the smaller black bars indicate the accuracy's standard deviation across the ensemble (for each *k*).

the *k*th most likely class given by a classifier for *x*; for example, for y_{pred_1} , one would get

$$y_{pred_1} = \arg \max_c P(C(x) = c) = \arg \max_c y_{pred_c},$$

whereas for c_{27} , one would have

$$y_{pred_{27}} = \arg \min_c P(C(x) = c) = \arg \min_c y_{pred_c}.$$

Based on this, the top-*k* accuracy metric then calculates for each data point,

$$acc_k(x) = \begin{cases} 1 & \text{if } y_{label} \in \{y_{pred_1}, \dots, y_{pred_k}\} \\ 0 & \text{otherwise.} \end{cases}$$

We then computed the categorical accuracy metric on the validation data for the 27 different values of *k*. Because we opted for a cross-validation train and test strategy (with an ensemble of ten classifiers), we show the mean value and standard deviation of the top-*k* accuracy across all members of the ensemble. The results of the metric can be seen in Fig. 6 (up to *k* = 27) and Table I (up to *k* = 5). In Table I, we show the results only up to *k* = 5 for ease of comprehension.

An analysis of Table I and Fig. 6 shows that the ensemble of ten neural networks achieves very good results on the classification task, with a mean top-5 accuracy score of 0.976 (out of a maximum score of 1) for validation data. This means that for any data point, the correct prior is found within the top-5 most likely outputs predicted by the network in 97.7% of cases. In practice, if one is exclusively interested in finding the single, most likely, prior, this result reduces the search space for the right hyperparameters from

TABLE I. Accuracy mean and standard deviation across the ensemble of ten neural networks, for validation data, up to top-5 accuracy.

	K				
	1	2	3	4	5
Mean	0.509	0.783	0.903	0.955	0.977
St. dev.	0.041	0.038	0.033	0.016	0.01

27 classes to 5. Should one be interested only in comparing different models for certain physical distributions, this result also allows for quickly estimating which priors are more or less likely. Furthermore, the standard deviation of the accuracy score demonstrates consistently low values, indicating that the choice of train/test split for our data did not significantly bias the achieved results; all neural networks in the ensemble behave similarly, even if tested on different data.

B. Sample reconstructions

In addition to evaluating the neural network's capacity for classification purposes, we also produced and evaluated tomographic reconstructions. To that end, we took, for each data point in the

validation dataset, the most likely class prediction given by the neural network; this class prediction maps to one of the models we have previously defined. We then computed, based on the class prediction and Eqs. (12) and (13), the posterior mean and covariance for each data point. We took the posterior means (i.e., the maximum *a posteriori* estimates) as the tomographic reconstructions of the plasma emissivity—each mean was a 2400-dimensional vector, where each entry μ_j denotes the most likely value for the plasma emissivity in a point j in the reconstruction grid. The posterior covariances allowed us to determine the error of the tomographic reconstruction, by taking the diagonal of the covariance matrix, which corresponds to the individual variance σ_{post}^2 of each pixel in the reconstruction grid; we converted the value of that variance into a percentage error by once again taking advantage of the $3 - \sigma$ rule and computing the said

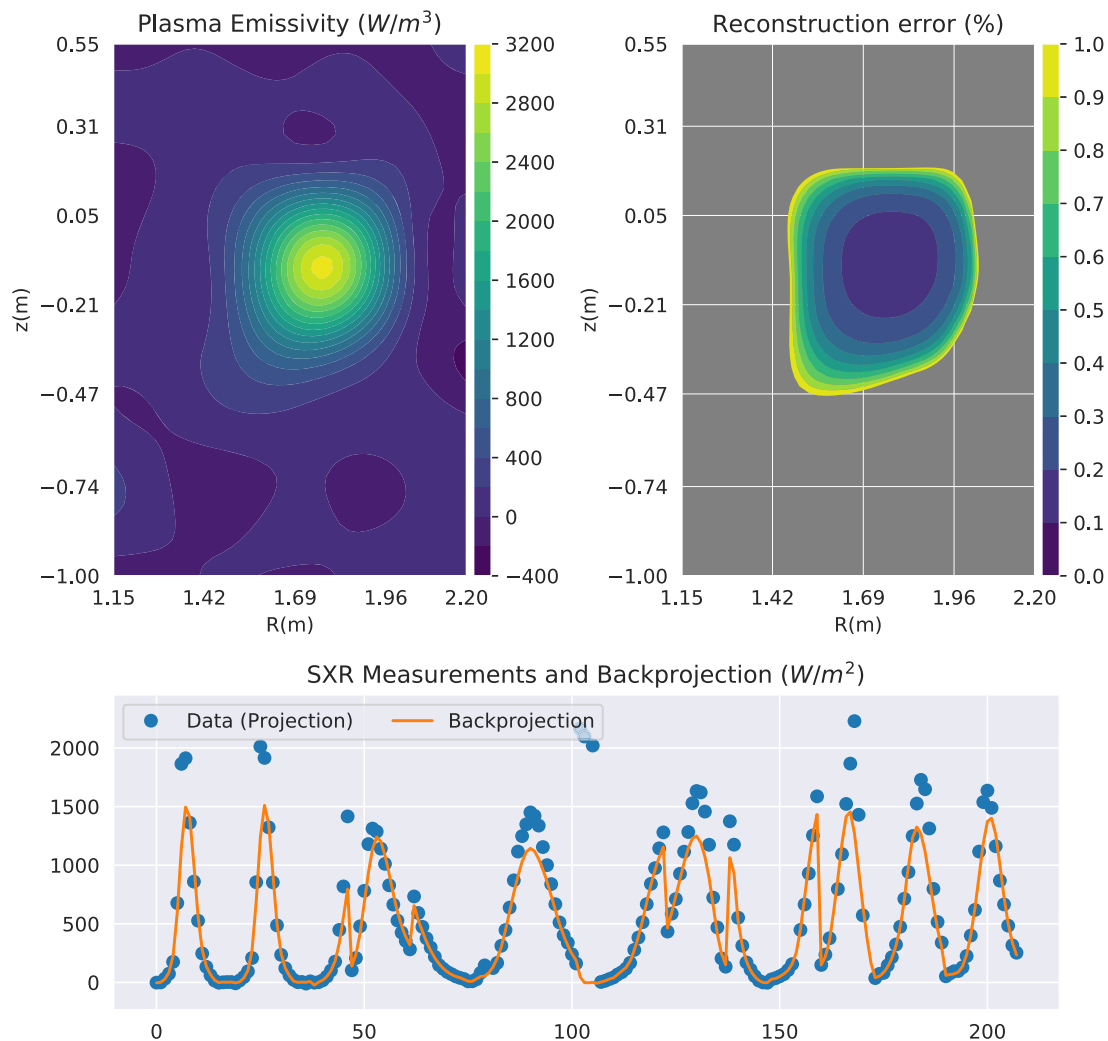


FIG. 7. Sample tomographic reconstruction and error, and comparison between the SXR measurement and the back-projected reconstruction, for ASDEX Upgrade shot No. 30857, $t = 4.0441$ s. The determined model hyperparameters by the classifier were $\theta_{err} = 0.75$, $\theta_t = 500$, and $\theta_x = 0.175$; 200 measurements (out of 208) were used for this reconstruction.

percentage, for pixel j , as

$$\%_{err_j} = 3 \frac{\sqrt{\sigma_{post_j}^2}}{\mu_j} \times 100\%.$$

Two sample results can be seen in Figs. 7 and 8. For the reconstruction error, we show only points where the percentage error was found to be below 100%. Note how in Fig. 8, despite the value of σ_f being 2500, a reconstruction with a much larger maximum intensity can still be produced. Furthermore, in both cases, the reconstruction error values are noticeably lower in the center of the grid. We explain this with two factors: the larger number of LOSs covering that region, which lowers the uncertainty in the reconstructed values, and the higher intensity of the plasma emissivity, which lowers the relative error.

C. Model complexity and data fit

To evaluate the quality of the models, we performed, for each maximum *a posteriori* (MAP) tomographic reconstruction, a pass through the forward model defined in Eq. (3) to obtain the corresponding back-projection, i.e., the projection of the reconstruction back into the measurement space. The marginalization procedure guarantees that from the ensemble of models that is evaluated for a data point, the simplest model that can fit the data will be chosen, and we have shown that the proposed convolutional neural network can in most cases do this as well. However, a problem can arise if the ensemble of models from which we sample is itself mostly composed of overly simple or overly complex models.

Computing the back-projections allowed us to see how the obtained MAP estimates fit the original SXR data. Our

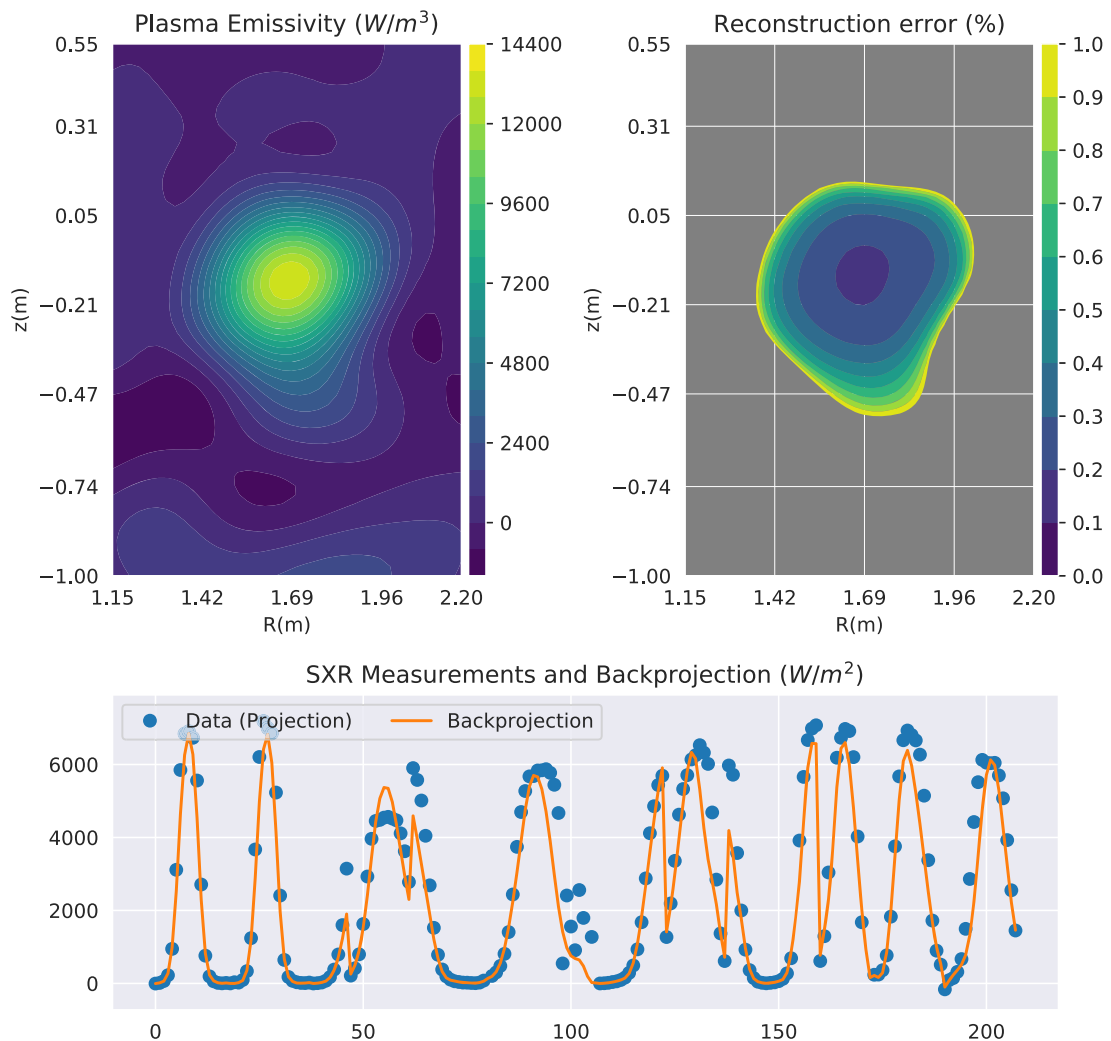


FIG. 8. Sample tomographic reconstruction and error, and comparison between the SXR measurement and the back-projected reconstruction, from ASDEX Upgrade shot No. 31238, $t = 3.2641$. The determined model hyperparameters by the classifier were $\theta_{err} = 1.0$, $\theta_f = 2500$, and $\theta_x = 0.175$; 199 measurements (out of 208) were used for this reconstruction.

expectation was that if the models we defined were too complex, we would observe very tight fits of the data, with very low deviations between it and the back-projections. Conversely, if the models were too simple, we would tend to see large differences between the back-projections and the data.

To check this, we computed the percent deviation between the back-projections and the original tomographic projections and did this for every measurement in every data point. We computed this value as

$$\text{err} = \frac{|\text{backprojection} - \text{measurement}|}{\text{measurement}} \times 100\%.$$

The histograms in Figs. 9(a) and 9(b) show the results of this evaluation, up to a deviation of 100%, a threshold that covers 99.38% of the validation data. Note that the deviation was computed by comparing the back-projection only with valid (non-faulty) measurements. On average, 91.25% of the 208 measurements in each data point were used to compute the tomographic reconstructions.

D. Discussion

Looking at the distribution of the deviations between back-projections and measurement data in Fig. 9, one can see that most back-projected values have relatively low deviations from the data—90% have a deviation lower than 50%. On the other hand, one can also see that some back-projected values have larger deviations, and in fact, a few had an error greater than 100% (though they are not represented in the figure for ease of visualization; they represent only 0.6% of cases). We interpret this as a good result: on one hand, the low back-projection deviations indicate that the models chosen for the tomographic reconstructions have the necessary rigidity to constrain the solutions to mostly match the data, while at the same time being flexible enough to allow for large deviations, when not doing so would render overly complex models. We would like to point out the fact that many other types of models (other than the ones we used) can be chosen. For example, we used the Euclidean distance and a single length scale for the covariance function. Nevertheless, it is possible to use more complex co-variance functions with different length scales in the R and z directions, or with a distance metric that uses the radial and angular coordinates of pixels in the reconstruction grid, to account for the fact that

points in the same flux surface are considered to be highly correlated. In principle, it is always possible to define more complex models that fit the data better and reduce the deviation between projection and back-projection; nevertheless, those models will not necessarily have the highest evidence when compared with simpler ones.

V. CONCLUSIONS

Gaussian process tomography makes it possible to obtain the most likely estimate for an unknown, potentially infinite-dimensional, quantity, given some assumptions about the underlying physical distribution and about the data generated by that distribution. The tomography problem, based on SXR measurement data from the ASDEX Upgrade tokamak, lends itself to investigation under this framework. If one assumes a fixed model for the behavior of the underlying physical distribution (i.e., the plasma emissivity) and for the data, for example, by specifying the length scales involved in the emission process and the expected fraction of noise in the measurements, Gaussian process tomography (GPT) inversion techniques readily yield the corresponding maximum *a posteriori* estimate of the plasma SXR emissivity in the two-dimensional tokamak cross section.

Nevertheless, this raises the issue of what models one would like to assume in the first place. Through Bayesian Occam's razor principle, GPT answers this question by computing the evidence for different possible models; out of which, the one with the highest score can then be selected. This can be useful if one wishes to test different assumptions regarding the data distribution, for example, what fraction of noise can be expected in the observations (measurements). However, in a setting such as SXR tomography with ASDEX Upgrade data, this task can become cumbersome due to the dimensionality of the tomographic projections. This is further compounded when the number of models under evaluation is large.

For these reasons, we developed a novel method for the automatic selection of the best model (out of 27 pre-defined ones) for the plasma SXR emissivity distribution and the corresponding data, for measurements from the ASDEX Upgrade tokamak. The individual models had different assumptions regarding the noise level in the collected data, the correlations between variables in the tomographic reconstruction grid, and the individual variances of those same variables. The method then consisted in training a convolutional neural network to perform the Bayesian model selection (marginalization) procedure and bypass the need to perform that task analytically. Our results show that the neural network achieved good classification results when compared to the analytical Bayesian marginalization step, with top-5 accuracy (out of 27 possible classes) reaching a value of 0.976 (out of a maximum of 1). Furthermore, while the marginalization procedure across the entire dataset (of 127 528 tomographic projections), through analytical methods, took ~48 h, the same computation, performed by the neural network, took only 43 s. Thus, the neural network approach can be particularly useful for high-dimensional data settings such as ours, as well as problems where the number of models under consideration is large, which would otherwise render the model comparison problem too slow through analytical methods. This can be particularly useful for settings where not only the

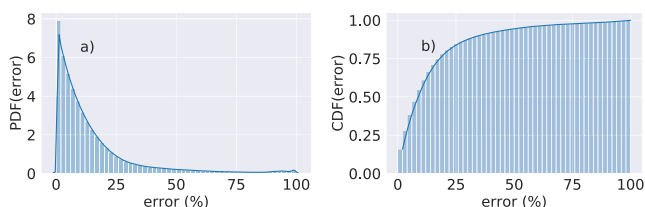


FIG. 9. Distribution of the deviations between the tomographic reconstructions' back-projections into the measurement space and the data. 54.4% of the individual back-projected measurements have a relative deviation lower than 10%, 93.83% have a deviation lower than 50%, and 99.38% have a deviation lower than 100%.

real-time inversion of tomographic profiles but also the real-time comparison of different models for certain physical distributions is a necessity.

ACKNOWLEDGMENTS

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom Research and Training Programme 2014-2018 and 2019-2020 under Grant Agreement No. 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

- ¹A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging* (IEEE Press, 1988).
- ²L. C. Ingesson, B. Alper, B. J. Peterson, and J.-C. Vallet, "Chapter 7: Tomography diagnostics: Bolometry and soft-x-ray detection," *Fusion Sci. Technol.* **53**, 528–576 (2008).
- ³J. Mlynar, V. Weinzettl, G. Bonheure, A. Murari, and JET-EFDA Contributors, "Inversion techniques in the soft-x-ray tomography of fusion plasmas: Toward real-time applications," *Fusion Sci. Technol.* **58**, 733–741 (2010).
- ⁴T. Odstrčil, T. Pütterich, M. Odstrčil, A. Gude, V. Igochine, U. Stroth, and ASDEX Upgrade Team, "Optimized tomography methods for plasma emissivity reconstruction at the ASDEX Upgrade tokamak," *Rev. Sci. Instrum.* **87**, 123505 (2016).
- ⁵J. Mlynar, T. Craciunescu, D. R. Ferreira, P. Carvalho, O. Ficker, O. Grover, M. Imrisek, J. Svoboda, and JET Contributors, "Current research into applications of tomography for fusion diagnostics," *J. Fusion Energy* **38**, 458–466 (2019).
- ⁶A. N. Tikhonov, "Regularization of incorrectly posed problems," *Dokl. Akad. Nauk. SSSR* **153**, 49 (1963).
- ⁷A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Dokl. Akad. Nauk. SSSR* **151**, 501 (1963).
- ⁸V. Loffelmann, J. Mlynar, M. Imrisek, D. Mazon, A. Jardin, V. Weinzettl, and M. Hron, "Minimum Fisher Tikhonov regularization adapted to real-time tomography," *Fusion Sci. Technol.* **69**, 505–513 (2016).
- ⁹D. R. Ferreira, P. J. Carvalho, and H. Fernandes, "Deep learning for plasma tomography and disruption prediction from bolometer data," *IEEE Trans. Plasma Sci.* **48**, 36 (2019).
- ¹⁰F. A. Matos, D. R. Ferreira, P. J. Carvalho, and JET Contributors, "Deep learning for plasma tomography using the bolometer system at JET," *Fusion Eng. Des.* **114**, 18–25 (2017).
- ¹¹A. Jardin, J. Bielecki, D. Mazon, J. Dankowski, K. Król, Y. Peysson, and M. Scholz, "Neural networks: From image recognition to tokamak plasma tomography," *Laser Part. Beams* **37**, 171–175 (2019).
- ¹²J. Svensson, "Non-parametric tomography using Gaussian processes," Technical Report No. EFDA-JET-PR(11)24, 2011.
- ¹³C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research Vol. 37, edited by F. Bach and D. Blei (PMLR, Lille, France, 2015), pp. 1613–1622.
- ¹⁴Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning* (PMLR, 2016), Vol. 48, pp. 1050–1059.
- ¹⁵A. Pavone, J. Svensson, A. Langenberg, N. Pablant, U. Hoefel, S. Kwak, R. C. Wolf, and Wendelstein 7-X Team, "Bayesian uncertainty calculation in neural network inference of ion and electron temperature profiles at W7-X," *Rev. Sci. Instrum.* **89**, 10K102 (2018).
- ¹⁶A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," [arXiv:1511.06434](https://arxiv.org/abs/1511.06434) (2015).
- ¹⁷T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems* (2016), pp. 2234–2242.
- ¹⁸J. Radon, "Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten," *Ber. Verh. Königlich-Säch. Gesellschaft Wiss. Leipzig, Math.-Phys. Kl.* **69**, 262 (1917).
- ¹⁹R. M. Lewitt, "Reconstruction algorithms: Transform methods," *Proc. IEEE* **71**, 390–408 (1983).
- ²⁰F. Matos, "Deep learning for plasma tomography," M.Sc. thesis, Técnico Lisboa, 2016.
- ²¹A. Murari, E. Joffrin, R. Felton, D. Mazon, L. Zabeo, R. Albanese, P. Arena, G. Ambrosino, M. Ariola, O. Barana *et al.*, "Development of real-time diagnostics and feedback algorithms for jet in view of the next step," *Plasma Phys. Controlled Fusion* **47**, 395 (2005).
- ²²V. Igochine, A. Gude, M. Maraschek, and ASDEX Upgrade Team, "Hotlink based soft x-ray diagnostic on ASDEX upgrade," Technical Report IPP 1/338, Max-Planck-Institut für Plasmaphysik, 2010.
- ²³A. Jardin, J. Bielecki, D. Mazon, J. Dankowski, K. Król, Y. Peysson, and M. Scholz, "Synthetic x-ray tomography diagnostics for tokamak plasmas," *J. Fusion Energy* 1–11 (2020).
- ²⁴L. C. Ingesson, P. J. Böcker, R. Reichle, M. Romanelli, and P. Smeulders, "Projection-space methods to take into account finite beam-width effects in two-dimensional tomography algorithms," *J. Opt. Soc. Am. A* **16**, 17–27 (1999).
- ²⁵M. Odstrčil, J. Mlynar, T. Odstrčil, B. Alper, A. Murari, and JET Contributors, "Modern numerical methods for plasma tomography optimisation," *Nucl. Instrum. Methods Phys. Res., Sect. A* **686**, 156–161 (2012).
- ²⁶M. Anton, H. Weisen, M. J. Dutch, W. Von der Linden, F. Buhlmann, R. Chavan, B. Marletaz, P. Marmillod, and P. Paris, "X-ray tomography on the TCV tokamak," *Plasma Phys. Controlled Fusion* **38**, 1849 (1996).
- ²⁷D. D. Carvalho, D. R. Ferreira, P. J. Carvalho, M. Imrisek, J. Mlynar, H. Fernandes, and JET Contributors, "Deep neural networks for plasma tomography with applications to JET and COMPASS," *J. Instrum.* **14**, C09011 (2019).
- ²⁸I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016), <http://www.deeplearningbook.org>.
- ²⁹A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. (O'Reilly Media, Incorporated, 2019).
- ³⁰D. Li, J. Svensson, H. Thomsen, F. Medina, A. Werner, and R. Wolf, "Bayesian soft x-ray tomography using non-stationary Gaussian processes," *Rev. Sci. Instrum.* **84**, 083506 (2013).
- ³¹D. J. C. MacKay, "Comparison of approximate methods for handling hyperparameters," *Neural Comput.* **11**, 1035–1068 (1999).
- ³²C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning* (MIT Press, 2006).
- ³³D. MacKay, "Bayesian methods for adaptive models," Ph.D. thesis, Caltech, 1991.
- ³⁴J. Svensson, A. Werner, and JET-EFDA Contributors, "Current tomography for axisymmetric plasmas," *Plasma Phys. Controlled Fusion* **50**, 085002 (2008).
- ³⁵C. Lechte, G. D. Conway, T. Görler, C. Tröster-Schmid, and ASDEX Upgrade Team, "X mode Doppler reflectometry k-spectral measurements in ASDEX upgrade: Experiments and simulations," *Plasma Phys. Controlled Fusion* **59**, 075006 (2017).
- ³⁶K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.
- ³⁷F. Chollet, Keras, <https://github.com/fchollet/keras> (2015).

³⁸S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research Vol. 37, edited by F. Bach and D. Blei (PMLR, Lille, France, 2015), pp. 448–456.

³⁹X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," in *2019 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR) (IEEE, 2019), pp. 2677–2685.

⁴⁰D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014).

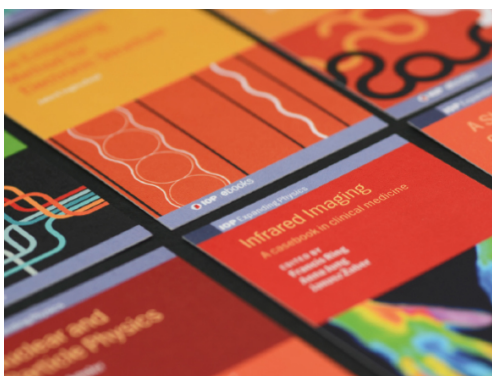
⁴¹G. James, *An Introduction to Statistical Learning: With Applications in R* (Springer, New York, NY, 2013).

PAPER • OPEN ACCESS

Plasma confinement mode classification using a sequence-to-sequence neural network with attention

To cite this article: F. Matos *et al* 2021 *Nucl. Fusion* **61** 046019

View the [article online](#) for updates and enhancements.



IOP | ebooks™

Bringing together innovative digital publishing with leading authors from the global scientific community.

Start exploring the collection—download the first chapter of every title for free.

Plasma confinement mode classification using a sequence-to-sequence neural network with attention

F. Matos^{1,*}, V. Menkovski², A. Pau³, G. Marceca³, F. Jenko¹ and the TCV Team^{3,a}

¹ Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany

² Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands

³ École Polytechnique Fédérale de Lausanne (EPFL), Swiss Plasma Center (SPC), CH-1015 Lausanne, Switzerland

E-mail: francisco.matos@ipp.mpg.de

Received 2 November 2020, revised 1 January 2021

Accepted for publication 5 February 2021

Published 12 March 2021



CrossMark

Abstract

In a typical fusion experiment, the plasma can have several possible confinement modes. At the tokamak à configuration variable, aside from the low (L) and high (H) confinement modes, an additional mode, dithering (D), is frequently observed. Developing methods that automatically detect these modes is considered to be important for future tokamak operation. Previous work (Matos *et al* 2020 *Nucl. Fusion* **60** 036022) with deep learning methods, particularly convolutional long short-term memory networks (conv-LSTMs), indicates that they are a suitable approach. Nevertheless, those models are sensitive to noise in the temporal alignment of labels, and that model in particular is limited to making individual decisions taking into account only the input data at a given timestep and the past data, represented in its hidden state. In this work, we propose an architecture for a sequence-to-sequence neural network model with attention which solves both of those issues. Using a carefully calibrated dataset, we compare the performance of a conv-LSTM with that of our proposed sequence-to-sequence model, and show two results: one, that the conv-LSTM can be improved upon with new data; two, that the sequence-to-sequence model can improve the results even further, achieving excellent scores on both train and test data.

Keywords: CNN, LSTM, H mode, L mode, dither, sequence-to-sequence, attention

(Some figures may appear in colour only in the online journal)


1. Introduction

During nuclear fusion experiments, the plasma can be described as being in one of several possible confinement states. At the tokamak à configuration variable (TCV) toka-

mak, it is typically classified as being in either low (L), dithering (D) or high (H) confinement mode. All shots, during the ramp-up phase of the plasma, begin in L mode. By applying sufficient heating power, the plasma spontaneously transitions into H mode [2] (typically at TCV this process lasts approximately 1 ms). This mode is termed high confinement because, once it is reached, one can observe significantly reduced transport of particles and energy from the plasma to the surrounding vessel walls. This allows for a larger energy confinement per input power; for this reason, most current designs for future tokamaks assume that they will regularly run in H-mode. In

* Author to whom any correspondence should be addressed.

^a See Coda *et al* 2019 (<https://doi.org/10.1088/1741-4326/ab25cb>) for the TCV team.

 Original content from this work may be used under the terms of the [Creative Commons Attribution 3.0 licence](https://creativecommons.org/licenses/by/3.0/). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

some cases the transition from L to H mode does not happen directly, but rather the plasma oscillates rapidly between the two confinement regimes. In this case, the plasma is considered to be in a dithering [3] mode.

Many studies have been done on the physical factors behind the transition between L and H mode, but the phenomenon is still not completely understood [4]. Furthermore, there is no simple set of rules that can be used to determine the plasma mode given the values of the signals during a fusion experiment. Nevertheless, most of the time, there are highly salient patterns in these measured signals that can be used by domain experts to determine the plasma mode with high confidence. For example, a transition from L to H mode can typically be identified by observing a sudden drop in the emitted $H\text{-}\alpha$ radiation. However, these data patterns can be rather complicated and ambiguous; for example, Dithers leave signatures in the emission of photons similar to that of type III edge localized modes (ELMs) [5], which are events that occur during H-mode.

This process of manually labeling the experimental data can be quite cumbersome in many cases, particularly when one wishes to conduct large studies and analyze many shots. For that reason, work has been put into developing tools capable of automating the task of detecting different confinement modes. In particular, in the past few years, research has been done on using machine learning [6–10] and, more recently, deep learning [1] for this task. These algorithms are particularly suitable for dealing with such challenges of extracting patterns from high-dimensional data collected during these experiments.

For example, when analyzing transitions between L and H modes, the type of correlations one expects to find in the data—localized, spatial correlations as well as long-term temporal ones—can be, respectively, efficiently discovered using convolutional [11, 12] and recurrent neural networks (RNNs) [13, 14]. Previous work with models of this sort, in particular with long short-term memory (LSTM) networks, indicates that they can be very accurate in this task [1]. One of the main challenges of these models, however, is that they have to produce a decision about the plasma mode at each timestep by looking only at a given context of the signals and their own past states, which represent past data values. Bi-directional LSTMs can look at future information, but they still only have access to the data when making decisions. In contrast, when a human expert faces a difficult decision, they not only look at the data, but also reason through several possible sequences of plasma confinement mode evolutions. They go back and forward through the input signal, consider the consequences of labeling a mode with a given value for all consecutive modes, and in doing so, frequently revise decisions until the most likely sequence of plasma confinement modes can be determined.

Conceptually, the tasks of automated language translation and the automated labeling of plasma confinement modes are closely related: one wishes to translate a sentence in a source language to a different sentence with the same meaning in a target language. In the case of automated labeling of plasma confinement modes, one can consider signal time traces to constitute the sentence in the source language, while the

corresponding confinement modes can be thought of as the ‘translated’ sentence in the target language.

For this reason, in this paper, we propose an approach that builds upon previous work with deep learning applied to automated detection of plasma confinement modes, by using recent developments in the field of neural machine translation (NMT). Broadly, in that field, one can talk of two main types of algorithms in use: sequence-to-sequence models based on RNNs, and transformers. Transformers are ideal when a large amount of data is available, since they parallelize and scale well with large amounts of data when compared to RNNs, and have achieved tremendous success in processing of short sequences [15].

However, those are not the conditions we face in our setting: the amount of data we have is (compared to typical NMT tasks) rather limited, and furthermore, the sequences we deal with (i.e., plasma shots) are quite long. This suggests that RNN-based models are a more suitable approach, given their capacity to process long sequences with their hidden states. A general, vanilla RNN, such as a convolutional long short-term memory network (conv-LSTM), is, by itself, incapable of producing decisions over sequences of outputs, and is limited to making a sequence of independent decisions based only on the observed data. It is also susceptible to noise produced by misaligned labels. However, sequence-to-sequence models can solve both of these problems. These models, as well as associated mechanisms such as attention [16–18], have considerably advanced the field of NMT and transduction in the past few years.

This paper is organized as follows. Section 2 provides an overview of the field of NMT, in particular by explaining the functioning of sequence-to-sequence models, and how we can adapt them to suit our task. Section 3 details our considerations regarding the data and the problem formulation, our preprocessing steps and the proposed model architecture. Section 4 shows some of the obtained results and scores, and in particular, we compare the results of this model with those obtained in [1]. We then wrap up with a discussion in section 5.

2. Background

2.1. Sequence-to-sequence models

Sequence-to-sequence models have achieved tremendous success in the field of NMT [19, 20]. These models are characterized by two separate components performing different tasks: an *encoder* that reads a sentence in the source language and produces an encoded representation of that sentence, and a *decoder* that, based on the *encoding*, produces an appropriate translation into the target language. The encoder and decoder can technically be any type of algorithm, though in most applications, they are built with RNNs [21] that are jointly trained.

In NMT, the encoder typically maps a word or sequence of words in the source language to a numerical representation of said words, as a function of a pre-defined size of the source language vocabulary. This is then followed by a recurrent layer, typically a LSTM layer, or a gated recurrent unit

that is trained to find sequential correlations in the embedded input sentences. These models can keep track of correlations between points that are far apart in time because they have internal *hidden states*, though their capacity deteriorates if the time interval is large enough. At every source timestep j , with $0 < j < k$ (where k is the length of the source sequence), the encoder computes a new hidden state, h_j ; each vector h_j in the sequence of hidden states (h_0, \dots, h_k) constitutes a summary of the information that the encoder has processed until timestep j , and the final hidden state of the encoder's recurrent layer (h_k) can thus be considered to be an encoded representation of the entire input sequence.

The decoder must, subject to the encoding produced by the encoder (h_k), produce an appropriate corresponding sequence of words in the target language. When using an RNN decoder, this is done by setting its initial hidden state to h_k . Therefore, unlike a simple RNN model, which only receives a part of the source data as input at each timestep, in the sequence-to-sequence model, the decoder can work with a representation of the entire source sequence. The decoder then outputs, at each decoding timestep, a probability distribution over the discrete set of possible outputs, conditioned on the source sequence. Furthermore, in the general formulation of the sequence modeling problem, where the input sequence is not aligned to the output (e.g., different sampling rates of the input and output), the model needs not only to determine the output sequence, but also to align the symbols of the output sequence with the input. Lastly, the decoder can be made autoregressive, by feeding it a selected output at the previous timestep as an input in the next.

Figure 1 illustrates this mechanism: the encoder produces an encoding of the input sequence; the decoder, using that encoding as a starting state, produces a translation into a target sentence. At timestep 1 of the decoding process, a `<start>` character is fed to the decoder; in subsequent steps, the selected output from the previous timestep is fed as input.

An autoregressive decoder can, at inference time (i.e., after training), evaluate several output sequences, by conditioning its prediction at each timestep on different past outputs. Therefore, it is not limited to outputting a single solution, but rather, it can produce a probability distribution of possible solutions.

In practice, sampling from the output joint probability distribution can be done by treating the distribution as a tree data structure, where each path in the tree represents a sample from the distribution, i.e., a different possible output sequence. Expanding a new node in that tree corresponds to sampling a different output from the previous timestep to condition future outputs; searching this tree for solutions (paths) of high probability can be done efficiently with a beam search algorithm [22]. In addition, the tree search can be done (and simplified) by explicitly incorporating domain knowledge, which can allow for pruning paths that are known to be impossible (for example, in our case, by discarding paths that start in H-mode). All of this contrasts with using a simple RNN for translation. In that case, one expects the model to output a (single) solution of high likelihood, there is no way (in inference time) to explicitly encode knowledge that rules out

certain solutions, and the source and target must be of the same length.

2.2. Attention

While encoder–decoder models achieve very good results, they can nevertheless still lose performance when translating long input sequences [16]. There are several reasons for this, but the main one is that the encoder is expected to be able to encode all the information of the source sentence in a single vector; in practice, especially for long input sentences, training such models with algorithms that back-propagate gradient updates can be challenging. For this reason, the *attention* mechanism was developed. The main idea behind it is to extend the decoder with an attention layer that can access the entire sequence of encoder hidden states $h = (h_0, \dots, h_k)$. The attention layer can then, at every decoding timestep i , compute an attention vector α_i , whose values are normalized to add up to 1, and which constitute a series of weights that are used to compute the decoder's *context vector* c_i , defined as:

$$c_i = \sum_{j=0}^k \alpha_{ij} h_j.$$

Intuitively, at every decoding timestep i , the corresponding context vector c_i will change to reflect the greater (or lesser) relevance of some components of h for computing the decoder output at that timestep. While the final encoder state h_k is still fed into the decoder as an initial hidden state, the decoder is no longer wholly dependent on it, and has access to a much richer context thanks to the attention layer, which is trained with the rest of the model. Moreover, the existence of the attention layer can give additional insight into the inner working of the model. At evaluation time, the attention vectors can be collected and used to visualize what parts of an input have been focused on by the model when generating a certain output. Attention significantly improved the results obtained by sequence-to-sequence models, and is also the main mechanism behind the functioning of transformer networks [15].

3. Methods

3.1. Problem formulation

The data used for this work comes from four different signals from the TCV tokamak: the photodiode (PD), measuring H- α radiation; the plasma current; the diamagnetic loop measuring the plasma toroidal flux; and the interferometer, measuring the line-integrated electron density. All of the signals are re-sampled to a frequency of 10 kHz; a more thorough description can be found in [1]. As in that work, we are generically interested in finding, for a given temporal sequence of measurements x_t , with $0 < t \leq N$ (which constitute a single shot), the most likely sequence of plasma confinement mode $\hat{z}_{1:N}$ that explain the observations $x_{0:N}$.

The approach proposed in this paper does this in two parts. On the one hand, a model is trained to estimate the joint probability distribution of the sequence of plasma modes $p(z_1, z_2, \dots, z_N | x_{0:N})$ for a given shot. Second, an algorithm

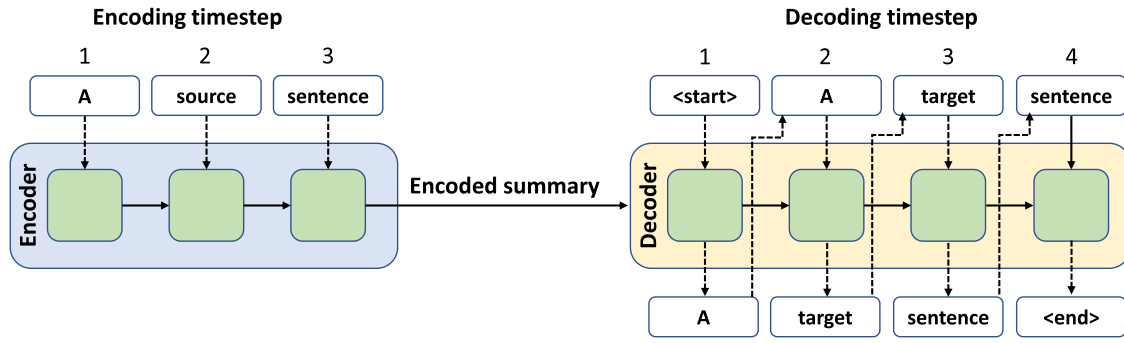


Figure 1. Representation of the flow of information in a sequence-to-sequence encoder–decoder model. Dashed lines denote model inputs and outputs, while solid lines denote hidden states. At decoding timestep 1, the output distribution gives highest probability to the word *A*, which is then fed as input at timestep 2.

finds a sequence $\hat{z}_{1:N}$, drawn from the joint distribution, with high probability. Formally, the task is to find:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} p(z_1, z_2, \dots, z_N | x_{0:N}).$$

with $z_t \in Z$ and $Z: \{‘low’, ‘dither’, ‘high’\}$, and $z_0 = \{‘low’\}$, since any shot is assumed to begin in L-mode.

In practice, in a real-time (RT) environment, we would not possess the entire sequence of measurements (the whole shot), but rather, only the signal values up to a certain point in time t . Thus, one of our requirements is to find a sequence of high probability up until t while looking only at past measurements. For this task, a simple RNN (LSTM) model can be used [1]. However, such models, when making a decision, rely only on the input data and their own internal state. They cannot take their own past outputs into account when making a decision, and therefore, are limited to producing a single point-like estimate for the output sequence of confinement states. In contrast, sequence-to-sequence models can explicitly take their own past outputs into account when making a decision. This way, at time t , a sequence-to-sequence model does not decide on a single output for t , but rather, it decides on the entire sequence of plasma mode evolutions up to that point in time; that is, the model computes the distribution $p(z|x)$ as:

$$\begin{aligned} p(z_1, z_2, \dots, z_N | x_{0:N}) &= p(z_1 | x_{0:1}, z_0) p(z_2 | x_{0:2}, z_{0:1}) \dots \\ &\quad \times p(z_N | x_{0:N}, z_{0:N-1}) \\ &= \prod_t p(z_t | x_{0:t}, z_{0:t-1}), \end{aligned}$$

where $p(z_t | x_{0:t}, z_{0:t-1})$ denotes the probability of observing mode z at time t , given the sequence of observed signals x from time 0 to time t , and the sequence of outputs until t . It is the additional conditioning on past outputs that allows the sequence-to-sequence model to approximate the full joint distribution $p(z|x)$.

One caveat of the sequence-to-sequence model architecture is that it requires that the model observe windows, or subsequences, of the input data (up until time t) of fixed size. This means that in a RT environment, for most values of t , a sequence-to-sequence model would have a delay when computing $p(z_t | x_{0:t})$. This delay corresponds to a pre-defined size

of the signal windows that the model receives, which therefore must be minimal.

With a sequence-to-sequence model, using the notation above, finding a sequence of high probability means finding:

$$\hat{z}_{1:N} = \arg \max_{z_{1:N}} \prod_t p(z_t | x_{0:t}, z_{0:t-1}). \quad (1)$$

The task of finding samples of high probability from the distribution is done, in the case of this work, with a beam search algorithm [22]. Because in our setting the sequences have potentially thousands of timesteps, computing their individual likelihoods using products as in equation (1) would yield numerically unstable results; thus, the beam search uses the logarithm of the probabilities:

$$\begin{aligned} \hat{z}_{1:N} &= \arg \max_{z_{1:N}} \log \left(\prod_t p(z_t | x_{0:t}, z_{0:t-1}) \right) \\ &= \arg \max_{z_{1:N}} \sum_t \log p(z_t | x_{0:t}, z_{0:t-1}), \end{aligned} \quad (2)$$

which allows it to use sums instead, and to look for the sequence whose log-probability is greatest.

3.2. Data engineering

Events such as, for example, the LH transition can be roughly pinpointed in a signal time trace. However, it is difficult to specify precisely, on a consistent basis, at which exact point in time the transition happens; for example, in some shots, the transition might be quite sudden, whereas in others, the transition signatures in the data can be more spread out over time. The typical time that a TCV shot takes to make an HL transition is on the order of 1 ms. Considering also that the sampling frequency of our signals is 10 kHz, this translates to an intrinsic uncertainty for the label determination of at least ten timesteps. So, for example, one expert might determine, for a shot, the start of the H mode at the point in time where the PD signal starts dropping, whereas another expert might claim that the H mode actually only starts at the point where the signal has already stabilized (perhaps 1 ms later). While the difference may sound trivial, this can become problematic in a supervised learning task such as the one we face in this work. In principle, if the amount of training data is sufficiently

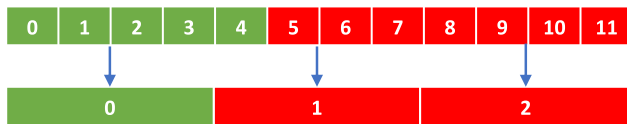


Figure 2. Representation of the computation of a block of target labels, where each bottom block corresponds to four source timesteps. The green color indicates a label of L (low confinement) while red represents H mode. A majority of the labels in the source timesteps 4–7 are in H mode, so the corresponding block at target timestep 1 is labeled as H.

large, small inconsistencies in labels (such as variations of 1 ms in the localization of transitions) will tend to be averaged out by a classifier. Nevertheless, these mismatches can produce instabilities during training and can ultimately degrade performance.

For that reason, one of the steps we took in this work is to reduce the temporal resolution of the sequence-to-sequence model’s outputs. This can be done thanks to the model’s architecture, which allows for a mismatch between the size of the input and output sequences. We do this by grouping the existing labels (i.e., sequences of shot classifications) in our dataset into *blocks* of a fixed size. In the pre-processing stage, each of those blocks is mapped to a certain plasma confinement mode (L, D or H); this mapping is done by computing the source label with the highest number of occurrences within that block (see figure 2). Our expectation is that this decrease in temporal resolution will yield better performance in both training and inference time, at a minimal cost to the physical validity of the results.

3.3. Model architecture

Our model’s architecture is based on existing architectures for NMT, in particular, the one proposed in [17]. The architecture consists of an encoder–decoder model with attention, which we detail in the following paragraphs.

Unlike most work with language translation, where one receives discrete units (words) as inputs, in our case, the inputs to the model are the continuous signal time-traces from TCV shots. In those time-traces, one expects to find not only localized, spatial correlations in the data—for example, a sudden drop in the PD signal—but also long-term contextual correlations, namely, which modes a shot may have been in in the past. For that reason, the encoder in our model consists of a convolutional RNN, much like the conv-LSTM used in [1]. We made slight adjustments, namely in the number of convolutions used, but otherwise preserved that model’s architecture, and used LSTM units for the recurrent layers. The inputs fed to the encoder are sequences of overlapping windows, which slide across the signal time-series. These windows were defined to have a size of 40 source timesteps, with a stride between windows of ten timesteps. This last detail, in particular, means that the full sequence-to-sequence model has approximately ten times fewer network parameters than the conv-LSTM in previous work.

The convolutional layers are trained to find local correlations in the windows, while the recurrent layers, based on the

output of the convolutions, keep track of long-term correlations in the input sequences. For example, the convolutional layers can detect a local shape in a signal window indicating a possible L–H transition, and the recurrent layers then use that, as well as their stored information about the current plasma mode (i.e. the long-term dependency), to determine whether a transition indeed has occurred.

In [1], the conv-LSTM was used to directly map the input sequence of measurements, x_t , into a sequence of outputs, z_t , indicating a plasma confinement mode at a particular point in time [see equation (3.1)]. In this work, the task of this particular submodel is instead to produce an encoded summary of x_t , which is stored in its internal hidden state vector h . During both training and inference time, we feed the encoder not with entire shots, but rather, subsequences of signals drawn from the shots. There are several reasons for this. On one hand, in RNNs, gradients can vanish when being backpropagated through time, which can be particularly problematic when working with long sequences; training with smaller subsequences mitigates this problem. On the other hand, the existing data is imbalanced with regard to the labels; for example, dithers tend to be much less frequent than L and H modes. Using subsequences allows us to feed the entire model, during the training process, with a more balanced number of samples for each class (by oversampling the dithers), thus preventing a potential source of biased results. Finally, any future usage of the methods proposed in this paper for RT plasma data analysis implies, by definition, that only information until a particular point in time, and not the entire shot, is available for classification.

In training time, the subsequences are drawn uniformly (with respect to the three classes) from our existing data ensemble. In inference time, for any given shot, the subsequences are drawn and fed to the encoder consecutively. In both cases, the encoder’s state is reset each time a new subsequence is fed to it, which means that the context vectors only hold information about the current subsequence under consideration.

In practice, we defined the subsequences drawn for training and inference to have a size of 300 source timesteps (this contrasts with a typical shot size of, at least, 10 000 timesteps at a 10 kHz sampling rate). This value also corresponds to the delay that the model would have in a RT setting. With a window size of 40 and a window stride of 10, these 300 timesteps are fed to the convolutional layers as sequences of 27 convolutional windows (further shrinking the total size of the sequence fed to the LSTM), with window 1 observing x_t from $t = 0$ to $t = 39$, and window 27 observing x_t from $t = 260$ to $t = 299$. The conv-LSTM processes these windows to produce the hidden state vector h , which has a length of 27 elements and is fed to the decoder. An illustration of this can be found in figure 3.

The task of the decoder is to approximate a probability distribution $p(z|x)$ of plasma confinement modes, subject to the summary given to it by the encoder. Our decoder is composed of a RNN (specifically, an LSTM layer), an attention layer, and a series of dense layers.

Each individual element of the output sequence is processed in a single timestep, with each target timestep corresponding

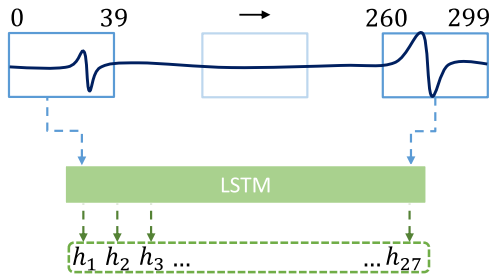


Figure 3. Illustration of the encoder architecture and how a single subsequence is processed and encoded in the encoder hidden state vector h . The dark blue line represents a subsequence of a signal timeseries. The sliding windows are in light blue.

to a single block of classifications. We defined the blocks as having a size of 10—that is, each target timestep yields a single classification for ten source timesteps.

In the first decoding timestep of each new subsequence, the decoder’s initial hidden state is set to the decoder’s final hidden state, h_k , for that subsequence. At each decoding timestep, the decoder receives as input its own last processed output (plasma confinement mode z_{t-1}), and the last output from its own LSTM cell; these are concatenated as suggested in [17] and fed to the decoder’s LSTM, which also updates its own internal state. The output of the LSTM is then concatenated to the output of the attention layer (which receives the entire encoder hidden state vector h), and fed through a series of dense layers to produce the final decoder output, which is a vector whose entries add up to 1 and which, individually, represent the computed probability of a given plasma confinement mode z_t (see figure 4). The decoder’s LSTM is built with a latent dimensionality of 32 units, that is, we process each target timestep with 32 LSTM cells. In terms of the attention layer, in [17], several different mechanisms for computing the alignment scores are proposed; we opted for the general form described in that paper.

One of our considerations when designing the decoder was to take into account how a human expert would label the signals. Our intuition was that, when looking at a time point of a shot, a labeler always takes into consideration information around that point—namely, the events happening immediately before and after. For that reason, we designed the decoder such that, for each incoming context vector (which encodes information regarding 300 source timesteps), the decoder produces a sequence of 18 blocks of labels (with a block size of 10, this corresponds to 180 source timesteps), which are the classifications for source subsequence steps [60 : 240]. The idea is that the extra information present in the remaining source timesteps would help improve the classification results. In evaluation time, this setup requires that the consecutive subsequences drawn from a shot overlap with each other, so that an output sequence can be produced for the entire shot. This also leads to the loss of the initial and final six target classification blocks of a shot (see figure 5), but we consider this to have no bearing on our results.

In evaluation time, the decoder always produces a distribution of possible plasma confinement states whose probabilities add up to 1. One possibility would be to use a greedy approach

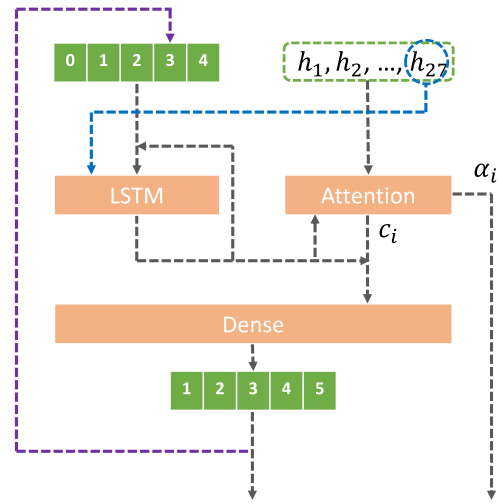


Figure 4. Schematic representation of the decoder’s architecture. Represented here is the sequence of operations carried out by the decoder to produce the output distribution of plasma confinement modes at decoding timestep $t = 2$. Gray arrows denote data flows at t . The purple arrow denotes the autoregressive feeding of the output at $t + 1$. The blue arrow denotes the initial setting of the decoder state to the last encoder state. Joining arrows denote concatenation, α_i and c_i are the attention weights and the context vector, respectively.

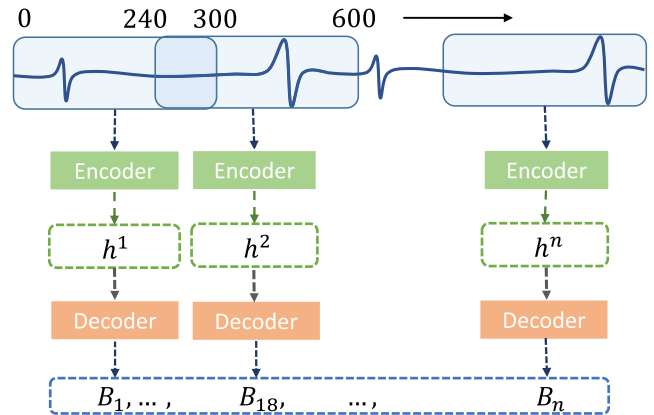


Figure 5. Representation of the full sequence-to-sequence model’s architecture. Notice how the subsequences (translucid blue) overlap with each other. The encoder’s state vectors, h^k (where k is the current subsequence), are fed to the decoder, which produces blocks of outputs.

and simply take, at each target timestep, the plasma state for which the output probability is highest, and feed that state at the next decoding timestep. This would yield a possible solution (i.e., a single sequence of plasma confinement states), but there would be no guarantee of it being optimal. For that reason, we use a beam search algorithm [22] to traverse the tree structure of possible solutions (different sequences of z), which allows for obtaining samples closer to the optimal \hat{z} . This is done by, at each target timestep, expanding the search tree for all previous outputs, and not just for the output of highest probability. Then, for each previous output z_{t-1} under consideration, the conditional probability, and the log-likelihood, defined in equation (3.1) are computed for the current timestep

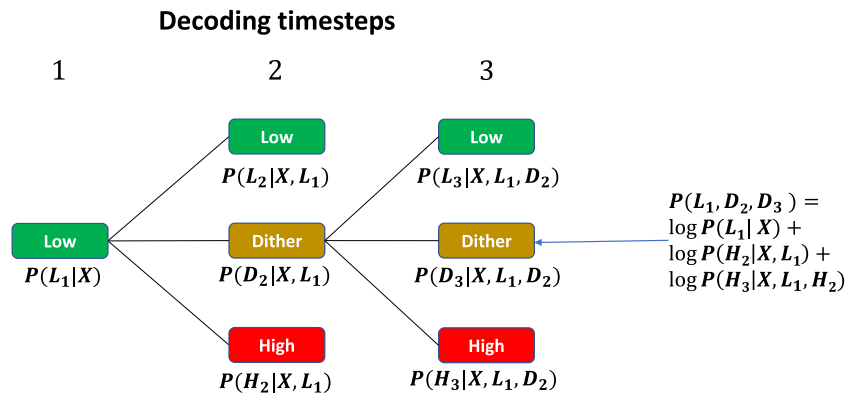


Figure 6. Illustration of the beam search algorithm in the first three target timesteps. X denotes the context vector and the final encoder state which condition the output. In timestep 1, the decoder computes the probability of a path starting in L mode, and we manually set that to be the only path to be expanded. In timestep 2, the conditional probabilities of three paths ($\{L_1L_2\}$, $\{L_1D_2\}$, $\{L_1H_2\}$) are computed; the paths are then expanded (for simplicity, only one expansion, that of $\{L_1D_2\}$, is shown here, but nine paths would be evaluated in timestep 3).

Table 1. κ -statistic scores for each plasma mode and as a mean, on training and test data, for the conv-LSTM model from [1] on the data used for this work.

		L	D	H	Mean
κ scores	Train	0.98	0.91	0.98	0.98
	Test	0.92	0.78	0.91	0.9

Table 2. κ -statistic scores for each plasma mode and as a mean, on training and test data, for the sequence-to-sequence model.

		L	D	H	Mean
κ scores	Train	0.99	0.99	0.99	0.99
	Test	0.94	0.86	0.96	0.94

t . Once all target timesteps have been processed, the value of z for which the likelihood is highest is returned.

Naturally, expanding the full tree at every timestep would quickly become unwieldy, owing to the large number of different paths to be processed. On the other hand, expanding a single beam at each timestep would be equivalent to a greedy search. For these reasons, we defined the beam search to have a maximum width of 20, i.e., at every step, only the 20 paths with the highest log-probability are expanded, while the rest are discarded. In addition, we encoded in the beam search a rule for expanding only those paths that start in L-mode, which simplifies the search; an illustration of this can be seen in figure 6.

3.4. Dataset preparation

For the work in [1], a total of 54 shots were used for training and testing the proposed models. In this work, we carried out a more careful treatment of the dataset preparation with respect to the previous publication. In the first place, the selection of the discharges for training and testing was done in order to cover as exhaustively as possible the space of the plasma confinement modes in TCV, accounting for the different temporal evolutions of the plasma. Using a dynamic time warping (DTW) [23] algorithm, we measured the similarity

between pairs of temporal sequences and assigned them to a given group, based on a similarity measure. The desired number of groups was obtained by applying a hierarchical clustering algorithm to univariate time sequences corresponding to the entire plasma discharges. A total of 293 discharges were selected and processed through the DTW, setting the number of clusters to 100. From each of the 100 clusters, shots were extracted and classified as an interesting (or not) shot from the physics point of view, as far as our problem (i.e., the presence of L/D/H transitions) is concerned. Some clusters were discarded since they consisted of disruptions without even achieving an H mode confinement state, while others presented technical issues in the ramp-up phase before reaching the stationary phase. Limiting ourselves to the interesting shots and maximizing the number of clusters where a given shot arose from, 88 discharges were selected for ground truth determination. For the latter, instead of having different annotations by different labelers for a particular shot, a consensus on a common convention between two experts was established to determine the label of each timestep for all shots. A detailed revision of the different transitions was performed with particular attention to the presence of short transitions on the order of 2 ms. The outcome was a unique, consistent, ground truth per shot. A test set to evaluate the final results of the model was carefully determined and fixed during all the experiments. A total of 27 shots were selected, each extracted from a different cluster. Out of 27 shots, 17 shots were ‘unpolluted’ cases (without the presence of type III ELMs), while the others 10 were special ‘noisy’ discharges with type III ELMs. The proportion between the noisy and ‘unpolluted’ discharges in the test set followed approximately the same proportion as in the complete dataset.

4. Results

We begin this section with a direct comparison between this sequence-to-sequence model, and the conv-LSTM used in [1]. To that end, we trained and tested the old model, preserving all the original architecture and hyperparameters, with the new train and test data (shots and labels) compiled for this

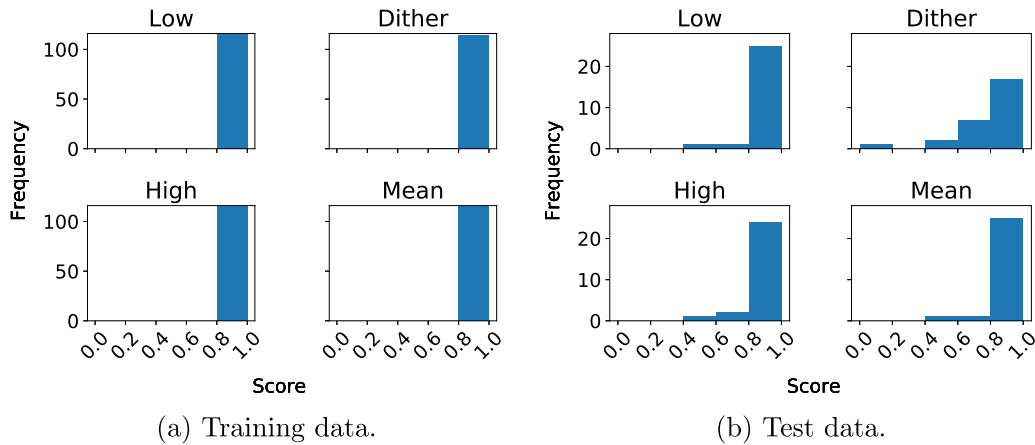


Figure 7. Distribution of the κ -statistic score on a per-shot basis.

work. Table 1 shows the results. As in the previous work, we performed the evaluation using Cohen’s Kappa-statistic coefficient [24], which gives an indication of the match between two sets of categorical data (with a score of 1 for a perfect match and 0 for no match). In our case, it reflects the match between the models’ outputs, and the labeled data. We computed the score on a per-class basis and also on a weighted mean basis, in order to indicate whether the classifications produced by the sequence-to-sequence model match the data’s labels. We designed the model with Tensorflow [25], and ran it on an NVIDIA Quadro RTX 5000 graphics processing unit (GPU).

A comparison between the results in table 1 and those described in [1] shows that the current dataset already improved the capacity of the old model. Nevertheless, it was still underperforming, particularly on dithers; even on training data, the mean dither score was 0.9.

We then ran the new sequence-to-sequence model. The results on both the train and test sets can be seen in table 2. They were obtained by training the network for 150 epochs, with each epoch consisting of 128 batches of data, and each batch consisting of 128 data samples, i.e., uniformly sampled subsequences of each of the existing classes drawn from the training shots. We downsampled the source labels to the same temporal resolution as the model’s output blocks to compute the scores. Figures 7(a) and (b) show the distribution of the scores on a per-class basis, for train and test data, as well as a weighted mean value, taking into account the relative frequencies of each class in the labels.

5. Discussion

In this section, we discuss the results in further detail, in particular, the cases where the sequence-to-sequence model’s performance was poorer.

With regards to the training data, the classification was excellent in all cases; all shots achieved scores, both on a per-mode basis and as a mean, above 0.8. The lowest mean train score for a shot was 0.965, while the lowest scores for

Table 3. Kappa statistic scores for shots with at least one mode whose score was lower than 0.8. The ‘fraction’ column indicates what percentage of the labels were labeled in a particular state for that shot.

Shot ID	L		D		H		Mean
	Fraction	Score	Fraction	Score	Fraction	Score	
42197	0.57	0.5	0.35	0.66	0.08	0.46	0.55
61057	0.5	0.72	0.04	0.68	0.46	0.73	0.72
61274	0.69	0.84	0.12	0.7	0.19	0.96	0.84
32911	0.52	0.84	0.28	0.79	0.20	0.97	0.85
61043	0.78	0.92	0.13	0.69	0.1	0.76	0.87
45105	0.42	0.91	0.02	0.52	0.56	0.94	0.92
34309	0.14	1	.03	0.8	0.83	0.96	0.96
33459	0.8	0.98	0.01	0.5	0.19	1	0.98
64376	0.92	0.9	.01	0.73	0.08	1	0.98
30268	0.24	1	0	0	0.76	1	1

L, D and H modes were, respectively, 0.96, 0.8 and 0.98. Nevertheless, for all modes, the mean score was 0.99, a result that indicates that the sequence-to-sequence model has the capacity to learn the underlying correlations in the data to make accurate predictions.

With regard to the test data, the scores are slightly lower. Table 3 shows the detailed breakdown of the scores for all shots where at least one mode had a score lower than 0.8. Some of those shots have low scores on more than one class: for example, #42197 had a score lower than 0.8 for both L and D modes. Notice how, even though there are more shots with lower dither scores, the overall mean values are in most cases above 0.8; this is due to the fact that dithers are rather less frequent in the data than L and H modes.

Figure 8 shows the classification results for shot #42197, together with the ground truth (label), shown in the lower panel of the figure (we show only the PD signal values for ease of comprehension). The classification has two major errors: the non-detection of a switch to L mode at the end of the shot, around $t = 1.2$ s, and the rapid oscillation between L and D modes from approximately $t = 0.8$ s to $t = 1.1$ s. This oscillation in particular is questionable because this dithering phase

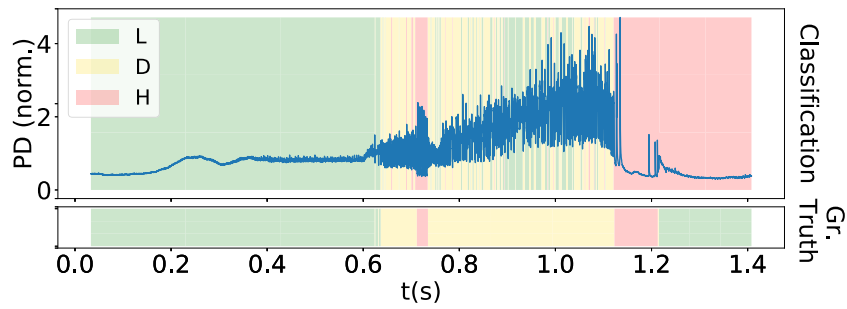


Figure 8. Detection results for shot #42197. The blue line denotes the PD signal, while the solid background color denotes the confinement states.

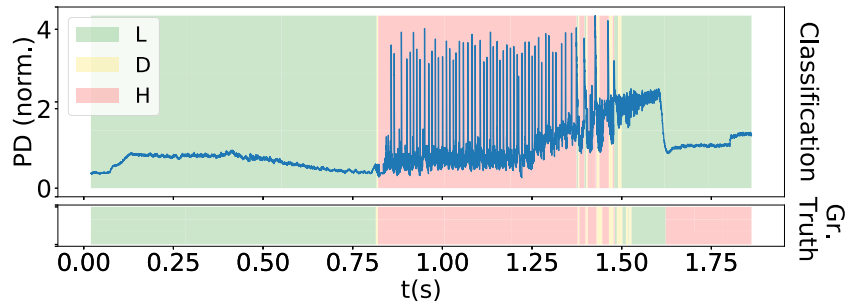


Figure 9. Detection results for shot #61057.

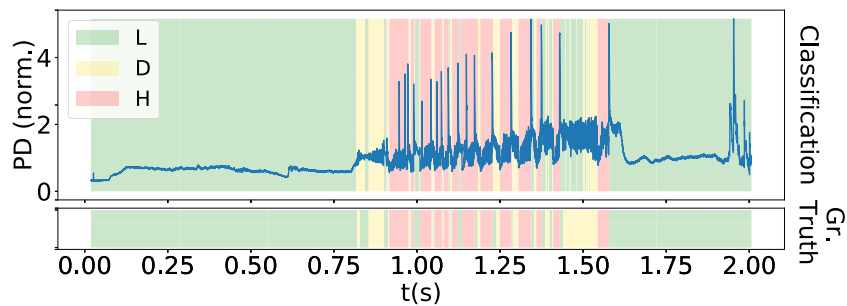


Figure 10. Detection results for shot #61274.

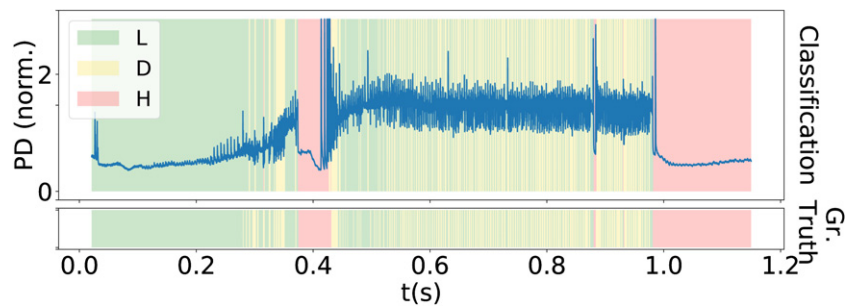


Figure 11. Detection results for shot #32911.

presented particularly odd fluctuations that were not a common behavior in our dataset.

For shot #61057, two short dither bursts near $t = 1.5$ s are missed, as well as the final transition back into H mode near $t = 1.6$ s (see figure 9).

For shot #61274, the overall classification score is already above 0.8; the model's largest mistake is in incorrectly classifying a region around $t = 1.5$ s as L mode, which explains both the lower score for that mode and for dither (see figure 10).

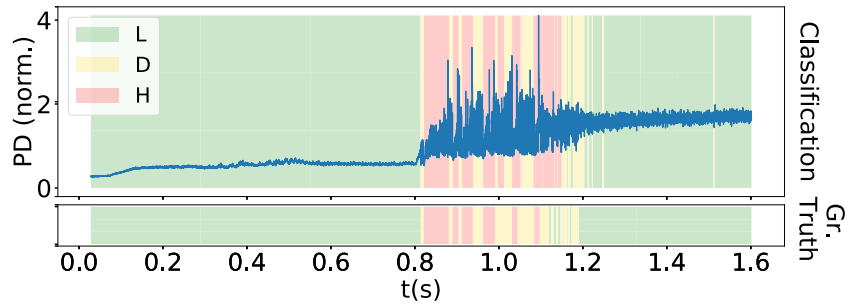


Figure 12. Detection results for shot #61043.

In shot #32911, the lower scores for dither and L mode are also due to rapid fluctuations between the two modes; we think that a big factor behind the lower score is the accumulation of many small mismatches between the labels and the classifications (see figure 11).

Shot #61043 contains several switches between H mode and dithering; the model incorrectly classifies some of these (see figure 12).

We consider the remaining test shots to be good classifications overall. We explain the occasionally lower dither scores by the fact that the labeled dither phases are very short (at most 3% of any given shot); any mismatch, even if small, between the label and dither classification produces a low score.

6. Conclusions

This work developed a sequence-to-sequence neural network model with attention for automated classification of plasma confinement modes at the TCV tokamak. Taking previous work [1] with a conv-LSTM for the same task as a baseline, our intuition was that one of the factors holding previous models back from achieving even better scores was the fact that vanilla LSTMs cannot reason over different past outputs, and are limited to the signal observations. This contrasts with the way a human labeler would perform the same task; typically, a labeler will reason over different possibilities for past confinement modes before deciding on a label for a given timestep, which is a process that a sequence-to-sequence model can emulate. In addition, we theorized that the varying length of transitions between plasma confinement modes, and resulting misalignment between data and labels, was producing instabilities during the training of the conv-LSTM. Finally, we also hypothesized that performance was limited by the lack of sufficient train data. Therefore, we extended the dataset used in previous work with more shots and chose a train/test split that thoroughly represented the operational shot space of TCV. In addition, we took particular care with the labeling process, to ensure a high quality of the data.

Testing the conv-LSTM from [1] on the new dataset increased its scores on both test and train data, suggesting that indeed the size and quality of the dataset was having an impact on the obtained results. Nevertheless, even with the new dataset, the conv-LSTM still failed to completely fit the training set, particularly with regards to dithers.

On the other hand, running the sequence-to-sequence model with the new data, we achieved excellent results on the train set (with a mean score of 0.99 out of 1) and likewise, very good results on the test set (with a mean score of 0.94 out of 1). Both of these scores are important. On the one hand, the training score shows that the sequence-to-sequence model can correctly fit all the training data. This is something that the previous model (the conv-LSTM) could not do, and suggests that that model, in spite of achieving good results, had indeed some limitations compared to the new one. This is particularly interesting given that the sequence-to-sequence model has a lower number of network parameters: approximately 100 000, compared to the conv-LSTM's 1 000 000 (i.e., an order of magnitude less). We explain the fact that the results are nevertheless better with three factors, which coincide with our initial assumptions: firstly, the larger context from the data available to the sequence-to-sequence model; secondly, the model's ability to deal with misaligned labels and with transitions of varying length; and finally, the autoregressive decoder which can reason over several sequences instead of independent outputs.

On the other hand, the test scores indicate that the sequence-to-sequence model can also generalize well. They are slightly lower, but we explain that with the fact that several test shots were particularly challenging to classify, even for a human labeler. Nevertheless, the results obtained indicate that the task of plasma confinement mode classification can be best carried out by the sequence-to-sequence model presented here, given that it achieves a significant improvement over the conv-LSTM, reaching almost domain expert accuracy.

Finally, we would like to make a comment regarding a RT implementation of this model, which is outside the scope of this publication, but should be possible. As we mentioned in section 3, the only fixed constraint, as far as an RT implementation would be concerned, is the size of the subsequences fed to the encoder, which induce a small minimum delay (30 ms) into any model prediction. Ideally, an RT implementation would make extensive use of parallelization, in particular in the beam search; that step could otherwise become a potential performance bottleneck in a non-parallel implementation, given that the time of the search scales with the number of beams. Also, an RT implementation would ideally require multiple central processing units (CPUs) or GPUs, such as what we used for developing the method, since deep learning frameworks can efficiently run convolutional layers in parallel. Nevertheless,

the model shown here, as it stands, can already be used for offline labeling of data. In addition, in light of the current research, within the field of neural language processing, on transfer learning with encoder–decoder networks, we believe that this model could also be suited for tasks of domain adaptation, for example with a view to creating useful databases for future fusion devices such as ITER and DEMO. It would also be interesting to investigate, in the future, applications of transformer networks to this and similar problems in fusion.

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under Grant Agreement No. 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

ORCID iDs

F. Matos  <https://orcid.org/0000-0002-3110-6639>
 A. Pau  <https://orcid.org/0000-0002-7122-3346>
 G. Marceca  <https://orcid.org/0000-0002-8850-614X>

References

- [1] Matos F., Menkovski V., Felici F., Pau A. and Jenko F. (the TCv Team and the EUROfusion MST1 Team) 2020 *Nucl. Fusion* **60** 036022
- [2] Xu G.S. et al 2014 *Nucl. Fusion* **54** 103002
- [3] Nielsen A.H., Xu G.S., Madsen J., Naulin V., Rasmussen J.J. and Wan B.N. 2015 *Phys. Lett. A* **379** 3097–101
- [4] Martin Y.R. and Takizuka T. (the ITPA CDBM H-mode Threshold Data Group) 2008 *J. Phys.: Conf. Ser.* **123** 012033
- [5] Ryter F. et al 1994 *Plasma Phys. Control. Fusion* **36** A99
- [6] Vega J., Murari A., Vagliasindi G. and Rattá G.A. (JET-EFDA Contributors) 2009 *Nucl. Fusion* **49** 085023
- [7] González S., Vega J., Murari A., Pereira A., Dormido-Canto S. and Ramírez J.M. (JET-EFDA Contributors) 2012 *Plasma Phys. Control. Fusion* **54** 065009
- [8] Murari A., Vagliasindi G., Zedda M.K., Felton R., Sammon C., Fortuna L. and Arena P. 2006 *IEEE Trans. Plasma Sci.* **34** 1013–20
- [9] Lukianitsa A.A., Zhdanov F.M. and Zaitsev F.S. 2008 *Plasma Phys. Control. Fusion* **50** 065013
- [10] Meakins A.J. and McDonald D.C. 2010 *Plasma Phys. Control. Fusion* **52** 075005
- [11] Krizhevsky A., Sutskever I. and Hinton G.E. 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* (Lake Tahoe, Nevada, US, December 2012) pp 1097–105 (<https://nips.cc/Conferences/2012>)
- [12] Ciresan D., Meier U. and Schmidhuber J. 2012 Multicolumn deep neural networks for image classification *2012 IEEE Conf. on Computer Vision and Pattern Recognition (IEEE)* (Providence, RI, USA, June 2012) pp 3642–9
- [13] Graves A., Mohamed A.R. and Hinton G. 2013 Speech recognition with deep recurrent neural networks *2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (IEEE)* (Vancouver, BC, Canada, May 2013) pp 6645–9
- [14] Elman J.L. 1990 *Cogn. Sci.* **14** 179–211
- [15] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L.U. and Polosukhin I. 2017 Attention is all you need *Advances in Neural Information Processing Systems* (Long Beach, CA, USA, December 2017) vol 30 ed I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (Curran Associates, Inc.) pp 5998–6008 (<https://nips.cc/Conferences/2017>)
- [16] Bahdanau D., Cho K. and Bengio Y. 2015 Neural machine translation by jointly learning to align and translate *Int. Conf. on Learning Representations* (San Diego, USA, May 2015) (<https://iclr.cc/archive/www/2015.html>)
- [17] Luong M.T., Pham H. and Manning C.D. 2015 Effective approaches to attention-based neural machine translation *Conf. on Empirical Methods in Natural Language Processing* (Lisbon, Portugal, September 2015) (<http://emnlp2015.org/>)
- [18] Xu K., Ba J., Kiros R., Cho K., Courville A., Salakhudinov R., Zemel R. and Bengio Y. 2015 Show, attend and tell: neural image caption generation with visual attention *Int. Conf. on Machine Learning* (Lille, France, July 2015) pp 2048–57 (<https://icml.cc/Conferences/2015/>)
- [19] Sutskever I., Vinyals O. and Le Q.V. 2014 Sequence to sequence learning with neural networks *Advances in Neural Information Processing Systems* (Montreal, Canada, December 2014) pp 3104–12 (<https://nips.cc/Conferences/2014>)
- [20] Stahlberg F. 2020 *J. Artif. Intell. Res.* **69** 343–418
- [21] Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H. and Bengio Y. 2014 Learning phrase representations using rnn encoder–decoder for statistical machine translation *The 2014 Conf. on Empirical Methods in Natural Language Processing* (Doha, Qatar, October 2014) (<https://emnlp2014.org/>)
- [22] Graves A. 2012 Sequence transduction with recurrent neural networks *Int. Conf. on Machine Learning* (Edinburgh, Scotland, June 2012) (<https://icml.cc/2012/>)
- [23] Łuczak M. 2016 *Expert Syst. Appl.* **62** 116–30
- [24] Landis J.R. and Koch G.G. 1977 *Biometrics* **33** 159–74
- [25] Abadi M. et al 2015 TensorFlow: large-scale machine learning on heterogeneous systems software available from tensorflow.org (<https://www.tensorflow.org/>)