Deep neural networks for prediction of solar flares

DD Krynauw

orcid.org/0000-0001-6077-5906

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor:	Prof Marelie H Davel
Co-Supervisor:	Dr Stefan Lotz

Graduation:	June 2021
Student number:	26013835

Declaration

I, Dewald Daniël Krynauw, hereby declare that the dissertation entitled "Deep neural networks for prediction of solar flares" is my own original work and has not already been submitted to any other university or institution for examination.

D.D. Krynauw

Student number: 26013835

Signed on the 19th day of April 2021 at Potchefstroom.

Acknowledgements

This research was performed within the Multilingual Speech Technologies (MuST) research group of the North-West University, which is a member of the Centre for Artificial Intelligence Research (CAIR) of the Department of Science and Innovation. It was supervised by Professor Marelie Davel and Doctor Stefan Lotz in association with the South African National Space Agency (SANSA). Working alongside world-class supervisors and distinguished researchers has made me humble, thankful and even more curious.

I would like to thank:

- Ulrike Janke, who took care of all the administrative workload and made sure I could do my best work
- The MuST group and the use of the *Teapot* server, without which none of the work would be possible
- My supervisors Prof. Marelie and Dr. Stefan, whose insight and knowledge into the subject matter steered me through this research. Your unending patience and goodwill are greatly treasured.
- Liu et al. who willingly provided their research and data, which we critically reviewed.

Lastly, I would like thank my friends and family for their support throughout this undertaking, and a special thanks to Jacques for being a brother-in-arms.

Genesis 1:3 (NKJV): Then God said, "Let there be light"; and there was light.

Abstract

Solar flares are enormous explosions on the solar surface that originates from sunspots, which could cause damage to satellites, power grids and radio communication systems. Having early-warning systems that could accurately predict the eruption of harmful flares are becoming more crucial, as our society grows more dependant on technology. Deep learning has brought about a new era of flare prediction tools and models, with astounding success compared to traditional techniques, although at a deficit of interpretability.

In this study, we present findings and study the effects of several deep neural network architectures, optimised with different optimisation techniques and scaling strategies. We aim to create simplified models with enough capacity to accurately predict flares, using image-derived sunspot features. We also investigate the predictive ability of these features using deep learning attribution methods.

We find that the task can be modelled such that the basic multilayer perceptron (MLP), with a small architecture, could accurately predict solar flares on par with its larger MLP counterparts. We find that including the temporal information of past observations does not increase performance, but smoothed the probabilistic forecasts in the case of our proposed one-dimensional causal convolutional (1D-CNN) network, compared to the MLP network. We also find that the attribution methods tested were able to extract the most relevant features relating to the eruption of flares, which is consistent with other findings in the literature. The study contributes towards open questions regarding imbalanced classification and probabilistic forecasting using deep neural networks, specifically giving an informed perspective on binary classification architectures, optimisation techniques, scaling strategies and attribution methods. In the process, we provide some additional insight into a well-known dataset, and develop a deployable solar flare prediction model.

Keywords: *Deep Neural Network, Solar Flare Prediction, Interpretability*

Contents

List of Figures

List of Tables

List of Acronyms

1D-CNN one-dimensional convolutional neural network

AI artificial intelligence

AIA atmospheric imaging assembly

ANOVA analysis of variance

AR active region

AUC area under curve

BACC balanced accuracy

BSS Brier skill score

CEA cylindrical equal area

CCD charge-coupled detector

CME coronal mass ejection

CNN convolutional neural network

DNN deep neural network

EM electromagnetic

EVE extreme ultraviolet (EUV) variability experiment

EUV extreme ultraviolet

GOES Geostationary Operational Environmental Satellite

HARP helioseismic and magnetic imager (HMI) active region patch

HMI helioseismic and magnetic imager

HP hyperparameter

HSS Heidke skill score

JSOC Joint Science Operations Center

LSTM long short-term memory

LR learning rate

MDI Michelson doppler imager

MI mutual information

MLP multilayer perceptron

NCEI National Centers for Environmental Information

NOAA National Oceanic and Atmospheric Administration

OCLR one-cycle learning rate

ReLU rectified linear unit

RF random forest

ROC receiver operating characteristic

SDO Solar Dynamics Observatory

SGD stochastic gradient descent

SHARP spaceweather HMI active region patch

SOHO Solar and Heliospheric Observatory

SSP skill score profile

SVM support vector machine

TCN temporal convolutional network

TSS true skill statistic

Chapter 1

Introduction

In this chapter we introduce the study and discuss why it is relevant and interesting. We discuss what forms part of the scope, as well as the research questions and objectives.

1.1 Background

The Sun has been praised for its radiance throughout the ages, and not without good reason; it provides light, life and warmth for all mankind. As quiet as the Sun may seem, it is actively blasting electromagnetic (EM) energy everywhere. Solar flares are gigantic explosions on the solar surface that originate from sunspots, also known as active regions (ARs). These flares produce a wide spectrum of photons releasing X-rays, visible light, ultra-violet light and energised protons. The largest solar storms come from coronal mass ejections (CMEs). CMEs launch massive plumes of plasma, containing billions of tons of solar particles, which have a magnetic field binding them. They usually take tens of hours to propagate the 1.495×10^8 km (= 1 astronomical unit = 1 AU) to reach Earth [1]. Just like hurricane forecasting, there are instruments actively trying to classify the severity of solar storms. Space weather forecasting tries to predict these storms by

observing the Sun for solar flares and CMEs.

Fortunately, the Earth's magnetic field helps to protect humans from some of the effects of solar storms, by deflecting the charged particles of the solar wind plasma around the Earth. However, strong solar storms can influence electrical systems such as power grids, satellites and radio communication systems. As our reliance on these technologies grows, so does our need to have accurate forecasting models [1].

The increased availability and cadence of quality solar data and the rise of computational power in the last decade have reignited the field of solar flare prediction using machine learning and deep learning techniques, with great success compared to traditional methods [2]–[10].

Deep learning is a subset of machine learning that leverages multiple layers and non-linear functional units to achieve outstanding performance in fields such as speech recognition, computer vision and natural language processing. Deep learning has excellent generalisation ability, which allows models to perform well on unseen data, but is still not well understood [11]. Deep learning is also inherently difficult to interpret and explain, and is still an active area of research [12].

1.2 Problem statement

Solar flares are eruptive events on the Sun that can have detrimental effects on various technological systems that are essential to modern society, such as power grids, satellites and various radio communication systems [13]. Accurate forecasts of solar flare occurrence and intensity is an unsolved problem, and being able to predict these events is becoming crucial for our infrastructure. Our goal is to create a deep neural network (DNN) that can predict the occurrence of a flare ahead of time, by using image-derived features of the solar disk and ARs. The secondary aim is to keep the model as simple as possible, not only to keep the model practically robust but also to optimise it to be interpretable, for

us to learn something about the underlying physics of flare formation.

1.3 Project scope

Given the problem statement above, we restrict the scope of the research to a limited number of datasets and architecture types:

- **Datasets:** The spaceweather HMI active region patch (SHARP) dataset provides vector and line-of-sight magnetogram images of sunspots, with metadata containing parameters explaining the physics of each AR. The Geostationary Operational Environmental Satellite (GOES) X-ray dataset supplies the class labels for the flare types. The aforementioned datasets are publicly available, as well as the processed dataset used by Liu et al. [4].
- **Architecture types:** The focus will be on starting with a simple architecture, namely multilayer perceptrons (MLPs) to establish a baseline, and then move on to a temporal convolutional network (TCN) variant.

1.4 Research questions

To investigate the forecasting capabilities of a DNN in terms of predicting solar flares, the following research questions are formulated:

- What is the simplest neural network architecture that can provide a solid baseline for solar flare prediction?
- What impact does additional architecture and optimisation extensions have on the network's prediction ability?
- From the trained networks, what can we learn about the importance of different features during the time period leading up to a flare?

1.5 Objectives of the study

The objective of this study is to investigate the feasibility of solar flare prediction with modern deep learning methods, while maintaining reproducibility and interpretability of the developed models. Particularly, the aims are to:

- Develop a baseline model that can perform to an acceptable degree of capability.
- Analyse the effect that more sophisticated methods have on performance, compared to the baseline model.
- Utilise the model training parameters to aid in the interpretation of the physical mechanisms responsible for driving flare activity.

1.6 Research methodology

This study consists of empirical research based on prior work on solar flare prediction using DNNs. It will explore the importance of certain physics-based features, derived from solar images. The following will form part of the study:

- **Literature review:** Gain a steadfast understanding of DNNs, focusing on models regarding time-series prediction. Comprehend the formation of solar flares. Investigate recent research regarding the prediction of solar flares using DNNs, as well as previous prediction methods.
- **Codebase development:** Develop a flare prediction codebase. The codebase will consist of methods to configure, train, evaluate and analyse the DNNs' performance in terms of flare prediction.
- **Experimental development:** Using the flare prediction codebase, DNNs will be trained to:
  - Determine the most essential features and parameters for flare production.

  - Identify specific hyperparameters (HPs) and optimise them for each architecture.
  - Analyse the network's predictive ability by comparing several importance ranking techniques on manually designed features, and investigate whether deep learning models can give us additional information about the physics of flare development.

- **Assess and discuss the experimental findings:** After executing the set of experiments, the results will be assessed and discussed in detail.

1.7 Dissertation overview

This dissertation is structured as follows:

- In Chapter 2 we introduce the field of space weather and machine learning. We review existing literature on flare prediction.
- In Chapter 3 we discuss the dataset, along with its input and target features.
- In Chapter 4 we perform an analysis of the input features. The different scaling strategies are also analysed.
- In Chapter 5 the MLP is optimised for different architectures, scaling strategies and optimisation techniques, to establish a baseline model.
- In Chapter 6 our proposed TCN-variant architecture is trained and evaluated.
- In Chapter 7 we apply attribution methods to our best MLP and TCN-variant models to determine the predictive capability of individual input features.
- In Chapter 8 we elaborate on the practicalities of deploying models such as ours.
- In Chapter 9 we deliberate on the objectives, summarise the key findings and their implications, and we discuss future work.

Chapter 2

Background

In this chapter we give background information from relevant literature and introduce several concepts that are essential in this study. We investigate similar studies and identify areas where further empirical analysis is required.

2.1 Introduction

The problem we look to solve is to predict solar flares with interpretable DNNs. We introduce the domain, namely solar physics, with the focus on flares and space weather; the means of predicting, namely DNNs and other models previously used; how the quality of the models is measured with performance metrics; and how important certain features are for flare prediction using attribution methods.

In Section 2.2 we cover space/solar physics, and in Section 2.3 we introduce relevant deep learning concepts and strategies. In Section 2.4 we describe previous work utilising traditional and machine learning techniques for flare prediction. In Section 2.5 we explain how DNNs can be used to explain the importance of features related to flare production.

Lastly, in Section 2.6, we list all the typical metrics used to evaluate solar flare prediction models.

2.2 Space weather

Space weather is a catch-all term for the effect that EM and particle emissions from the Sun have on technology through its influence on the geomagnetic field, on the ground and in space. The most severe effects of space weather are driven by solar flares and CMEs that project bursts of radiation and plumes of magnetised gas outwards from the sun. Spacecraft, communications and power transmission systems may be compromised by these space weather phenomena. [14]. The direct effect of a solar flare is not harmful to humans, because of the Earth's magnetosphere protecting us from most of the radiation. However, as our dependence on technology grows, this could become more life-threatening soon.

Solar flares are enormous explosions of radiation from the solar surface and corona, emitting energy across the entire EM spectrum [13]. Flares can be small and harmless or vicious and catastrophic (see [15], where solar storms cause communication outages of air traffic and disaster relief efforts during an extreme hurricane season). Large flares are sometimes accompanied by CMEs, which cause the bulk of geomagnetic disturbances observed at earth.

The largest flare in the last 15 years was on September 6, 2017. Figure 2.1 shows the false colour, full-disk image of the Sun around the time of eruption taken by the atmospheric imaging assembly (AIA) instrument on board the Solar Dynamics Observatory (SDO) spacecraft. Small flares are common but larger flares, such as this, are rare.

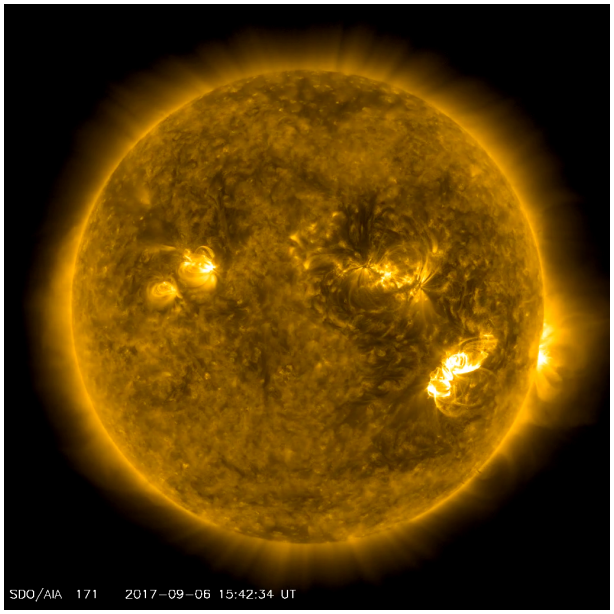


Figure 2.1: SDO/AIA full-disk image of class X9.3 flare. Taken 2017/09/06. Retrieved from [16].

Flares occur in ARs which are regions of strong complex magnetic fields that evolve rapidly over time [13]. These regions are colder than the surrounding solar surface, hence the distinctive spots. These ARs are assigned unique National Oceanic and Atmospheric Administration (NOAA) numbers to identify these regions as they appear on the earth-facing solar disk. An image of the photosphere or solar surface a day before the flare (from Figure 2.1) can be seen in Figure 2.2, with four distinct ARs traced in red, this time taken by the HMI instrument on board the SDO spacecraft.

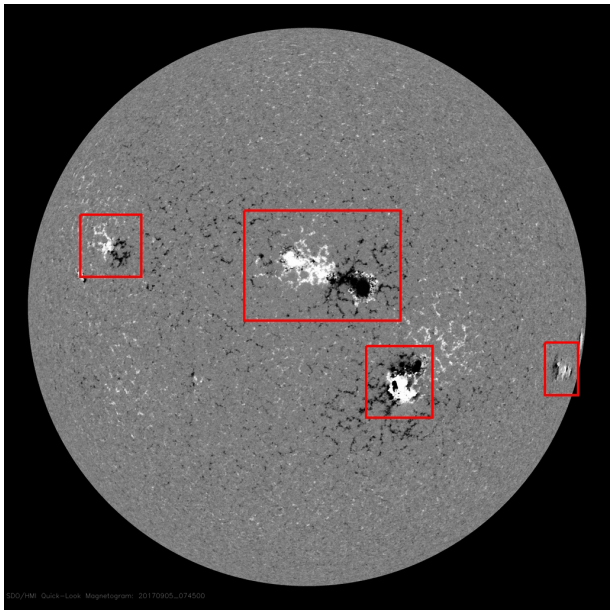


Figure 2.2: SDO/HMI magnetogram with ARs traced inside red squares. White regions are magnetic fields coming out. Black regions are magnetic fields going in. Gray is zero magnetic field. Taken 24 hours before flare eruption. Retrieved from [16].

Flares release energy much like earthquakes, slowly building up stress and quickly releasing it. Figure 2.3 shows this flare energy as the integrated X-ray emissions from the Sun, as measured by a series of satellites known as the GOES. The x-axis is time and the y-axis shows the Watts per square metre on a log scale. The two different colours are the two different wavelength bands ($0.5 - 4.0$Å (blue) and $1.0 - 8.0$Å (orange)). One can see that the Sun emits a baseline level of flux, but on September 6 at 12:00, increased the output in a few orders of magnitude; that was the flare. The abrupt vertical lines around 09:00 is missing data.

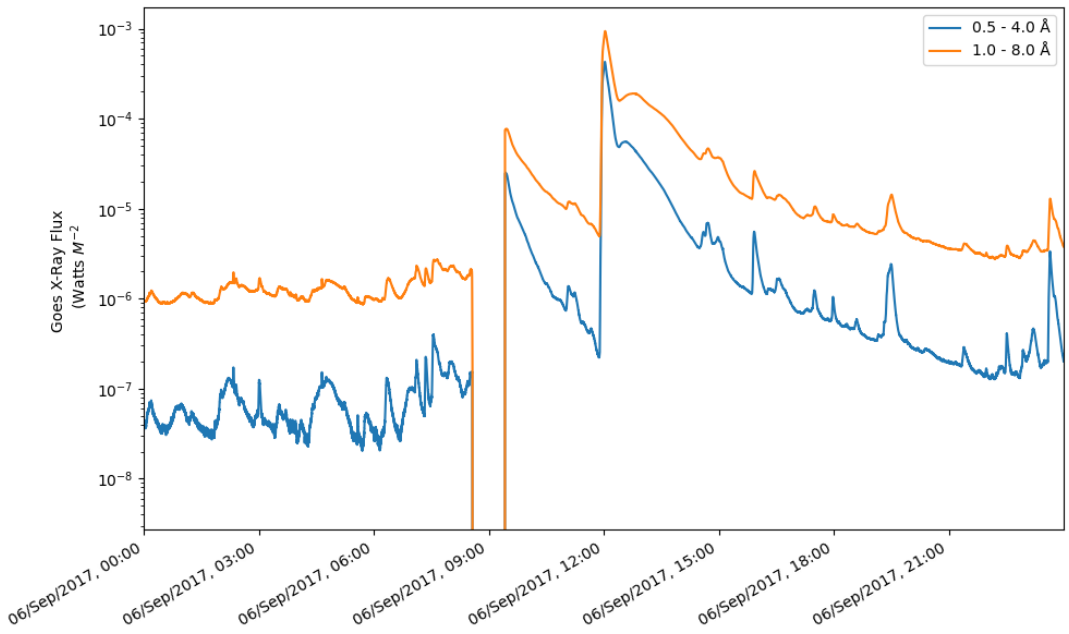


Figure 2.3: GOES X-ray flux on September 6, 2017. Retrieved with Sunpy [17].

Flares are classified logarithmically by peak flux, measured in the 1-8Å pass-band, into classes A, B, C, M, and X (see Table 2.1). Flares that release more than $5 \times 10^{-5} Wm^{-2}$ or ≥M5-class are considered large flares, and the Space Weather Prediction Center (SWPC) sends out alerts for these flares, since they are more likely to cause disturbances in satellite communication and radio transmissions [18].

Table 2.1: Solar flare classification by X-ray flux at geostationary orbit. The letter classes are further subdivided into decades.

Class	min flux $[Wm^{-2}]$	max flux $[Wm^{-2}]$
A	$<$	10^{-7}
B	10^{-7}	10^{-6}
C	10^{-6}	10^{-5}
M	10^{-5}	10^{-4}
X	10^{-4}	$>$

According to the standard model of flare eruption, a flare is triggered when plasma instabilities below the solar surface result in the rise of a section of plasma (known as a prominence) above the surface. As the section of very hot plasma rises, the magnetic

field lines 'frozen' into the plasma starts breaking apart and reconnects to lower sections. This results in a rapid release of radiation at the reconnecting point, resulting in the broadband emission of EM energy, and plasma flung out into interplanetary space. See [13] for a detailed description of flare dynamics.

2.2.1 Solar flare prediction

Solar flare prediction can be thought of as the same as predicting terrestrial weather such as analysing cloud patterns can be used to forecast rain. Similarly, magnetic field patterns on the Sun can be used to predict whether there will be a flare, today or tomorrow.

If we take a closer look at a single one of these ARs in Figure 2.4, there are some strong borders between the field lines going in and out of the Sun (indicated by orange arrows). Flares regularly originate from these regions [13].

If we isolate the pixels very closely, as in Figure 2.4, the magnetic field through the given area can be calculated, for example the magnetic flux for this specific AR at this specific time. This makes up one feature, and there are numerous studies on the different types of features that can be used to predict flares, see Section 2.4.

The current understanding of which features cause flares have to do with: (a) the size, (b) the complexity and (c) evolution of the flare [13].

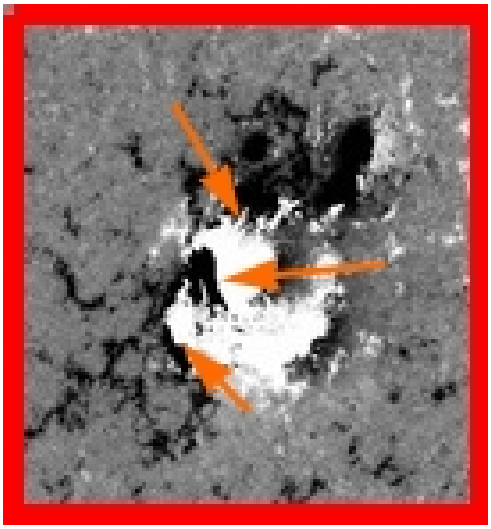


Figure 2.4: Soon-to-be-flaring AR from magnetogram in Figure 2.2

2.2.2 Solar satellites and instruments

We have come a long way as a species from first observing the Sun with a telescope in the 17th century, to having satellites taking stunning close-up images with the Parker solar probe. Through the years, many satellites have been launched to study the Sun and observe its inner and outer workings. In the paragraphs that follow, the most relevant of these satellites and the instruments they carry are explained, following [19].

Probably the most groundbreaking solar satellite was the Solar and Heliospheric Observatory (SOHO), launched in 1995. It was stationed at the Earth's first Lagrangian point (L1). At this point, the combined gravity of the Sun and Earth keeps the object locked near the Earth-Sun line, roughly 1.5 million kilometres from Earth. It was designed to study the Sun's internal structure and outer atmosphere, as well as the origin of the solar wind. It had 12 instruments to measure different kinds phenomena, of which the Michelson doppler imager (MDI) is the most popular for solar flare prediction, since it measures the longitudinal component of the Sun's magnetic field. These images that measure the strength of the solar magnetic field are called magnetograms. SOHO also aided in the discovery of coronal waves, solar tornadoes and our adroitness to forecast space weather. SOHO/MDI was decommissioned in 2013 [20].

The SDO [21] is the successor of SOHO, launched in 2010, purposefully designed to understand the impact of solar variability on Earth. It has three instruments: (1) AIA [21] takes full-disk images of the Sun in 10 different wavelengths every 12 seconds. (2) HMI [22] also takes magnetogram and continuum images of the solar surface like SOHO/MDI, but at a higher 4 k resolution with a charge-coupled detector (CCD) camera. (3) EUV variability experiment (EVE) measures the extreme ultraviolet spectral irradiance to understand differences in timescales that influence Earth's climate.

GOES is a series of satellites formally started in 1975 by NOAA, with GOES-17 being the latest to the series, launched in 2017. Each of the satellites differs greatly, but most of them had some form of Earth-facing, Sun-facing and space environment sensors. The sun-facing instruments include the X-Ray Sensor (XRS) and the EUV Sensor. The XRS consists of an ion chamber that measures X-ray flux in the 0.5-4 and 1-8 Angstrom (Å) wavelength bands. This allows for the accurate surveying of the start and evolution of solar flares. These instruments provide a gauge on the intensity of flares [18].

Other missions used for solar observations that are relevant, but are not examined in this study include: Transition Region and Coronal Explorer (TRACE) [23], Reuven Ramaty High-Energy Solar Spectroscopic Imager (RHESSI) [24], Solar Radiation and Climate Experiment (SORCE) [25], Hinode [26] and Solar Terrestrial Relations Observatory (STEREO) [27]

2.2.3 Solar cycles

The number of sunspots fluctuates in 11-year cycles (see Figure 2.5). This makes up one solar cycle [28], [29]. At the beginning of a solar cycle, the sunspots are located in higher latitudes up to 40°. Over the course of the solar cycle, the sunspots move closer to the equator. This behaviour is exemplified in the Maunder butterfly diagram in Figure 2.6.

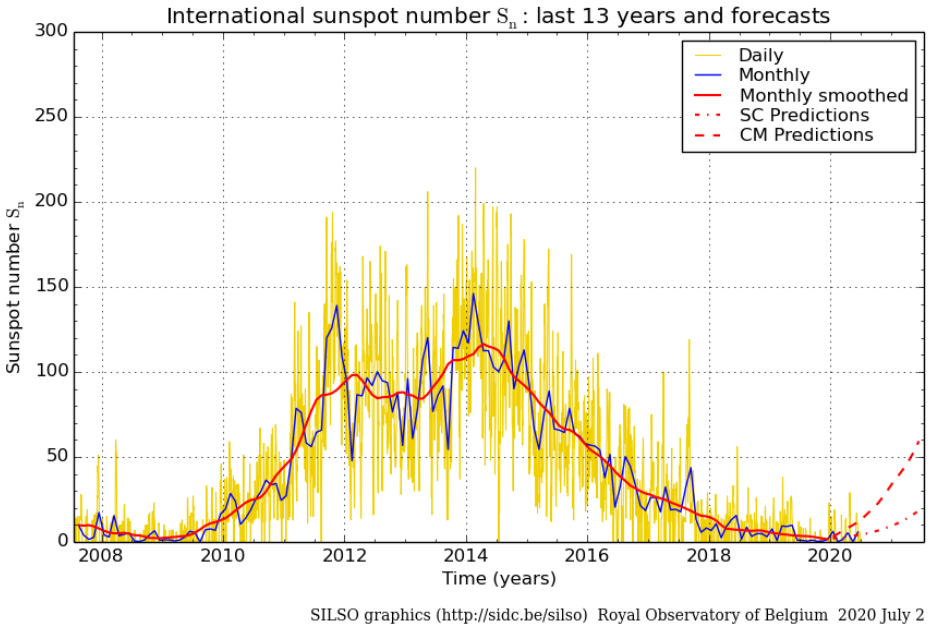


Figure 2.5: Daily and monthly sunspot count (last 13 years). Solar cycle 24. SC: Waldmeier's standard curves prediction method, based on the sunspot number series [30]. CM: K. Denkmayr and P. Cugnon's method, using regression on the sunspot number series along with the geomagnetic index [31]. Retrieved from [32]

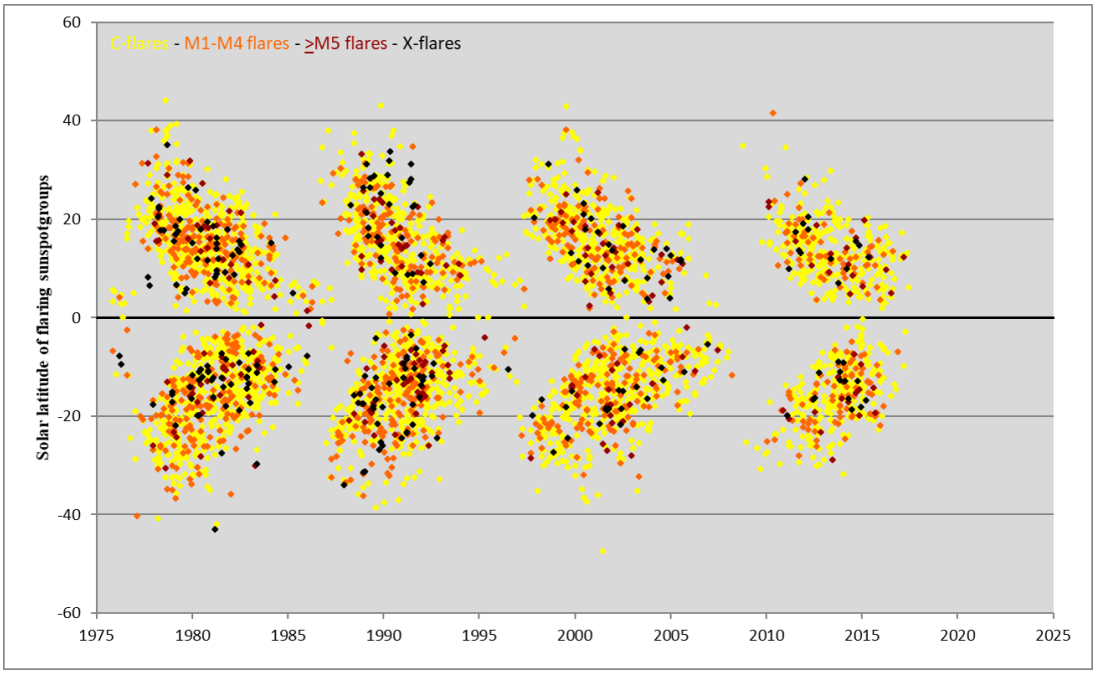


Figure 2.6: Maunder butterfly diagram of flare-producing sunspot groups. Sunspot latitudes over time. Retrieved from [33]

2.3 Machine learning and DNNs

"To deal with hyper-planes in a fourteen-dimensional space, visualize a 3D space and say 'fourteen' to yourself very loudly. Everyone does it." – Geoffrey Hinton

2.3.1 A brief overview of DNNs

Artificial intelligence (AI) has been the dream of inventors through the ages, but unlike the Mechanical Turk (1769) that thwarted great minds such as Napoleon and Benjamin Franklin at chess, which turned out only to be a man in a box [34], today, humanity has made great strides towards AI in solving menial tasks, understanding speech and images, as well as really beating the greatest chess players, to name but a few.

Representation learning is a set of methods that allows raw data to be fed into an algorithm that determines the representation needed for classification. Deep learning is basically representation learning that makes use of multiple layers of transformations. Deep learning is part of the larger machine learning family based on artificial neural networks [35]. With the rise of computing power and abundance of data in the last decade, the field of deep learning has gained plenty of traction, due to its pattern recognition ability that surpasses older machine learning techniques on tasks such as image recognition.

The adjective *deep* comes from the models having layers that extract simpler representations of the raw data. When coupled with non-linear activation functions, these networks can map inputs to outputs in a non-linear fashion. The term *neural* was loosely derived from the understanding of neurons at that time (1960s) but has changed since then [36].

DNNs most likely owe their fame to their ability to *generalise* well. The term "generalisation" points to a model's ability to perform well on data that it has not seen. The reason why DNNs generalise so well is still unclear and is an active area of research at present [11].

MLPs, also known as fully connected feedforward networks, are the most basic deep

learning architectures and we elaborate more on them in Chapter 5. Background on the MLP architecture provided in Section 5.2. One drawback of MLPs are that they do not take into account the temporal information of solar flares, unless something like sliding windows are used.

2.3.2 DNNs for time-series prediction

The nature of solar flares is time related, implying that past observations are relevant for future prediction. DNNs exist that have been designed for time-series prediction called recurrent neural networks (RNNs), the most popular being the long short-term memory (LSTM) [37]. Unlike the MLP, the RNN has connections that feed back to previous units, making it able to remember past information. The addition of hidden cell units allow the LSTM also to forget redundant information. LSTMs were also designed to combat the vanishing gradient problem of the normal RNN.

Recently, non-recurrent networks that have also shown promise, are the TCN [38] and our proposed one-dimensional convolutional neural network (1D-CNN). (Background on the TCN and our 1D-CNN is presented in Section 6.2). TCNs use dilated causal convolutions and skip connections to learn the temporal information of the data, instead of feeding back information to previous units like the LSTM, it uses convolution kernels with large receptive fields to remember past observations. The TCN can also be used as a drop-in replacement of a LSTM. The traditional 1D-CNN is the rawest form of convolutional networks that uses one dimensional data, instead of two dimensional data that is typically used for image and video classification, to extract the local information across the kernels [39]. They are used in many other applications such as electrocardiograms (ECG) classification or early motor fault detection [40], [41]. Convolutions are also easier to probe than LSTMs.

2.4 Previous studies on solar flare prediction

In this section we discuss the traditional approaches to flare prediction and how machine learning allowed us to use many features and large datasets. We also highlight the different approaches taken to predict flares. Many techniques have been applied successfully, of which we show a few examples. We establish and comment on a baseline model used for this study and discuss our deductions from previous works.

2.4.1 Traditional methods

Traditionally most flare prediction studies only used one or two features with simple statistical techniques; for example, if the fractal dimension of AR images is ≥ 1.25, the occurrence of an X-class flare is most likely [42]. These statistical methods often proved somewhat unreliable and did not scale well, especially when evaluated on different datasets.

2.4.2 Changes brought about by machine learning

Since the launch of the SDO in 2010 and the availability of data and computation power giving rise to machine learning, there has been a surge in solar-based prediction research. Instead of using a few features, we can now use hundreds of features in machine learning models, while also using enormous datasets such as the one generated by the SDO. We highlight some of the most groundbreaking approaches which include:

- A support vector machine (SVM) that looks only at the AR images of the solar magnetic fields and calculate 25 features from it (for example magnetic flux or how much energy is stored in the magnetic field) and then view it as a binary classification problem: whether the AR will produce a flare, yes or no [43]. This model achieves accuracy of 0.924 ± 0.007, which seems to be good, but for predicting rare events this can be deceitful because a high accuracy can be achieved by just

predicting 'no', all the time. This is why the researchers also used the true skill statistic (TSS) (recommended for flare prediction by [44], discussed in Section 2.6), that can quantify better how well the model is doing with imbalanced data (see Chapter 3). This skill score ranges from minus one to one, where minus one predicts everything wrong, zero is random guessing and one is perfect prediction. In this case, the researchers obtained a TSS = 0.761 ± 0.039. From a physics perspective, they deduced that ARs with a twisted or curled magnetic field are more likely to flare.

- Another approach tried to look not only at the magnetic field on the surface on the Sun but also directly at the image data at different wavelengths, as well as its atmosphere [45]. The researchers used a convolutional kernel network [46] on 3.6 million images of the solar surface and various layers of the solar atmosphere and ended up with a TSS = 0.814 ± 0.026, which is still within the error of the previous model. It is therefore unclear whether using this much extra data was a real improvement. They mention possible reasons for the small improvement: (1) They did not consider previous predictions and that something like an auto-regressive component could help. (2) They only looked at ARs individually, but perhaps the different ARs are influencing each other; this concept is not new and is called sympathetic flaring [47]. Sympathetic flaring is another technique that is currently being studied with machine learning.

- Similarly, [5] used a LSTM model and tried to derive features from the magnetic field images employing an auto-encoder. The authors found that their machine-derived features could predict almost as well as the physics-based features (SHARP, discussed in Chapter 3) derived from physical understanding. LSTM and other temporally sensitive networks are useful, since they capture the flare dynamics better.

- Ambitious endeavours include using the entire Earth-Sun model, which includes solar image data, solar wind, the Earth's magnetic field etc. This is currently being researched by the SOLSTICE team [48].

2.4.3 Summary of previous results

The most significant models for flare prediction (with their original proposers) are the MLP [49], SVM [43], [49]–[53], Jordan Network (JN) [54], Deep Flare Net (DeFN) [2], random forest (RF) [49], [55], [56] and LSTM network [4].

We briefly summarise the key results in Table 2.2 below. The table is comprised of the popular models and their respective scores (recall, precision, accuracy, balanced accuracy (BACC), Heidke skill score (HSS) and TSS, respectively) achieved on Liu et al.'s [4] dataset for a given flare class (adapted from Liu et al. with permission).

There are a few caveats that should be mentioned here:

- Only the architecture is given: Liu et al. did not mention any specifics on the HPs or optimisation strategy used. The extent of optimisation on models other than the LSTM is unclear.
- No standard deviation/error is given nor any indication that the networks are stable across multiple seeds.
- No validation score is given, only a test/evaluation score. This makes it difficult to reproduce the optimisation of their models.
- These results are not from their cross-validation evaluation technique. (See Appendix A.1 for more information on this technique.)

Table 2.2: Previous machine learning flare prediction models and results (reproduced from Liu et al. [4] with permission). Metrics include: recall, precision, accuracy, BACC, HSS and TSS, respectively

Metric	**Model**	**≥M5.0**	**≥M**	**≥C**
Recall	MLP	0.944	0.812	0.637
	SVM	0.644	0.692	0.746
	JN	0.923	0.851	0.701
	DeFN	0.889	**0.891**	0.761
	RF	**1.000**	0.850	0.727
	LSTM	0.978	0.881	**0.762**
Precision	MLP	0.037	0.143	0.451
	SVM	0.014	0.106	0.497
	JN	0.033	0.178	0.543
	DeFN	0.037	0.173	0.497
	RF	0.034	**0.252**	0.532
	LSTM	**0.038**	0.222	**0.544**
ACC	MLP	0.901	0.855	0.778
	SVM	0.818	0.824	0.803
	JN	0.882	0.884	0.826
	DeFN	**0.907**	0.872	0.801
	RF	0.886	**0.924**	0.822
	LSTM	0.899	0.909	**0.829**
BACC	MLP	0.922	0.834	0.725
	SVM	0.732	0.76	0.781
	JN	0.903	0.868	0.779
	DeFN	0.898	0.881	0.786
	RF	**0.943**	0.888	0.786
	LSTM	0.938	**0.895**	**0.803**
HSS	MLP	0.064	0.204	0.389
	SVM	0.020	0.141	0.472
	JN	0.056	0.26	0.502
	DeFN	0.064	0.253	0.476
	RF	0.059	**0.361**	0.502
	LSTM	**0.074**	0.347	**0.539**
TSS	MLP	0.845	0.669	0.449
	SVM	0.464	0.52	0.562
	JN	0.806	0.736	0.558
	DeFN	0.796	0.763	0.572
	RF	**0.886**	0.776	0.572
	LSTM	0.877	**0.790**	**0.607**

2.4.4 Baseline model

According to Liu et al.'s results, their LSTM model performs the best compared to the other models with the binary classification task. This research builds on the work of Liu et al. [4], using their dataset (see Chapter 3) and results to evaluate our model's performance. Their best model made use of attention mechanisms, which is beyond the scope of this study. They also only used 20 of the 40 available features after doing feature elimination. During our analysis of [4] we found some discrepancies and possible issues that we will highlight and address as they occur.

2.4.5 Key points and unsolved problems with flare prediction

The key points that previous research reveals are:

- Machine learning does hold a significant advantage compared to traditional methods. Open-source data and software have sparked a significant improvement in the field [43].
- "More data" and "bigger models" do not yield better predictive capacity, but rather more information-rich data, and well-optimised models [45].

Three main problems remain unresolved in the general field of solar flare prediction and machine learning:

- **Class imbalance problem** - Many real-world problems suffer from this issue, where the class that one is trying to predict is outweighed by another. Various possible solutions exist including the use of generative adversarial networks [57], to generate extra data.
- **Benchmarking problem** - There are many different datasets and models, but no standard to test against or a benchmark to compare model performance. This could be solved by using large inter-calibrated datasets.

- **Interpretability problem** - Deep learning models are notorious for being difficult to interpret. From a science perspective, understanding the dynamics of the Sun would prove more useful than a model that can only predict flares accurately. This is more the focus of the current study than the previous two problems.

2.5 Interpretability and explainability of DNNs

DNNs are notoriously difficult to interpret and are likened to a "black box"; the model can give excellent results, but it is unknown how it arrives at that conclusion. This is often undesirable in medical, science and financial fields, where trust and the reason for decision-making are crucial. In recent years, research has surged towards more explainable and interpretable DNNs [58].

The terms 'explainability' and 'interpretability' are used in different ways in different contexts in the literature. Explainability is often used to describe the importance of features that lead to a decision, for example heat-maps on images that highlight the most important pixels for image classification. Interpretability can be thought of mapping an abstract concept to human understanding. This is often coupled with what an algorithm bases its choices on [12]. For this study, we define interpretability as identifying the relative importance of features that contribute to flare production.

Traditionally, decision trees were the preferred method for interpretable machine learning, but performed unsatisfactorily when dealing with higher-order multi-dimensional problems such as image classification [59], hence, the need for models and techniques that achieve good performance but remain explainable and interpretable.

Some approaches that try to understand DNNs include, architectures designed to reveal the importance of features [60], attribution methods [61] or even model-agnostic approaches, such as local interpretable model-agnostic explanations (LIME) [62]. Many have tried to compare the different techniques but thus far, no single method has been found satisfactory or close to perfect [12], [61], [63]–[65]

2.5.1 DNN attribution methods

Attribution methods try to quantify the relevance of input features in a network, usually accomplished with one of two manners, perturbation-based or gradient-based methods [61].

Perturbation-based methods calculate the attribution (how much a feature attributes to the prediction) by changing an input feature and measuring the output with the original. Typical perturbation-based methods include:

- **Ablation** [66], which determines the importance based on the magnitude of change in the output when the inputs are replaced by a baseline (typically zero).
- **Shapely value sampling** [67], derives the feature importance based on the number of sample permutations on all the features, by adding each permutation to the baseline and computing the magnitude of change in the output; these results are then averaged over all permutations.

Gradient-based methods determine the attribution of all the input features after doing a forward and backward propagation through the network. There are a growing number of gradient-based techniques, such as:

- **DeepLIFT (deep learning important features)** [68] describes the difference in the non-linear activation's outputs in relation to the differences in inputs from a baseline.
- **Integrated gradients** [69] approximates the integral of gradients across the network path and from the baseline to inputs.
- **Input $\times$ gradient** [68] was initially proposed to sharpen attribution maps. It calculates the attribution by taking the inputs and multiplying these with the signed partial derivatives of the output with respect to the input.

The Captum library [70] was designed to implement the listed attribution methods in a network easily, which we use and discuss in more detail in Chapter 7.

2.6 Performance metrics

To quantify the performance of our models, we incorporate the typical metrics used for flare prediction in binary classification, as well as probabilistic forecasting [2], [4], [6], [7], [9], [43]–[45], [49], [56], [71].

2.6.1 Binary classification metrics

Binary classification problems can be analysed using a confusion matrix, which consists of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). These values can then be arranged to determine specific metrics. The metrics typically used for flare prediction are,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.1}$$

$$\text{Precision } = \frac{\text{TP}}{\text{TP } + \text{ FP}} \tag{2.2}$$

$$\text{Accuracy (ACC)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2.3}$$

$$\text{Balanced accuracy (BACC)} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \tag{2.4}$$

$$\text{Heidke skill score (HSS)} = \frac{2(\text{TP} \times \text{TN} - \text{FP} \times \text{FN})}{(\text{TP} + \text{FN})(\text{FN} + \text{TN}) + (\text{TP} + \text{FP})(\text{FP} + \text{TN})} \tag{2.5}$$

$$\text{True skill statistic (TSS)} = \frac{\text{TP}}{\text{TP} + \text{FN}} - \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{2.6}$$

Accuracy is inadequate for unbalanced classification; BACC is better. As suggested by [44], we use TSS as our main metric, which is the recall minus the false alarm rate. The HSS indicates how much improvement there is over random guessing. The higher BACC, TSS and HSS are, the better the model performs.

2.6.2 Probabilistic forecasting metrics

Any binary classification task can be converted to a probabilistic forecasting model by applying a soft-max function to the output layer to squeeze the outputs between zero and one, this gives a probability of flare eruption. This is often better for climatology forecasting such as flare prediction. Common metrics include the Brier score (BS) and Brier skill score (BSS).

The BS is typically used as a measurement for accuracy for probability forecast; it is merely the mean-squared error (MSE) of n number of forecast-observation pairs [72],

$$\text{BS} = \frac{1}{\text{n}} \sum_{\text{i}=1}^{\text{n}} (\text{y}_\text{i} - \text{x}_\text{i})^2 , \qquad (0 \leq BS \leq 1) \tag{2.7}$$

with the forecast (x_i) as a probability.

BSS is adjusted to give the skill or relative accuracy based on the event relative frequency, given as [73],

$$\text{BSS} = \frac{\text{BS} - \text{BS}_\text{clim}}{0 - \text{BS}_\text{clim}} = 1 - \frac{\text{BS}}{\text{BS}_\text{clim}}, \qquad (BSS \leq 1) \tag{2.8}$$

where,

$$\text{BS}_\text{clim} = \text{BS for the reference forecast} \tag{2.9}$$

The reference forecast is the probability at which a relevant event is likely to occur.

For more on the practicalities of forecasting, refer to Chapter 8.

2.7 Conclusion

Solar flare prediction is crucial for protecting mankind's technological infrastructure. Traditionally basic statistical and machine learning techniques were used to predict solar flares by only using a single or a few features, but the advent of deep learning has sparked new interest in the space weather field with a host of new promising techniques with improved performance. These novel techniques should be used carefully when developing and measuring model performance.

Deep learning also comes with a lack of interpretability that is not desirable when trying to uncover the scientific principles that govern flare production, although some techniques such as attribution methods, can reveal the importance of certain features.

Chapter 3

Dataset

In this chapter we describe the dataset, how it is acquired, and what the input and output features are, as well as giving general statistics of the data.

3.1 Description

The dataset describes a binary classification problem: whether or not a Γ-class flare will erupt within the next 24 hours, where Γ is $\geq C$, $\geq M$ and $\geq M5.0$ class. For the remainder of the study, we will only focus on $\geq M5.0$-class flares, since their classified as large flares by the National Centers for Environmental Information (NCEI). The dataset consists of input and target features. Input features describe the evolution of an identified AR of the Sun using 25 physical features and 15 history features. Physical features are of two types: SHARP describes the magnetic-field characteristics of an active patch, as estimated from images of the Sun's magnetic field (the SHARP features) [74], and Lorentz force components are functions of the SHARP parameters that capture dynamic behaviour given estimated magnetic field characteristics (the Lorentz features) [75], [76]. History features of a specific AR encapsulates (part of) its flaring history. This is important

because complex ARs can produce multiple flares (see September 6, 2017, NOAA: 12 673, for an example). Target parameters describe whether or not (True/False) a flare $\geq M5.0$-class (in our case) will occur within the next 24 hours, for the given input features. For a description of input and output parameters, see Section 3.3.

3.2 Source

The dataset that is used in the remainder of this study (with a few tweaks and fixes, see Section 3.5 and 3.3.4), is open source and available on GitHub under the GNU General Public License. It is sourced, processed and compiled by Liu et al. [4]. All of their data and models are available for download at the link provided[1]. The SHARP data series was released at the end of 2012 and was instantiated by the SDO/HMI team [74]. It can be queried at Joint Science Operations Center (JSOC)[2] as *hmi.sharp*. The Lorentz force components can be queried as *cgem.Lorentz*. By querying the website, one can specify the desired time or ARs, and receive all extracted features for the given data series (for example *hmi.sharp* or *cgem.Lorentz*).

Target parameters are derived from the GOES X-ray flare catalogues, provided by the NCEI. The catalogues include the flare time and magnitude pertaining to each AR.

The input and target parameters are obtained for the period of May 2010 and May 2018 from the JSOC website using the SunPy package [17], at a cadence of 1 hour. Similar to what Bobra and Couvidat [43] do, the AR whose samples are $\pm 70°$ from the central meridian or whose features are incomplete are discarded. These edge cases are discarded, since the parameters derived from the images could be misleading.

[1] https://web.njit.edu/wangj/LSTMpredict/
[2] http://jsoc.stanford.edu/

3.3 Input and target parameters

In this section we examine all the input and target features, to understand what they are and how they are prepared for the deep learning models. For the input parameters, there are two groups of predictive parameters. The first group relates to the 25 physical features described in [43], that portrays magnetic field properties. The second group consists of 15 features related to flaring history. We elaborate on this in the following sections. For a description of each input parameter, see Table A.1 in Appendix A.2.

3.3.1 Physical features

The HMI instrument [22] takes full-disk images of the Sun's magnetic field, as shown in Figure 2.2. An automated processing pipeline extracts the HMI active region patch (HARP) at 720-second intervals throughout its lifetime [74]. Finally, the SHARP parameters are derived from the cut-out images, which are available in helio-projective Cartesian CCD image coordinates or the remapped cylindrical equal area (CEA) projection [77]. SHARP describes the magnetic-field distribution and its deviation from a potential-field configuration [74]. The Lorentz force components are the integrated Lorentz force measured over the outer solar atmosphere, which is perpendicular to the motion of the charged particles and magnetic field [76], [78]. The SHARP parameters and the additional Lorentz force components that are based on the SHARP data make up the first 25 physical features of the dataset.

3.3.2 Flare history features

The first nine out of 15 history features describe the AR at data sample x_t for time step t as defined by Nishizuka et al. [79]. These features include the total number of B-class (C-class, M-class, X-class, respectively) flares that developed in the same AR before the observed sample at time step t, named as *Bhis* (*Chis*, *Mhis*, *Xhis*, respectively). Similarly, features *Bhis1d* (*Chis1d*, *Mhis1d*, *Xhis1d*, respectively) represent the number of relevant

class flares 24 hours before time point t in the same AR. *Xmax1d* represents the maximum X-ray flux in the last 24 hours before time point t.

The last six of the 15 history parameters are related to time decay values, as proposed by Jonas et al. [45]. The decay value for data sample x_t at time step t with respect to the B-class (C-class, M-class, X-class, respectively), expressed as *Bdec* (*Cdec*, *Mdec*, *Xdec*, respectively), is calculated as,

$$\mathrm{Bdec}\,(x_t) = \sum_{f_i \in F_B} e^{-\frac{t-t(f_i)}{\tau}} \tag{3.1}$$

$$\mathrm{Cdec}\,(x_t) = \sum_{f_i \in F_C} e^{-\frac{t-t(f_i)}{\tau}} \tag{3.2}$$

$$\mathrm{Mdec}\,(x_t) = \sum_{f_i \in F_M} e^{-\frac{t-t(f_i)}{\tau}} \tag{3.3}$$

$$\mathrm{Xdec}\,(x_t) = \sum_{f_i \in F_X} e^{-\frac{t-t(f_i)}{\tau}} \tag{3.4}$$

where F_B (F_C, F_M, F_X, respectively) depicts the corresponding flare that happened in the same AR before time step t. $t(fi)$ is the time of flare f_i, and τ is the decay constant, where τ is set to 12 as empirically determined by Jonas et al. [45]. The last two time decay parameters take into account all flare classes before time step t and are calculated as follows,

$$\mathrm{Edec}\,(x_t) = \sum_{f_i \in F} E\,(f_i) \cdot e^{-\frac{t-t(f_i)}{\tau}} \tag{3.5}$$

$$\log \mathrm{Edec}\,(x_t) = \sum_{f_i \in F} \log\left(E\,(f_i)\right) \cdot e^{-\frac{t-t(f_i)}{\tau}} \tag{3.6}$$

where $F = F_B \cup F_C \cup F_M \cup F_X$ and $E(f_i)$ is the magnitude of flare f_i.

Figure 3.1 is a simple representation of what the decay values look like.

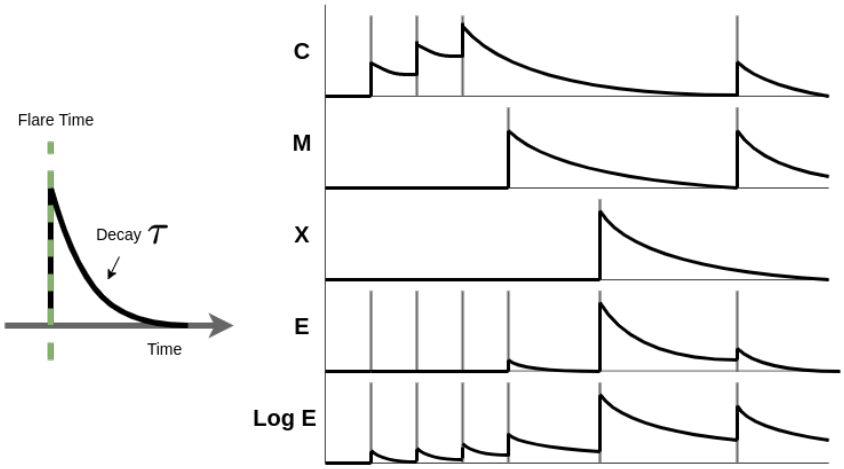


Figure 3.1: Simplified decay value illustration. Each class decays at constant rate τ after a flare. E is the sum of the decay values of all the classes. Log E is the logarithm of E. Class B is omitted for illustration purposes. Replicated from source [45]

3.3.3 Target parameters

Figure 3.2 depicts the labelling scheme used to create the target parameters. The flares are labelled positive 24 hours before the start of the Γ-class flare, where Γ is $\geq C$, $\geq M$ and $\geq M5.0$-class.

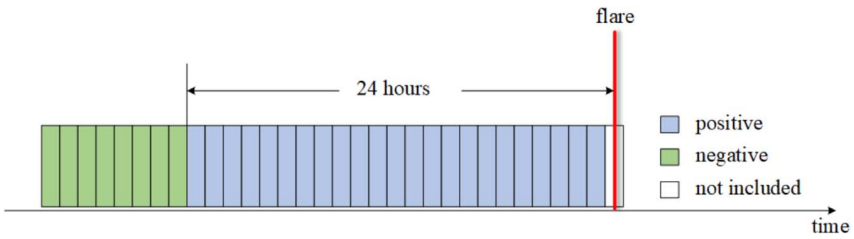


Figure 3.2: Illustration of positive and negative labelling for the prediction task. Every rectangular box represents a sample at one-hour intervals. The red line indicates the starting time of a Γ-class flare. Data samples 24 hours before the flare are labelled as positive, represented by the blue rectangles. The other samples belong to the negative class, as shown by the green boxes. Samples are not included from the white rectangular box. Sourced with permission from Liu et al. [4]

3.3.4 Normalisation

Liu et al. [4] state that because the features each has different units and scales, varying in orders of magnitude, the physical features are normalised with z-norm (Equation 3.7) and the history features are minimum-maximum (min-max) scaled (Equation 3.8). (We argue that performing z-norm on all the features is sufficient and elaborate more on this topic in Chapter 4.) Nevertheless, the normalisation that Liu et al. originally used is done as follows: For the 25 physical features, z_i^k stands for the normalised value of the ith features of the kth data sample. Such that

$$z_i^k = \frac{v_i^k - \mu_i}{\sigma_i} \tag{3.7}$$

where v_i^k is the value of the ith of the kth data sample, μ_i is the mean of the ith feature, and σ_i is the standard deviation of the ith feature.

For the 15 flare history features, we have

$$z_i^k = \frac{v_i^k - \min_i}{\max_i - \min_i} \tag{3.8}$$

where max_i and min_i are the maximum and minimum values of the ith feature, respectively.

The i values used by [4] are somewhat controversial, and discussed in Section 4.4.1.

3.4 Dataset partitioning and statistics

In this section, we explain how the dataset is partitioned into the required training, validation and testing sets. We also extract a few general statistics from the dataset, such as the number of flares, sunspots, and ARs with their corresponding lifetimes.

3.4.1 Dataset partitioning

The dataset is prepared specifically to solve a binary classification problem; whether or not a Γ-class flare will erupt from an AR within the next 24 hours, where Γ is divided into three classes separately: $\geq M5.0$, $\geq M$ and $\geq C$ class. This follows the previous attempts made by Bobra & Couvidat [43], Jonas et al. [45] and Nishizuka et al. [2], using historical observations of an AR to predict its future flaring activity. All the aforementioned studies considered $\geq M$ and $\geq C$ class flares. Additionally, the $\geq M5.0$ class is considered in this dataset since there are so few X-class flares.

Supervised learning requires that the dataset be divided into some form of training, validation and test set. The training and testing set should be disjoint to ensure so that the predictions are on samples it has never seen before. The number of positive and negative samples per flare class for each partition can be seen in Table 3.1.

Table 3.1: Number of positive and negative samples for each flare class in every partition. Reproduced from Liu et al [4] with permission.

	$\geq C$ class		$\geq M$ class		$\geq M5.0$ class	
	Positive	Negative	Positive	Negative	Positive	Negative
Training (2010-2013)	18 266	66 311	2 710	81 867	633	83 944
Validation (2014)	7 055	19 418	1 347	25 126	292	26 181
Testing (2015-2018)	8 732	35 957	1 278	4 341	180	44 509

The data can also be sequenced before feeding to a model, as in the case of an LSTM or TCN. This means that for time step t we use the samples from $(t - s)$, where s is the sequence length. When sequencing the data, one needs to append zeroes before an AR for the length of the sequence, so as not to leak data from one AR to the next.

3.4.2 Flares per class

Table 3.2 shows the number of flares per class in the entire dataset.

Table 3.2: Flares per class type

Class Flare	Flares
B	4 203
C	6 768
M	704
X	49

3.4.3 Class $\geq M5.0$ dataset statistics

The $\geq M5.0$ dataset will be used, since is it generally regarded as a major class flare. Therefore, Table 3.3 shows the relevant information on the dataset labelled positive for flares $\geq M5.0$ in magnitude. For each dataset partition (training, validation, testing), we list the number of ARs that produced a $\geq M5.0$ flare, the number of samples that are labelled positive for being $\geq M5.0$, the number of samples of those ARs that eventually produce a flare $\geq M5.0$ and the total number of samples. Lastly, the fraction of imbalance is also measured by taking the positive labelled samples and dividing these by the total number of samples. This gives a sense of how unbalanced the dataset is, meaning how much the positive samples outweigh the negative samples.

Table 3.3: $\geq M5.0$-class flare dataset partitioning and statistics

Category	Train Set	Validation Set	Test Set
ARs >M5	23	12	8
Positive labelled samples	633	292	180
Samples of ARs that eventually flare >M5	4 362	2 183	1 476
Total samples	84 577	26 473	44 689
Fraction of imbalance	0.75%	1.1%	0.4%

3.4.4 Active region lifetimes

Since the Sun is not a solid mass, the duration of rotation depends on the distance from the equator (At equator: 25 days [77]). Each AR has different locations that it can occupy

on the Sun, which can determine the AR's size and duration. This also changes depending on the time in the solar cycle [13] (see: butterfly diagram in Figure 2.6). Including the coordinates could also prove useful for developing a flare prediction model; unfortunately, these are not included in the Liu et al. [4] dataset.

Data are sampled at a 1-hour cadence, thus if we count the samples, we can determine the number of days these ARs last. Naturally, since only a part of the Sun is seen, if the AR moves behind the edge of the Sun, it is not possible to know its duration past that point. To get a rough estimate, the samples for flares for each class are counted and the number is averaged in Table 3.4. It can be seen that ARs producing X- and M-class flares seem to have longer lifetimes than those producing less intense flares. Each AR can flare multiple times in its lifetime. This statistic is strictly specific to this dataset and is not a study on AR lifetimes. For more information on sunspot lifetimes, refer to [80].

Table 3.4: Average AR duration per flare class

Flare Class	Samples per AR ($\mu \pm \sigma$)	Days ($\mu \pm \sigma$)
B	152 ± 57	6.3 ± 2.4
C	167 ± 49	7.0 ± 2.1
M	184 ± 37	7.6 ± 1.5
X	182 ± 28	7.6 ± 1.2

3.4.5 Annual sunspot count

The number of ARs per year are tallied and presented in Table 3.5. The peak of the solar cycle can be seen to have been reached in 2013-2014 (refer to the annual sunspot count in Figure 2.5). This dataset contains the bulk of the solar cycle, but not all.

Table 3.5: Annual sunspot count in entire dataset

Year	Number of ARs
2010	72
2011	195
2012	201
2013	235
2014	184
2015	191
2016	130
2017	62
2018	9

3.5 Errata for Liu et al.'s dataset

There is a minor fault in the original dataset used by Liu et al. [4]. It does not invalidate their models but changes the ranking of their features. The problem is that some of the physical SHARP features are incorrectly labelled and do not correspond to the JSOC names. This issue possibly originated from their data queries to JSOC, since the issue is already in their raw data. See Figures 3.3a and 3.3b, where the value in red (a) is the same as in (b) but (a) has the incorrect feature name. In this example *SAVNCPP* (a) should be *USFLUX* (b). This is the same issue for the other physical features.

timestamp	NOAA	HARP	TOTUSJH	SAVNCPP	ABSNJZH
2013-12-28T01:10:09.20Z	11937	3542	790.637	1.507953E+022	2421783000000
2013-12-28T02:10:09.30Z	11937	3542	801.765	1.572881E+022	1576670000000
2013-12-28T03:10:09.30Z	11937	3542	843.463	1.556715E+022	1589849000000
2013-12-28T04:10:09.30Z	11937	3542	873.5	1.521064E+022	2646049000000
2013-12-28T05:10:09.30Z	11937	3542	855.808	1.467145E+022	2860127000000
2013-12-28T06:10:09.30Z	11937	3542	815.75	1.44592E+022	1881047000000
2013-12-28T07:10:09.30Z	11937	3542	819.588	1.486197E+022	1846910000000
2013-12-28T08:10:09.30Z	11937	3542	823.751	1.514216E+022	2338215000000
2013-12-28T09:10:09.30Z	11937	3542	842.898	1.527455E+022	3432347000000

(a) Screenshot of Liu et al.'s dataset. Incorrect feature name and value highlighted in red.

RecordName	USFLUX	MEANGAM	MEANGBT
hmi.sharp_720s[3542][2013.12.28_01:12:00_TAI]	1.507953e+22	34.594	90.073
hmi.sharp_720s[3542][2013.12.28_01:24:00_TAI]	1.529990e+22	34.640	89.335
hmi.sharp_720s[3542][2013.12.28_01:36:00_TAI]	1.553838e+22	34.492	90.042
hmi.sharp_720s[3542][2013.12.28_01:48:00_TAI]	1.567458e+22	34.625	89.628
hmi.sharp_720s[3542][2013.12.28_02:00:00_TAI]	1.559958e+22	34.036	89.534
hmi.sharp_720s[3542][2013.12.28_02:12:00_TAI]	1.572881e+22	34.191	88.976

(b) Screenshot of JSOC query. Correct feature name and value highlighted in red.

Figure 3.3: Example of incorrect labelling in Liu et al.'s dataset.

We present a quick fix for the data in Table 3.6. Each feature name should just be

renamed to the correct feature name. All of our subsequent experiments are done with the names of the new features.

Table 3.6: Errata for Liu et al.'s dataset

Old feature	New feature
SAVNCPP	USFLUX
ABSNJZH	SAVNCPP
TOTPOT	TOTUSJZ
TOTUSJZ	ABSNJZH
TOTBSQ	TOTPOT
USFLUX	AREA_ACR
AREA_ACR	MEANPOT
MEANPOT	R_VALUE
SHRGT45	MEANGAM
MEANSHR	MEANJZH
MEANJZD	MEANALP
MEANJZH	MEANSHR
MEANGAM	MEANGBT
MEANGBT	MEANGBZ
MEANALP	TOTBSQ
R_VALUE	SHRGT45
MEANGBZ	MEANJZD

3.6 Conclusion

For the rest of this study, we proceed to use the $\geq M5.0$ class dataset, unless stated otherwise, since a $\geq M5.0$ class is generally considered a major class flare. The fixed dataset according to the errata is also used. When referring to the training, validation and test set, we mean the original partition of 2010-2013, 2014 and 2015-2018 respectively. We stick to the original partition to keep our models comparable with those of Liu et al. [4].

For a full description of each parameter and its formula, see Appendix A.2 Table A.1.

Chapter 4

Feature analysis

In this chapter we perform an analysis on the dataset, in order to understand the features and their dependencies with each other better, as well as their predictive ability.

4.1 Introduction

"Data is like garbage. You'd better know what you are going to do with it before you collect it" - Mark Twain

Before training of a neural network can commence, the data need to be analysed, to prepare it better for the model development. In Chapter 3, the dataset was probed for a few general statistics, but it does not tell us anything about how the features interact with one another, how useful certain features are for flare prediction or the distributions of the features.

The aim is to understand the data better by analysing the distribution and correlation for each feature. We also inspect how predictive each feature is to the target. The impact of different scaling strategies is also analysed.

We start by determining the distribution of each feature in Section 4.2. In Section 4.3, the correlation of all features with one another are calculated. We study the effect of different scaling strategies in Section 4.4. Lastly, we analyse how predictive certain features are related to the target in Section 4.5.

4.2 Feature distribution analysis

Before we can determine how predictive each feature is to the target, the features need to be checked to see if they are normally distributed. If the features are not normally distributed then either non-parametric methods need to be used to determine predictive capability, or the data need to be transformed with something like power transforms [81].

First, we discuss how one can test for normality, then we apply the test to our data. Based on our observations, we decide which methods can be used to give an estimate of how predictive features are.

4.2.1 Tests for normality

Typical normality tests include graphically looking at the histograms or drawing a quantile-quantile plot of each feature. This is less viable when dealing with many features. Another alternative is performing statistical tests, such as the D'Agostino-Pearson Test, which measures skewness and kurtosis [82]. However, statistical tests come with their own set of caveats, for example, being very sensitive to the number of samples and having assumptions [83]; in this case, we settle on a histogram to obtain an indication of feature distribution.

4.2.2 Feature distribution plots

To obtain the distributions of our features, we plot a histogram for each feature of the training set, along with its maximum likelihood Gaussian distribution, in Figure 4.1 below.

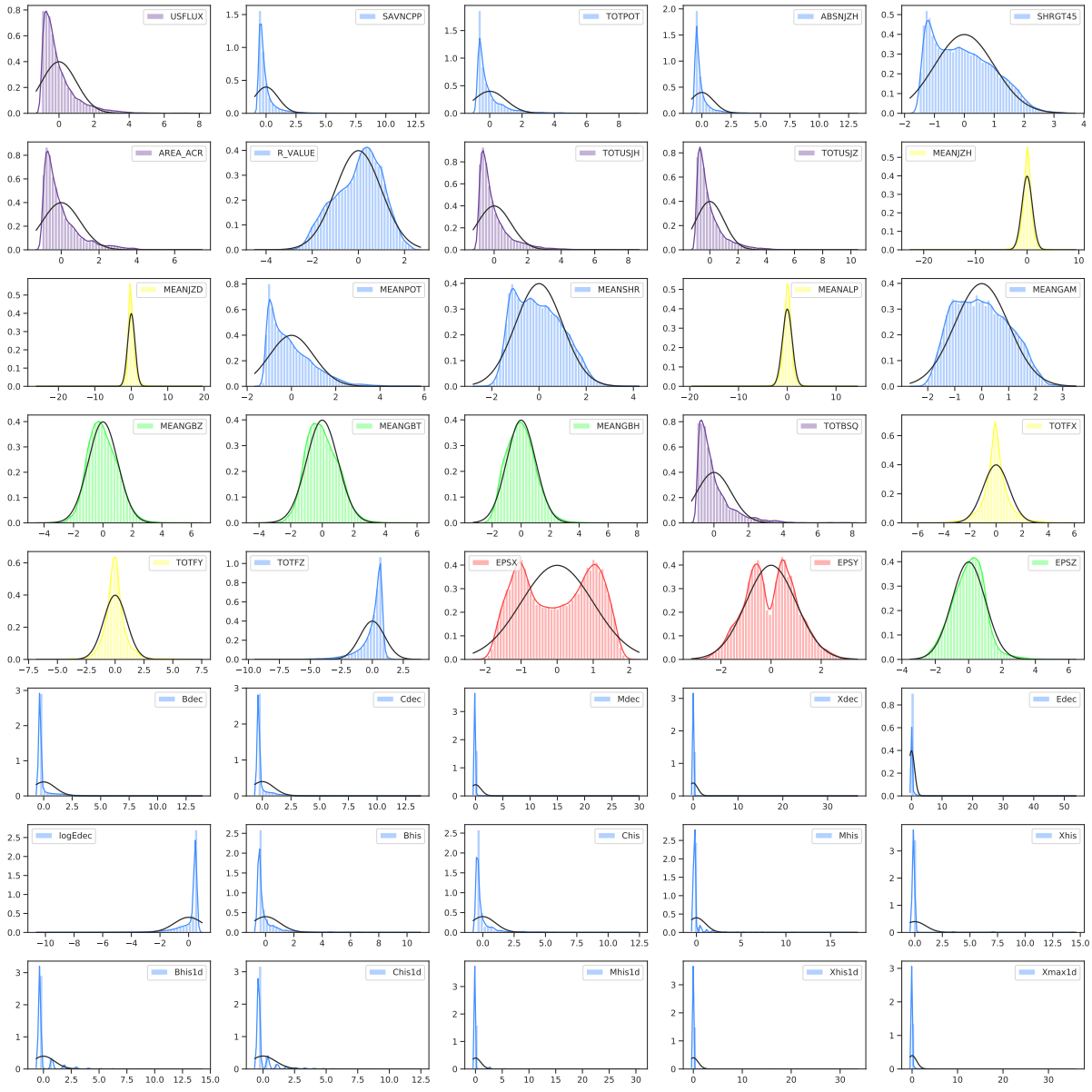


Figure 4.1: Histogram of each feature (coloured) and maximum likelihood Gaussian distribution (black). Colours indicate: Strong normal (green), log-normal (magenta), weak leptokurtic normal (yellow), bi-modal (red). Z-normalised to have zero mean and equal unit variance on all features before plotting. Training set. (Best viewed as pdf.)

From Figure 4.1, the maximum likelihood Gaussian distribution (black curve) is an estimate of how the histogram (coloured curve) should be for the data to be normally distributed. The closer the coloured curve resembles that black curve; the more normally distributed the feature. Features are visually sorted into the groups indicated in Table 4.1:

Table 4.1: Distribution categories of strong normal, weak leptokurtic normal, bi-modal and non-normal features.

Categories	**Condition**	**Features**
Strong normal (Green)	If histogram fits the black curve closely	*MEANGBZ*, *MEANGBT*, *MEANGBH*, *EPSZ*
Log-normal (Magenta)	If the distribution is strong log-normally distributed	*USFLUX*, *AREA_ACR*, *TOTUSJH*, *TOTUSJZ*, *TOTBSQ*
Weak leptokurtic normal (Yellow)	If the distribution is not skewed with only a small spike making it somewhat leptokurtic	*MEANJZH*, *MEANJZD*, *MEANALP*, *TOTFX*, *TOTFY*
Bi-modal (Red)	If there are clearly two peaks	*EPSX*, *EPSY*
Non-normal (Blue)	The other 24 features are not normally distributed i.e. large kurtosis and/or skewness	Remaining

Not all of the features in the dataset are normally distributed, as confirmed by Figure 4.1. Most of the features suffer from very strong skewness or kurtosis, especially history features. Some features seem to be log-normally distributed, indicating multiplicative random processes instead of additive process as is the case with normally distributed

phenomena. See [84] for interpreting log-normal distribution in biological systems. The features are not uni-modal, normal or univariate, therefore analysis of variance (ANOVA) would not work with these data [85]. Even if we were to power transform our data to be more normally distributed, the ANOVA results would still not be very reliable (see supplemental Figures A.5 in Appendix A.3). An alternative would be to use non-parametric methods such as mutual information (MI) for feature ranking, which we discuss further in Section 4.5.

4.3 Feature correlation

To understand more about the relationships between the features themselves, the correlation between each feature and every other feature can be determined to tell us more about how related they are to each other. In this section, we touch on the different kinds of correlations and determine the Pearson correlation between all the features with one another, displayed as heat-maps.

4.3.1 Correlation methods

Correlation measure the linear association between two variables [86]. If the correlation is strong between two features, it means that they are closely associated with each other. The most popular and relevant two correlations are Pearson's and Spearman's rank correlation, each having unique conditions and advantages.

Pearson correlation coefficient

This measures how strong the linear relationship is between two variables [87], quantified between minus one and one, where one is a perfect linear relationship, zero is no relationship and minus one is a perfect negative relationship. It also assumes that both variables are normally distributed when dealing with rankings, but do not need to be normally

distributed when only calculating linear relationships [88].

Spearman's rank correlation coefficient

Often in the real world, values are not linearly correlated, which is where Spearman's correlation is useful. Unlike Pearson's correlation it does not measure the strength of the linear relationship, but the strength and direction of the monotonic relationship. A monotonic relationship implies that when one value increases so does the other and vice versa [89]. It makes no assumptions about the distribution of the data and is non-parametric.

4.3.2 Feature correlation plots

Pearson correlation measures the strength of the linear correlation between two distributions [87]. Therefore, a grid of Pearson correlation coefficients can be made of all the input features in combination with one another. This grid reveals how strongly each feature is linearly correlated with another. The grid is visualised as a heat-map in Figure 4.2 below. The features are grouped into clusters where certain features within a group are closely linearly correlated with one another. The feature pairs that have a Pearson correlation $\rho > \pm 0.7$ are then inspected, since a correlation of $\rho > \pm 0.7$ indicates a strong linear correlation. All other feature pairs are indicated as zero for simplicity.

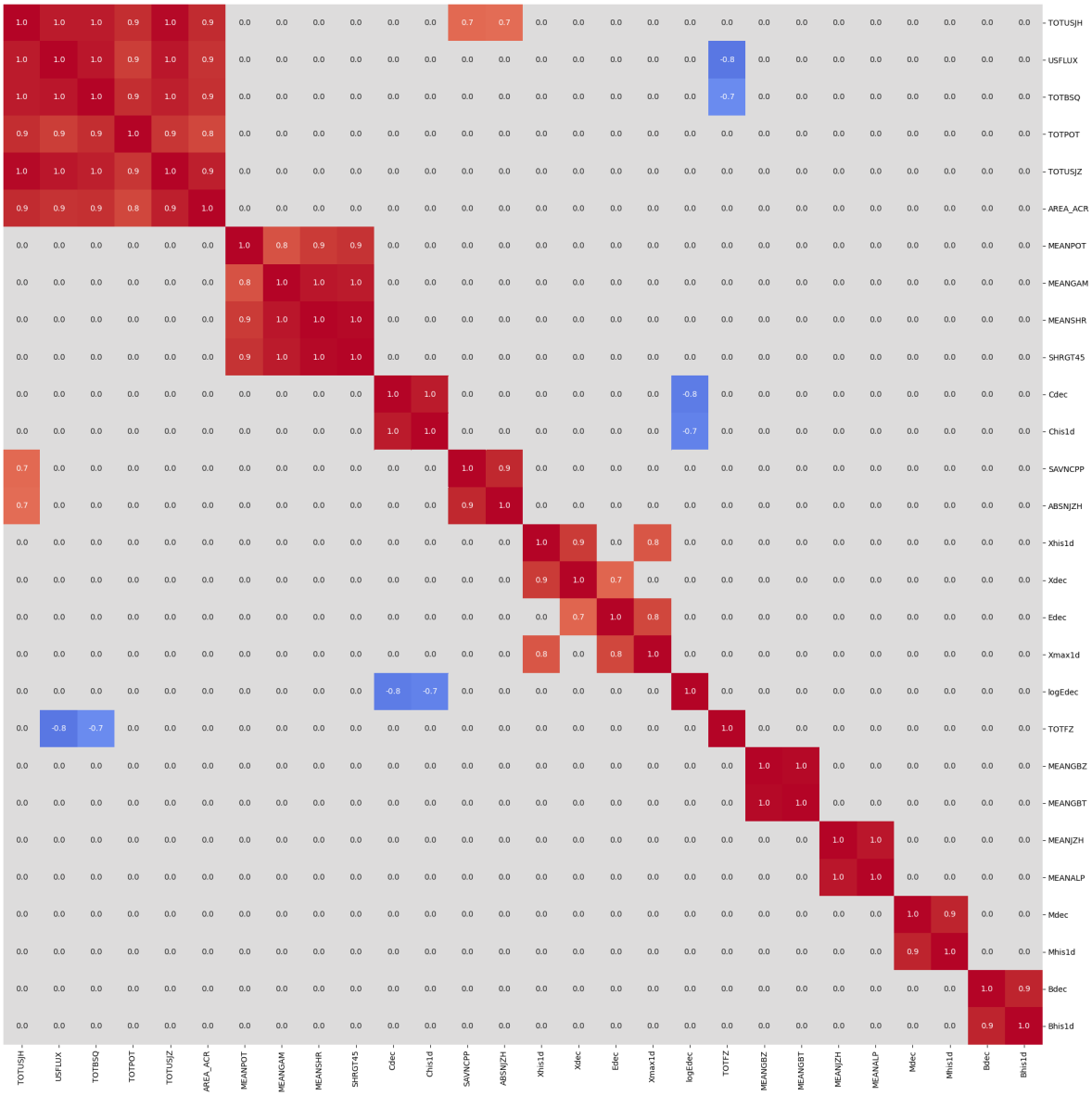


Figure 4.2: Feature correlation (Pearson) heat-map of all samples in training set. Features that are closely linearly correlated are clustered together. Only correlations larger than $\rho > \pm 0.7$ that are not only correlated with themselves are shown, for example 28 of 40 (for aesthetic purposes). Correlations are rounded up to the first decimal.

These correlation heat-maps display the mutual correlation between features, not correlation to flare occurrence. The one-day history and decay features would correlate according to mathematical definition. Various other pairs of parameters share high degree of correlation due to similar physical attributes quantified by both parameters. The parameters

measuring total flux, helicity, energy, etc. all share high correlation. Similarly, parameters quantifying mean angles and gradients of magnetic field are also correlated.

Features that correlate larger than 0.95 essentially carry the same information, therefore it would be possible to keep only one of these features per cluster and the prediction model should still perform the same. Doing this, the number of features can be reduced from 40 to 32, which could significantly speed up the training process as well (see supplemental Figure A.2 in Appendix A.3).

When we observe the correlation for the samples 24 hours leading up to the $\geq M5.0$ class flares (see the supplemental Figures A.3 and A.4 in Appendix A.3); we see that R_VALUE starts strongly correlating with the first cluster with $TOTPOT$, and $MEANPOT$ with $MEANSHR$ and $SHRGT45$. This suggests that there are pre-flaring mechanisms driving these changes.

4.4 Comparison of scaling strategies

Liu et al. [4] scaled their dataset with z-normalisation (z-norm, Eq. 3.7) on the physical features and min-max scaling (Eq. 3.8) on their history features. They state that two different strategies are required because the values between physical features and history features vary in orders of magnitude. They also scaled and transformed on the entire dataset (as confirmed in Section 4.4.1). This could bias the model to generalise better *on the specific test set*, and is considered bad practice in the machine learning community. It is better practice to scale the data only on the training set, and transform the validation and test set based on those scaler properties. By using z-norm, all the features are scaled to a mean of zero and a unit variance.

In the rest of this section, we investigate the differences between scaling strategies with the help of box-plots to visualise the distributions of the features and draw likely conclusions.

4.4.1 Confirmation of scaling used

If Liu et al. [4] scaled on the entire dataset, features where min-max scaling was applied would have exactly a minimum of zero and a maximum of one. We examined all the features and show *Cdec* as a test case in Table 4.2. Because the minimum and maximum is precisely zero and one, confirms that the scaling was performed on the entire dataset.

Table 4.2: *Cdec* feature statistics

	Cdec - Entire dataset	*Cdec* - Train set
count	155 739	84 577
mean	0.0269	0.0251
std	0.0729	0.0697
min	0	0
max	1.0000	0.9643

4.4.2 Distribution of scaling strategies

There are two reasons why we want to use a different scaling method from [4]: As we are interested in interpretability, all the features should be treated the same so that the importance of features is not influenced by the different scaling strategies. Secondly, scaling parameters are calculated only on the training set. When data from the validation and test sets are scaled, the parameters calculated from the training set are used, so that the validation and test data are not biased.

The three scaling strategies in question are:

- Liu et al.'s scaling strategy. Physical features are scaled with z-norm. History features are scaled with min-max scaling. Scaled on the entire dataset. (*z_minmax_all*)
- Physical features are scaled with z-norm. History features are scaled with min-max scaling. Scaled on the training set. (*z_minmax_train*)
- All features are scaled with z-norm. Scaled on the training set. (*z_train*)

The raw dataset is transformed with these three scaling strategies and a box-plot is drawn for each strategy, to visualise the differences in feature distributions (see Figures 4.3, 4.4 and 4.5). All plots are shown only on the training set data.

Each box is drawn with the box edges being Q1 and Q3, the 25th percentile and 75th percentile, respectively. The median is the solid line within the box (green) and the mean is the dotted line (red). The minimum whisker is Q1 - 1.5*IQR and the maximum whisker is Q3 + 1.5*IQR, where IQR = Q3-Q1. The outliers are not shown. The magenta line divides the physical features (left) from the history features (right).

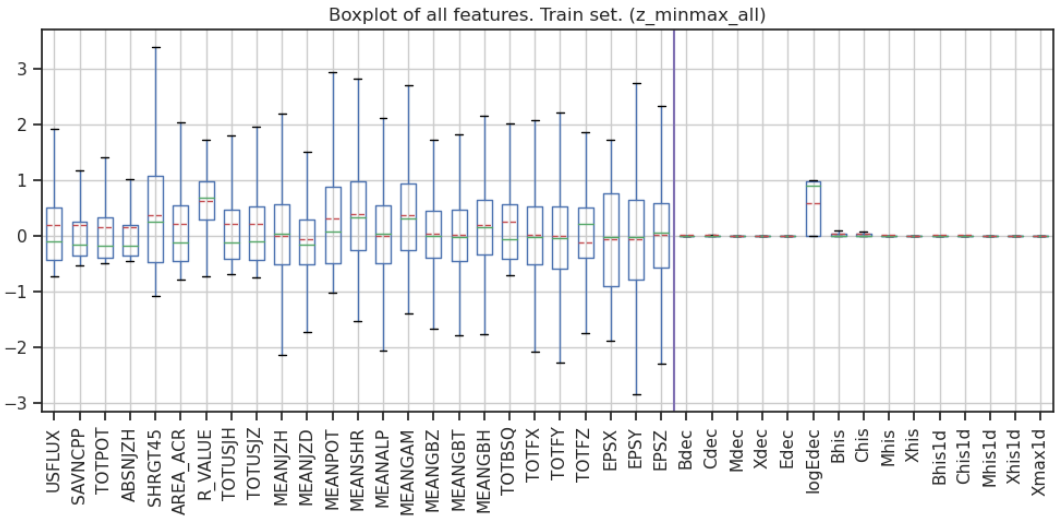


Figure 4.3: Box-plot of features with z-norm on physical features and min-max scaling on history features. Scaled on entire dataset (*z_minmax_all*). The median is the solid green line. The mean is the dotted red line. Outliers are not shown. The magenta line divides physical and history features.

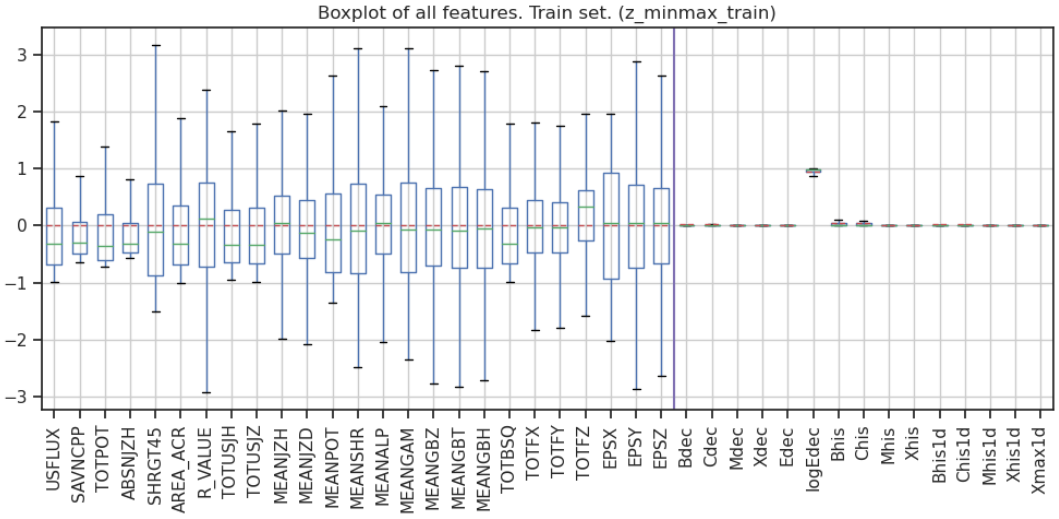


Figure 4.4: Box-plot of features with z-norm on physical features and min-max scaling on history features. Scaled on training set (*z_minmax_train*). Same format as Figure 4.3.

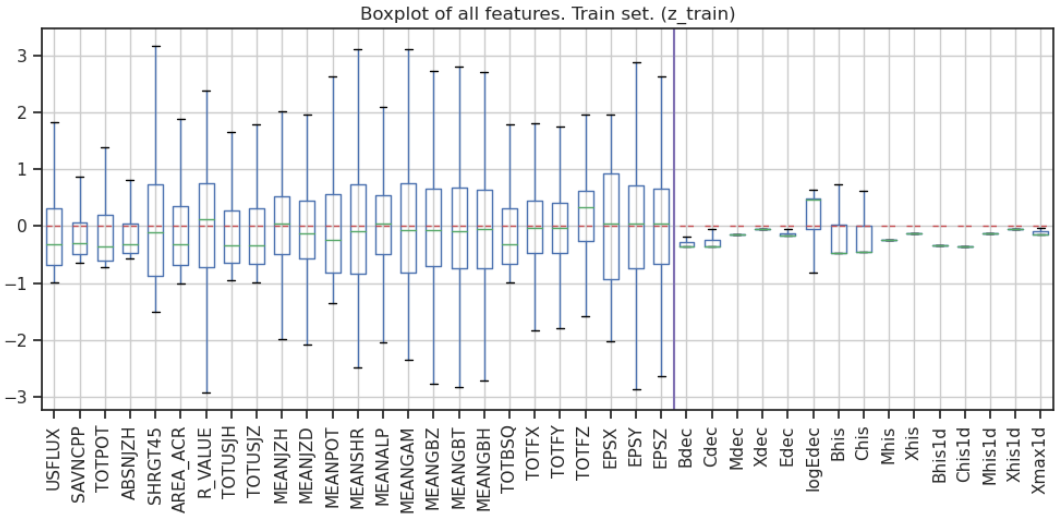


Figure 4.5: Box-plot of features with z-norm on all features. Scaled on train dataset (*z_train*). Same format as Figure 4.3.

Min-max scaling on the history features (*z_minmax_train*) is mostly zero compared to only z-norm (*z_train*). Having mostly zero values using min-max scaling for a feature could potentially have some merit, by overfitting sooner, but did not yield the higher scores (see Section 5.7).

The *z_train* strategy is chosen for the remainder of the research, so as to not bias the model and keep the importance of features the same.

4.5 Predictive capacity of features

From Section 4.2, we saw that features are not all normally distributed and sometimes bi-modal, eliminating parametric methods for determining predictive capability. Since we have numerical input features with a categorical output, a viable option would be to use MI to rank our features according to how well they estimate the target value.

MI, also known as information gain [90], measures the mutual dependence between two variables. It is zero when the two variables are independent, and higher for a larger dependency. Originally it was defined in terms of discrete random variables but has since been extended to continuous variables [91]. We calculate the MI using the k-nearest neighbours entropy estimation method as described in [92], [93], and implemented in *sklearn* [94].

We determine the MI for each feature with respect to the target value, on the training set, using a nearest neighbours value of 6, shown in Figure 4.6. We sorted the features according to the importance found by [4], which is similar to that found by [43], [56]. Both scaling strategies (*z_train* and *z_minmax_train* explained in Section 4.4) are shown.

The discrepancies between the history feature importance for *z_train* (blue) and *z_minmax_train* (orange) are due to the k-nearest neighbours estimation method. The distribution of the data for both scaling strategies are the same, and the scaling strategies only scale the data linearly, therefore the MI is not expected to change significantly.

The 16 features that have high importance, is similar to what [4] found, although not exactly in the same order. This is a good starting point, but not enough convincing evidence to explain the predictive capability of each feature. In Chapter 7, we extend our investigation to attribution methods.

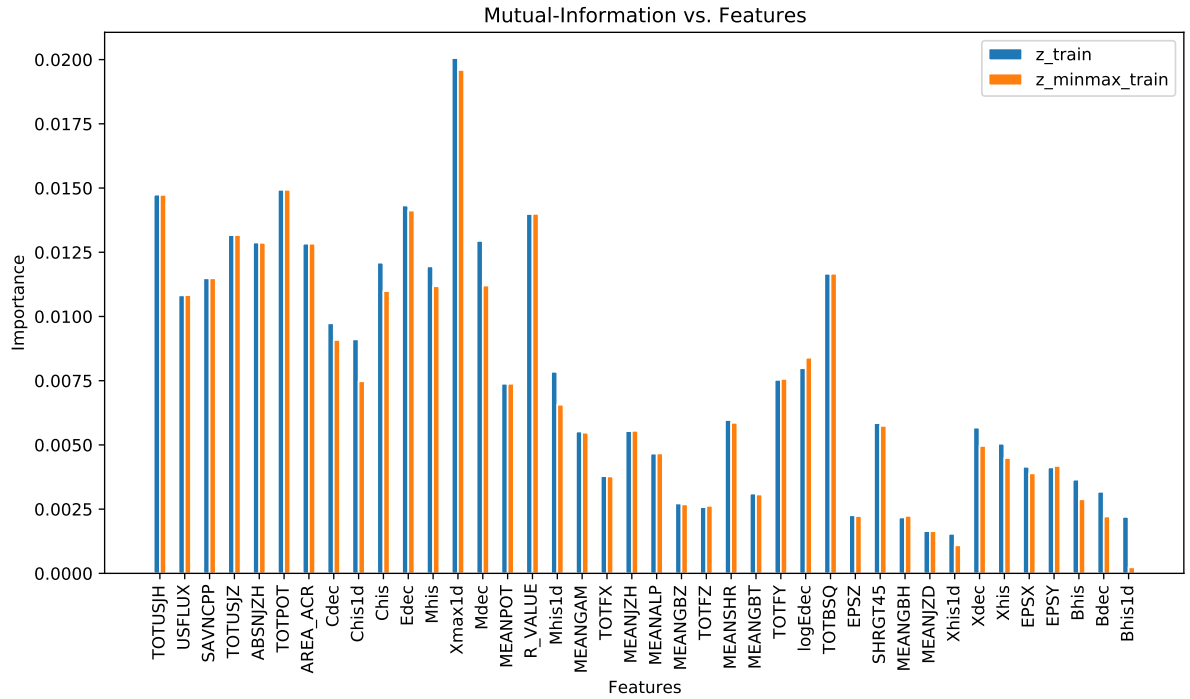


Figure 4.6: MI of input features with respect to target features. Sorted most to least important according to Liu et al.'s [4] feature importance. Nearest neighbours of 6.

4.6 Conclusion

Feature correlations exposed certain clusters of features that strongly correlate linearly. Several features in a cluster share high degree of correlation due to similar physical attributes quantified by both parameters. The parameters measuring total flux, helicity, energy, etc. all share high correlation. Similarly, parameters quantifying mean angles and gradients of magnetic field are also correlated. Some features could potentially be removed by keeping only one feature per cluster, while theoretically retaining the same information.

Histograms confirmed that not all the features in the dataset are normally distributed, allowing only non-parametric ranking of features, such as MI. The MI analysis revealed that the scaling strategy does affect the importance of features, and also regarded the top 16 features of [4] as relevant for flare prediction. MI is certainly not exhaustive enough

for determining feature importance and alternative methods are tested in Chapter 7.

Different scaling strategies were introduced, of which we only use the *z_train* in our research, unless stated otherwise. *Z_train* is z-normalisation (Eq. 3.7) fitting only on the training set, which does not bias the validation and test set and treats all features equally.

Chapter 5

Multilayer Perceptron baseline

In this chapter we create and optimise a baseline model for flare prediction using MLPs. We also evaluate the one-cycle learning rate (OCLR) policy and compare scaling strategies.

5.1 Introduction

"Onions have layers... Ogres have layers..." - Shrek

We aim to establish a baseline with the MLP, since it is the simplest form of DNN, for predicting flares larger than class $M5$, based on research previously done [2], [4], [49]. Different optimisation techniques are evaluated in an attempt to facilitate the optimisation process, as optimisation is a very time-consuming process. We also investigate the effect of the different scaling strategies on performance (see scaling strategies in Section 4.4).

First, the MLP architecture is introduced and its functionality is explained in Section 5.2, then the MLP used by [4] is replicated as closely as possible and optimised in Section 5.3. We then simplify and optimise the model while retaining its predictive capacity in Section 5.4. In Section 5.5 the OCLR policy is studied to see if it can eliminate the strenuousness

of model optimisation while also boosting performance. All the models are evaluated and compared in Section 5.6. Lastly, in Section 5.7, we inspect and compare the effect of the different scaling strategies on model performance.

5.2 MLP architecture

MLPs, also known as fully connected feedforward networks, are the most basic deep learning architectures. They learn parameters $\boldsymbol{\theta}$ that best approximate the function $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ [36]. An example of these networks can be seen in Figure 5.1.

Learning in supervised DNNs takes place via (stochastic) gradient descent, by trying to find the minima in the loss landscape. The networks attempt to find the minima by doing forward- and backward propagation. Forward propagation entails multiplying the input node values or features with the weights, passing the sum through some activation function, all the way until the final output layer. With supervised learning, the error between the predicted and true output values is calculated with a loss function. The backward propagation algorithm efficiently determines the gradient of the loss function with regard to the parameters being optimised. The weights are updated using this gradient, and this process is repeated until the network has effectively minimised the loss. Once the entire dataset has been processed in this manner it counts as one *epoch* [36].

Next, we explain each of the sub-components in more detail.

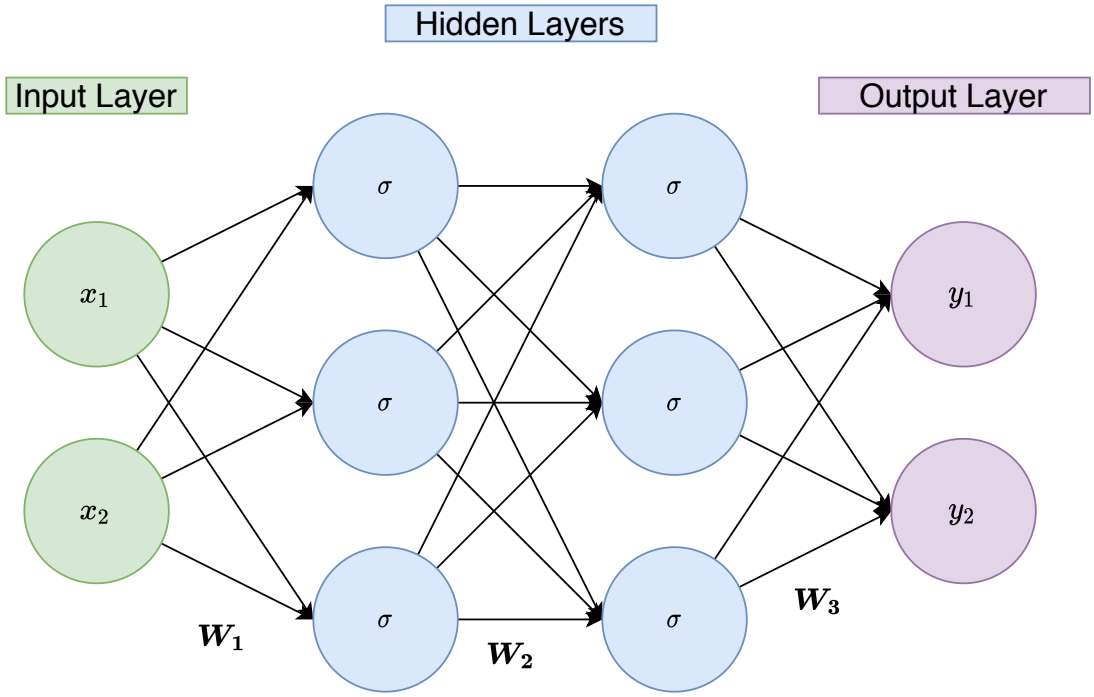


Figure 5.1: MLP illustration: Vector $\boldsymbol{x}$ as input nodes, $\boldsymbol{W_i}$ as the weights between nodes, σ as the activation function at the hidden nodes and $\boldsymbol{y}$ as the output nodes. The number of nodes and hidden layers can vary.

Activation functions

On each of the hidden layers' nodes, an activation function σ is applied on the linear combination of weights and values from the previous layer. Non-linear activation functions are used, typically rectified linear unit (ReLU) [95], which is a piece-wise linear function that zeros values smaller than zero, and passes through values larger than one, defined as $y = max(0, x)$.

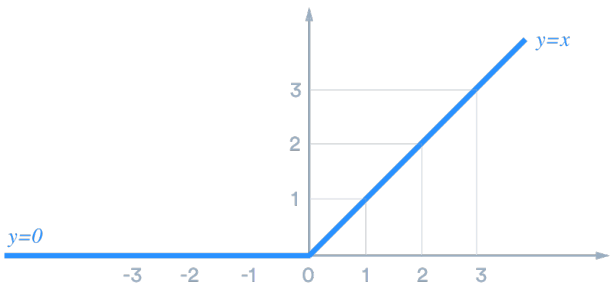


Figure 5.2: ReLU activation function.

Loss functions

For binary classification problems such as ours, cross-entropy is the most common loss function [36]. Cross-entropy loss, or log loss, measures the difference between two probability distributions. When dealing with unbalanced data, one can also specify a weight connected to the loss function as described in [96],

$$J_{wcce} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k \times y_m^k \times \log\left(h_\theta\left(x_m, k\right)\right) \tag{5.1}$$

where M is the number of training examples, K is the number of classes, w_k is the weight for class k. y_m^k is the target label for training example m for class k. x_m is the input for training example m. h_θ is the model with neural network weights θ. Parameter J_{wcce} is the weighted categorical cross-entropy loss. This assumes that there are a output node per class.

Optimisers and Optimisation

Optimisers are responsible for minimising the loss function and updating the weights; the most bare-boned of these is stochastic gradient descent (SGD). With classical gradient descent each parameter is updated according to the gradient of the loss function with respect to the parameters θ for the entire dataset. Mini-batch SGD increases the computational efficiency by updating the parameters for n number of random samples in each mini-batch. Mini-batch size is referred to as batch size throughout the rest of this dissertation. SGD is formally defined in [97] as,

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{5.2}$$

where $\nabla_\theta J$ is the gradient of the loss function with respect to the parameters θ, for some learning rate (LR) of η, and with n the number of training samples in batch $i : i + n$.

Another popular variation of SGD uses the addition of Nesterov momentum. Nesterov momentum introduces some velocity component v, that assists in avoiding local minima

as well as speeding up in directions of strong improvement. According to [98], the update rule is changed such that,

$$\begin{aligned} v_{t+1} &= \gamma v_t - \eta \nabla_\theta J\left(\theta_t + \gamma v_t\right) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{5.3}$$

with γ as the momentum term. Parameters $x^{(i:i+n)}$ and $y^{(i:i+n)}$ left out for clarity.

The parameters θ are the weight values for the connections between nodes that are updated by the optimiser during training. The HP are the parameters than can be changed outside of the learning algorithm to change its performance. Optimisation of neural networks entails the tuning of HPs, for example, number of layers, number of nodes, mini-batch size, LRs, etc. to achieve optimal performance. There is a very strong interplay between HPs, especially batch size and LR. This is still an active area of research [99], [100].

Regularisation

Regularisation is defined as any modification made to the learning algorithm that restricts the training process to boost generalisation [36]. Overfitting happens when the model learns to fit the training data too well, and when unseen data are presented, the model performs poorly, hence, the need for splitting data into training, validation and test sets.

Dropout [101], is one of the most popular regularisation techniques. It works by "dropping out" random hidden layer nodes during training, which hurts the training process but reduces overfitting, and can in turn improve the model's generalisation ability.

Weight decay [102] is another regularisation technique that has been shown to improve generalisation by suppressing noise on the targets. It applies a regularising term to the loss function, to bring the weights closer to the origin.

Batch normalisation (BatchNorm) [103], normalises layer inputs as each batch comes in

such that,

$$\hat{x}^d = \frac{x^d - E\left[x^d\right]}{\sqrt{\text{Var}\left[x^d\right]}} \tag{5.4}$$

where E is the mean and Var is the variance along dimension d for samples x in the batch. BatchNorm allows the use of larger batch sizes and higher LRs [104].

There are some concerns regarding the order of BatchNorm and dropout, regarding which should precede which. This is still an active area of research, but we placed our dropout after BatchNorm as suggested by [105], otherwise it could cause more instabilities.

5.3 Optimising Liu at al.'s MLP with static LR

Liu et al. [4] achieved a test TSS of 0.845 with their MLP on the $\geq M5.0$-class problem (see Table 2.2). They did not report on standard error, so we do not know whether their methods were tested across multiple seeds. Standard error (SE) measures the uncertainty around the estimate of the mean, given as $SE = \sigma/\sqrt{n}$, where σ is the standard deviation and n is the number of samples (or seeds) [106].

Their MLP architecture consisted of two layers, with 200 and 500 nodes respectively, with all 40 features as inputs and two nodes as output. All other HPs used in their MLP are unknown.

We aim to optimise and retest their MLP architecture to see if we can obtain similar performance across multiple seeds, using the *z_train* scaling strategy, not to bias our data or change the feature importance (refer to section 4.4 on scaling strategies).

5.3.1 Model setup

We create an MLP with 40 input nodes, two hidden layers, each with 500 nodes and an output layer with two nodes, denoted as 40_500_500_2 or simply 2_500. The following setup is kept the same throughout all our MLP experiments unless specifically stated otherwise. Hidden layers have ReLU activation functions [95]. Weights are initialised using Kaiming initialisation, which has proven to improve performance when using ReLU activation functions [107]. Seeds describe the pseudo-random initialisation of weights before training commences. No bias nodes are included. Batch normalisation with an affine transform is used to stabilise training [103]. Weighted cross-entropy is used as loss function, since the data are extremely unbalanced. SGD with Nesterov momentum of 0.9 is implemented as optimiser [98].

We optimise for TSS with early stopping. Early stopping entails stopping training if the model has not improved on the desired metric (training loss in our case) in the last x epochs, called patience (40 in our case), and selects the model at the epoch with the best validation TSS. This allows the model to train until the training loss converges and uses the model at the epoch that has the best validation TSS.

The elements that can vary per experiment are: the number of layers and nodes, batch size, dropout, weight decay, seed, LR and scaling strategy.

All models are built with the Pytorch library [108], and results logged using "weights and biases" [109].

5.3.2 Static LR optimisation procedure

Optimisation is performed by searching over a grid of possible HPs with static LRs (does not change during training), where each parameter is trained in combination with all the other parameters. After a region is found to have good performance, the grid is refined to find the optimal HPs, by extending edge cases and searching among good HPs.

Three grid searches are performed to find the optimal model, each with a goal of:

- Finding an appropriate batch size, that yields stable and high validation TSS (grid search 1 - GS1).
- Adding regularisation to stabilise training even more, since larger batch size are still somewhat unstable (grid search 2 - GS2).
- Refining regularisation and LRs (grid search 3 - GS3).

The HP we search over for each grid-search is in Table 5.1.

Table 5.1: Experimental setup for MLP architecture (2_500)

Hyperparameters	GS1: No regularisation	GS2: Adding dropout and weight decay	GS3: Refining
Goal	*Finding a reliable batch size*	*Adding regularisation*	*Refine dropout and LR*
Architecture	40_500_500_2	40_500_500_2	40_500_500_2
Seeds (random)	15, 49, 124	15, 49, 124	15, 49, 124
Scaling strategy	z_train	z_train	z_train
Batch size	256, 1 024, 8 192, 65 536	65 536	65 536
Learning rates	0.0001 - 0.1	0.001 - 0.3	0.14 - 0.6
Dropout	0	0, 0.4, 0.8	0.8, 0.9, 0.95, 0.98
Weight decay	0	0, 0.001, 0.01	0

5.3.3 Static LR optimisation results

The MLP is trained over GS1 in Table 5.1 with batch sizes ranging from the typical 256 to 65 536, which is the largest that can fit into the training set and memory. Figure 5.3 presents the different batch sizes with the training, validation and test TSS versus LRs. The curves are the average TSS across three seeds with a standard error shadow. The training, validation and test TSS is evaluated at the epoch with the highest validation TSS.

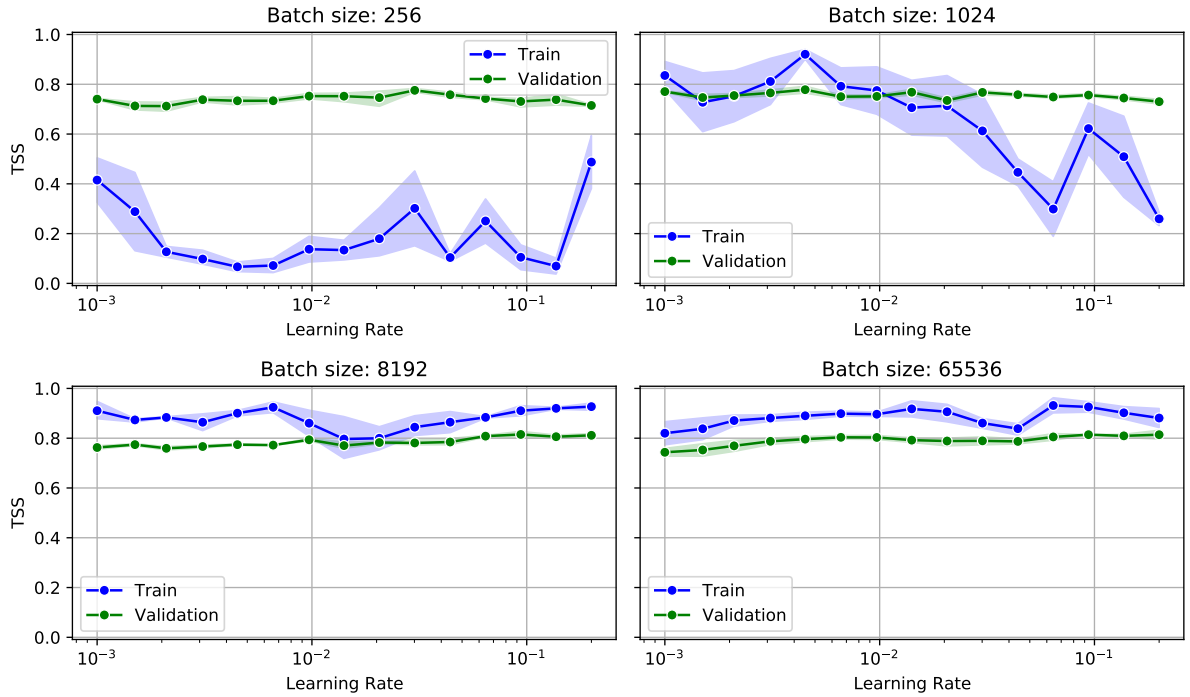


Figure 5.3: TSS versus LR plot for different batch sizes. No regularisation. 2_500 MLP architecture. Training set (blue), validation set (green).

It is clear that the smaller batch sizes are more unstable and do not reach as high validation TSS. Batch size 65 536 is selected for subsequent experiments, it is also at the upper limit of what could be practically chosen.

When studying the validation TSS training curve, smaller batch sizes exhibit much more 'spiky' behaviour than the larger batch sizes (see Figure 5.4). Adding regularisation could help smooth out the training curves.

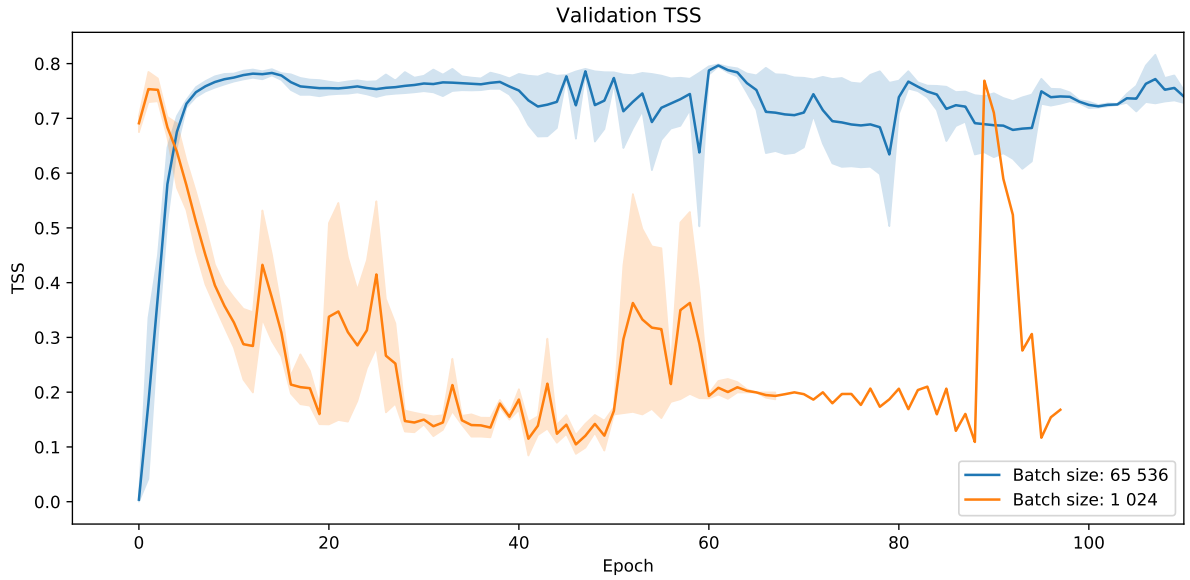


Figure 5.4: Validation TSS versus epochs between batch size 1 024 (orange) and 65 536 (blue), with LR of 0.0007 and 0.1, respectively. Best HPs with highest validation TSS shown. Batch size 1 024 reveals extremely spiky behaviour. Average across three seeds with standard error shadow.

Another grid-search (GS2) is performed over different degrees of dropout and weight decay (refer to Section 5.2), with the batch size of 65 536. A dropout of 0.4 means that 40% of the nodes in the MLP are randomly zeroed for each batch that is propagated. The TSS versus different LRs and degrees of regularisation is found in Figure A.6. The validation TSS for the model at the best learning rates can be seen in Table 5.2.

It seems that the higher the dropout, the more stable the validation TSS. Weight decay of 0.001 seems nearly identical to zero weight decay and 0.01 weight decay is only harmful

Table 5.2: Best validation TSS for different dropout and weight decay values. Mean and standard error over three seeds.

Validation TSS	Dropout: 0	Dropout: 0.4	Dropout: 0.8
Weight decay: 0	0.8237 ± 0.0041	0.8212 ± 0.0017	0.8258 ± 0.0025
Weight decay: 0.001	0.8309 ± 0.0076	0.8242 ± 0.0017	0.8287 ± 0.0029
Weight decay: 0.01	0.8289 ± 0.0047	0.8195 ± 0.0073	0.8261 ± 0.0078

to performance. Therefore, we discard weight decay, as it shows no clear advantage. The edge of the grid in the grid search gave the best scores and needs to be extended and refined to include larger dropouts and higher LRs. This bring us to GS3, with the TSS versus LR plot in Figure 5.5.

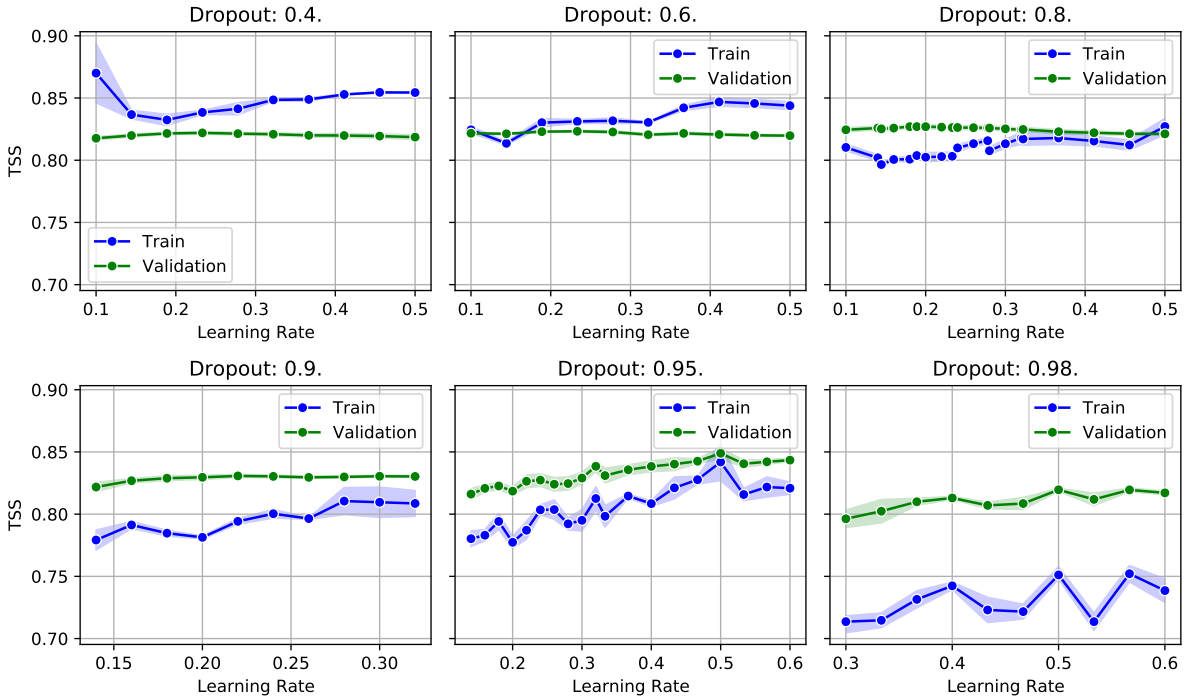


Figure 5.5: TSS versus LR plot for degrees of dropout. 2_500 MLP architecture. Training set (blue), validation set (green)

The best validation TSS achieved is with a dropout of 0.95 at a LR of 0.5. The final scores can be found in Table 5.7. Our test TSS of 0.747 is less than the 0.845 of the MLP found by [4], which we investigate further in Section 5.7. The training curve of the training and validation TSS can be seen in Figure 5.6.

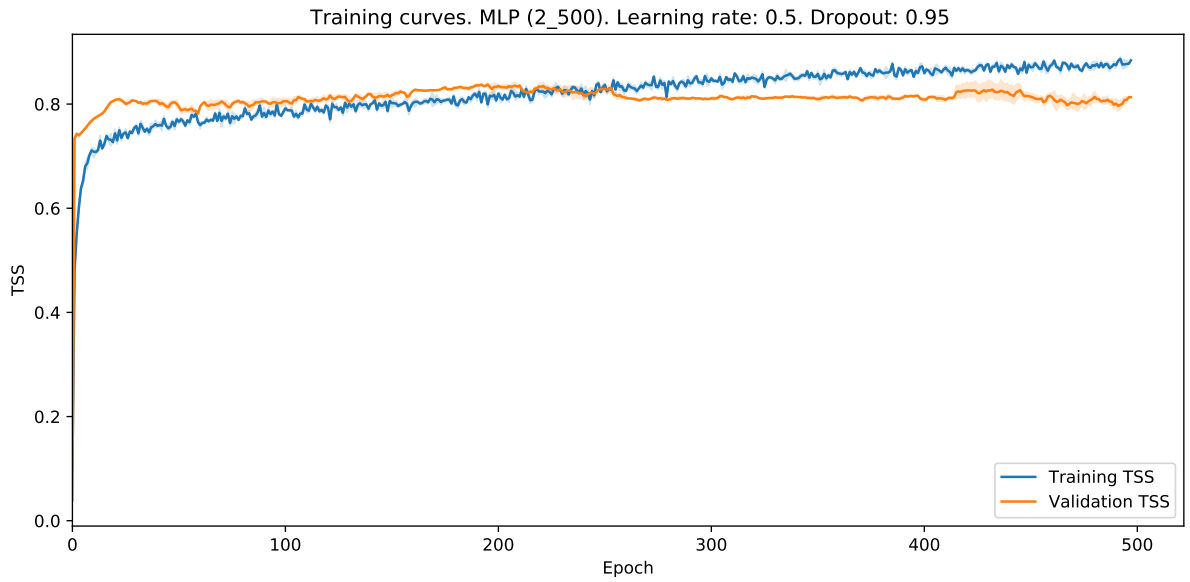


Figure 5.6: Training and validation TSS versus epochs of best 2_500 MLP. LR: 0.5. Dropout: 0.95. Average across three seeds with standard error shadow. Best model at epoch 235 with TSS of 0.8478 for seed 49.

5.4 Simplifying the MLP

In the previous section we saw that the two hidden layer with 500 nodes (2_500) architecture did very well with an extremely high degree of dropout, probably due to over-parameterisation. Therefore, in this section, we seek a single hidden layer alternative with significantly fewer parameters that can achieve similar or better performance than the larger model. Having a smaller architecture could also make interpreting it easier.

5.4.1 Static LR optimisation procedure

The model is set up exactly like the previous model in Section 5.3.1, except that the number of layers has been reduced to one and we perform a search for the number of nodes.

Two grid-searches are performed to approximate the best network by:

- Performing a rough search over a wide range of nodes, dropout and LRs, with only one seed (GS1)
- Refining the LRs across three seeds, with selected HP from the initial probe (GS2).

Table 5.3: Experimental setup for smaller MLP architecture

Hyperparameters	GS1: Rough GS	GS2: Refine LR
Goal	*Estimation of good HPs*	*Learning rate refined*
Number of nodes (one layer)	10, 25, 50, **100**	100
Seeds (random)	15	15, 49, 124
Scaling strategy	z_train	z_train
Batch size	65 536	65 536
Learning rates	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1	0.01 - 1.4
Dropout	0.4, **0.8**, 0.9, 0.95	0.8
Weight decay	0	0

5.4.2 Static LR optimisation results

The model is trained over all combinations of HP in GS1 from Table 5.3. In Table A.2 is a snippet of the combination of HP that consistently gave the best validation TSS. The best LR seems to lie between 0.01 and 1 for a dropout of 0.8 and 100 nodes. We refine the search with a finer grain of LRs, as well as across more seeds in GS2, seen in Figure 5.7.

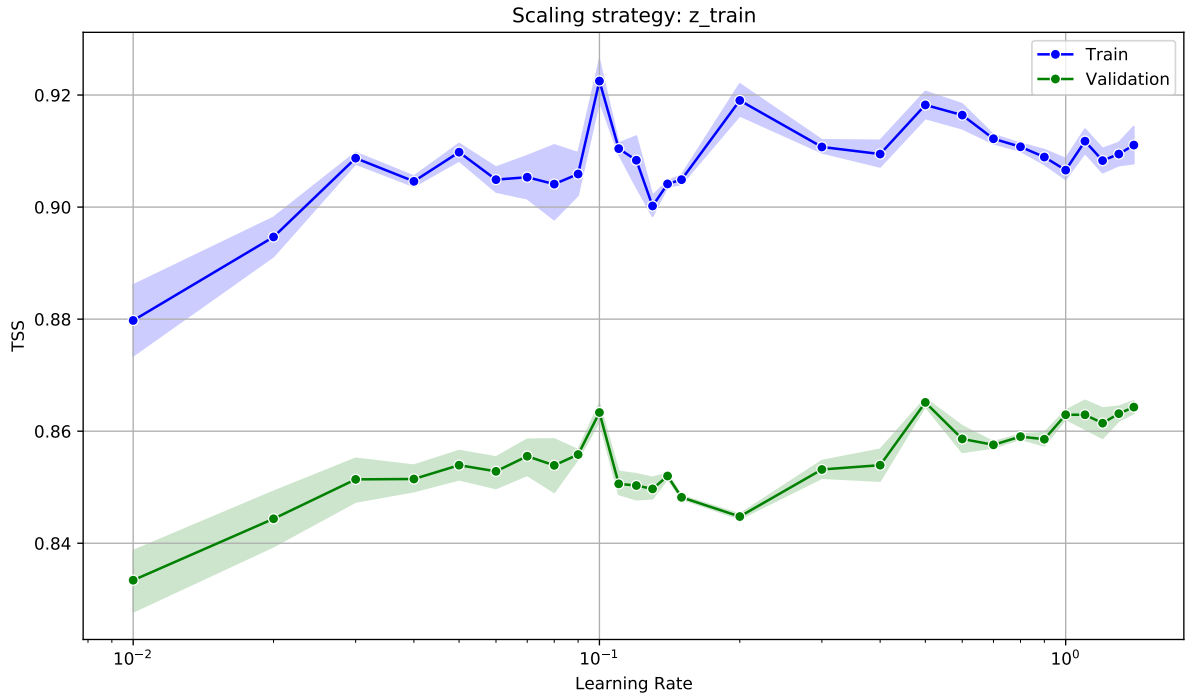


Figure 5.7: TSS versus LR plot. MLP 1_100. Dropout: 0.8. Best validation TSS of 0.8489 ± 0.0067 at a LR of 0.5

The best LR that has the highest validation TSS and smallest standard error is at 0.5. We compare this model to the previous larger model in Section 5.6, Table 5.7. The smaller (1_100) MLP is better in terms of training and validation TSS but is poorer in its generalisation ability (test TSS). Since the smaller network is better on validation TSS, we proceed with it. The training curve of the model can be seen in Figure A.15 in Appendix A.4.

5.5 Optimising the simplified MLP with OCLR

In this section we aim to optimise the architecture from the previous section by means of the OCLR policy, compare the performance and see whether the policy can reduce the effort of searching for optimal LRs to speed up the process of finding optimal HPs.

5.5.1 OCLR policy

In our problem of trying to optimise a DNN for flare prediction, the most difficult task is certainly finding the best HPs, of which the batch size and the LR are the most tedious.

Recently, a few techniques have been developed that could potentially alleviate the burden of incrementally searching for batch sizes and LRs, one of which is the OCLR policy introduced in [110]. In their research, these authors describe the phenomenon coined 'super-convergence', which is when neural networks can be trained tens of times faster than static LRs when using their LR policy. One can potentially achieve 'super-convergence' by first doing a pre-training test and then applying the OCLR scheduling technique when training. Pre-training includes the LR range test, to compute an estimate of the optimal LR for the OCLR policy. They also suggest using the largest batch size that can fit into memory. Training commences with the previously determined batch size and LRs using the OCLR scheduling. The concept of cyclical LRs such as OCLR is a combination of curriculum learning [111] and simulated annealing [112].

Learning rate range test

The LR range test is used to determine whether super-convergence is likely for a given architecture. There are two versions of this approach: (1) Smith's original [110] and (2) fast.ai's tweaked version [113]. We follow fast.ai's approach in the range test.

The LR range test is a pre-training run, starting with a very small LR close to zero that is exponentially increased while measuring the loss. This provides some insight into how well a network can be trained over the range of LRs. A typical curve (from fast.ai's approach) of the LR range test can be seen in Figure 5.8.

As the LR increases, the loss dips into a minimum and diverges again. Fast.ai suggests the LR at which the curve has the steepest descent for static LRs. For the OCLR scheduler, they suggest anywhere in the region of the minimum, even LRs much larger than at the minimum loss, but they warn that larger maximum LRs need a longer cycle [113].

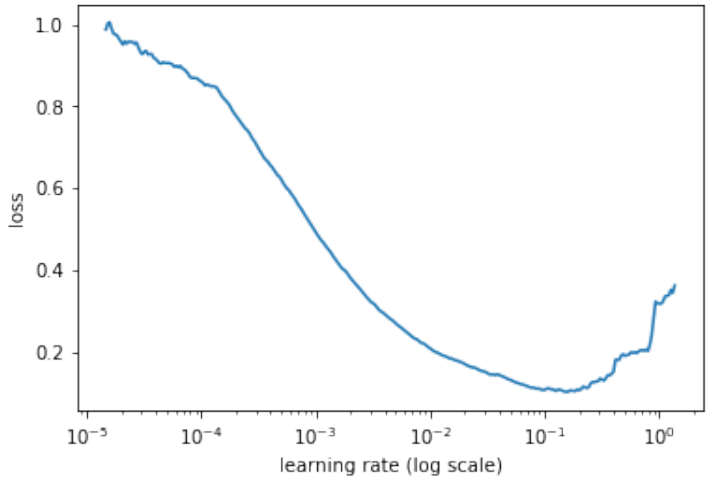


Figure 5.8: Typical range test. Geometrically spaced steps from 10^{-5} to 10^{-0}. Adapted from [113]

OCLR scheduling

In short, the OCLR scheduling anneals the LR from an initial LR (*min_lr*) up to some maximum LR (*max_lr*, warm-up phase) and then back down to the initial LR (cool-down phase) and then much lower than the initial LR (annihilation phase). This makes up one cycle (see Figure 5.9).

The *max_lr* is chosen from the range test, *min_lr* is typically a tenth of *max_lr*. Each iteration is a batch, therefore adjusting the number of epochs changes the length of a cycle. No early stopping is applied either; the model is selected at the end of training for evaluation.

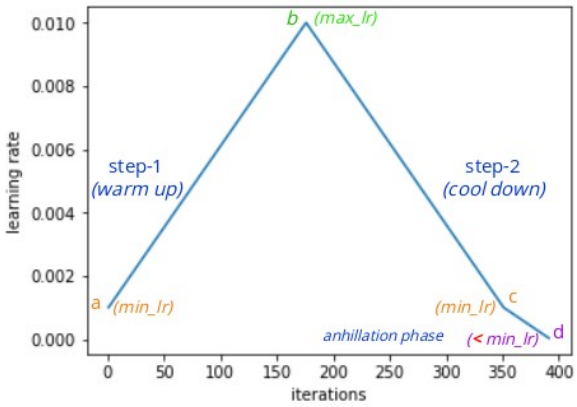


Figure 5.9: OCLR scheduling process. Each iteration is a batch.

5.5.2 OCLR optimisation procedure

Model setup

The same architecture is used as in the previous section, with a single layer and 100 nodes (1_100). The model has ReLU activation functions, with Kaiming initialisation, batch normalisation with affine transforms, weighted cross-entropy as loss function, SGD with Nesterov momentum of 0.9 optimiser, and is optimised for validation TSS.

Early stopping is not used since that would mean cutting short the full effect of the OCLR policy. Batch size is fixed to 65 536, as OCLR requires the largest size that can fit into memory.

LR range test setup

For the LR range test, the LR starts at 10^{-5} and ends at 10, with 200 geometrically spaced iterations in between, as used by fast.ai [113]. No smoothing is applied. The higher the number of iterations in the range test, the better the results, but the longer it takes to compute. 100 iterations were too few to capture the correct shape and 300 took extremely long too compute, therefore 200 was a perfect trade-off between the two (see supplemental Figure A.7 for decision).

For a range test, we observe the training and validation TSS as well as the loss versus LRs. Our decision on the maximum LR (*max_lr*) obtained from the range test is based on the validation TSS and selected at the point before rippling begins. (See section 5.5.3 for decision).

OCLR scheduling setup

For the actual OCLR scheduler the division factor is set to the default of 25, making the minimum LR (*min_lr*) equal to the maximum LR (*max_lr*) divided by the division factor. The total number of steps is equal to the maximum number of epochs times the number of batches that fit into the training set. The LR is adjusted every batch with a cosine annealing strategy, which rapidly increases or decreases the LR in curved fashion instead of a straight line as in Figure 5.9. The turning point percentage is at 30% of the total steps.

OCLR optimisation procedure

Then the models are trained utilising a grid-search to

- Find the appropriate number of epochs (which affects the number of total steps) and degree of dropout (GS1).
- Validate whether the max_lr of one was a sound assumption to make (GS2).

The HPs searched over are in Table 5.4.

Table 5.4: MLP OCLR grid searches

HP	GS1: Epochs and dropout	GS2: Max_lr
Goal	Find number of epochs and dropout	Find best max_lr
Max_lr	1	0.4, 1, 4.66
Epochs	50, 100, **200**, 400	200
Dropout	0.4, 0.8, **0.9**, 0.95	0.9
Architecture (layers_nodes)	1_100	1_100
Seed	15	15, 49, 124
Scaling strategy	z_train	z_train

5.5.3 OCLR results

Determining the best number of epochs

First, the LR range test is performed over the various degrees of dropout and plotted in Figure 5.10.

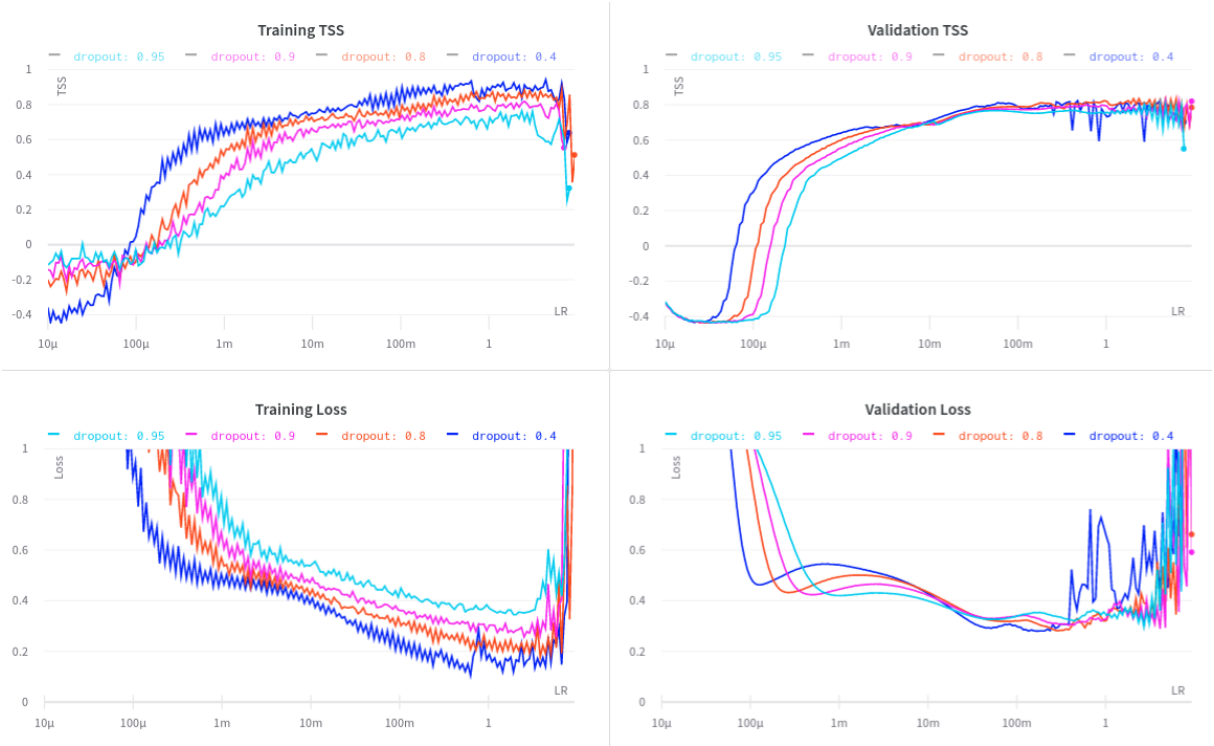


Figure 5.10: LR range test of MLP (1_100) across multiple dropout values. Batch size: 65 536. 200 logarithmically spaced iterations. Parameters μ and m are $\times 10^{-6}$ and $\times 10^{-3}$, respectively.

Larger dropouts seem to be able to utilise higher LRs before diverging. Dropouts 0.8, 0.9 and 0.95 start to ripple around LR=1 and diverge around LR=10 on validation TSS, whereas dropout=0.4 becomes unstable at about LR=100m. Fast.ai suggests choosing a *max_lr* anywhere between the minimum loss and before divergence; we chose our max_lr = 1, right before rippling begins and after the minimum loss. We evaluate this decision empirically in Section 5.5.3.

Now training can commence with OCLR scheduling using our chosen max_lr = 1 over the HPs from GS1, to obtain the results seen in Table 5.5.

Table 5.5: GS1: Finding optimal number of epochs and dropout, with max_lr = 1.

Nodes	Epochs	Dropout	Training TSS	Validation TSS
100	50	0.4	0.8420	0.7943
		0.8	0.7866	0.7931
		0.9	0.7545	0.7924
		0.95	0.7015	0.7897
	100	0.4	0.8849	0.8114
		0.8	0.8655	0.8090
		0.9	0.7849	0.7943
		0.95	0.7471	0.7900
	200	0.4	0.9658	0.6758
		0.8	0.9263	0.8175
		0.9	0.8828	**0.8292**
		0.95	0.8096	0.7993
	400	0.4	0.9816	0.5257
		0.8	0.9367	0.7897
		0.9	0.8992	0.8070
		0.95	0.8009	0.8141

From the rough search, we find that a dropout of 0.9 and 200 epochs produces the highest validation TSS. Next, we will test across multiple seeds.

Obtaining the best maximum LR (*max_lr*)

To see whether a *max_lr* of one is the best choice for the selected HPs, we train the model across three seeds, using OCLR scheduling, for three *max_lr* choices,

- Minimum validation loss before rippling begins (0.4).
- Maximum validation TSS before rippling (1).
- Maximum validation TSS after rippling (4.66).

The LR range test for the model is depicted in Figure 5.11.

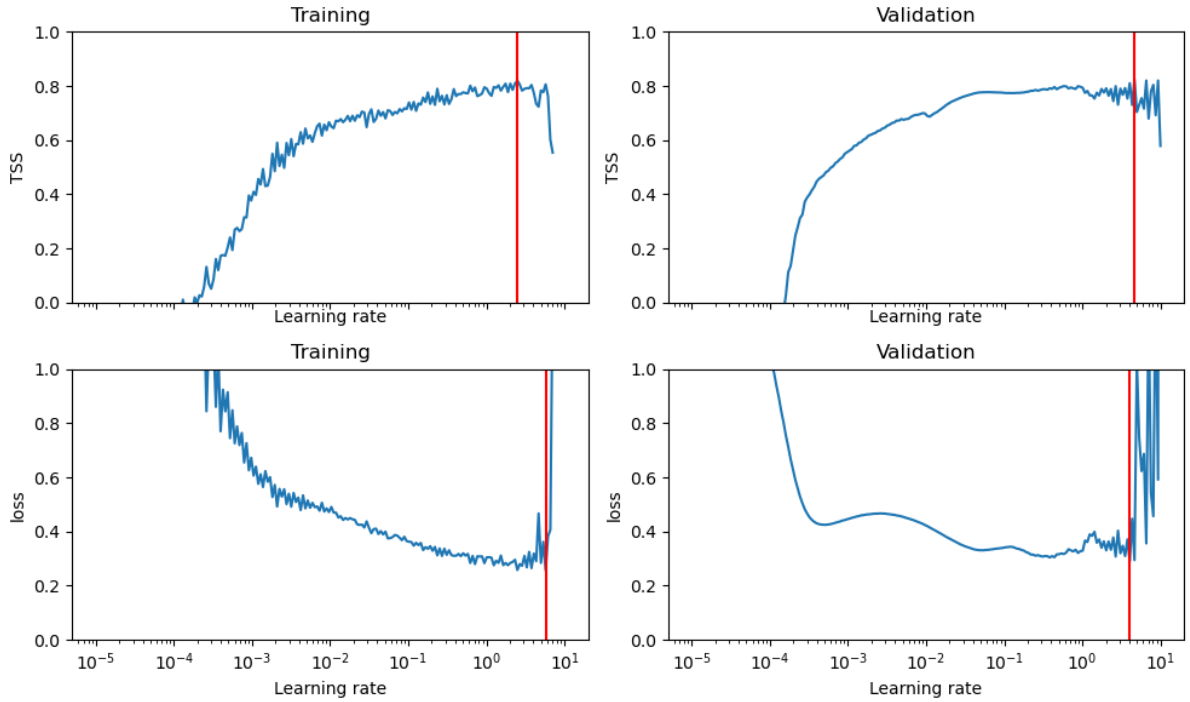


Figure 5.11: LR range test for best OCLR model. Dropout: 0.9. Epochs: 200. MLP (1_100)

The results from GS2, for the different *max_lr* values are found in Table 5.6.

Table 5.6: Max_lr TSS comparison

Max LR	Train TSS	Validation TSS	Test TSS
0.4	0.8690 ± 0.0091	0.8275 ± 0.0044	0.7852 ± 0.0025
1	0.8770 ± 0.0053	**0.8317 ± 0.0029**	0.7673 ± 0.0106
4.66	0.8535 ± 0.0091	0.8171 ± 0.0034	0.7572 ± 0.0116

In terms of the validation TSS a *max_lr*=1 did best but by a small margin compared to the smaller *max_lr* of 0.4. The larger *max_lr* of 4.66 did experience more adverse effects, which is understandable, since the rippling could cause more instability. Fast.ai mentioned that networks are not too sensitive to the choice of the maximum LR [114].

The best model's training curves are visualised in Figure A.16 in Appendix A.4.

5.6 Model evaluation and comparison

For this section, we compare our three different MLP models by means of performance metrics and other helpful diagrams that aid in model evaluation. The binary classification models are also evaluated as probabilistic forecasting models on selected ARs.

5.6.1 Performance comparison

In Table 5.7 below we calculate the average recall, precision, BACC, HSS and TSS across three seeds with standard error, which are similar to the metrics used to evaluated flare prediction performance by [2], [4], [45], [49], [55], [115]. We calculate the performance for the large and small MLP optimised with static LRs and our small MLP optimised using the OCLR policy.

The 1_100 static MLP did best for all the metrics on the validation set but produced lower recall, BACC and TSS on the test set. All our models have higher precision and HSS than the MLP of [4], but are worse on recall and TSS. (This is explored further in Section 5.7). The smaller architecture optimised with the OCLR policy is less prone to overfitting and has better generalisation, as expected by [110]. When optimising with the OCLR policy, the smaller MLP did slightly worse on the validation set with a TSS of 0.8317 ± 0.0029 but on par with the larger architecture with a test TSS of 0.7673 ± 0.0106. We did not benefit from any super-convergence.

Table 5.7: MLP performance comparison. Average and standard error over three seeds.

Metric	Model	Training	Validation	Test
Recall	Static (2_500)	0.9289 ± 0.0253	0.9281 ± 0.0034	0.7815 ± 0.0018
	Static (1_100)	0.9921 ± 0.0009	0.9281 ± 0.0000	0.6630 ± 0.0491
	OCLR (1_100)	0.9789 ± 0.0032	0.9292 ± 0.0023	0.8111 ± 0.0111
	MLP ([4])	-	-	0.944
Precision	Static (2_500)	0.1464 ± 0.0108	0.1162 ± 0.0062	0.0843 ± 0.0037
	Static (1_100)	0.2095 ± 0.0118	0.1413 ± 0.0033	0.1053 ± 0.0005
	OCLR (1_100)	0.1260 ± 0.0040	0.0962 ± 0.0022	0.0697 ± 0.0008
	MLP ([4])	-	-	0.037
BACC	Static (2_500)	0.9438 ± 0.0138	0.9245 ± 0.0034	0.8735 ± 0.0015
	Static (1_100)	0.9818 ± 0.0014	0.9326 ± 0.0008	0.8201 ± 0.0237
	OCLR (1_100)	0.9638 ± 0.0023	0.9159 ± 0.0015	0.8837 ± 0.0053
	MLP ([4])	-	-	0.922
HSS	Static (2_500)	0.2429 ± 0.0173	0.1906 ± 0.0101	0.1460 ± 0.0062
	Static (1_100)	0.3374 ± 0.0163	0.2306 ± 0.0050	0.1757 ± 0.0018
	OCLR (1_100)	0.2127 ± 0.0065	0.1574 ± 0.0036	0.1219 ± 0.0014
	MLP ([4])	-	-	0.064
TSS	Static (2_500)	0.8878 ± 0.0276	0.8489 ± 0.0067	0.7470 ± 0.0030
	Static (1_100)	0.9636 ± 0.0028	0.8651 ± 0.0017	0.6402 ± 0.0474
	OCLR (1_100)	0.9276 ± 0.0047	0.8317 ± 0.0029	0.7673 ± 0.0106
	MLP ([4])	-	-	0.845

5.6.2 Evaluation plots

For our three MLP models, we draw the following three plots, which we call evaluation plots, on the test set (train and validation in Appendix A.4):

a) Skill score profile (SSP) over the range of probability thresholds. Before the evaluation plot is drawn, the model outputs are passed through a soft-max layer, squeezing the outputs between zero and one. This creates a probabilistic forecasting model, at which a probability threshold can be selected to predict either a binary zero or one.

b) Receiver operating characteristic (ROC) curve. The ROC curve illustrates the relationship between the TP rate and the FP rate, whereas the area under curve (AUC) gives a measure of how well the model can differentiate between two classes, with a maximum of one [116].

c) The reliability diagram is often used in climate forecasting to explain the observed frequency of a flare (blue) against the predicted probability of a flare [117]. Note that this is a fraction matching the left-most y axis. A histogram (red) is used to show the sample count per bin, irrespective of actual flare status. The histogram matches the right-most y axis. Plots below the perfectly calibrated dotted line in the reliability diagram imply over-forecasting [71]. The climatological event rate is the probability of an event occurring, which is set to the fraction of imbalance on our training set, which is 0.75% (see Table 3.3). This is used to calculate the BSS, which is also indicated in the label.

These evaluation plots were also used by [4], [49], [71] for their flare prediction models. The evaluation plots for the large (2_500), simplified (1_100) and simplified OCLR networks, on the test set, are shown in Figures 5.12, 5.13 and 5.14, respectively. The plots on the training and validation set can be found in Appendix A.4.

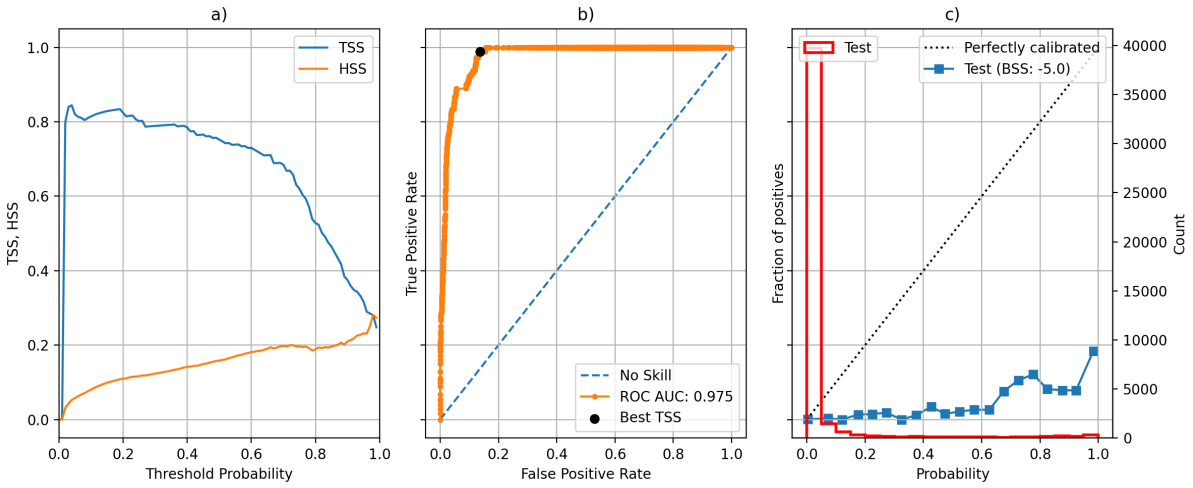


Figure 5.12: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP 2_500.

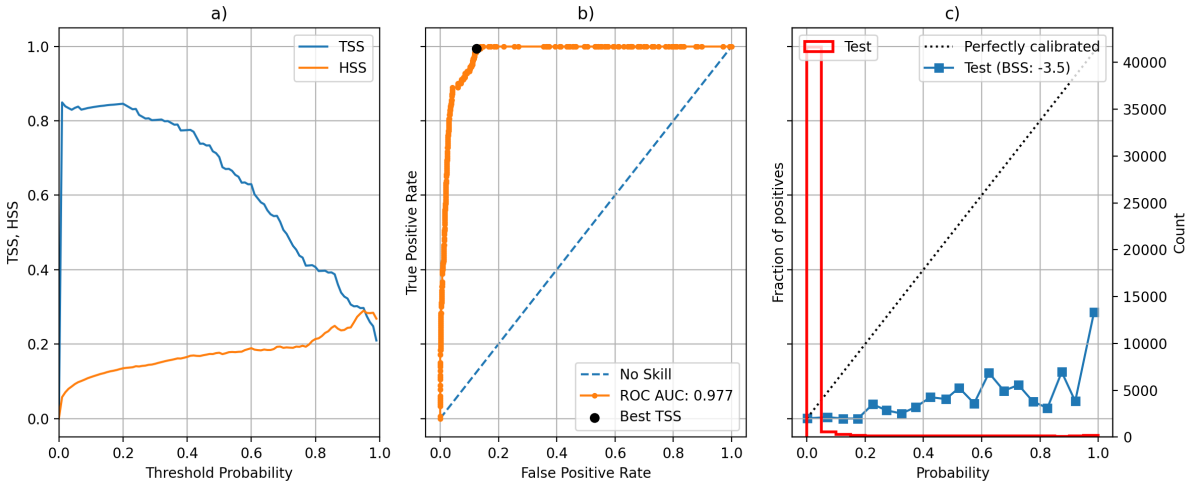


Figure 5.13: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP 1_100.

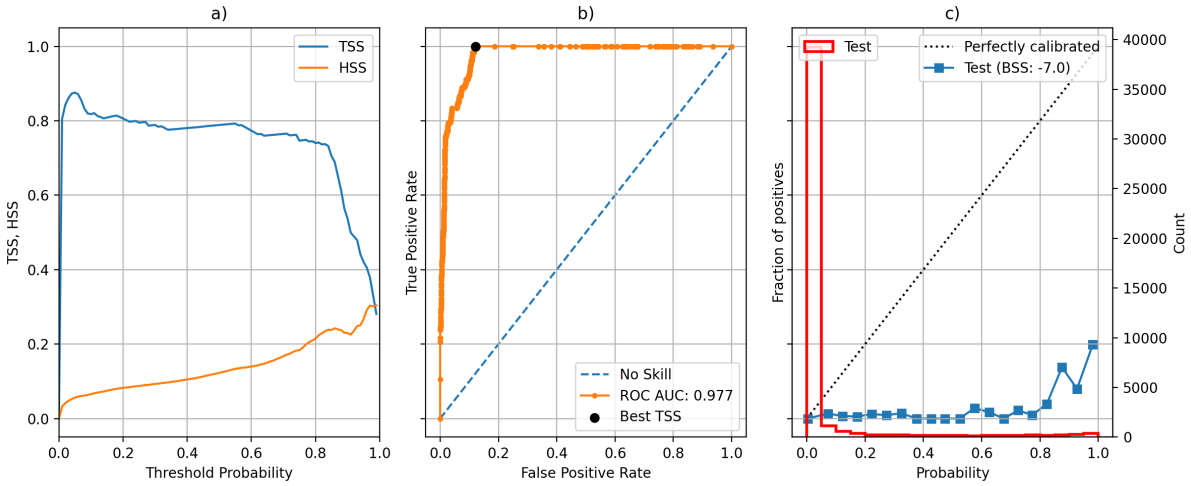


Figure 5.14: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP 1_100; OCLR.

According to the SSP plots, the MLP trained with the OCLR policy had a higher test TSS at the threshold of 0.5 with a much more constant TSS across the different probabilities; this could indicate that the OCLR policy did have some regularising effect that improved generalisation, as mentioned by [110]. We also observe TSS peaking with every model at the probability threshold around 0.007, which is close to the climatological event rate. This is consistent with other flare prediction research [44], [55], [115].

For the ROC curves, all three models have a nearly identical AUC, which is able to

separate the classes really well; however, ROC curves are deceiving when working with imbalanced data. The alternative is precision-recall curves [118] (see Appendix A.4, Figure A.14, for a precision-recall curve with static MLP (1_100) on the test set).

All our MLP models are poorly calibrated, as seen in the reliability diagrams, with curves positioned far below the perfectly calibrated line, thus indicating that our models are over-predicting that flares occur. It should also be noted that a larger BSS does not correspond with a more calibrated reliability diagram [71]. Reliability diagrams become more important when talking about the practical implementation of such a model, which we elaborate on in Chapter 8, but are not as crucial when considering the quality of a binary classifier.

5.6.3 Error analysis

We perform an error analysis with our best model (MLP 1_100, static LR), using what we call prediction plots, which consist of two diagrams. The first shows the predicted probability that a flare larger than class M5 will occur in the next 24 hours, along with the peak flux at the time of the flare. The second depicts the binary output: when the probability passes the threshold (into the blue region, first diagram), the predicted output is one, likewise for the flux (into the red region, $\geq M5.0$). Otherwise, the output is zero. The target values in the second diagram are labelled positive 24 hours before flare eruption as originally created by [4].

We quantify the number of flares predicted accurately ahead of time for the validation and test set using the prediction plots. If the predicted probability is above the 0.5 threshold for at least three hours ahead of flare time, the flare is counted as a 'hit'. If the predicted probability is under the 0.5 threshold or late, the flare is counted as a miss. Only $\geq M5.0$-class flares are counted. The tally can be seen in Table 5.8.

Table 5.8: Number of flares correctly and incorrectly predicted. Validation and test set. MLP (1_100)

Partition	Hit	Miss	Total
Validation	17	3	20
Test	9	4	13

We select three ARs from all those that flare larger than class M5, based on our static simplified MLP, which represent the main recurring categories that we observe:

- "The Good" (NOAA: 12 017) - This has a large flare in the centre of the AR's observed lifetime. Probabilities easily predict the impending flares at the right times. See Figure 5.15, is has seven flares: [four C-class, M2, X1, M2 and another C-class.]
- "The Bad" (NOAA: 12 027) - In this case the large flare happens early in the AR observation. The model's prediction is late and misses the flare, either due to the AR being on the limb of the Sun or simply miss-predicting. See Figure 5.16.
- "The Ugly"(NOAA: 12 242) - These ARs exhibit anomalous behaviour, usually with probabilities spiking drastically everywhere. See Figure 5.17.

It is clear from Figures 5.15, 5.16 and 5.17 that the model over-predicts after a flare has erupted, causing the distinct double-peak. The MLP does not know it has already flared, as it has no memory. The history features might contribute to the sharp rise in probability after a flare occurred. The model successfully ignores the C-class flare, which it was trained to do.

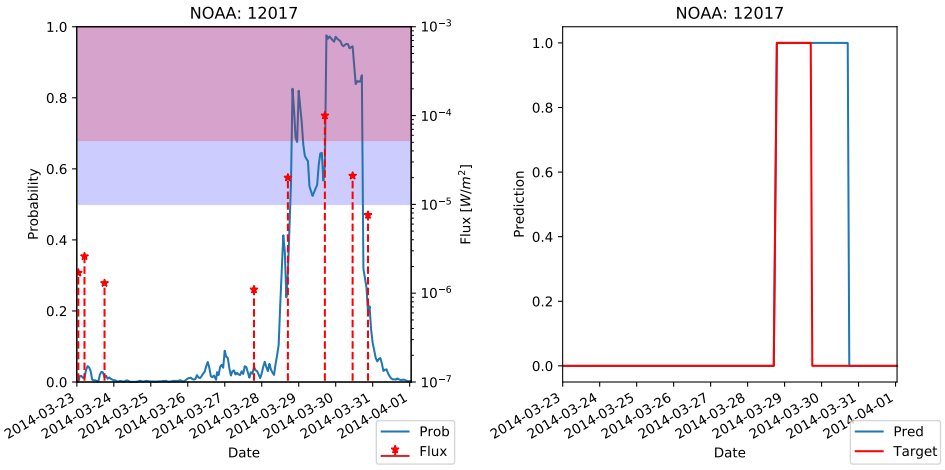


Figure 5.15: "The Good" MLP prediction plot of NOAA: 12 017. (Left-hand image) Predicted probability (blue) and peak flare flux (red). (Right-hand image) Predicted binary output (blue) and target value (red). Probability inside blue shade predicted as one. Pending flux in red shade observed as one, otherwise zero. From validation set. Flare counted as a hit.

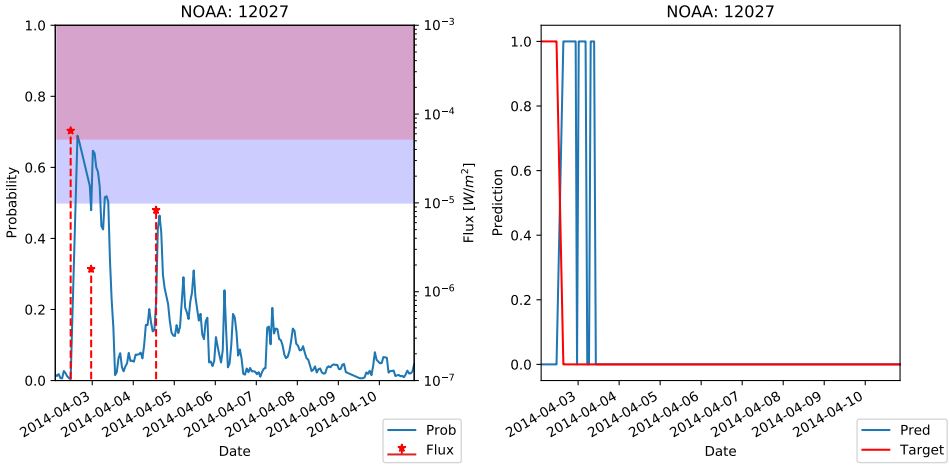


Figure 5.16: "The Bad" MLP prediction plot of NOAA: 12 027. From validation set. Flare counted as a miss. Same format as Figure 5.15

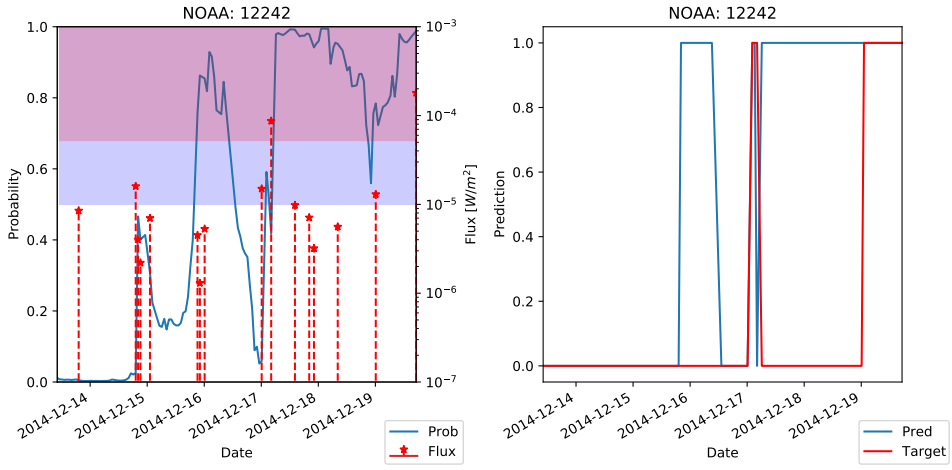


Figure 5.17: "The Ugly" MLP prediction plot of NOAA: 12 242. From validation set. First flare counted as a miss. Second flare counted as a hit. Same format as Figure 5.15

5.6.4 Incorrect labelling count

Only after our study was completed, a detailed error analysis revealed that some labels did not span the entire 24 hours before flare eruption, as made evident in Figure 5.17. This could be caused by data cleanup, where there are missing values or simply miss-labelling which was done by [4]. The number of incorrectly labelled $\geq M5.0$-class flares are counted and tabulated in Table 5.9. Most of our analysis was done on the validation set, but having four out of 13 incorrectly labelled flares on the test set, is concerning. This would likely affect measured generalisation performance but is not changed, both due to time constraints and to be able to compare results directly with [4]. Ideally, we would have tested our best network on the corrected test data to quantify the difference, if found earlier in the study.

Table 5.9: Number of incorrectly labelled flares, per partition, out of total number of $\geq M5.0$-class flares.

Partition	Number of incorrectly labelled flares	Total number of $\geq M5.0$-class flares
Training	7	42
Validation	2	20
Test	4	13

5.7 Comparison with Liu et al. results

In this section, we compare the results from our various models, scaling strategies and optimisation techniques with that of Liu et al. [4].

5.7.1 Scaling strategies

Up to this point, our models could not achieve the test TSS of 0.845 for the MLP of [4]. Liu et al. [4], used two scaling strategies for their dataset, where we only used one. It is also possible that they also stopped training before a maximum validation TSS was reached, since they trained only for seven epochs. To see if their scaling strategies hold any clear advantage, the previous simplified model (1_100) was retrained and optimised on the scaling strategy used by [4] and compared with our previous results. We optimised the models using the static LR.

Similar to the GS2 in Table 5.3, the (1_100) MLP model is re-optimised on the different scaling strategies that were employed by Liu et al. Figure 5.18 presents the TSS versus LRs plot, for the training, validation and test set. It is clear that *z_train* reaches a higher validation than the other strategies, but the other strategies are able to reach much higher test TSS.

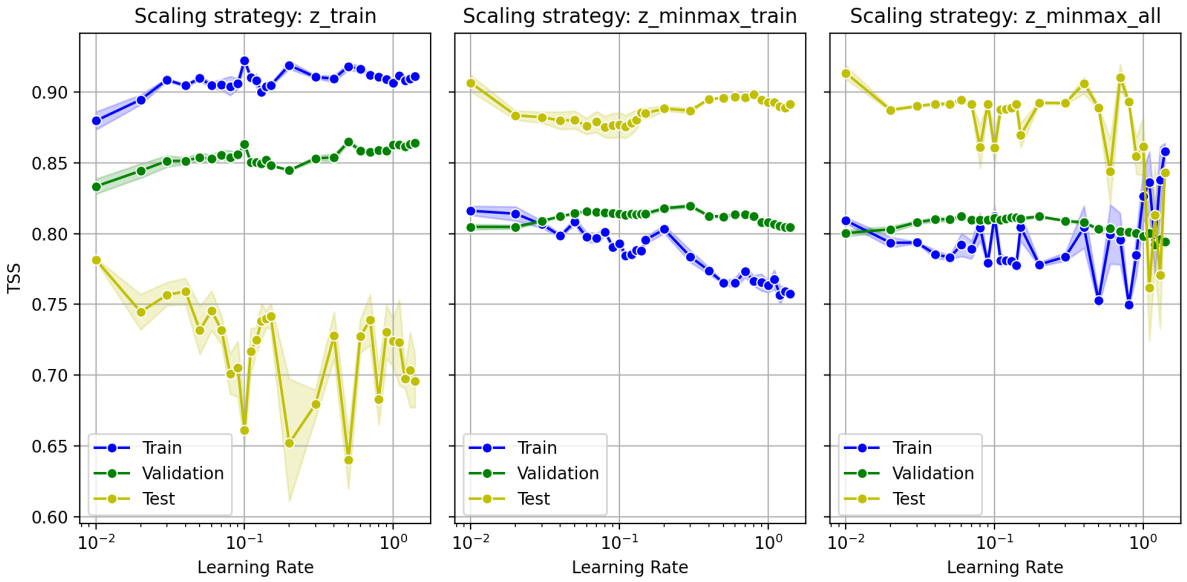


Figure 5.18: TSS versus LR plot for different scaling strategies on MLP (1_100) with dropout of 0.8.

To understand better where this drastic difference comes in, we can look at the training curves of the validation TSS for the best model from *z_train* and *z_minmax_train* in Figure 5.19. The *z_minmax_train* strategy starts overfitting much sooner, leading to a better test TSS; we see why this happens next, in Section 5.7.2.

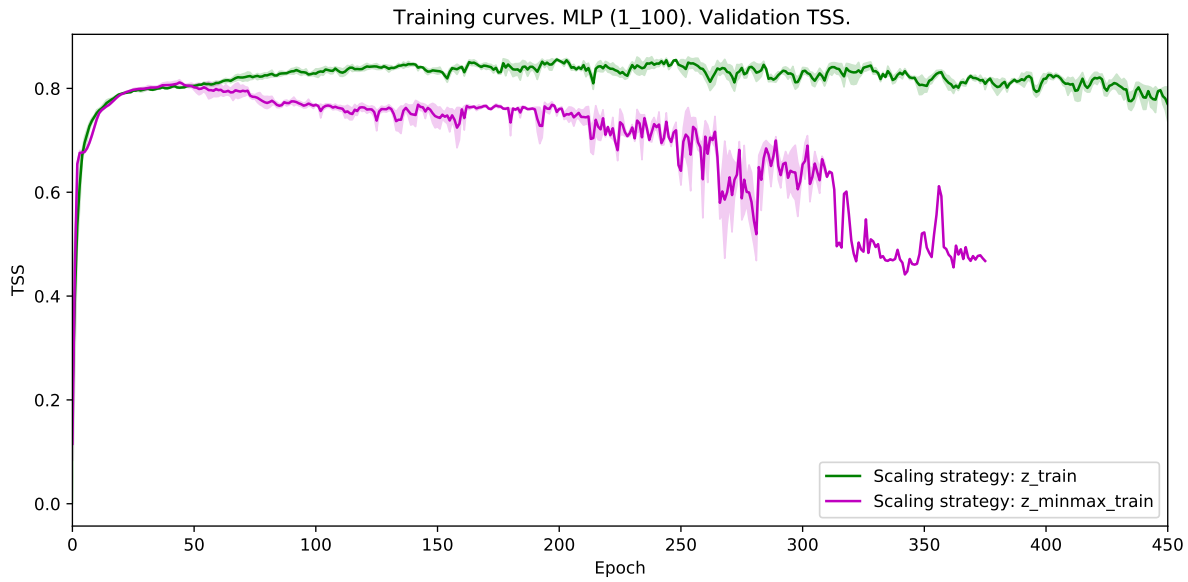


Figure 5.19: Training curve of validation TSS between best MLP model optimised with *z_train* and *z_minmax_train*. LR of 0.5 and 0.3 respectively. Average across three seeds with standard error shadow.

5.7.2 Early stopping

None of our models achieved the test TSS of 0.845 as presented by [4], but for analysis purposes only, if we plot the test TSS on the training curve (see Figure 5.20) and stop training early, before the maximum validation TSS has been reached, we see that we can indeed achieve up to 0.880 test TSS. This is why the models that tend to overfit sooner have better test TSS. This is also what we observe the authors in [4] did when they trained their models; they stopped training at a certain epoch, before the best validation TSS could be reached. If not using train or validation accuracy to guide this process, it is not clear how the specific epoch was selected.

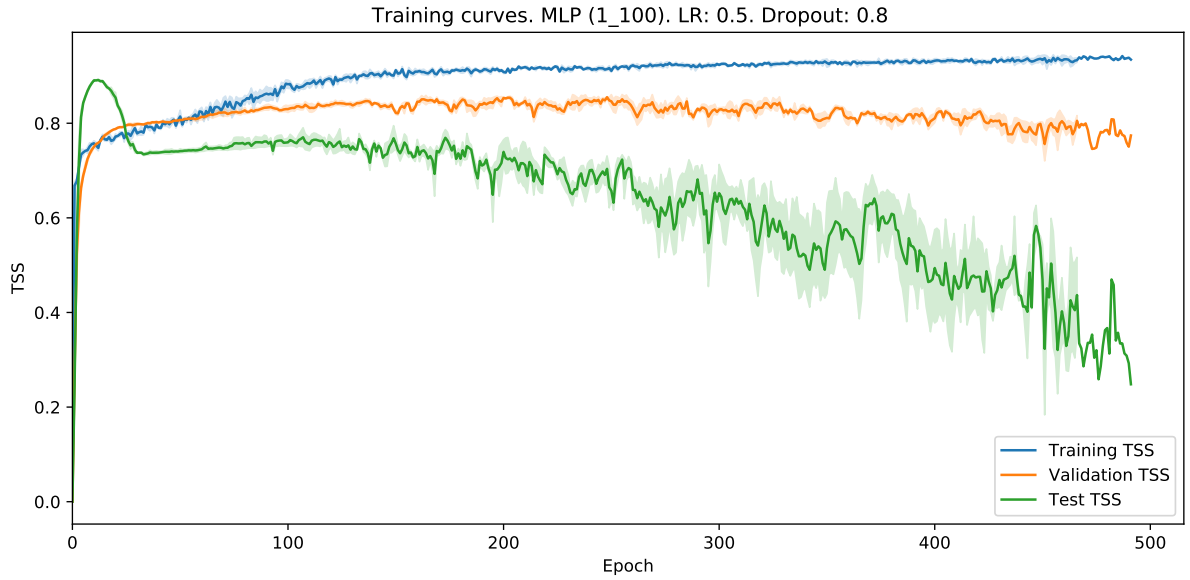


Figure 5.20: Typical training curves of training, validation and test TSS for illustration purposes. High test TSS can be obtained by stopping early, but validation TSS is not yet maximized. Average across three seeds with standard error shadow.

It becomes clear that our simple models are able to achieve similar performance to the MLP of [4] using *z_minmax_train* and *z_minmax_all*. However, the *z_train* scaling strategy achieves higher validation TSS and should therefore be used in the model selection process, which leads to a lower test TSS.

5.8 Conclusion

The smaller single-layer, 100-node, MLP with static LRs model reached the highest validation TSS compared to the 2_500 and OCLR model, although it had a much lower and more unstable test TSS. The MLP (1_100) trained with static LRs on *z_train* will be the MLP model that we use in Chapter 7, since it performed the best in the model selection process.

The scaling strategy used by [4] (*z_minmax_all*) overfits sooner, inhibiting the maximum TSS the model can reach on the validation set and giving better results on the test set.

However, when doing model selection, all choices should be based on the validation set, in which our *z_train* strategy proved best. Stopping training early, before the maximum validation TSS is reached, will give a higher test TSS, but should not be done as it imposes prior knowledge about the test set into the model selection process.

One known limitation of the MLP is that it does not consider the evolution of the AR across time, which we aim to address next in Chapter 6.

Chapter 6

TCN-variant Investigation

In this chapter we propose a TCN-variant for flare prediction. We optimise and evaluate the network's predictive capability.

6.1 Introduction

"People assume that time is a strict progression of cause to effect, but actually, from a nonlinear, non-subjective viewpoint, it's more like a big ball of wibbly-wobbly, timey-wimey ... stuff." - The Tenth Doctor (Doctor Who)

Previously we trained an MLP for a time-related problem, but MLPs do not take into account any past observations unless one adds an auto-regressive component such as sliding windows [119]. With flare prediction, we know that the evolution of an AR is relevant [13], but with an MLP no temporal or time-related information is retained, except except for explicitly engineered history features such as *Xdec*, *Xhis*, etc.

LSTMs are often used for time-series modelling [120], but recently it has been shown that TCNs can operate on par with LSTMs [38], with the advantage of having better

interpretability, since they use convolution kernels, which are inherently easier to dissect.

In this chapter, we propose our 1D-CNN architecture, which is a simplified version of the traditional TCN, to use the temporal information of the data in an attempt to improve flare prediction performance, while keeping our model easy to interpret in Chapter 7.

In Section 6.2 the traditional TCN architecture is explained and our 1D-CNN is proposed and elaborated on. The 1D-CNN is optimised in Section 6.3 and evaluated in Section 6.4.1. We discuss and conclude our findings in Sections 6.5 and 6.6.

6.2 1D-CNN architecture

This section introduces the TCN, what it is and how it works. We explain how our 1D-CNN differs from the traditional TCN and why we use it instead. We describe all the adjustable HPs as well as how the input data are prepared in sequences and fed to the network.

6.2.1 The TCN

The TCN was proposed by Bai et al. [38], specifically for sequence modelling of time series data. The TCN is founded on the time delay neural network published 30 years ago by Waibel et al. [121], along with zero-padding to ensure that the size of all the layers are equal, and dilated convolutions and residual/skip connections, to get a larger history.

The TCN architecture is best described as several residual blocks in sequence [122]. Each residual block contains dilated causal 1D convolutions, weight normalisation [123], ReLU activation [95] and dropout [101]. As the number of blocks i increases, so does the dilation with $d = 2^i$. There are basically two ways to increase the receptive field (history) of the TCN: increasing the number of residual blocks i or the kernel (or filter) size k.

6.2.2 Our 1D-CNN

In short, our 1D-CNN the same as a normal 1D-CNN but with causal convolutions. It is a TCN without the skip connections, dilation and weight normalisation. The purpose of using this instead of the full TCN is to begin with the a simpler architecture and make the model easier to interpret.

The 1D-CNN architecture (see Figure 6.1) consists of an input layer of time series data, causal convolutional layer with several kernels and a ReLU activation function with dropout. Since the output of the convolutional neural network (CNN) layer is a sequence the same length as the input, we take the sample at the last time-step and feed it to a fully connected layer that connects to the final two output nodes to create the binary classifier.

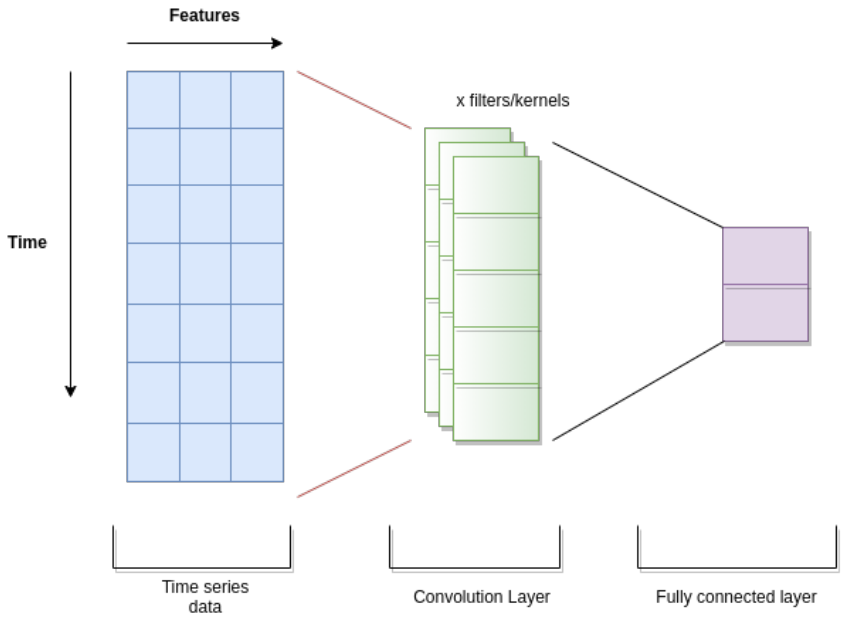


Figure 6.1: 1D-CNN architecture

To demonstrate how a normal 1D-CNN works (without causal convolutions), we create a small example in Figure 6.2, with input features *X, Y, Z*, a sequence size of four and a kernel size of two.

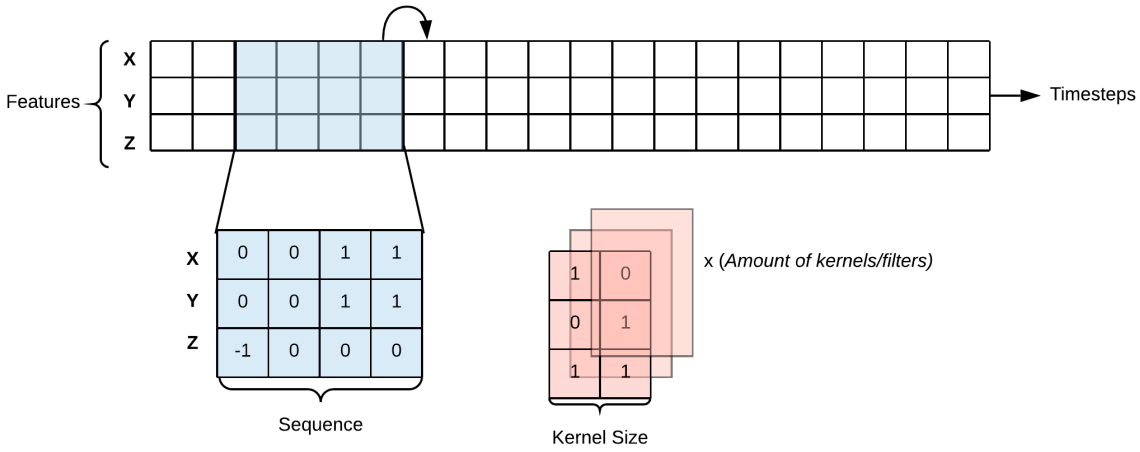
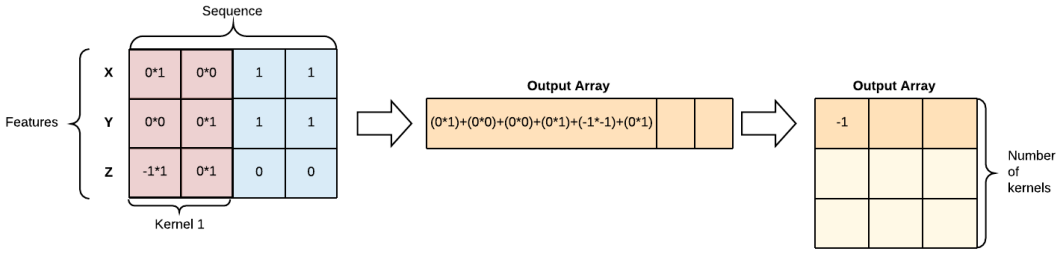
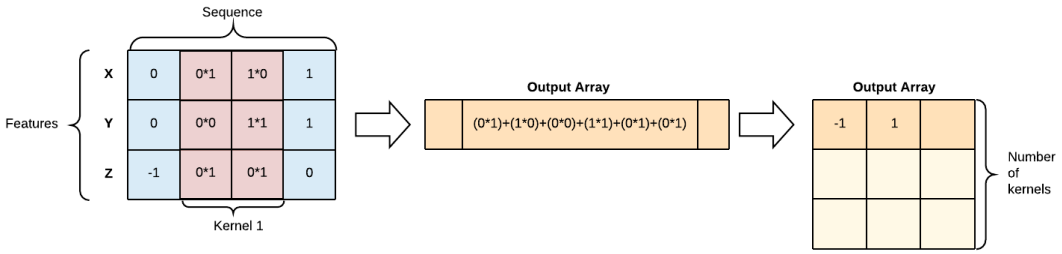


Figure 6.2: 1D-CNN functional illustration.

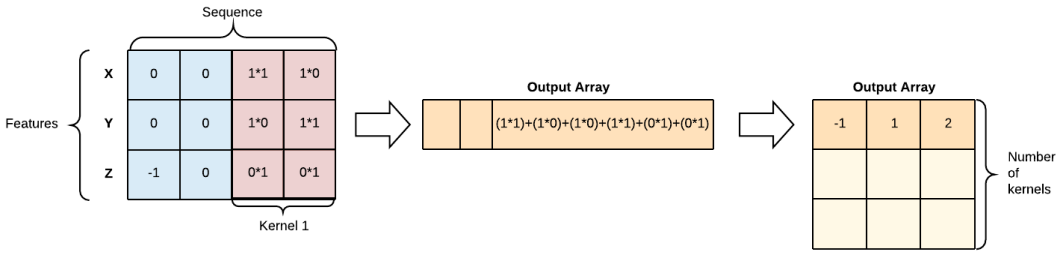
At a time-step, the sequence is convolved with all of the kernels, as seen in Figure 6.3. Figures 6.3a, 6.3b and 6.3c show the kernel (from Figure 6.2) as it strides along the sequence, computing the output. The last column of the output array is passed through a fully connected layer to the final binary output nodes.



(a)



(b)



(c)

Figure 6.3: 1D-CNN convolution process. Without causal convolutions.

Currently, in this example the model would be at time-step $t = 3$ (refer back to Figure 6.2), but clearly time-steps $i = 4, 5, 6$ are leaked into the current time-step. With causal convolutions this is avoided by simply using the sequence of values before the time-step, for example $t = 0, 1, 2, 3$ and zero-padding for the missing values at $t = 0$.

6.2.3 Sequencing solar flare data

The data is sorted per AR and if a sequence of samples is selected, then parameters from different AR can leak into the sequence, causing inaccuracies. Therefore, we zero-pad sequences when changing ARs, such that no information is shared.

6.3 Optimisation

In this section we prepare the 1D-CNN model and optimise it using static LRs. We provide less detail in this section, but follow a similar overall process as in Section 5.3.

6.3.1 Model setup

For the 1D-CNN architecture, we begin using only a single level, which is the smallest possible architecture. We use a stride of one with no dilation on the kernels. The sequence length of the input data is equal to the receptive field, which is equal to the kernel size in our setup. Weights are initialised with Kaiming initialisation [107] since they used ReLU activation functions. Dropout is used on the convolutional layers.

The HP that can vary are: batch size, kernel size, number of kernels, dropout, LR and seed.

6.3.2 Optimisation procedure

From the results in Chapter 5 we found that using the OCLR policy did not yield a performance increase; therefore, the 1D-CNN is optimised by searching for the best static LR, in a similar fashion as the MLP. To optimise the 1D-CNN we first search over a wide range of likely HPs for a single seed (GS1, Table 6.1), and once a reasonable range of HPs has been identified that gives the highest validation TSS, we refine the search (GS2,

Table 6.1).

Table 6.1: Experimental setup for 1D-CNN. Bold items indicate the best validation TSS results.

Hyperparameters	GS1	GS2
Goal	*Rough grid search for possible range of HPs*	*Refining dropout and LR*
Batch size	256, 1 024, **65 536**	65 536
Levels	1	1
Kernel size	2, 3, 7, 13	3, **7**, 13
Number of kernels	20, **40**, 80, 160	40
Weight decay	0	0
Dropout	0, 0.4, **0.8**	0.8
LR	0.001, 0.01, 0.1, 1	0.05 - 0.23
Seed	15	15, 49, 124

6.3.3 Optimisation results

From GS1, again the largest batch size, with a kernel size of 7, the number of kernels equal to the number of features (40) and a large dropout (0.8) gave the highest validation TSS.

The results from GS2 are shown in Figure 6.4, with the best validation TSS reached with a kernel size of 7, 40 kernels, dropout of 0.8 and at a LR of 0.066.

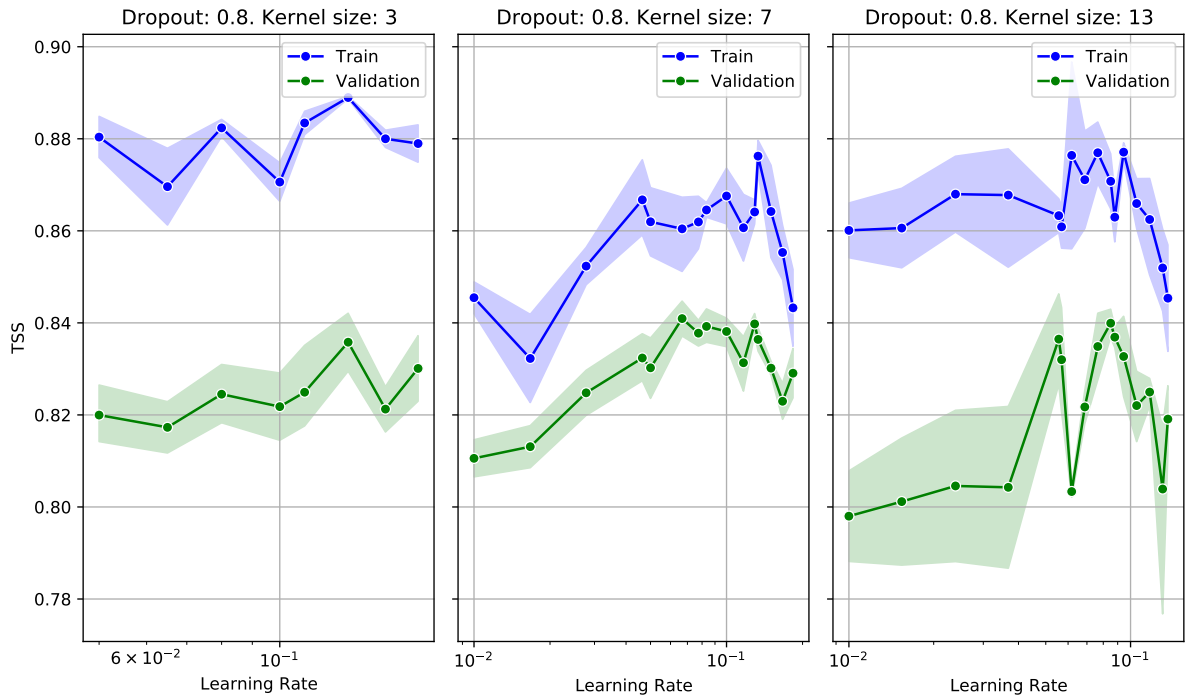


Figure 6.4: TSS versus LR plot of 1D-CNN model. Kernel sizes: 3, 7 and 13. Dropout: 0.8. Scaling strategy: *z_train*.

6.4 Model evaluation and comparison

Similar to the MLP, we evaluate our best 1D-CNN on different performance metrics, evaluation plots and prediction plots. We also compare our model to the LSTM described by [4].

6.4.1 Performance comparison

All the performance metrics for the best 1D-CNN model and LSTM [4] can be found in Table 6.2.

Table 6.2: 1D-CNN performance metric comparison. Average and standard error over three seeds

Metric	Model	Training	Validation	Test
Recall	1D-CNN	0.9947 ± 0.0010	0.9224 ± 0.0050	0.7944 ± 0.0224
	LSTM ([4])	-	-	0.978
Precision	1D-CNN	0.1428 ± 0.0201	0.1165 ± 0.0159	0.0764 ± 0.0094
	LSTM ([4])	-	-	0.038
BACC	1D-CNN	0.9736 ± 0.0047	0.9202 ± 0.0051	0.8770 ± 0.0085
	LSTM ([4])	-	-	0.938
HSS	1D-CNN	0.2387 ± 0.0317	0.1902 ± 0.0261	0.1326 ± 0.0158
	LSTM ([4])	-	-	0.074
TSS	1D-CNN	0.9473 ± 0.0093	0.8403 ± 0.0102	0.7539 ± 0.0170
	LSTM ([4])	-	-	0.877

The 1D-CNN validation TSS was 0.8% (absolute) lower than that of the larger MLP (2_500), but with almost double the standard error. The 1D-CNN test TSS was 0.7% (absolute) better than that of the large MLP (2_500). From these results, there is no clear gain for including temporal information using the 1D-CNN architecture.

Our 1D-CNN did not achieve the same performance as the LSTM of [4], but we know from Section 5.7 and 5.7.2 that scaling strategies and premature stopping of training based on test results can lead to high test TSS of 0.8968 ± 0.0028. However, this is bad practice.

6.4.2 Evaluation plots

The best 1D-CNN's evaluation plot on the test set is displayed in Figure 6.5. Training and validation sets can be found in Appendix A.5, Figures A.19 and A.20. For a description of our evaluation plots, refer back to Section 5.6.2.

The 1D-CNN evaluation plots are nearly identical to our MLP models, with the test TSS again peaking at the climatological event rate and a reliability diagram that over-forecasts.

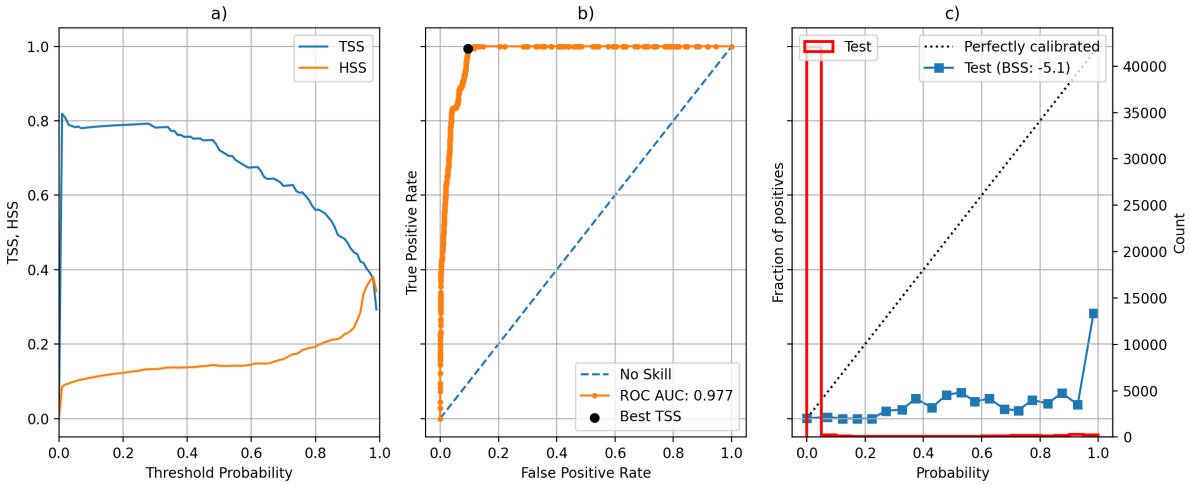


Figure 6.5: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. 1D-CNN.

6.4.3 Error analysis

The number of accurately predicted flares are tallied and "The Good, The Bad and The Ugly" prediction plots (refer back to Section 5.6.3 for a detailed description of the plots) are redrawn for our best 1D-CNN model, based on the highest validation TSS it achieved, as well as for the best LSTM model by [4]. We examine whether the addition of temporal information can help the models perform better in an operational setting for the selected ARs.

1D-CNN

Using the prediction plots, we counted the number of flares predicted accurately in the validation and test set for the 1D-CNN in Table 6.3. The flares predicted are the same as the MLP, except one more flare is correctly predicted on the test set.

Table 6.3: Number of flares correctly and incorrectly predicted. Validation and test set. 1D-CNN.

Partition	Hit	Miss	Total
Validation	17	3	20
Test	10	3	13

From the prediction plots in Figures 6.6, 6.7 and 6.8, it is clear that the 1D-CNN also over-predicts after a flare has occurred, similar to the MLP, however, the 1D-CNN probabilities are less sporadic.

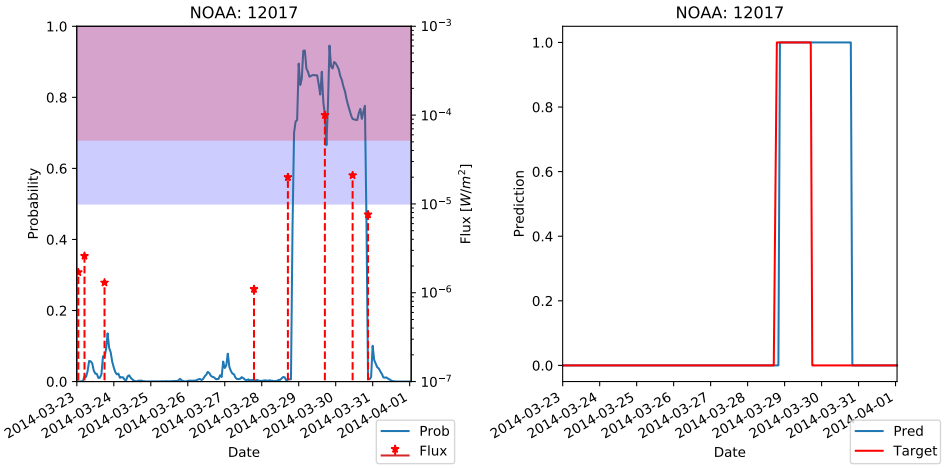


Figure 6.6: "The Good" 1D-CNN prediction plot of NOAA: 12 017. Validation set. Counted as a hit. Same format as Figure 5.15

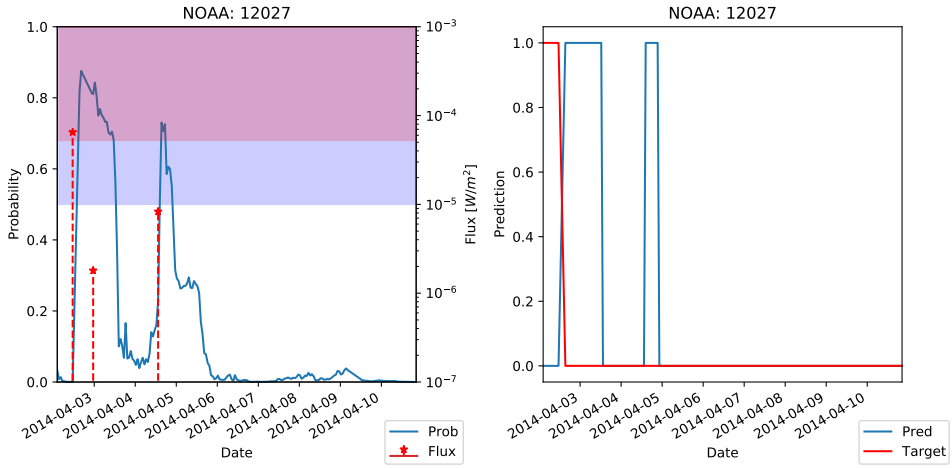


Figure 6.7: "The Bad" 1D-CNN prediction plot of NOAA: 12 027. Validation set. Counted as a miss. Same format as Figure 5.15

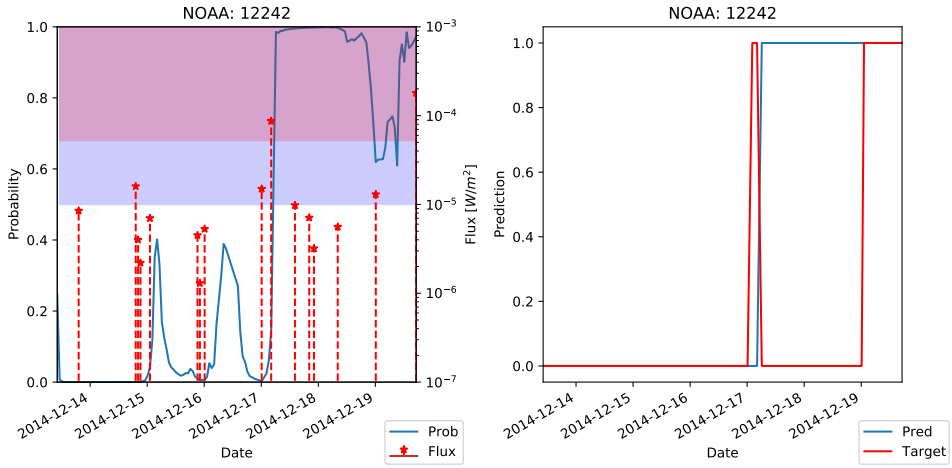


Figure 6.8: "The Ugly" 1D-CNN prediction plot of NOAA: 12 242. Validation set. First flare counted as a miss. Second flare counted as a hit. Same format as Figure 5.15

LSTM from [4]

We took the final pretrained LSTM model used by the authors of [4], on their dataset, using only their top 20 features (as used by [4]) and with an attention layer as originally used, and drew the prediction plots for a threshold of 0.5.

The tally of the predicted flares for the LSTM can be found in Table 6.4. It is evident

that the LSTM accurately predicts more flares ahead of the eruption, than our MLP and 1D-CNN. This disparity could be due to the architecture, scaling strategy, number of features, optimisation strategy or attention mechanisms.

Table 6.4: Number of flares correctly and incorrectly predicted. Validation and test set. Best LSTM by [4]

Partition	Hit	Miss	Total
Validation	18	2	20
Test	12	1	13

The prediction plots in Figures 6.9, 6.10 and 6.11, reveal that the LSTM probability does not spike after a flare has occurred. The probability also rises and falls more gradually. The LSTM does over-predict more severely than our MLP and 1D-CNN.

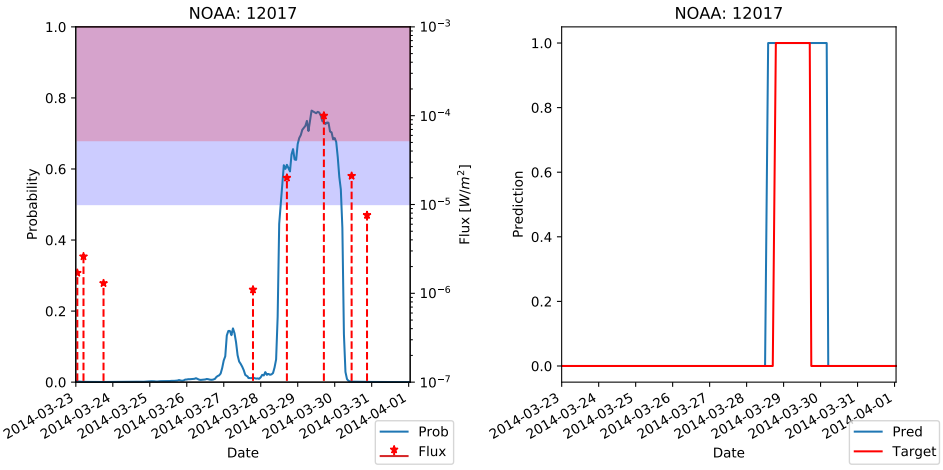


Figure 6.9: "The Good" LSTM plot of NOAA: 12 017. Validation set. Counted as a hit. Same format as Figure 5.15

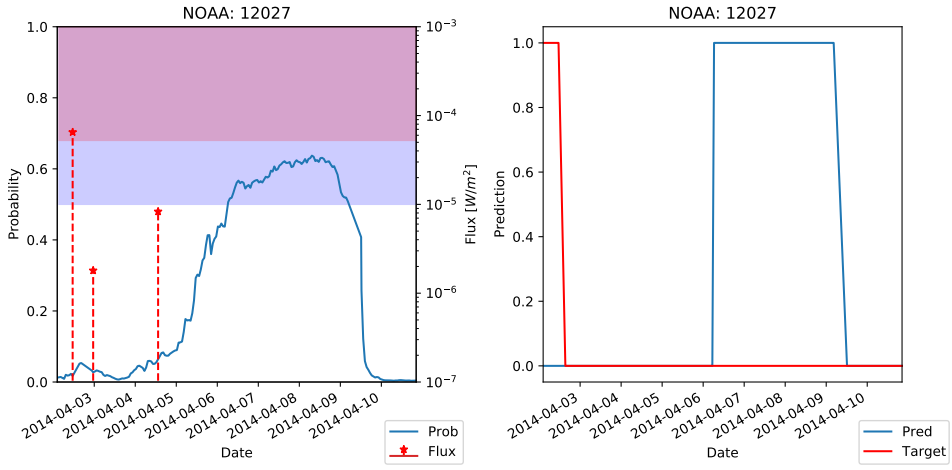


Figure 6.10: "The Bad" LSTM prediction plot of NOAA: 12 027. Validation set. Counted as a miss. Same format as Figure 5.15

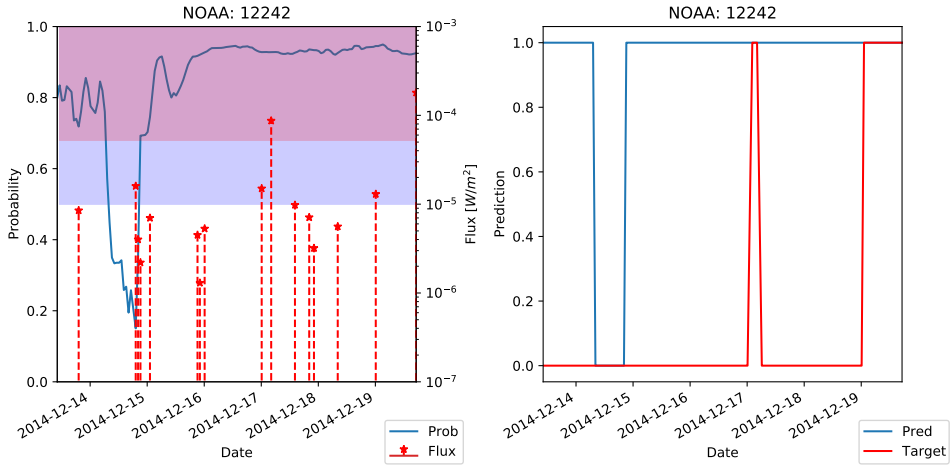


Figure 6.11: "The Ugly" LSTM prediction plot of NOAA: 12 242. Validation set. Both flares counted as a hit. Same format as Figure 5.15

6.5 Discussion

From the performance metric results in Table 6.2, it is clear that the single-layer 1D-CNN has no clear advantage over the MLP even when accounting for past observations. Other research has shown that the models trained on shuffled data actually performed better [2], thus completely disregarding the temporal element. This is counter-intuitive, since we

know from the physics and observation that the evolution of the flare is crucial to the production of flares [13].

Liu et al. [4] used an attention layer for their final LSTM network and saw an increase in performance. They also optimised their final LSTM model using their own version of cross-validation (see Appendix A.1, Figure A.1). They did not use cross-validation with the model results shown in Table 2.2. Our 1D-CNN did not use any attention mechanism or form of cross-validation. Cross-validation such as *k-fold* is typically used when datasets are not large (or information-rich) enough to solve real-world tasks, making it ideal for flare prediction [124]. Introducing cross-validation could potentially be beneficial, to combat the large class imbalance.

6.6 Conclusion

The single-layer 1D-CNN with a receptive field of either 7 or 13 did not perform better in terms of the validation and test TSS it achieved compared to our MLP. The addition of temporal information using the 1D-CNN yielded no significant improvement with the performance metrics, but did show a smoothing effect on the predicted probabilities in an operational setting.

The addition of attention mechanisms and cross-validation could potentially improve performance, as shown in [4], but the largest bottleneck seems to lie in the data itself and the general issue with unbalanced data.

These models are nevertheless useful and could still be used to with reasonable certainty to tell us more about the features they learnt to be most relevant for flare prediction, which we elaborate on next in Chapter 7.

Chapter 7

Interpretability

In this chapter we carry out a case study on a specific AR and investigate the feature attribution methods with the MLP and 1D-CNN. We compare the different techniques with existing research and domain knowledge.

7.1 Introduction

Space weather scientists are more interested in which solar mechanics drive the eruption of flares than accurate prediction models. While DNNs are able to predict flares with gusto, how they arrive at those choices is still not well understood. However, there are empirical techniques, such as attribution methods, that, while not physics-based, are able to estimate how important the model regards certain input features.

In this chapter we aim to use our previous best MLP and 1D-CNN to calculate the importance of all the features during the observed life-cycle of an AR. The different models and attribution techniques will be compared to each other and to previous research.

In Section 7.2 we explain the different attribution methods, the specific AR observed and

the results from the attribution methods applied to our MLP and 1D-CNN models. The findings are discussed in Section 7.3 and concluded in Section 7.4.

7.2 Feature attribution

First, we explain the attribution methods we use for our analysis and the specific samples to which these are applied. Then we calculate the feature importance for each attribution method on both models.

7.2.1 Feature attribution methods

In Section 2.5, we briefly describe five attribution methods, two perturbation-based and three gradient-based. The perturbation-based methods are ablation and shapely value sampling [67]. The gradient-based methods are DeepLIFT [68], integrated gradients [69] and input $\times$ gradient [68]. All the techniques mentioned will be evaluated. Most of these techniques require a baseline, which represents a zero activity case. We use a zero baseline, since similar results were found when defining the baseline as the non-events arising from NOAA 12 252, which never produced a flare. We describe the main methods, following the descriptions in [70]:

Ablation [66]

This technique determines the feature importance by swapping out each input feature with some baseline (zero in our case) and measuring the change in the output. Features can be grouped together, but are studied individually in our analysis.

Shapely value sampling [67]

Shapely values stem from cooperative game theory. The method works by adding every permutation of the input features sequentially to a baseline. The difference in the output after adding each feature coincides with its contribution, and the average is calculated for these differences over all permutations to get the attribution.

This method is more computationally expensive, depending on the number of features. Shapely value sampling eliminates some of the computation by sampling random permutations and calculating the average marginal contribution from these permutations. Features can also be grouped.

DeepLIFT [68]

DeepLIFT tries to quantify the difference in the outputs from a baseline with regard to the difference in inputs from a baseline. It does this with the theory of multipliers that criticises certain neurons for the change in the output. These multipliers are defined by [68] as,

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x} \tag{7.1}$$

where Δx is the difference between the input neuron x and the baseline, and Δt is the difference between the target neuron t and the baseline. C is the contribution of Δx to Δt.

Integrated gradients [69]

Integrated gradients approximate the integral of gradients with respect to inputs across the network path from the baseline to inputs. Originally proposed by [69] as,

$$\text{IntegratedGrads}_{\ i}(x) := (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha \tag{7.2}$$

where integrated gradients are calculated along the ith dimension from baseline x' to input x, and α is the scaling coefficient. $F(x)$ is the function that represent the DNN and $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i dimension.

Saliency [125]

Saliency is a straightforward method for determining attribution, by calculating the gradient of the output in relation to the inputs. It can be thought of as taking a first-order Taylor expansion at the inputs, and the coefficients are the gradients in the linear representation of the model. The importance of the features is simply the absolute value of the coefficients. One downfall of this technique is that is only has positive importance.

Input $\times$ gradient [68]

Input $\times$ gradient is an addition to saliency that computes the outputs related to the inputs and multiplies it with the input features. The product of the coefficient from the gradient, with the input is the attribution of the feature.

7.2.2 Case study

The AR we will be observing is NOAA: 12 673. This was a class X9 flare, the largest in the last 15 years. It is the same flare described in Section 2.2.

This AR was specifically selected because it is massive and both our MLP and 1D-CNN were able to predict the flare accurately ahead of time. If the models have learnt what causes extremely large flares, then extracting which features are important for large flares could give valuable feedback on which features should be looked for at certain stages of an AR's life-cycle to determine whether it might erupt.

The prediction plots for NOAA: 12 673, for both the MLP and 1D-CNN, can be found in Appendix A.6, Figures A.21 and A.22, respectively.

7.2.3 Attribution over active region life-cycle (MLP)

The Captum library [70] was designed to use attribution methods to analyse DNNs. We use this library to calculate the feature attribution. For our best MLP model we create a diagram that plots all the different attribution methods for each feature over the duration of the AR, as seen in Figure 7.1.

Each subplot represents the specific input features (labelled by title) and the five attributions. The five attribution colours are listed in the bottom right-hand panel (*logEdec*). The red-shaded region shows the period from $t - 24$ hours up to t, where t is the time of the flare eruption. The plot starts at $t - 150$. The importance shown is the average importance over three seeds with a standard error shadow. Some attribution methods peak long before flare eruption and may indicate some build-up to a flare, or something unrelated to flare eruption.

7.2.4 Attribution over active region life-cycle (1D-CNN)

We repeat the previous section by recreating the attribution plot by using our best 1D-CNN model in Figure 7.2.

7.2.5 Comparison of attribution methods

In Table 7.1, we look at the features in the time leading up to the flare and sort them into three trends: upwards, downwards and noisy. If the trend is upwards before flare production, it means that the model regards the specific feature as relevant for predicting flares. We only consider trends that rise above zero to be upwards trending. If the trend is noisy or downwards before the flare eruption time, the model considers the feature as

unimportant.

TOTUSJH, *TOTUSJZ* and *TOTPOT* are consistently found to trend upwards, preceding the larger flare for both models, which corresponds to the findings of [4], [43], [56]. These parameters are indicative of the shape (*TOTUSJH*), magnitude (*TOTPOT*) and current density (*TOTUSJZ*) across the AR, all indicative of the dynamics involved in the eruption triggering mechanisms [43] The 1D-CNN upwards-trending features are the same ones that the authors in [4] found in their top 17 features. Unlike the previous research mentioned, our attribution methods also highlight the importance of features during certain stages of the ARs.

Table 7.1: Feature trends of attribution method between MLP and 1D-CNN

Trend	**Features (MLP)**	**Features (1D-CNN)**
Upwards	*SAVNCPP*, *TOTUSJH*, *TOTUSJZ*, *TOTPOT*, *Cdec*, *Mdec*, *logEdec*, *Bhis*, *Chis*, *Mhis*	*SAVNCPP*, *TOTUSJH*, *TOTUSJZ*, *TOTPOT*, *ABSNJZH*, *USFLUX*, *TOTBSQ*, *Cdec*, *Mdec*, *Edec*, *Mhis*, *Chis1d*, *Mhis1d*
Downwards	*TOTFX*, *MEANGAM*, *MEANJZH*, *MEANALP*, *MEANGBZ*, *MEANGBT*, *EPSX*, *EPSY*, *EPSZ*, *Chis1d*, *Mhis1d*	*TOTFX*, *MEANGAM*, *SHRGT45*, *MEANSHR*, *Bdec*, *Bhis1d*
Noise	Remaining	Remaining

7.3 Discussion

Some magnitudes dominate after the flare had already occurred, for example *Xmax1d*, which can be misleading, hence the study of trends instead of magnitudes. Plotting the

attribution versus time highlights the importance of features during certain stages of the AR life-cycle. DeepLIFT and integrated gradients are found to be near identical in our analysis.

The 1D-CNN's attribution seemed to be most consistent with [4], with its upwards-trending features contained in their top 17 features. This suggests that our attribution methods can extract predictive features, which can be investigated further.

These attribution methods could also be used on the image-based flare prediction models to help explain the choices the models make, based on the formation of the AR, in future work.

7.4 Conclusion

We applied attribution methods to our best MLP and 1D-CNN models to derive the importance of features during certain stages of an AR life-cycle. Our 1D-CNN upwards-trending features were consistent with the top 17 features of [4], and revealed that the importance of features change drastically as the AR evolves. Some feature importance differed between model types, but *TOTUSJH*, *TOTUSJZ* and *TOTPOT* was repeatedly found important in both, which agrees with previous research [4], [43], [56]. This suggests that the vertical current, current helicity and photospheric magnetic free energy density are very relevant for large flare prediction.

Monitoring the individual importance of these predictive features can also be beneficial for flare prediction, as there are very noticeable trends in features, such as *TOTUSJH*, before flares occur.

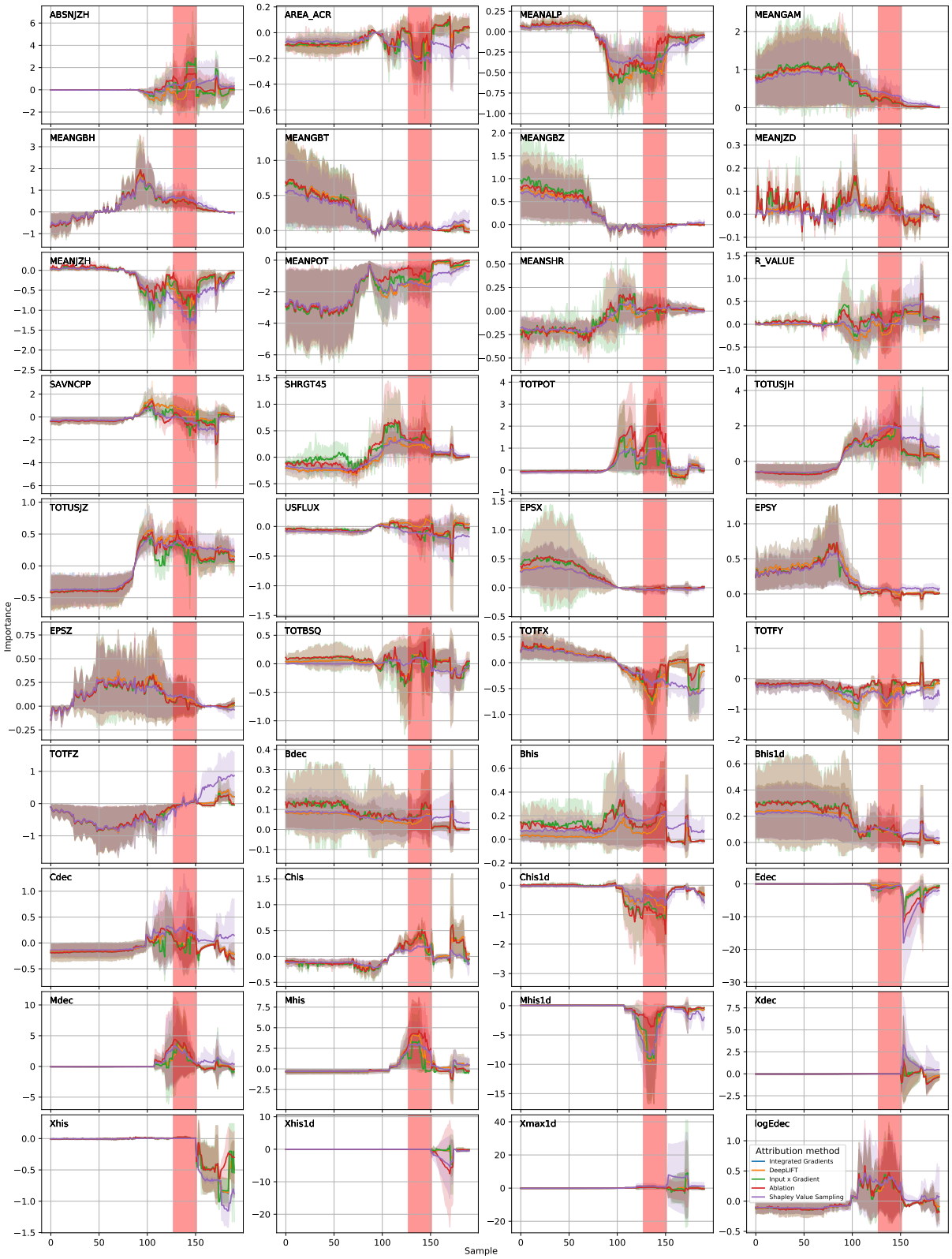


Figure 7.1: Attribution versus time for best MLP (1_100) model. 24 hours before flare (red region). NOAA:12 673

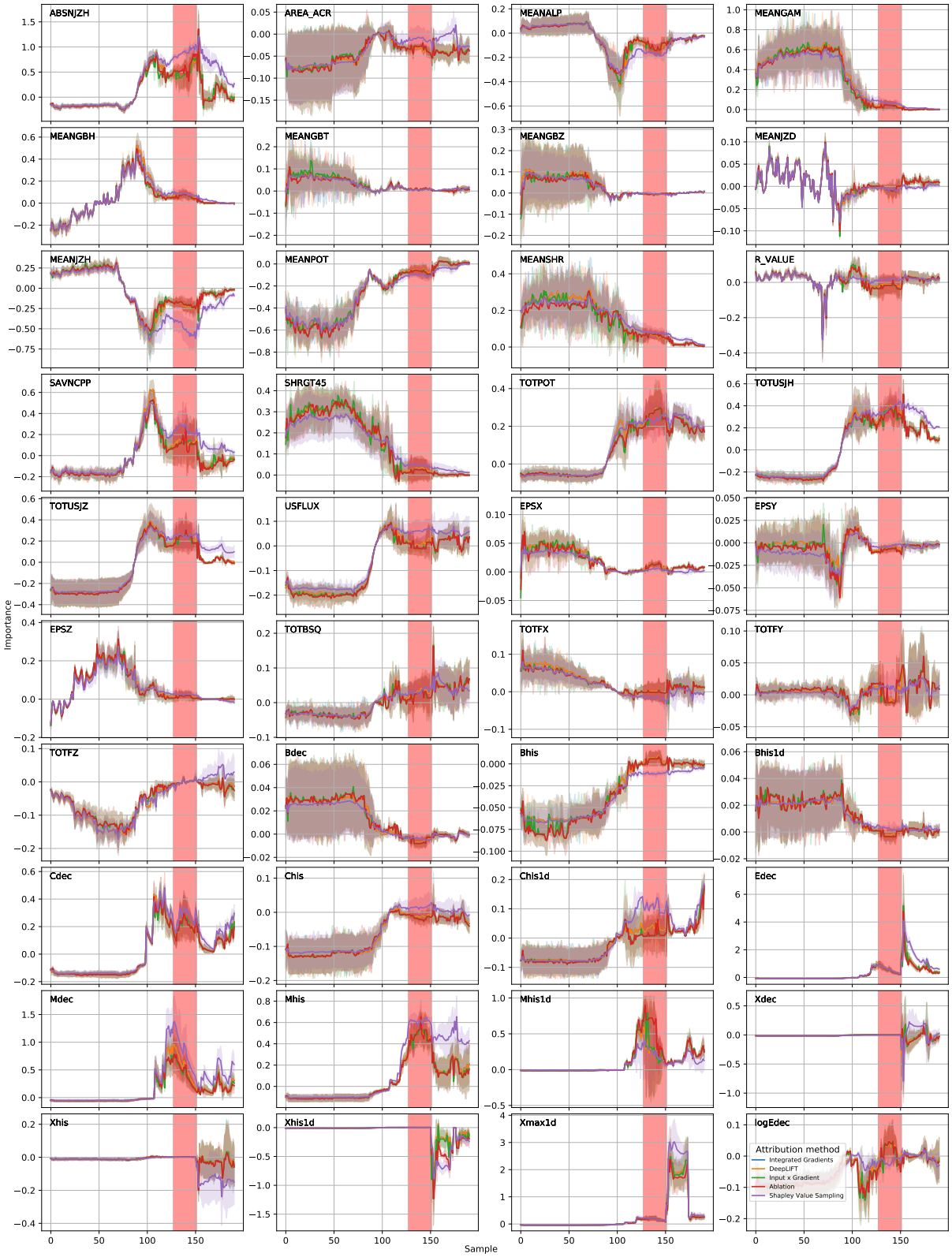


Figure 7.2: Attribution versus time for best 1D-CNN model. 24 hours before flare (red region). NOAA:12 673

Chapter 8

Deployment Practicalities

In this chapter we elaborate on the practicalities, should this or any other model be deployed for near real-time flare prediction.

8.1 Introduction

"How long is a piece of string?" - Engineers

Ultimately space observations require early-warning systems that can predict solar phenomena ahead of time, in order to take the necessary precautions. Ideally, for solar flare prediction we would like to have a model that can give a probability on how likely a certain class of flare is within a given time span for every AR.

In this chapter, we explain the process of acquiring data and doing near real-time predictions with probabilistic models. First, the difference between probabilistic and binary classification models are discussed in Section 8.2. In Section 8.3 we go through the data acquisition pipeline we developed, followed by how the acquired data can be used to do near real-time predictions in Section 8.4.

8.2 Probabilistic vs. binary classification models

Any binary classification model can be converted into a probabilistic forecasting model by passing the final nodes through a soft-max layer to squeeze the outputs between zero and one. Many DNNs have recently been developed that increased the TSS achievable for flare prediction ([2]–[10]), but networks with a high TSS tend to lack BSS and reliability (in the sense of reliability diagrams), and is not well understood [55].

Typically in terrestrial weather forecasting, models are calibrated [126], but this is not favoured in the space weather forecasting community. The opinion from the authors of [71] is that models should already be trained to reliable performance. It has been shown mathematically that reliable prediction models maximise TSS at the climatological event rate [127].

Producing reliable DNNs for solar flare prediction has not been thoroughly discussed in the literature, but at the time of writing the authors of [71] aimed to find a more reliable model, using what they called the Deep Flare Net-Reliable (DeFN-R) model. The DeFN-R is an MLP with skip connections. They were able to achieve near-perfect calibration in their network by simply using cross-entropy instead of weighted cross-entropy for the loss function and optimising for BSS. These are new results, developed in parallel with ours.

8.3 Data acquisition pipeline

Liu et al. [4] created a dataset that incorporates SHARP features [74], Lorentz features [75], [76], history data [79] and decay values [45], from 2010 to 2018.

We created our own pipeline to generate an equivalent dataset to that of Liu et al. Depending on the desired model (a) with or (b) without the Lorentz features (which cannot be queried in real time, see Section 8.4), the dataset can be generated for training, validation and testing. The process for generating the Liu et al. equivalent dataset is depicted by the solid lines following path 'a' in Figure 8.1, and works as follows:

- SHARP and Lorentz features are obtained from JSOC.
- The GOES flare list is acquired and used to calculate the history and decay values, as well as determine the labels for the target values.
- The data is cleaned for any missing and edge values, as well as scaled.
- The dataset is partitioned into the desired training, validation and testing sets.
- The model is optimised and used to make predictions.

8.4 Near real-time prediction and inference

There are a few issues that need addressing before near real-time predictions can be made.

First, a model needs to be trained and ready to use for the specific task, for example predicting flares larger than class M5 in the next 24 hours. Then, the near real-time data need to be queried from JSOC. The SHARP data have a near real-time option called *hmi.sharp_720s_nrt*, but the Lorentz force components do not, and are typically only available after a few days. So there are two options for deployment: (a) manually calculate the real-time Lorentz features based on the real-time SHARP data as done by [76], which requires technical expertise; or (b) leave out the Lorentz features, with a probable slight loss in accuracy (since *TOTBSQ* is important for flare classification according to Chapter 7, but might not be too severe, as strongly it correlates strongly with other important features as seen in Figure 4.2), and have a model for immediate deployment. Either way, the rest of the deployment process should remain relatively similar. For the sake of this study, we explain the process without the Lorentz features.

Once the SHARP data have been acquired for the current time and AR, the history values would need to be calculated, but this would require a record of previous queries and history features for each AR to be stored, for example, in a database. The database can keep a record of each AR over its lifetime, which is currently on the Sun. Records of past ARs can be discarded once an AR has expired, to conserve storage. Our previous

pipeline can be tweaked such that new entries are appended without labels to do this. The history values are thus derived from the current and previous entries in the record for the current AR. The new input entry is normalised, passed through the model and a soft-max function to determine the probability of flare eruption at the current time.

Figure 8.1 shows how the model development and real-time deployment would run. Solid lines indicate the process for model creation and dashed lines denote the process for real-time deployment. The Lorentz features are in dashed box as it indicates optional input features that may not always be available.

We developed a live flare prediction model as a proof of concept, using a MLP[1]. It is deployed using Flask and Heroku. It currently only uses the SHARP features to predict the probability of a class M5 flare within in next 24 hours, for every HARP. Updates can be made as required.

[1] `http://deep-flare-prediction.herokuapp.com/`

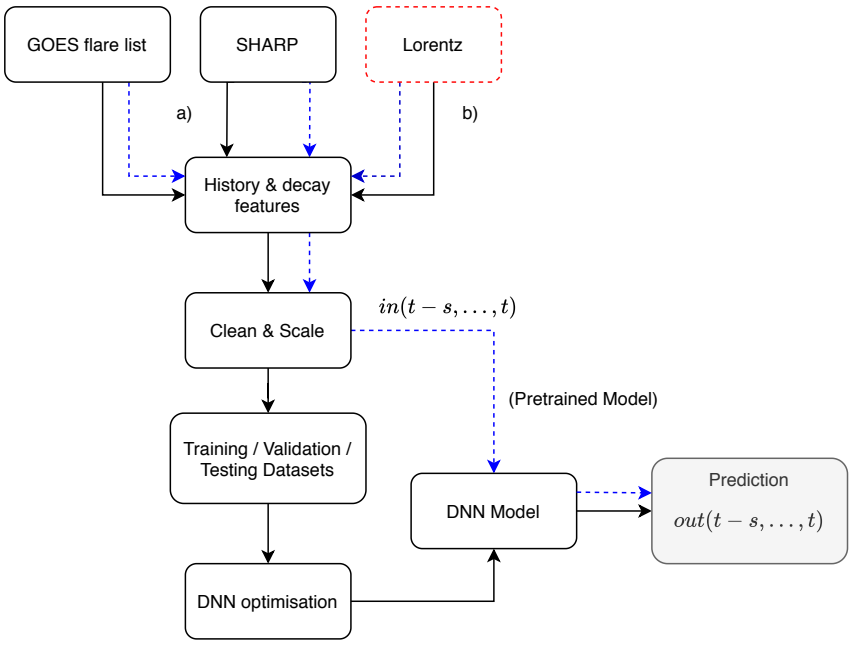


Figure 8.1: Schematic for training and deploying prediction models. Solid lines (black) are meant for the model training and selection process. Dashed lines (blue) denote the flow for doing near real-time prediction. Input data are denoted as $in(t-s,...,t)$, for a sequence s hours in the past, and the predicted output denoted as $out(t-s,...,t)$ (sequences only applicable to 1D-CNN). MLP only uses values at time-step t. Lorentz features are optional (red), depending on availability, denoted as path (b).

8.5 Conclusion

We have created a data acquisition pipeline that is able to generate the dataset similar to [4], which includes physical and history features. Our binary classification models can be converted to probabilistic forecasting models by passing the output nodes through a soft-max function to squeeze the results between zero and one. Near real-time prediction is possible either by leaving out the Lorentz features or building the necessary implementation to calculate them. Leaving out Lorentz features might come with a slight loss in performance, since $TOTBSQ$ is regarded as relatively important from Chapter 7.

Chapter 9

Conclusion

In this chapter we summarise key findings and elaborate on the implications of these findings. We also reflect on future work.

9.1 Introduction

As stated in Chapter 1, the aim of this dissertation was to develop neural networks with limited parameters and straightforward architectures, specifically limiting recurrence, for flare prediction; to test different optimisation techniques; and to interpret these models to explore flare driving mechanisms. We summarise the key findings, examine their implications and propose future work.

9.2 Key findings and implications

After training and optimising several models with different optimisation techniques, and interpreting each feature's predictive capability, we observed the following key findings:

Comparison with Liu et al.'s [4] result. Since this work was based on the dataset kindly provided by Liu et al. and described in [4], we list some important considerations:

- Datasets:
  - We believe some features were labelled incorrectly in the dataset. Utilising the labels as-is would cause models to report incorrectly on certain features. We re-labelled these and provided an errata (Section 3.5). Our models are trained on the adjusted dataset. (Note that this has no effect on performance, only on feature importance.)
  - Some flares were incorrectly labelled by [4], by not spanning the entire 24 hours before flare eruption. This would inhibit model performance on the test set (in other words the out-of-sample or real-world performance). This was examined and described in Section 5.6.4.
- Differences in approaches to model development between Liu et al. and what we consider best practise was found:
  - Liu et al. did not provide enough information regarding validation scores or error across training seeds to enable a fair comparison with our model selection procedures. All model development and model selection should conclude before measuring the effect of any decision using the evaluation set.
  - The dataset used by Liu et al. was scaled using measurements from the entire dataset instead of just from the training set, which could bias the validation and test results. Features should rather be scaled according to only the training set, and the test set should remain unobserved until the final evaluation.
  - Liu et al. stopped training at a specific epoch, which is acceptable practice, however we do not know how the epoch was decided and whether it was based on the validation set. Prematurely stopping training before the highest validation TSS is reached gives a higher test TSS. Our models could reach their reported test TSS of 0.880 if the training should be stopped around the epoch

where test TSS reaches a maximum, but this would be exceedingly bad practice.

- Early-stopping allows model selection at the epoch where the validation TSS is the largest, which is perhaps a healthier way of doing model selection, rather than stopping at a set epoch. Note that, validation TSS could still spike during training and cause the models to be selected on unstable epochs. To circumvent this, we used dropout as a regulariser.
- The scaling strategy also affects the model performance. Applying z-norm on the physical features and min-max scaling on the history features, that is scaled according to the training set (*z_minmax_train*), overfits much sooner on validation TSS than applying z-norm on all features (*z_train*), and as previously seen; leads to a higher test TSS. However, *z_train* reaches a higher validation TSS, and should be selected during the model selection process, if only validation set results are considered.
- We only compared our models to those by [4] without cross-validation, as we did not use cross-validation. Using k-fold cross-validation could potentially help improve performance and bring stability to the very limited dataset. Again, final evaluation should then be performed on a held-out set, not only using the results from the k-fold test results.

Simpler models reach good performance

- The simplified MLP (1_100), optimised with static LRs, reached a higher validation TSS of 0.865, compared to the validation TSS of 0.849 with the larger MLP (2_500). Unfortunately, the test TSS was much lower for the smaller MLP. This indicates that a high validation TSS does not correlate with a high test TSS, and that the validation and test set might not sufficiently be representative of the real-world data distribution.
- Using the novel OCLR policy [110], our simplified MLP could reach a validation TSS of 0.832 but with a higher test TSS of 0.767, compared to the larger MLP with

a test TSS of 0.747. This suggests that the smaller model can still perform on par with the larger model, which significantly reduces the complexity of the model.

- The OCLR policy helped with performance, but the process of optimisation and finding the optimal HP was still burdensome. The OCLR policy is efficient for finding HPs that work well, but still needs plenty of tuning for finding the best HPs.
- The addition of temporal information with our proposed 1D-CNN model did not yield a significant improvement on the performance compared to the MLP. However, the 1D-CNN had a smoothing effect on the predicted probabilities, suggesting that past observation help stabilize probabilistic forecasting.

Attribution methods

- The features that trend upwards 24 hours before a large flare using attribution methods with the 1D-CNN are also in the top 17 most important features found by [4], [43], [56].
- Both models (MLP and 1D-CNN) found, using attribution methods, that *TOTUSJH*, *TOTUSJZ* and *TOTPOT* are the most relevant features for large flare prediction, which is consistent with other literature [4], [43], [56]. This mean that the vertical current, current helicity and photospheric magnetic free energy density are very relevant for large flare prediction.
- Ablation, shapely value sampling, DeepLIFT, integrated gradient and input x gradient all gave similar trends in our analysis, as found by [61].
- DeepLIFT and integrated gradient gave near identical results.

Development of real-time pipeline

- Lorentz features are not available in near real-time from JSOC. To deploy our current best models, either the Lorentz features should be discarded, or a custom

implementation should be made to calculate them. Leaving out Lorentz features could result in a loss of performance, since *TOTBSQ* is regarded as important by the attribution methods (see Section 7.2). However, the loss might not be too severe as it is also strongly correlated with the other important features like *TOTUSJH* (see Section 4.3). Including the Lorentz features would require an additional step to calculate the parameters from the real-time SHARP features before the model is executed.

- Smaller models are less computationally expensive to deploy in near real-time settings.

While our models could reasonably predict flare events, we highlighted areas in the dataset and with the optimisation strategies used, which could be improved upon; we do not expect to create perfect flare prediction models.

9.3 Contributions

By completing this study, we demonstrate the use of simple DNNs for the prediction of flares, acceptable optimisation strategies and model interpretation using attribution methods.

The contributions of this study are thus:

- Smaller models that have similar predictive capacity than larger models.
- Favourable scaling strategies and optimisation techniques for finding an optimal flare prediction model.
- The most important features that contribute to large flare production.
- Pipeline for creating Liu et al. equivalent datasets.
- Live, proof of concept, flare prediction model.

The objectives listed in Chapter 1 were achieved. The performance and effects of different architectures, optimisation techniques, scaling strategies and attribution methods were investigated and compared. Key findings were outlined and their implications discussed. A perspective on the optimisation and interpretation of solar prediction using DNNs was given.

9.4 Future work

This study complements and supplements research themes concerning binary classification and DNN interpretation, specifically for flare prediction. Subjects that we might want to examine later on include:

- Approach the flare prediction task as a regression problem.
- Retest the LSTM by [4] using our optimisation strategy.
- Adapt the current approach to predict CMEs-associated flares with DNNs.
- Add attention mechanisms to the TCN for the flare prediction task.
- Study attribution methods on a wider range of architectures and examples.
- Investigate whether calibrated networks (in the sense of reliability diagrams) is desirable from a practical perspective.
- Incorporate a larger dataset from multiple sources using solar images and image-derived features.
- Explore techniques to balance the data.

9.5 Conclusion

The task of flare prediction has inspired a host of new approaches and advancements using DNNs versus traditional methods, but still lacks the reliability, performance and interpretation desired. Some approaches used very large and convoluted architectures. This study showed that smaller architectures can achieve similar performance, addressed a few questionable practices from previous studies, and used attribution methods for obtaining the importance of physics-based features responsible for large flare production. We hope that this study will contribute to the improvement and understanding of flare prediction models.

"If only I could be so grossly incandescent!" — Solaire of Astora

Bibliography

[1] NOAA, *Solar Flares (Radio Blackouts) — NOAA / NWS Space Weather Prediction Center*, 2019. [Online]. Available: https://www.swpc.noaa.gov/phenomena/solar-flares-radio-blackouts.

[2] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, and M. Ishii, "Deep Flare Net (DeFN) model for solar flare prediction," *The Astrophysical Journal*, vol. 858, no. 2, p. 113, May 2018. DOI: 10.3847/1538-4357/aab9a7.

[3] E. Park, Y.-J. Moon, S. Shin, *et al.*, "Application of the Deep Convolutional Neural Network to the Forecast of Solar Flare Occurrence Using Full-disk Solar Magnetograms," *The Astrophysical Journal*, vol. 869, no. 2, p. 91, 2018. DOI: 10.3847/1538-4357/aaed40.

[4] H. Liu, C. Liu, J. T. L. Wang, and H. Wang, "Predicting Solar Flares Using a Long Short-term Memory Network," *The Astrophysical Journal*, vol. 877, no. 2, p. 121, Jun. 2019. DOI: 10.3847/1538-4357/ab1b3c.

[5] Y. Chen, W. B. Manchester, A. O. Hero, *et al.*, "Identifying Solar Flare Precursors Using Time Series of SDO/HMI Images and SHARP Parameters," *Space Weather*, vol. 17, no. 10, pp. 1404–1426, 2019. DOI: 10.1029/2019SW002214.

[6] Y. Zheng, X. Li, and X. Wang, "Solar Flare Prediction with the Hybrid Deep Convolutional Neural Network," *The Astrophysical Journal*, vol. 885, no. 1, p. 73, 2019. DOI: 10.3847/1538-4357/ab46bd.

[7] K. Domijan, D. S. Bloomfield, and F. Pitié, "Solar Flare Forecasting from Magnetic Feature Properties Generated by the Solar Monitor Active Region Tracker," *Solar Physics*, vol. 294, no. 1, p. 6, 2019. DOI: 10.1007/s11207-018-1392-4.

[8] K. Yi, Y.-J. Moon, G. Shin, and D. Lim, "Forecast of Major Solar X-Ray Flare Flux Profiles Using Novel Deep Learning Models," *The Astrophysical Journal*, vol. 890, no. 1, p. L5, 2020. DOI: 10.3847/2041-8213/ab701b.

[9] X. Li, Y. Zheng, X. Wang, and L. Wang, "Predicting Solar Flares Using a Novel Deep Convolutional Neural Network," *The Astrophysical Journal*, vol. 891, no. 1, p. 10, 2020. DOI: 10.3847/1538-4357/ab6d04.

[10] B. Panos and L. Kleint, "Real-time flare prediction based on distinctions between flaring and non-flaring active region spectra," *The Astrophysical Journal*, vol. 891, no. 1, p. 17, Nov. 2019. DOI: 10.3847/1538-4357/ab700b.

[11] D. Jakubovitz, R. Giryes, and M. R. Rodrigues, "Generalization Error in Deep Learning," *Applied and Numerical Harmonic Analysis*, pp. 153–193, Aug. 2019. DOI: 10.1007/978-3-319-73074-5{_}5.

[12] G. Montavon, W. Samek, and K. R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing: A Review Journal*, vol. 73, pp. 1–15, Jun. 2018, ISSN: 10512004. DOI: 10.1016/j.dsp.2017.10.011.

[13] S. Toriumi and H. Wang, "Flare-productive active regions," *Living Reviews in Solar Physics*, vol. 16, no. 1, p. 3, Apr. 2019. DOI: 10.1007/s41116-019-0019-7.

[14] D. N. Baker, E. Daly, I. Daglis, J. G. Kappenman, and M. Panasyuk, "Effects of Space Weather on Technology Infrastructure," *Space Weather*, vol. 2, no. 2, n/a–n/a, Feb. 2004. DOI: 10.1029/2003sw000044.

[15] R. J. Redmon, D. B. Seaton, R. Steenburgh, J. He, and J. V. Rodriguez, "September 2017's Geoeffective Space Weather and Impacts to Caribbean Radio Communications During Hurricane Response," *Space Weather*, vol. 16, no. 9, pp. 1190–1201, Sep. 2018. DOI: 10.1029/2018SW001897.

[16] W. D. Pesnell, "Solar Dynamics Observatory (SDO)," in *Handbook of Cosmic Hazards and Planetary Defense*, Cham: Springer International Publishing, 2014, pp. 1–15. DOI: 10.1007/978-3-319-02847-7{_}16-1.

[17] S. J. Mumford, S. Christe, D. Pérez-Suárez, *et al.*, "SunPy - Python for solar physics," *Computational Science and Discovery*, vol. 8, no. 1, 2015. DOI: 10.1088/1749-4699/8/1/014009. [Online]. Available: http://sunpy.org.

[18] *GOES X-ray Flux — NOAA / NWS Space Weather Prediction Center*. [Online]. Available: https://www.swpc.noaa.gov/products/goes-x-ray-flux.

[19] B. Dennis and R. Milligan, "Solar Satellites," *Scholarpedia*, vol. 5, no. 7, p. 6139, 2010. DOI: 10.4249/scholarpedia.6139.

[20] V. Domingo, B. Fleck, and A. I. Poland, "The SOHO mission: An overview," *Solar Physics*, vol. 162, no. 1-2, pp. 1–37, Dec. 1995. DOI: 10.1007/BF00733425.

[21] W. D. Pesnell, B. J. Thompson, and P. C. Chamberlin, "The Solar Dynamics Observatory (SDO)," *Solar Physics*, vol. 275, no. 1-2, pp. 3–15, Jan. 2012. DOI: 10.1007/s11207-011-9841-3.

[22] J. Schou, P. H. Scherrer, R. I. Bush, *et al.*, "Design and Ground Calibration of the Helioseismic and Magnetic Imager (HMI) Instrument on the Solar Dynamics Observatory (SDO)," *Solar Physics*, vol. 275, no. 1-2, pp. 229–259, Jan. 2012. DOI: 10.1007/s11207-011-9842-2.

[23] B. N. Handy, L. W. Acton, C. C. Kankelborg, *et al.*, "The transition region and coronal explorer," *Solar Physics*, vol. 187, no. 2, pp. 229–260, 1999. DOI: 10.1023/A:1005166902804.

[24] R. P. Lin, B. R. Dennis, G. J. Hurford, *et al.*, "The Reuven Ramaty High-Energy Solar Spectroscopic Imager (RHESSI)," *Solar Physics*, vol. 210, no. 1-2, pp. 3–32, 2002. DOI: 10.1023/A:1022428818870.

[25] G. Rottman, *The SORCE mission*, Aug. 2005. DOI: 10.1007/s11207-005-8112-6.

[26] T. Kosugi, K. Matsuzaki, T. Sakao, *et al.*, "The Hinode (Solar-B) mission: An overview," *Solar Physics*, vol. 243, no. 1, pp. 3–17, Jun. 2007. DOI: 10.1007/s11207-007-9014-6.

[27] M. L. Kaiser, T. A. Kucera, J. M. Davila, O. C. Cyr, M. Guhathakurta, and E. Christian, "The STEREO mission: An introduction," *Space Science Reviews*, vol. 136, no. 1-4, pp. 5–16, Apr. 2008, ISSN: 00386308. DOI: 10.1007/s11214-007-9277-0.

[28] S. H. Schwabe, "Die Sonne," *Astronomische Nachrichten*, vol. 20, no. 17, pp. 283–286, Jan. 1843. DOI: 10.1002/asna.18430201706.

[29] D. H. Hathaway, "The solar cycle," *Living Reviews in Solar Physics*, vol. 12, no. 1, p. 4, Sep. 2015. DOI: 10.1007/lrsp-2015-4.

[30] H. Künzel, "M. Waldmeier: The sunspot-activity in the years 1610-1960. Zürich 1961 : Verlag Schulthess u. Co. AG," *Astronomische Nachrichten*, vol. 286, no. 6, pp. 285–286, Mar. 2006. DOI: 10.1002/asna.19622860613.

[31] K. Denkmayr and P. Cugnon, "About sunspot number medium-term predictions," in *Solar-Terrestrial Prediction Workshop V, edited by: Heckman, G., Maruboshi, K., Shea, MA, Smart, DF, and Thompson, R., Hiraiso Solar Terrestrial Research Center, Japan*, vol. 103, 1997.

[32] B. SILSO data/image Royal Observatory of Belgium, *Daily and monthly sunspot number (last 13 years) — SILSO*, 2019. [Online]. Available: http://sidc.be/silso/dayssnplot.

[33] J. Janssens, *Butterfly-Diagram*. [Online]. Available: http://users.telenet.be/j.janssens/SC24web/SC24.html#Butterflyflares.

[34] The Editors of Encyclopaedia Britannica, *The Mechanical Turk: AI Marvel or Parlor Trick?* [Online]. Available: https://www.britannica.com/story/the-mechanical-turk-ai-marvel-or-parlor-trick.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. DOI: 10.1038/nature14539. [Online]. Available: http://www.nature.com/articles/nature14539.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016, pp. 164–223. [Online]. Available: http://www.deeplearningbook.org.

[37] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[38] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv*, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.01271.

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[40] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, Nov. 2016. DOI: 10.1109/TIE.2016.2582729.

[41] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional Neural Networks for patient-specific ECG classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, vol. 2015-Novem, IEEE, Aug. 2015, pp. 2608–2611, ISBN: 978-1-4244-9271-8. DOI: 10.1109/EMBC.2015.7318926.

[42] R. T. J. McAteer, P. T. Gallagher, and J. Ireland, "Statistics of Active Region Complexity: A Large-Scale Fractal Dimension Survey," *The Astrophysical Journal*, vol. 631, no. 1, pp. 628–635, Sep. 2005. DOI: 10.1086/432412.

[43] M. G. Bobra, S. Couvidat, and W. W. Hansen, "Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm," *Astrophysical Journal*, vol. 798, no. 2, p. 135, 2015. DOI: 10.1088/0004-637X/798/2/135.

[44] D. S. Bloomfield, P. A. Higgins, R. T. McAteer, and P. T. Gallagher, "Toward reliable benchmarking of solar flare forecasting methods," *Astrophysical Journal Letters*, vol. 747, no. 2, p. L41, Feb. 2012. DOI: 10.1088/2041-8205/747/2/L41.

[45] E. Jonas, M. Bobra, V. Shankar, J. Todd Hoeksema, and B. Recht, "Flare Prediction Using Photospheric and Coronal Image Data," *Solar Physics*, vol. 293, no. 3, p. 48, Aug. 2018, ISSN: 1573093X. DOI: 10.1007/s11207-018-1258-9.

[46] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," *Advances in Neural Information Processing Systems*, vol. 3, no. January, pp. 2627–2635, Jun. 2014. [Online]. Available: http://arxiv.org/abs/1406.3332.

[47] G. PEARCE and R. HARRISON, "Sympathetic flaring," *Astronomy and astrophysics (Berlin. Print)*, vol. 228, no. 2, pp. 513–516, 1990, ISSN: 0004-6361.

[48] T. Gombosi and the SOLSTICE Team, "SOLSTICE: Space Weather Modeling Meets Machine Learning," *EGU General Assembly 2020*, vol. EGU2020-52, 0. DOI: https://doi.org/10.5194/egusphere-egu2020-5224.

[49] K. Florios, I. Kontogiannis, S. H. Park, *et al.*, "Forecasting Solar Flares Using Magnetogram-based Predictors and Machine Learning," *Solar Physics*, vol. 293, no. 2, p. 28, Jan. 2018, ISSN: 1573093X. DOI: 10.1007/s11207-018-1250-4.

[50] R. Qahwaji, T. Colak, M. Al-Omari, and S. Ipson, "Automated prediction of CMEs using machine learning of CME-Flare associations," in *Solar Image Analysis and Visualization*, 2, vol. 248, Springer Netherlands, Apr. 2009, pp. 261–273, ISBN: 9780387981536. DOI: 10.1007/978-0-387-98154-3{_}19.

[51] Y. Yuan, F. Y. Shih, J. Jing, and H.-M. Wang, "Automated flare forecasting using a statistical learning technique," *Research in Astronomy and Astrophysics*, vol. 10, no. 8, pp. 785–796, Aug. 2010. DOI: 10.1088/1674-4527/10/8/008.

[52] L. E. Boucheron, A. Al-Ghraibah, and R. T. J. McAteer, "Prediction of solar flare size and time-to-flare using support vector machine regression," *The Astrophysical Journal*, vol. 812, no. 1, p. 51, Oct. 2015. DOI: 10.1088/0004-637X/812/1/51.

[53] T. Muranushi, T. Shibayama, Y. H. Muranushi, *et al.*, "UFCORIN: A fully automated predictor of solar flares in GOES X-ray flux," *Space Weather*, vol. 13, no. 11, pp. 778–796, Jul. 2015, ISSN: 15427390. DOI: 10.1002/2015SW001257.

[54] M. I. Jordan, "Chapter 25 Serial order: A parallel distributed processing approach," in *Advances in Psychology*, C, vol. 121, Elsevier, Jan. 1997, pp. 471–495. DOI: 10.1016/S0166-4115(97)80111-2.

[55] G. Barnes, K. D. Leka, C. J. Schrijver, *et al.*, "A Comparison of Flare Forecasting Methods. I. Results From the "All-Clear" Workshop," *The Astrophysical Journal*, vol. 829, no. 2, p. 89, Aug. 2016. DOI: 10.3847/0004-637x/829/2/89.

[56] C. Liu, N. Deng, J. T. Wang, and H. Wang, "Predicting solar flares using sdo/hmi vector magnetic data product and random forest algorithm," *arXiv*, vol. 843, no. 2, p. 104, Jul. 2017. DOI: 10.3847/1538-4357/aa789b.

[57] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, Jun. 2020. DOI: 10.1145/3422622.

[58] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Muller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer Nature, 2019, vol. 11700, p. 435, ISBN: 9783030289539. [Online]. Available: http://arxiv.org/abs/1606.06565.

[59] A. Wan, L. Dunlap, D. Ho, *et al.*, "Nbdt: Neural-Backed decision trees," *arXiv*, 2020. [Online]. Available: http://arxiv.org/abs/2004.00221.

[60] R. Agarwal, N. Frosst, X. Zhang, R. Caruana, and G. E. Hinton, "Neural Additive Models: Interpretable Machine Learning with Neural Nets," 2020. [Online]. Available: http://arxiv.org/abs/2004.13912.

[61] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for Deep Neural Networks," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1711.06104.

[62] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, New York,

NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144, ISBN: 9781450342322. DOI: 10.1145/2939672.2939778.

[63] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Toward Interpretable Machine Learning: Transparent Deep Neural Networks and Beyond," 2020. [Online]. Available: http://arxiv.org/abs/2003.07631.

[64] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable Machine Learning for Scientific Insights and Discoveries," *IEEE Access*, vol. 8, pp. 42 200–42 216, May 2019. DOI: 10.1109/ACCESS.2020.2976199.

[65] C. Rudin, *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*, May 2019. DOI: 10.1038/s42256-019-0048-x.

[66] L. Merrick, "Randomized ablation feature importance," *arXiv*, Sep. 2019. [Online]. Available: http://arxiv.org/abs/1910.00174.

[67] E. Štrumbelj and I. Kononenko, "An efficient explanation of individual classifications using game theory," *Journal of Machine Learning Research*, vol. 11, pp. 1–18, 2010, ISSN: 15324435. [Online]. Available: https://www.jmlr.org/papers/volume11/strumbelj10a/strumbelj10a.pdf.

[68] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *34th International Conference on Machine Learning, ICML 2017*, vol. 7, pp. 4844–4866, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1704.02685.

[69] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *34th International Conference on Machine Learning, ICML 2017*, vol. 7, pp. 5109–5118, Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.01365.

[70] *Captum · Model Interpretability for PyTorch*. [Online]. Available: https://captum.ai/.

[71] N. Nishizuka, Y. Kubo, K. Sugiura, M. Den, and M. Ishii, "Reliable Probability Forecast of Solar Flares: Deep Flare Net-Reliable (DeFN-R)," *The Astrophysical Journal*, vol. 899, no. 2, p. 150, 2020. DOI: 10.3847/1538-4357/aba2f2.

[72] G. W. BRIER, "Verification of Forecasts Expressed in Terms of Probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, Jan. 1950. DOI: 10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2.

[73] D. S. Wilks, "Sampling distributions of the Brier score and Brier skill score under serial dependence," *Quarterly Journal of the Royal Meteorological Society*, vol. 136, no. 653, pp. 2109–2118, Oct. 2010, ISSN: 00359009. DOI: 10.1002/qj.709.

[74] J. T. Hoeksema, Y. Liu, K. Hayashi, *et al.*, "The Helioseismic and Magnetic Imager (HMI) Vector Magnetic Field Pipeline: Overview and Performance," *Solar Physics*, vol. 289, no. 9, pp. 3483–3530, Apr. 2014, ISSN: 1573093X. DOI: 10.1007/s11207-014-0516-8.

[75] G. H. Fisher, D. J. Bercik, B. T. Welsch, and H. S. Hudson, "Global forces in eruptive solar flares: The lorentz force acting on the solar atmosphere and the solar interior," *Solar Flare Magnetic Fields and Plasmas*, pp. 59–76, Jun. 2012, ISSN: 00380938. DOI: 10.1007/s11207-011-9907-2.

[76] X. Sun, "The CGEM Lorentz Force Data from HMI Vector Magnetograms," May 2019. [Online]. Available: http://arxiv.org/abs/1405.7353.

[77] W. T. Thompson, "Coordinate systems for solar image data," *Astronomy & Astrophysics*, vol. 449, no. 2, pp. 791–803, Apr. 2006. DOI: 10.1051/0004-6361:20054262.

[78] M. Moldwin, *An introduction to space weather*. Cambridge: Cambridge University Press, Jan. 2008, pp. 1–142, ISBN: 9780511801365. DOI: 10.1017/CBO9780511801365.

[79] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, S.-I. Watari, and M. Ishii, "Solar Flare Prediction Using Machine Learning with Multiwavelength Observations," *Proceedings of the International Astronomical Union*, vol. 13, no. S335, pp. 310–313, 2017. DOI: 10.1017/s1743921317007293.

[80] SWS, *SWS - The Sun and Solar Activity - Sunspots and Sunspot Groups*. [Online]. Available: https://www.sws.bom.gov.au/Educational/2/2/2%0Ahttps://www.sws.bom.gov.au/Educational/2/2/3.

[81] R. A. Johnson, "A new family of power transformations or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954–959, 2010. DOI: 10.2307/2673623.

[82] R. D'Agostino and E. S. Pearson, "Tests for Departure from Normality. Empirical Results for the Distributions of b 2 and b 1," *Biometrika*, vol. 60, no. 3, p. 613, Dec. 1973, ISSN: 00063444. DOI: 10.2307/2335012.

[83] A. Ghasemi and S. Zahediasl, "Normality Tests for Statistical Analysis: A Guide for Non-Statisticians," *International Journal of Endocrinology and Metabolism*, vol. 10, no. 2, pp. 486–489, Dec. 2012. DOI: 10.5812/ijem.3505.

[84] E. Limpert, W. A. Stahel, and M. Abbt, "Log-normal distributions across the sciences: Keys and clues," *BioScience*, vol. 51, no. 5, pp. 341–352, 2001. DOI: 10.1641/0006-3568(2001)051[0341:LNDATS]2.0.CO;2. [Online]. Available: https://academic.oup.com/bioscience/article/51/5/341/243981.

[85] Real-Statics.com, *Assumptions for ANOVA — Real Statistics Using Excel*, 2018. [Online]. Available: http://www.real-statistics.com/one-way-analysis-of-variance-anova/assumptions-anova/.

[86] C. Zaiontz, *Basic Concepts of Correlation — Real Statistics Using Excel*. [Online]. Available: http://www.real-statistics.com/correlation/basic-concepts-correlation/.

[87] Kent State University Libraries, {*SPSS*} *Tutorials: Pearson Correlation*, Jul. 2013. [Online]. Available: https://libguides.library.kent.edu/SPSS/PearsonCorr.

[88] P. Schober and L. A. Schwarte, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia and Analgesia*, vol. 126, no. 5, pp. 1763–1768, May 2018. DOI: 10.1213/ANE.0000000000002864.

[89] Lund Research ltd., *Spearman's Rank-Order Correlation - A guide to when to use it, what it does and what the assumptions are.* 2018. [Online]. Available: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php.

[90] C. E. Shannon and W. Weaver, "The mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 623–656, 1949.

[91] L. F. Kozachenko and N. N. Leonenko, "Sample Estimate of the Entropy of a Random Vector.," *Problems of information transmission*, vol. 23, no. 2, pp. 95–101, 1987, ISSN: 00329460. [Online]. Available: http://mi.mathnet.ru/ppi797.

[92] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 69, no. 6, p. 16, May 2004. DOI: 10.1103/PhysRevE.69.066138.

[93] B. C. Ross, "Mutual information between discrete and continuous data sets," *PLoS ONE*, vol. 9, no. 2, e87357, Feb. 2014, ISSN: 19326203. DOI: 10.1371/journal.pone.0087357.

[94] *scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation.* [Online]. Available: https://scikit-learn.org/stable/.

[95] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010, pp. 807–814, ISBN: 9781605589077.

[96] Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020. DOI: 10.1109/ACCESS.2019.2962617.

[97] L. Bottou, "On-line Learning and Stochastic Approximations," in *On-Line Learning in Neural Networks*, Cambridge University Press, Jan. 1999, pp. 9–42. DOI: 10.1017/CBO9780511569920.003.

[98] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning, ICML 2013*, 2013, pp. 2176–2184. [Online]. Available: http://www.cs.toronto.edu/~fritz/absps/momentum.pdf.

[99] S. L. Smith, P. J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv*, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1711.00489.

[100] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," Tech. Rep., 2017. [Online]. Available: https://arxiv.org/abs/1609.04836.

[101] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014, ISSN: 15337928. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[102] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," pp. 950–957, 1992. [Online]. Available: https://papers.nips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[103] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, Feb. 2015. [Online]. Available: http://arxiv.org/abs/1502.03167.

[104] S. De and S. L. Smith, "Batch Normalization Has Multiple Benefits: an Empirical Study on Residual Networks," Tech. Rep., Sep. 2019.

[105] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 2677–2685, Jan. 2019. DOI: 10.1109/CVPR.2019.00279.

[106] D. G. Altman and J. M. Bland, "Standard deviations and standard errors," *BMJ*, vol. 331, no. 7521, p. 903, Oct. 2005. DOI: 10.1136/bmj.331.7521.903.

[107] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1026–1034, Feb. 2015. DOI: 10.1109/ICCV.2015.123.

[108] *PyTorch*. [Online]. Available: https://pytorch.org/.

[109] L. Biewald, *Experiment Tracking with Weights & Biases*, 2020. [Online]. Available: https://www.wandb.com/.

[110] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," *arXiv*, p. 36, Aug. 2017. DOI: 10.1117/12.2520589.

[111] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, 2009, pp. 41–48, ISBN: 9781605585161. [Online]. Available: https://mila.quebec/wp-content/uploads/2019/08/2009_curriculum_icml.pdf.

[112] R. Gallego, A. Alves, A. Monticelli, and R. Romero, "Parallel simulated annealing applied to long term transmission network expansion planning," *IEEE Transactions on Power Systems*, vol. 12, no. 1, pp. 181–188, 1997. DOI: 10.1109/59.574938.

[113] fast.ai Community, *callbacks.lr_finder — fastai*. [Online]. Available: https://fastai1.fast.ai/callbacks.lr_finder.html%20https://docs.fast.ai/callbacks.lr_finder.html.

[114] S. Guggel, *Another data science student's blog – The 1cycle policy*, 2018. [Online]. Available: https://sgugger.github.io/the-1cycle-policy.html.

[115] K. D. Leka, S. H. Park, K. Kusano, *et al.*, "A comparison of flare forecasting methods. II. Benchmarks, metrics and performance results for operational solar flare forecasting systems," *arXiv*, vol. 243, no. 2, p. 36, Jul. 2019. DOI: 10.3847/1538-4365/ab2e12.

[116] C. Marzban, "The ROC curve and the area under it as performance measures," *Weather and Forecasting*, vol. 19, no. 6, pp. 1106–1114, Dec. 2004. DOI: 10.1175/825.1.

[117] J. Bröcker and L. A. Smith, "Increasing the reliability of reliability diagrams," *Weather and Forecasting*, vol. 22, no. 3, pp. 651–661, Jun. 2007. DOI: 10.1175/WAF993.1.

[118] P. Branco, L. Torgo, and R. Ribeiro, "A Survey of Predictive Modelling under Imbalanced Distributions," 2015. [Online]. Available: http://arxiv.org/abs/1505.01658.

[119] H. Tawfeig, V. S. Asirvadam, and N. Saad, "Sliding-window learning using MLP networks with data store management," in *2011 National Postgraduate Conference - Energy and Sustainability: Exploring the Innovative Minds, NPC 2011*, 2011, ISBN: 9781457718847. DOI: 10.1109/NatPC.2011.6136391.

[120] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014. [Online]. Available: http://arxiv.org/abs/1412.3555.

[121] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar. 1989. DOI: 10.1109/29.21701.

[122] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, no. APR, pp. 770–778, Dec. 2016. DOI: 10.1109/CVPR.2016.90.

[123] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909. [Online]. Available: https://arxiv.org/abs/1602.07868.

[124] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," 2018. [Online]. Available: http://arxiv.org/abs/1811.12808.

[125] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings*, Dec. 2014. [Online]. Available: http://arxiv.org/abs/1312.6034.

[126] T. Gneiting, F. Balabdaoui, and A. E. Raftery, "Probabilistic forecasts, calibration and sharpness," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 2, pp. 243–268, Apr. 2007. DOI: 10.1111/j.1467-9868.2007.00587.x.

[127] Y. Kubo, "Why do some probabilistic forecasts lack reliability?" *Journal of Space Weather and Space Climate*, vol. 9, A17, Apr. 2019. DOI: 10.1051/swsc/2019016.

Appendix A

Supplemental Figures

This appendix contains supplemental figures, referred to in the main text.

A.1 Appendix: Chapter 2

A.1.1 Liu et al.'s [4] cross-validation technique

Figure A.1: Visual depiction of Liu et al.'s cross-validation technique. The ' ' parameter indicates that all the folds except the one indicated are trained and validated on. Each partition in the dataset is split into ten folds. Each iteration, the model is trained on nine of the ten folds. Folds are shuffled before training. Based on [4].

A.2 Appendix: Chapter 3

Table A.1: Overview of the 40 Features (25 SDO/HMI magnetic parameters [74], [75], 15 flare history features [45], [79]). Reproduced from Liu et al [4] with permission.

Keyword	Description	Formula
TOTUSJH	Total unsigned current helicity	$H_{c_{total}} \propto \sum \lvert B_z \cdot J_z \rvert$
TOTPOT	Total photospheric magnetic free energy density	$\rho_{tot} \propto \sum (\boldsymbol{B}^{Obs} - \boldsymbol{B}^{Pot})^2 dA$
TOTUSJZ	Total unsigned vertical current	$J_{z_{total}} = \sum \lvert J_z \rvert dA$
ABSNJZH	Absolute value of the net current helicity	$H_{c_{\mathrm{abs}}} \propto \lvert \sum B_z \cdot J_z \rvert$
SAVNCPP	Sum of the modulus of the net current per polarity	$J_{z_{\mathrm{sum}}} \propto \left\lvert \sum^{B^+} J_z dA \right\rvert + \left\lvert \sum^{B_z^-} J_z dA \right\rvert$
USFLUX	Total unsigned flux	$\Phi = \sum \lvert B_z \rvert \, dA$
AREA_ACR	Area of strong field pixels in the active region	Area $= \sum$ Pixels
MEANPOT	Mean photospheric magnetic free energy	$\bar{\rho} \propto \frac{1}{N} \sum \left(\boldsymbol{B}^{\mathrm{Obs}} - \boldsymbol{B}^{\mathrm{Pot}} \right)^2$
R_VALUE	Sum of flux near polarity inversion line	$\Phi = \sum \lvert B_{\mathrm{Los}} \rvert \, dA$ within R mask
SHRGT45	Fraction of area with shear >45°	Area with shear $> 45°$/ total area
MEANSHR	Mean shear angle	$\bar{\Gamma} = \frac{1}{N} \sum \arccos \left(\frac{\boldsymbol{B}^{\mathrm{Obs}} \cdot \boldsymbol{B}^{\mathrm{Pot}}}{\lvert \boldsymbol{B}^{\mathrm{Obs}} \rvert \lvert \boldsymbol{B}^{\mathrm{Pot}} \rvert} \right)$
MEANGAM	Mean angle of field from radial	$\bar{\gamma} = \frac{1}{N} \sum \arctan \left(\frac{B_h}{B_z} \right)$
MEANGBT	Mean gradient of total field	$\lvert \nabla B_{\mathrm{tot}} \rvert = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B}{\partial x} \right)^2 + \left(\frac{\partial B}{\partial y} \right)^2}$
MEANGBZ	Mean gradient of vertical field	$\lvert \nabla B_z \rvert = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B_z}{\partial x} \right)^2 + \left(\frac{\partial B_z}{\partial y} \right)^2}$
MEANGBH	Mean gradient of horizontal field	$\lvert \nabla B_h \rvert = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B_h}{\partial x} \right)^2 + \left(\frac{\partial B_h}{\partial y} \right)^2}$
MEANJZH	Mean current helicity	$\overline{H_c} \propto \frac{1}{N} \sum B_z J_z$
MEANJZD	Mean vertical current density	$\overline{J_z} \propto \frac{1}{N} \sum \left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} \right)$
MEANALP	Mean characteristic twist parameter	$\alpha_{\mathrm{total}} \propto \frac{\sum J_z B_z}{\sum B_z^2}$
TOTBSQ	Total magnitude of Lorentz force	$F \propto \sum B^2$
TOTFX	Sum of x-component of Lorentz force	$F_x \propto -\sum B_x B_z dA$
TOTFY	Sum of y-component of Lorentz force	$F_y \propto \sum B_y B_z dA$
TOTFZ	Sum of z-component of Lorentz force	$F_z \propto \sum \left(B_x^2 + B_y^2 - B_z^2 \right) dA$
EPSX	Sum of x-component of normalized Lorentz force	$\delta F_x \propto \frac{\sum B_x B_z}{\sum B^2}$
EPSY	Sum of y-component of normalized Lorentz force	$\delta F_y \propto \frac{-\sum B_y B_z}{\sum B^2}$
EPSZ	Sum of z-component of normalized Lorentz force	$\delta F_z \propto \frac{\sum \left(B_x^2 + B_y^2 - B_z^2 \right)}{\sum B^2}$
Bdec	Time decay value based on the previous B-class flares only	$\mathrm{Bdec}\,(x_t) = \sum_{f_i \in F_B} e^{-\frac{t - t(f_i)}{\tau}}$
Cdec	Time decay value based on the previous C-class flares only	$\mathrm{Cdec}\,(x_t) = \sum_{f_i \in F_C} e^{-\frac{t - t(f_i)}{\tau}}$
Mdec	Time decay value based on the previous M-class flares only	$\mathrm{Mdec}\,(x_t) = \sum_{f_i \in F_M} e^{-\frac{t - t(f_i)}{\tau}}$
Xdec	Time decay value based on the previous X-class flares only	$\mathrm{Xdec}\,(x_t) = \sum_{f_i \in F_X} e^{-\frac{t - t(f_i)}{\tau}}$
Edec	Time decay value based on the magnitudes of all previous flares	$\mathrm{Edec}\,(x_t) = \sum_{f_i \in F} E_i \cdot e^{-\frac{t - t(f_i)}{\tau}}$
logEdec	Time decay value based on the log-magnitudes of all previous flares	$\log \mathrm{Edec}\,(x_t) = \sum_{f_i \in F} \log (E_i) \cdot e^{-\frac{t - t(f_i)}{\tau}}$
Bhis	Total history of B-class flares in an AR	...
Chis	Total history of C-class flares in an AR	...
Mhis	Total history of M-class flares in an AR	...
Xhis	Total history of X-class flares in an AR	...
Bhis1d	1 day history of B-class flares in an AR	...
Chis1d	1 day history of C-class flares in an AR	...
Mhis1d	1 day history of M-class flares in an AR	...
Xhis1d	1 day history of X-class flares in an AR	...
Xmax1d	Maximum X-ray intensity one day before	...

A.3 Appendix: Chapter 4

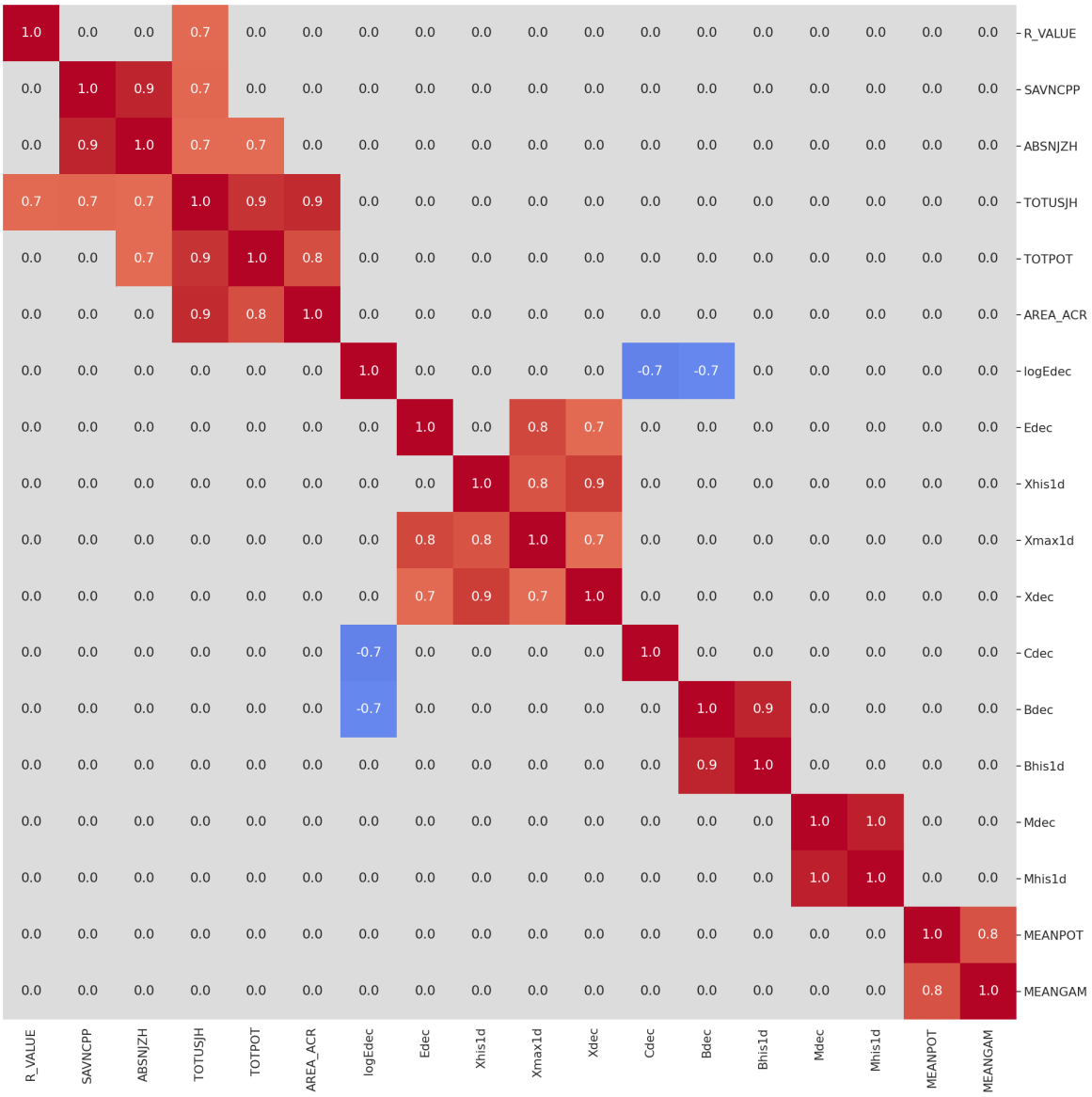


Figure A.2: Feature correlation (Pearson) heat-map of all samples in training set, keeping only one feature per cluster with $\rho > \pm 0.9$. Features are clustered together that are closely linearly correlated. Only correlations larger than $\rho > \pm 0.7$ that are not only correlated with itself are shown. Correlations are rounded up to nearest decimal place.

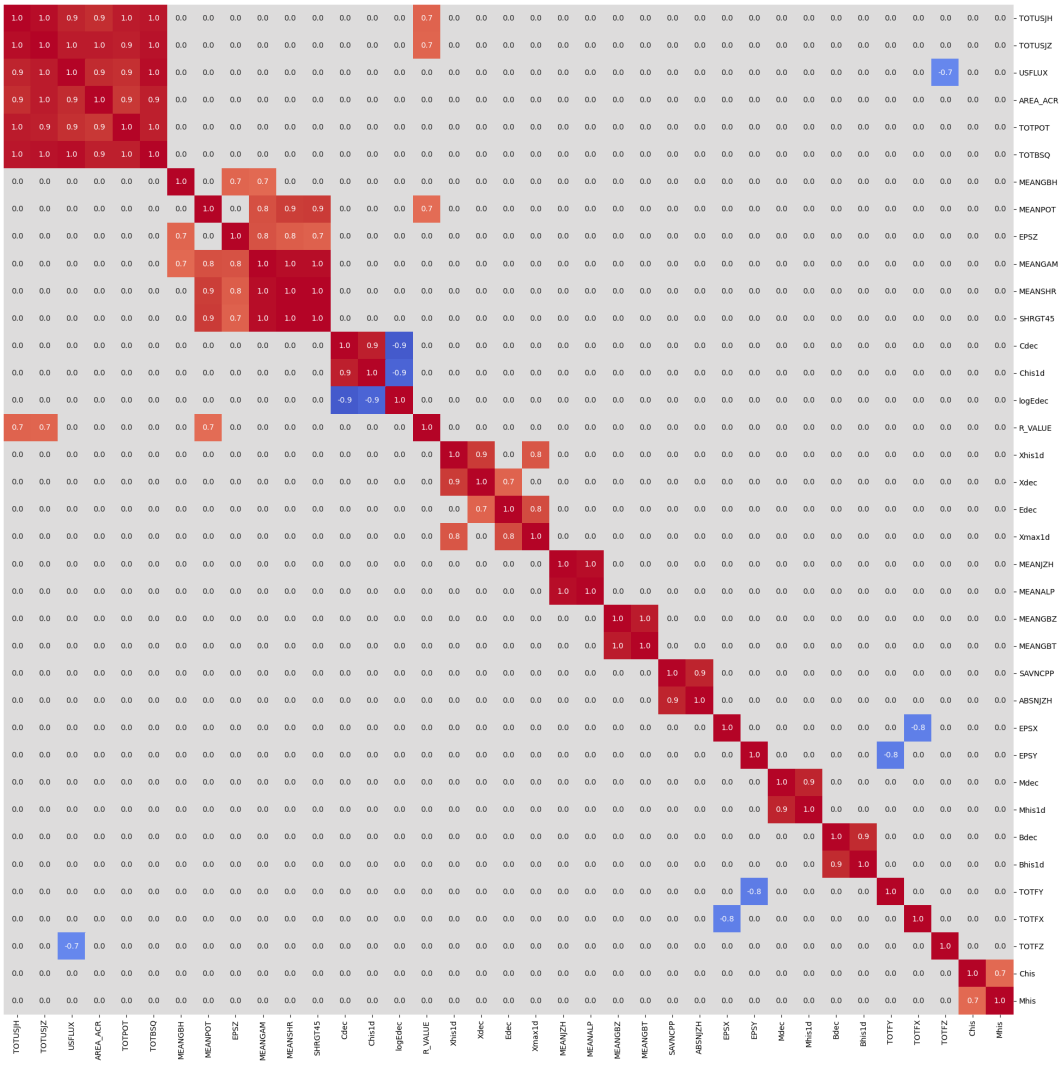


Figure A.3: Feature correlation (Pearson) heat-map of samples of ARs that would eventually flare $\geq M5.0$ in training set. Features are clustered together that are closely linearly correlated. Only correlations larger than $\rho > \pm 0.7$ that are not only correlated with itself are shown. Correlations are rounded up to nearest decimal place.

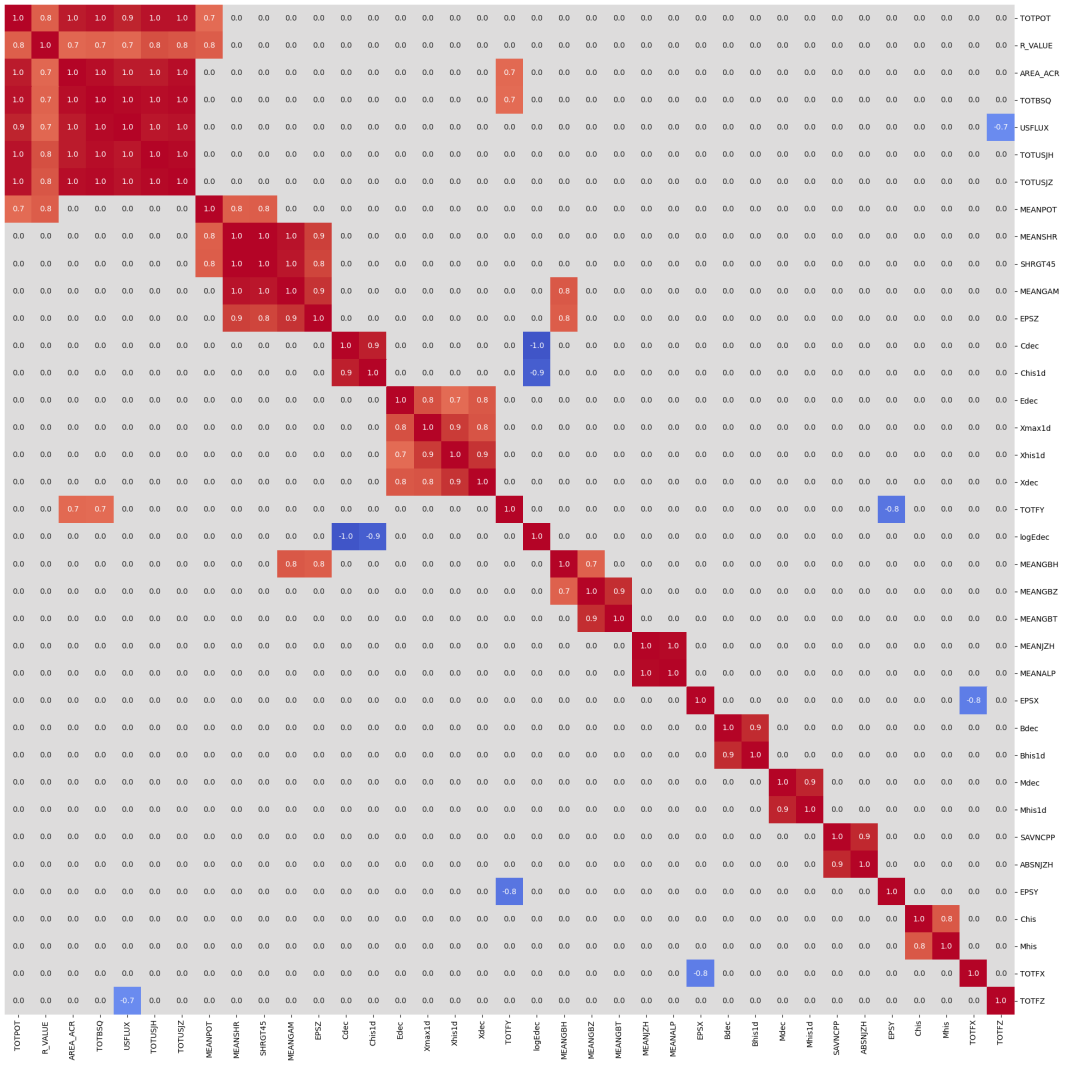


Figure A.4: Feature correlation (Pearson) heat-map of samples 24h before the flare in training set. Features are clustered together that are closely linearly correlated. Only correlations larger than $\rho > \pm 0.7$ that are not only correlated with itself are shown. Correlations are rounded up to nearest decimal place.

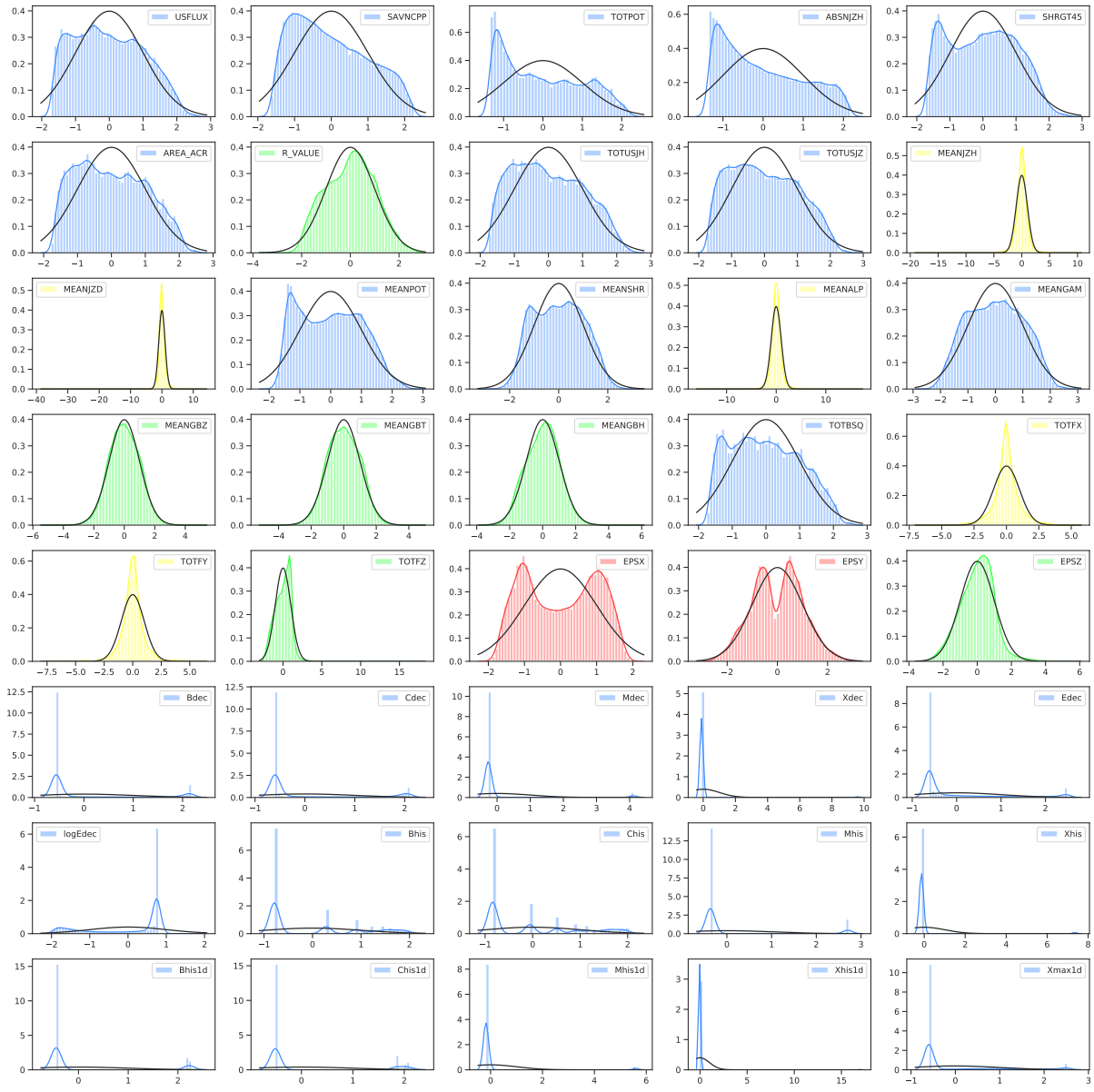


Figure A.5: Histogram of each feature (coloured) and maximum likelihood Gaussian distribution (black). Z-normalized to have zero mean and equal unit variance on all features before plotting. Training set. After Yeo-Johnson power transforms.

A.4 Appendix: Chapter 5

A.4.1 Model optimisation

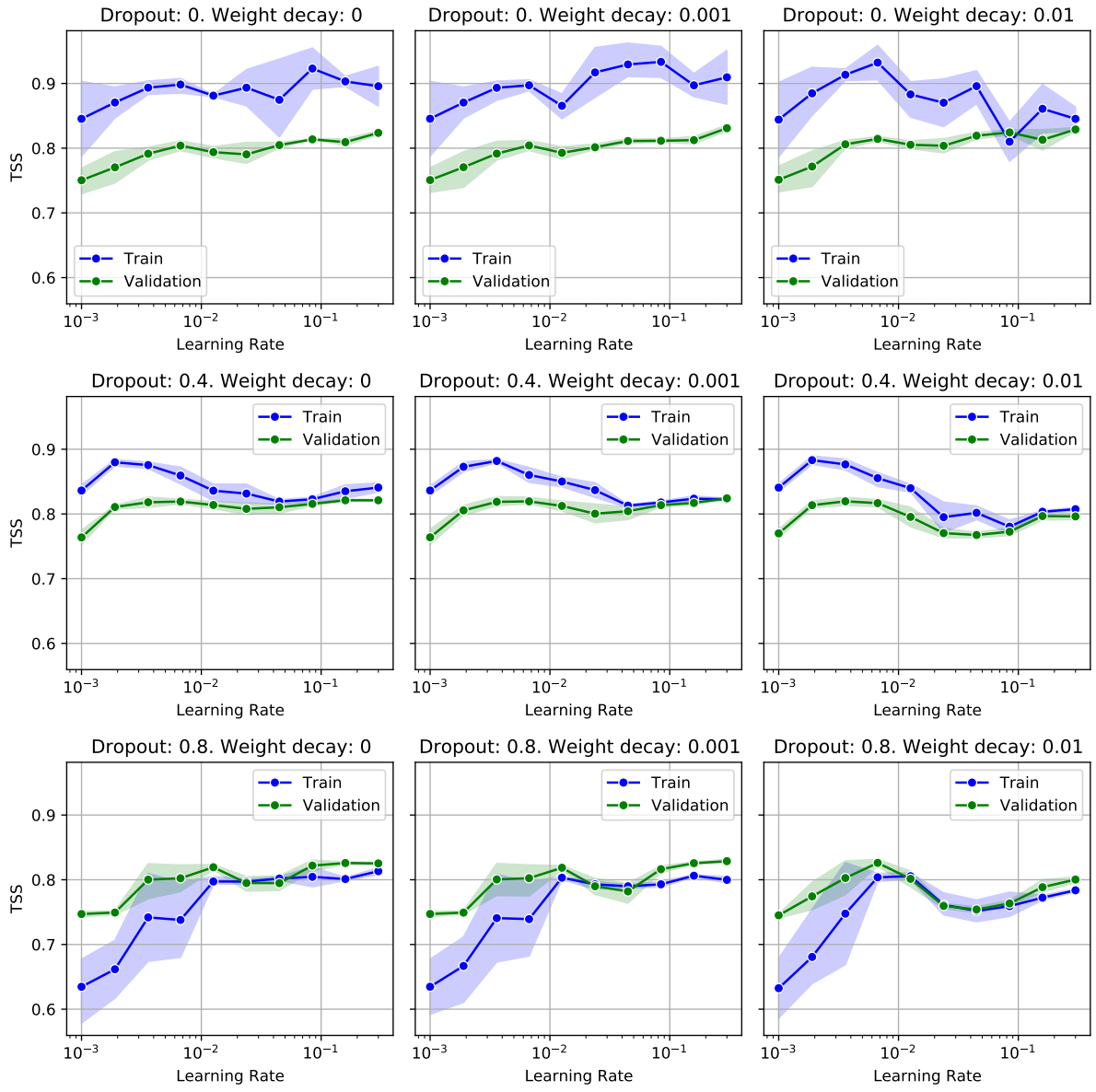


Figure A.6: TSS over LR plot for degrees of regularisation. 2_500 MLP architecture. Training set (blue), validation set (green)

Table A.2: Example of MLP (1_100) GS1 results for 100 nodes and dropout of 0.8 for different learning rates

Nodes	Dropout	Learning Rate	Train TSS	Validation TSS
100	0.8	0.001	0.8039	0.7864
		0.005	0.8991	0.8475
		0.01	0.9147	0.8605
		0.05	0.9093	0.8636
		0.1	0.9269	0.8616
		0.5	0.9005	0.8559
		1	0.8904	0.8570

A.4.2 LR range test. Best number of iterations test.

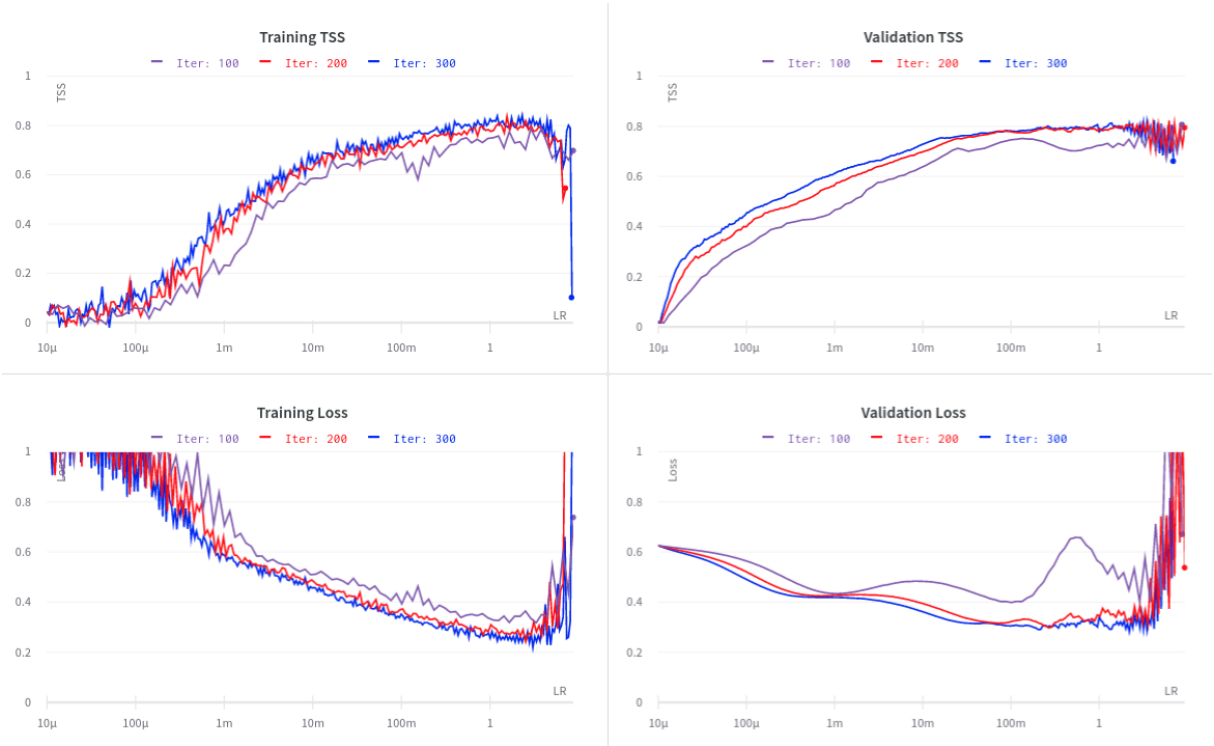


Figure A.7: Best number of iterations (Iter) for LR range test. Higher is better, but computationally expensive. 200 Iterations is a good trade-off.

A.4.3 Evaluation plots

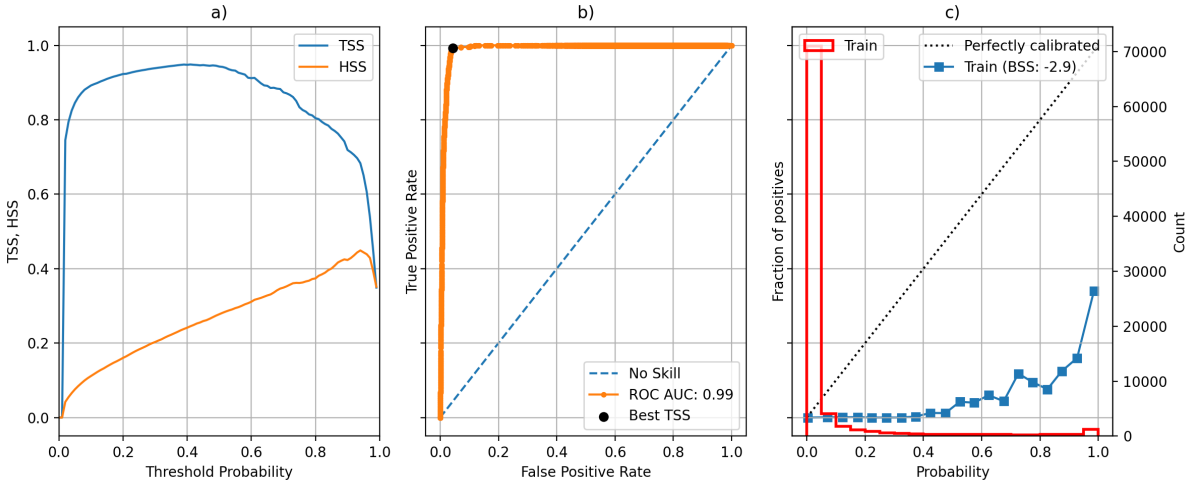


Figure A.8: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (2_500). Training set.

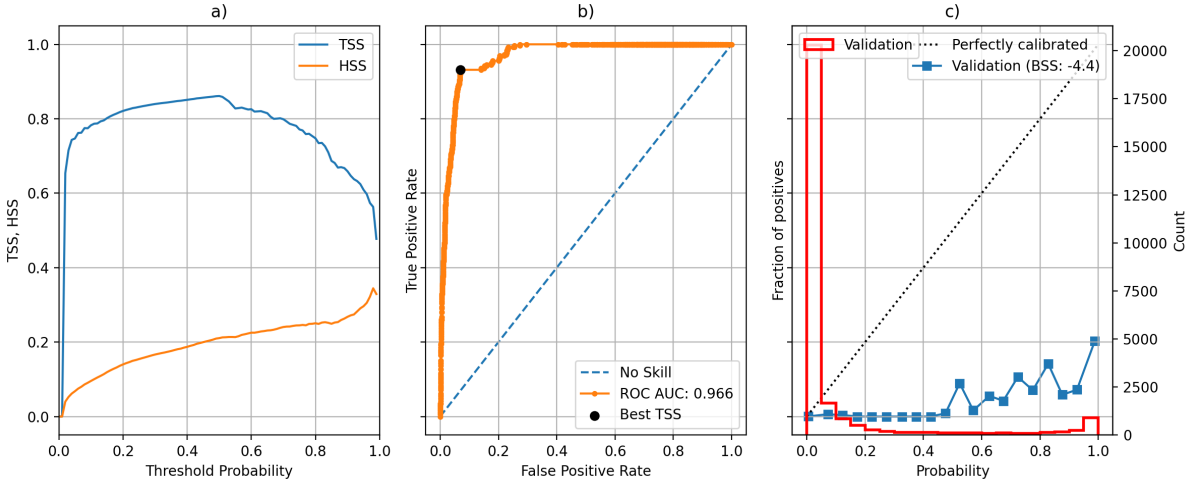


Figure A.9: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (2_500). Validation set.

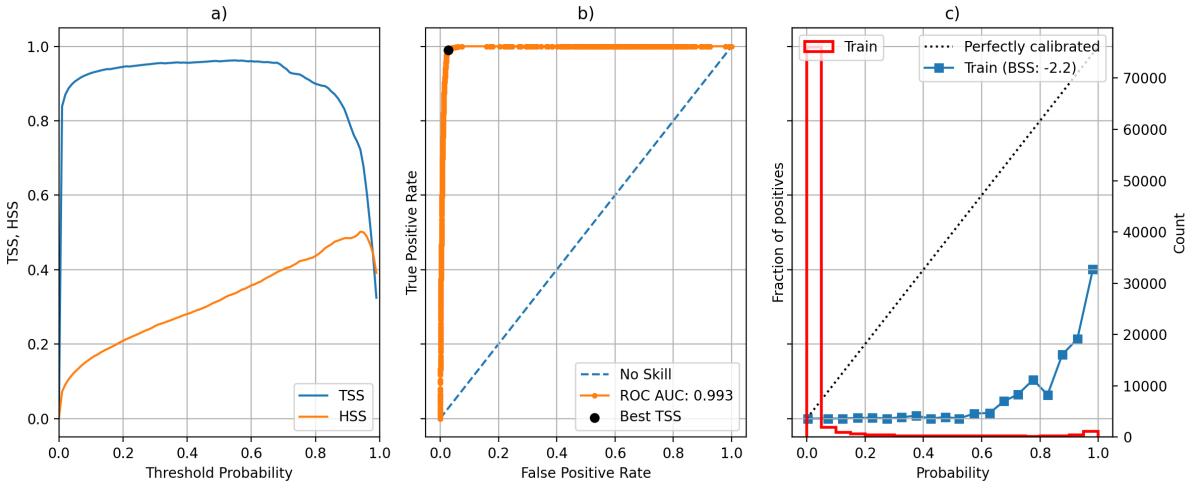


Figure A.10: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (1_100). Training set.

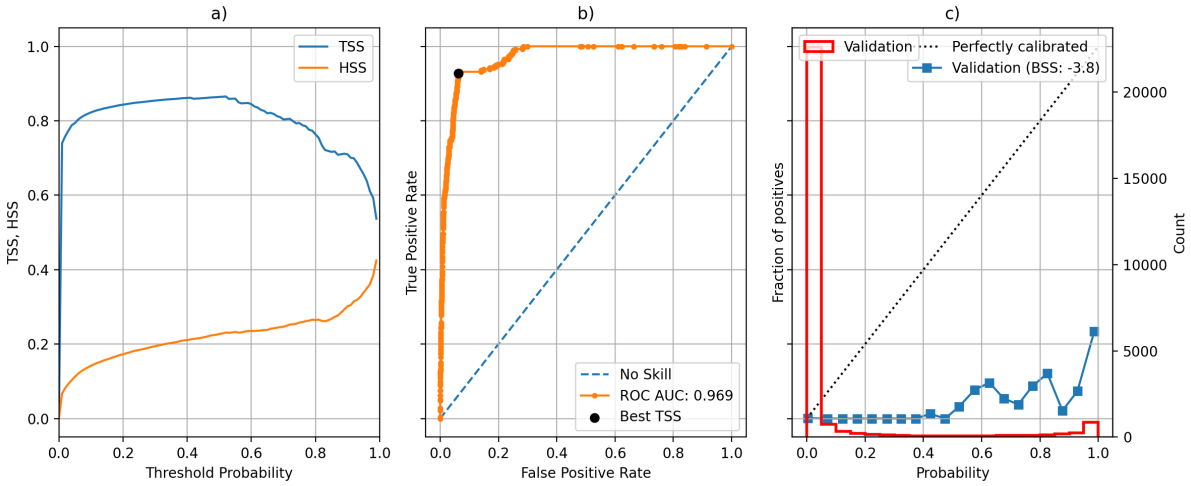


Figure A.11: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (1_100). Validation set.

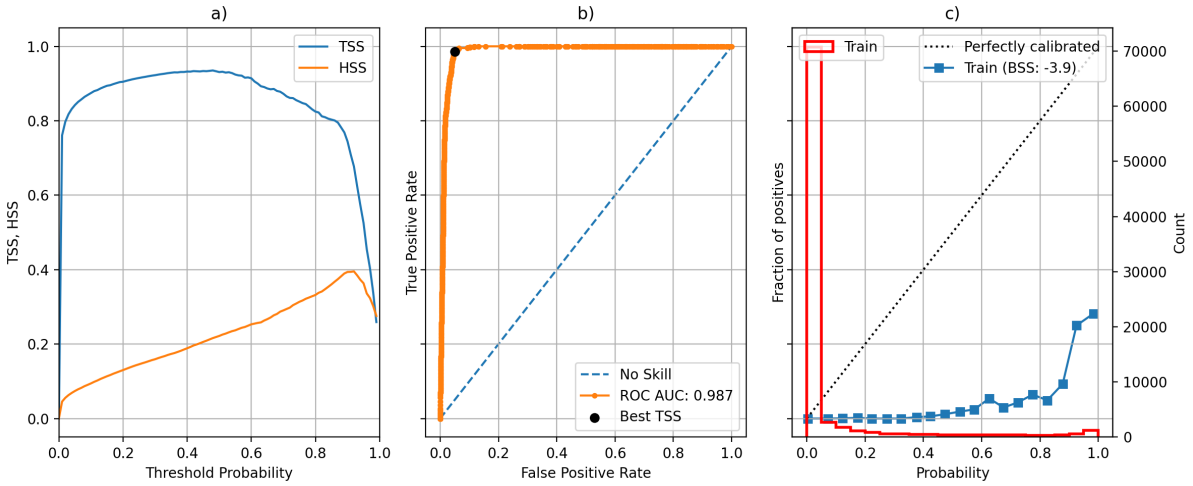


Figure A.12: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (1_100) OCLR. Training set.

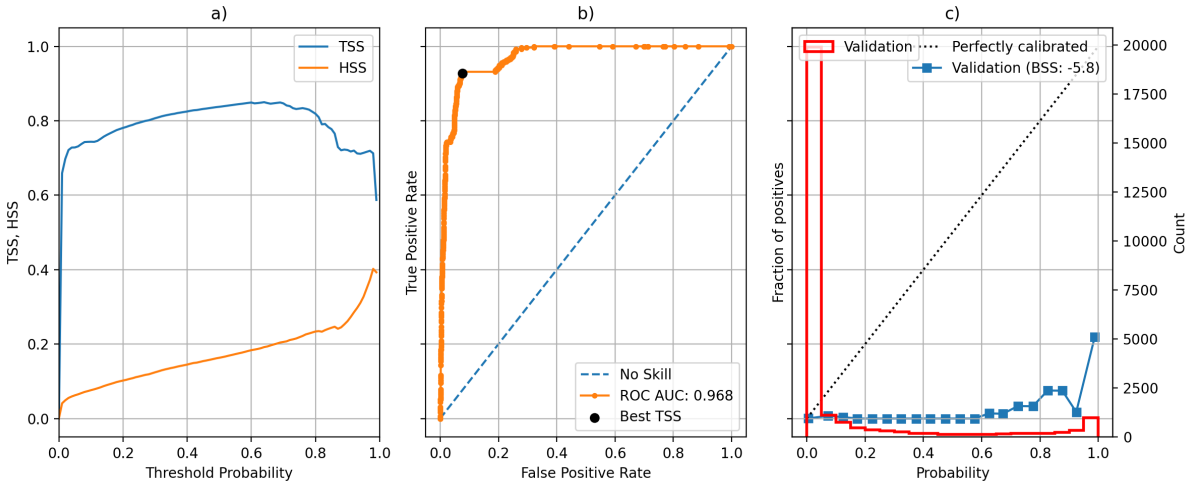


Figure A.13: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. MLP (1_100) OCLR. Validation set.

A.4.4 Precision-Recall curves

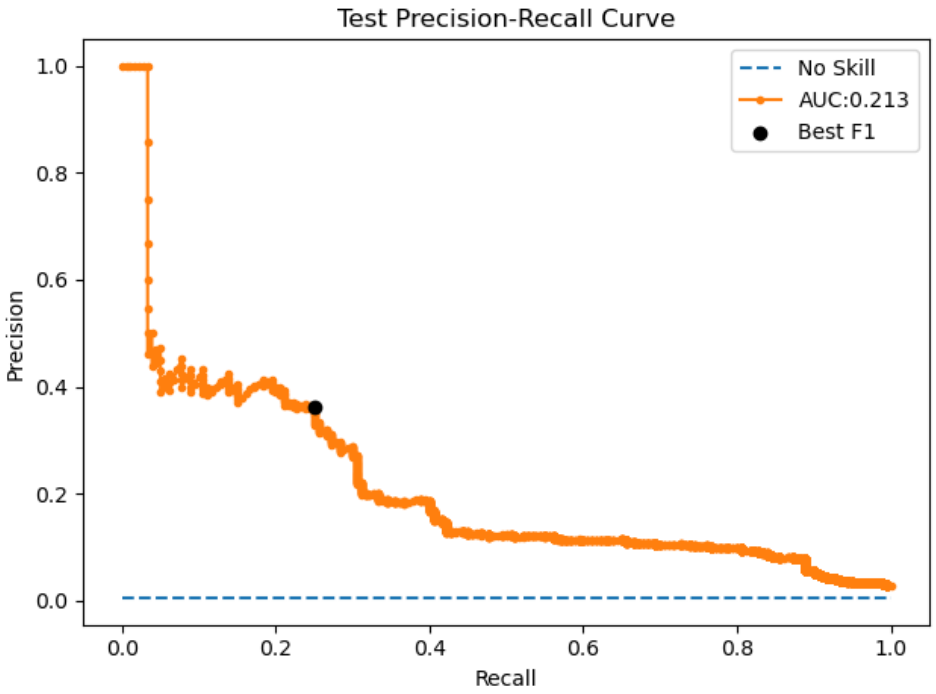


Figure A.14: Precision-Recall curve of MLP (1_100). On test set. Best F1-score location marked (black).

A.4.5 Training curves

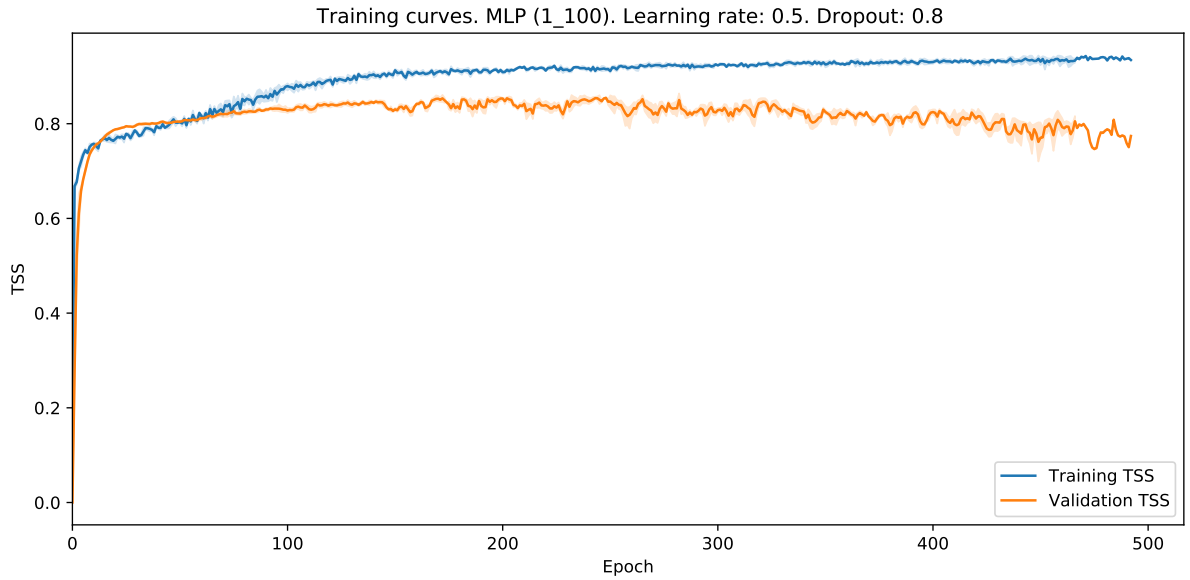


Figure A.15: Training curves of 1_100 best MLP with static LR. Average over three seeds with standard error shadow. Best model at epoch 203 with validation TSS of 0.8640 for seed 49.

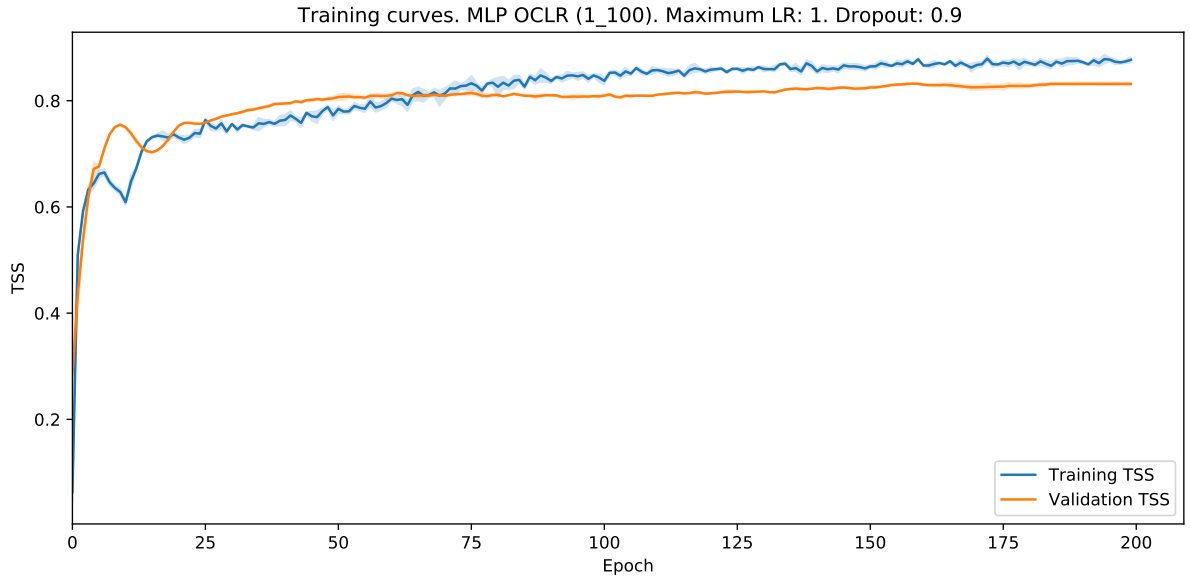


Figure A.16: Training curves of 1_100 best MLP with OCLR. Average over three seeds with standard error shadow. Best model at epoch 200 with validation TSS of 0.8375 for seed 124.

A.5 Appendix: Chapter 6

A.5.1 Learning rate range test results

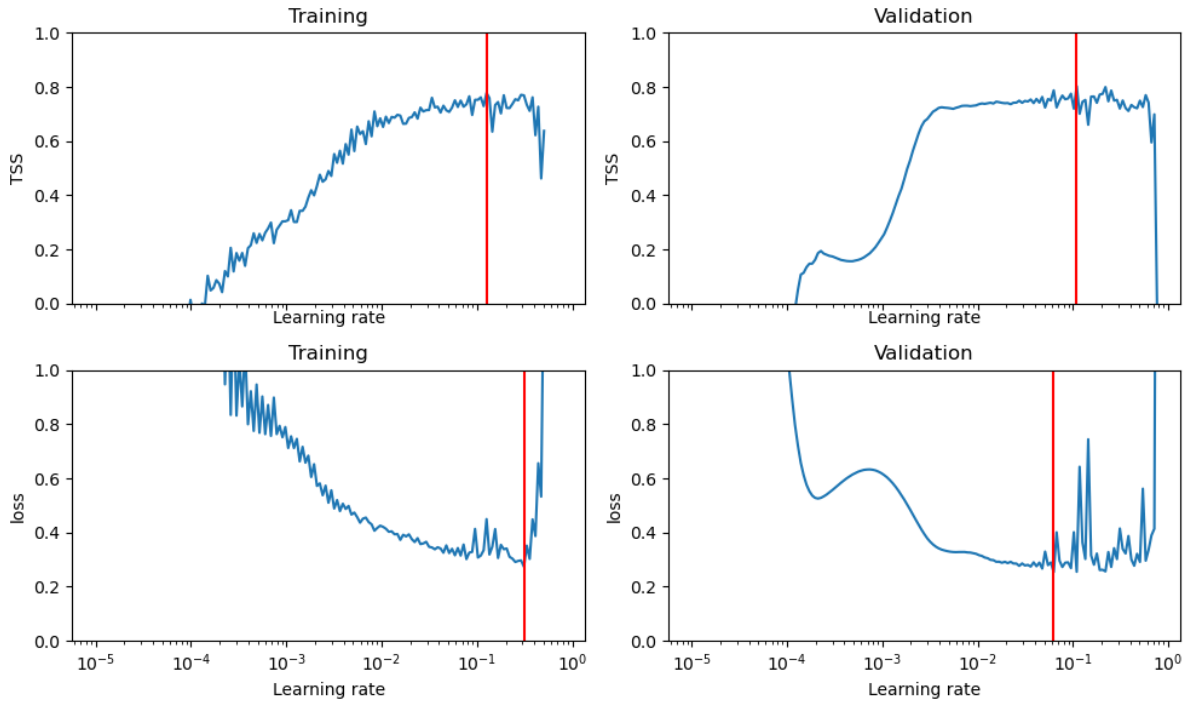


Figure A.17: LR range test for 1D-CNN. Dropout: 0.8. Kernel size: 7. Number of filters: 40

A.5.2 Training curves

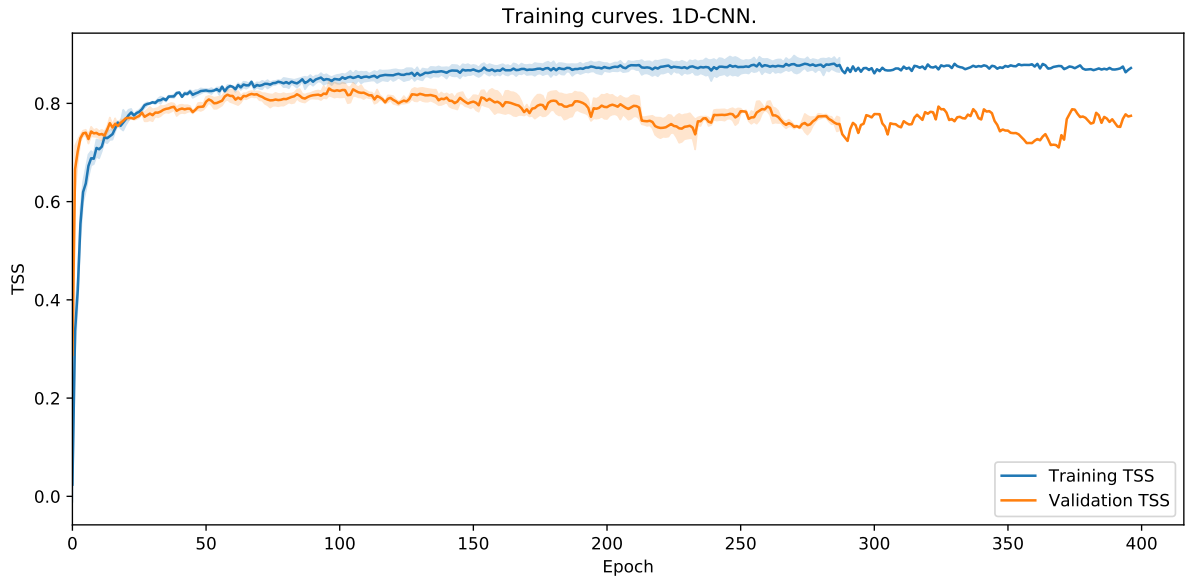


Figure A.18: Training curve of best 1D-CNN model. Dropout: 0.8. Kernel size: 7. Number of filters: 40. LR: 0.066. Average over three seeds with standard error shadow.

A.5.3 Evaluation Plots

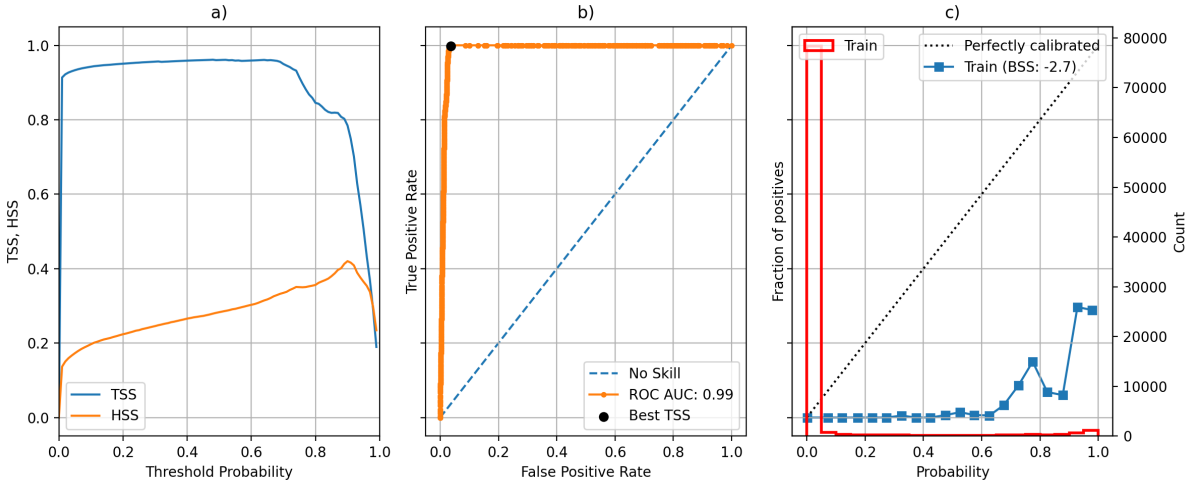


Figure A.19: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. 1D-CNN. Training set

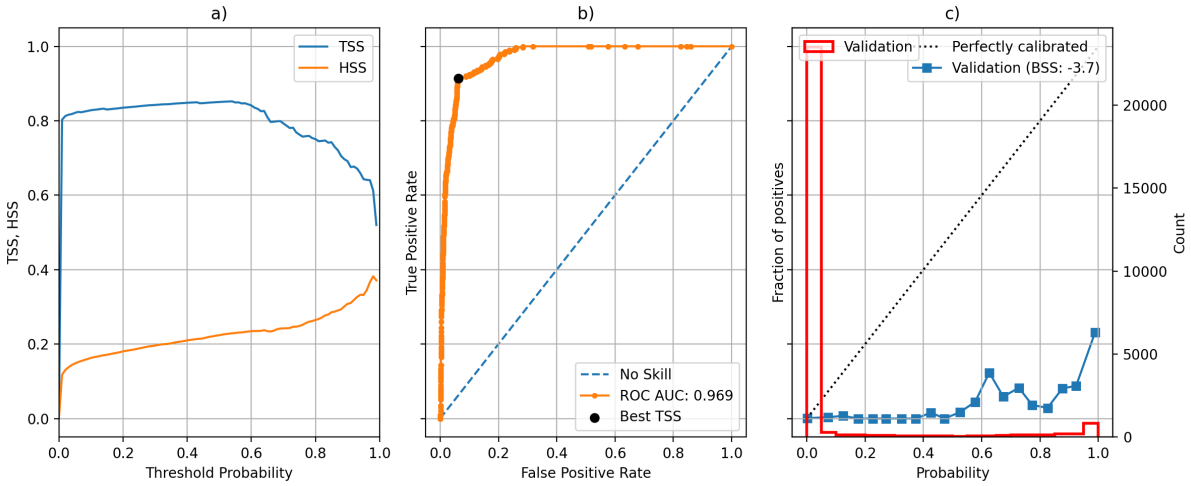


Figure A.20: (a) SSP of TSS and HSS, (b) ROC curve, (c) Reliability diagram. 1D-CNN. Validation set

A.6 Appendix: Chapter 7

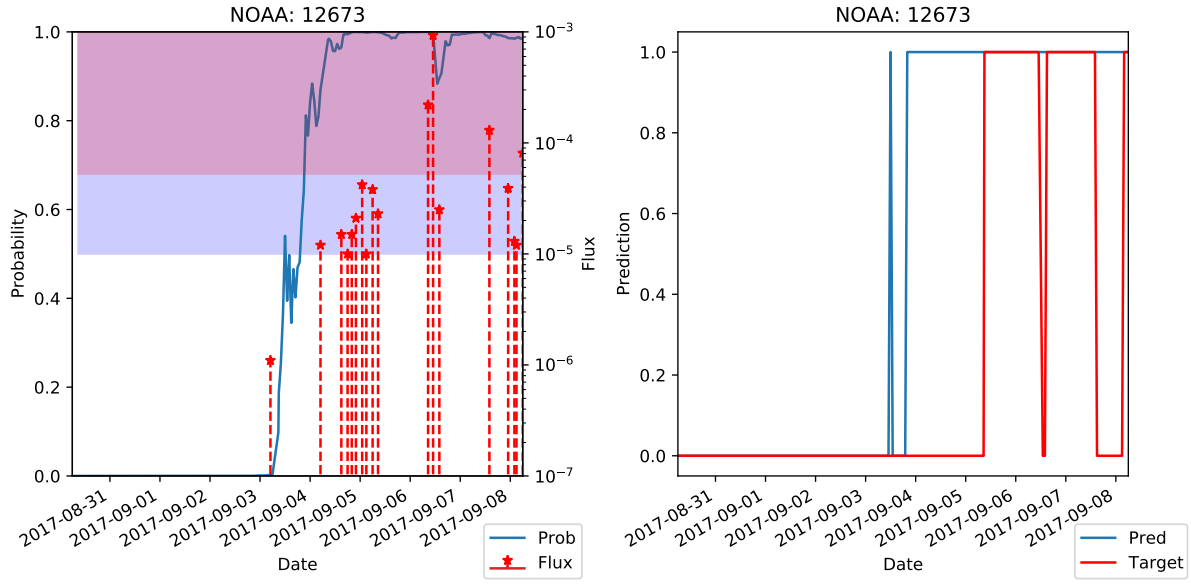


Figure A.21: Prediction plot. Same format as 5.15. 1_100 MLP. Test set. (NOAA:12 673)

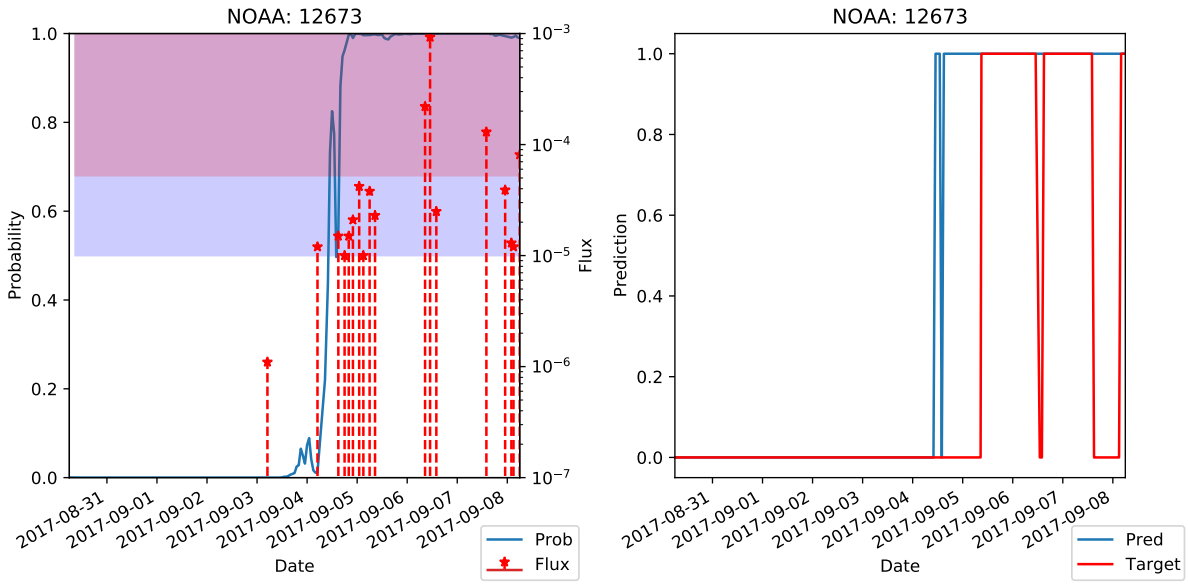


Figure A.22: Prediction plot. Same format as 5.15. 1D-CNN. Test set. (NOAA:12 673)