POLITECNICO DI TORINO

Corso di Laurea Magistrale
in ICT For Smart Societies

Tesi di Laurea Magistrale

Deep learning for the prediction of geomagnetic events

Relatore
Prof. Enrico Magli

Candidato
Maurizio Lo Schiavo
Matr. s281250

Anno Accademico 2021-2022

Acknowledgements

Al termine di questo percorso accademico, desidero ringraziare la mia famiglia, che mi è sempre stata accanto, i miei amici di Angri, i colleghi conosciuti qui in università e tutti quelli che mi sono stati accanto.
Un grazie va anche al politecnico, ai professori che mi hanno insegnato tanto, al mio relatore per l'opportunità datami. Ad Maiora!

Abstract

In the last decades, the rising Big Data revolution together with the consolidation of High Computing capacity led to the evolution of computer systems, able now to perceive their environment thanks to the extraction of meaningful information, developing learning capacities to be self-adaptable.
The depicted context has laid the perfect foundations of "Machine Learning", a subfield of Artificial Intelligence, considered at the forefront for the construction of a smart society. Its spreading is being affected all domains, finding application in medicine, transport, environment, and all industrial sectors, with the objective not only of automatizing static activities performed by human beings but also supporting them in taking decisions for more complex and dynamic challenges, capturing valuable knowledge in some cases not directly human-accessible.
In such digital enhancement, neural networks are becoming more and more popular as an ML technique since, in case of a sufficient amount of information and computation capacities, they result effective whichever the field. Characterized by a simple implementation phase, they ensure flexibility in structure and configuration, adopting a suited version according to the peculiarities of the problem. Furthermore, their learning process does not require physical knowledge about the phenomenon of interest.
A challenging task for these algorithms is represented by the Forecasting of specific events: the ML tool may leverage the previous history characterizing a phenomenon with the purpose of predicting its future behaviors. Mostly used in the weather and market domain, this implementation may result very useful also in the space weather field, as the prediction of specific geomagnetic events affecting the Earth may help in preventing harmful consequences.
The goal of this thesis work is that of implementing neural networks for a specific class of geomagnetic events, called Coronal Mass Ejection, trying to predict future trends and classify their severity based on a time series dataset containing measurements taken in situ L1 [1]. The project has been carried out in collaboration with the Osservatorio Astronomico di Torino (OATo), here interested in obtaining in advance future information about the strength of the ring current around Earth caused by solar protons and electrons (DST).

---

[1] "The L1 point is perhaps the most immediately significant of the Lagrangian points, which were discovered by mathematician Joseph Louis Lagrange. It lies 1.5 million kilometres inside the Earth's orbit, partway between the Sun and the Earth. Lagrangian points are where all the gravitational forces acting between two objects cancel each other out and therefore can be used by spacecraft to 'hover'." [1]

In the following work two main phases may be distinguished: at the beginning, analysis and manipulation of the dataset have been performed, trying to catch its characteristics and adopting augmentation techniques to transform it to be ready for the prediction step. Then the implementation of ML techniques has been deployed, where multiple NN architectures and settings were tested for different prediction tasks, so to prove the effectiveness of deep learning algorithms in reaching the desired goal.

.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
BS	Batch Size
LR	Learning Rate
CM	Confusion Matrix
CNN	Convolutional Neural Network
SVM	Support Vector Machine
LSTM	Long-Short Term Model
MLP	Multi-Layer Perceptron
RESNET	RESidual NETwork
BLSTM	Bi-directional Long-Short Term Model
LSTMFCN	Long-Short Term Model Fully Convolutional Network
ESA	European Space Agency
NOAA	National Oceanic and Atmospheric Administration
CME	Coronal Mass Ejection
LASCO	Large Angle and Spectrometric COronagraph
HPC	High Performance Computing
DST	Disturbance Storm Time
AE	Auroral Electrojet
SYM-H	SYMmetric for H component
MHD	MagnetoHydroDynamic

SSM	Single Step Model
MSM	Multi Step Model
DTW	Dynamic Time Warping
SMOTE	Synthetic Minority Oversampling TEchnique
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Chapter 1

Introduction

Modern society is being invested in a digital revolution that is reshaping the daily routine. Such transformation mostly relies on the abundance of data, massively generated by sensors and electronic devices and whose manipulation allows the creation of tailored applications, according to the customer's needs. However, dealing with a huge amount of data requires computing systems to be high performing, so that during their processing they can extract relevant information about the phenomenon of interest in a reasonable amount of time.
On these premises, machines are acquiring a new type of "artificial intelligence", nowadays known as Machine Learning" or "Deep Learning", allowing them to execute more challenging tasks and solve dynamic problems, offering valid support to humans for whichever domain of application.
The achieved smartness recently caught the attention of the space weather community, interested in studying phenomena related to the sun and their interaction with the Earth. In particular, scientists are concerned about geomagnetic disturbances -like solar flares and Coronal Mass Ejection- and their possibility of impacting the terrestrial surface, causing potentially severe consequences. Here Machine Learning enters the picture, allowing the forecast of future geomagnetic disturbances by analyzing its previous trends, basing the learning process upon past experiences.
A crucial advantage given by some ML tools is the unnecessity of physically describing the phenomena, an aspect considered of paramount importance in the space weather due to the intrinsic difficulty in explaining behaviors and interactions through physics-based models.
Given those circumstances, neural networks appear particularly suited for the prediction task, putting in place their ability to recognize correlations in raw data, and classify it, constructing mathematical formulas not necessarily meaningful from the physical point of view.
The following sections illustrate a general overview of the context, acknowledging

the statement of the problem, together with the consequences and risks derived from the solar activities of interest.
The most popular forecasting methods are presented, analyzing differences and limitations. Finally, the contribution offered by Deep learning is illustrated, showing how recent studies implemented those tools, many of which with a similar intention to the one set for this thesis work.

1.1 Space weather

Earth's atmosphere is constantly interacting with factors of various natures, many of which may alter the equilibrium of the terrestrial surface under different aspects. Among all possible perturbations, the ones derived from solar activities have been so relevant to come to the attention of the scientific community, interested in understanding and analyzing their possible effects.
Space weather field includes all these complex Sun-Earth interactions, whose phenomena are studied by scientists [10]" aiming to forecast potential geo-effective events occurring in the geo-space and caused by the release of solar energy into the Earth's magnetosphere during geomagnetic storms, and all related phenomena". This means that the Sun influences conditions in the near-Earth environment, including the magnetosphere, ionosphere, and thermosphere, and its radiation can pose a persistent hazard to space or ground-based stations and human health.
More deeply, D.Telloni [10] exposes the hypothetical harms caused by solar activities, mentioning "potential slowdown and orbital decay of the low-Earth-orbiting satellites (due to an additional aerodynamic drag force induced by solar activity), induction of very harmful electric currents in power transmission grids and pipelines, disruption of satellite signal propagation with severe implications for positioning systems, and unrecoverable failures of electronics onboard spacecraft". The ionosphere reflectivity can also be altered by the arrival of solar energetic particles, impairing radio communication systems. Finally, Space Weather deals with radiation produced by solar storms that can endanger the astronauts' health.

1.1.1 Coronal Mass Ejection

Coronal Mass Ejections are the largest expressions of solar activity, defined as large eruptions of magnetized plasma from the sun's atmosphere -the corona- into interplanetary space [11], which occur much more frequently at solar maximum than at minimum.
Their origin is similar to solar flares -bursts of electromagnetic radiation- deriving from the twisting and realignment of the sun's magnetic field.
In the case of tangling, they produce strong localized magnetic fields which can break through the surface of the sun at active regions, generating CMEs.

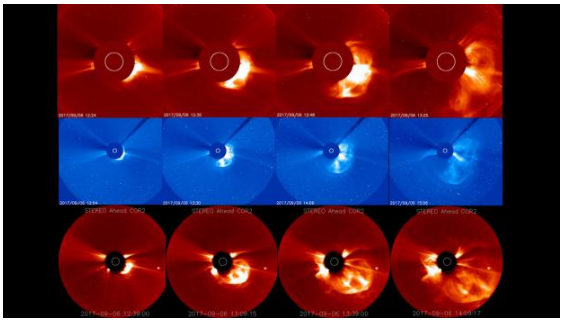

Figure 1.1: CME phenomena illustration [2]

Mentioning [12], " these interplanetary structures can be seen as propagating regions of space of enhanced density and magnetic field strength. They are characterized by an intense and long-lasting South-directed magnetic field, which magnetically reconnects with the oppositely oriented Earth's magnetic field. This process allows a net transfer of energy from the solar wind to Earth, triggering the most severe geomagnetic disturbances."
The magnetic field is not the only crucial parameter for these phenomena characterization, since also the transported kinetic/magnetic energy [13], the dynamic pressure [14], and turbulence [15] give a relevant contribution in driving the geomagnetic activity.
Compared to solar flares, whose travel occurs at the speed of light and reaches Earth in just over 8 minutes, CMEs travel at a more leisurely gait. Mentioning [2], "at their highest speeds of almost 1,900 miles per second (3,000 kilometers per second), CMEs can reach Earth in about 15 to 18 hours whilst slower CMEs traveling around 155 mi/s (250 km/s) can take several days to arrive". Continuing [2], "larger CMEs can reach a size comprising nearly a quarter of the space between Earth and the sun by the time it reaches our planet. If a CME is large enough and travels faster than the solar wind it generates a shock wave whereby accelerated charged particles travel ahead of the CME further disturbing space weather conditions and intensifying geomagnetic storms."
CMEs usually take place around sunspot agglomeration and are often accompanied by a solar flare, though the two do not always occur in tandem.
In this regard, scientists are still not entirely sure how the two events are related, with the University Corporation for Atmospheric Research's Center for Science Education (UCAR) [16] stating that "CMEs -like solar flares- are more common during solar maximum, a period in the sun's 11-year cycle of activity when the star is at its most active". After CMEs are released, they swell in dimensions as they travel away from the sun.

Although CMEs are positively welcomed by skywatchers worldwide, given the exceptional aurorae visible at latitudes beyond their normal polar range, in some cases they lead to significant damages, wreaking havoc with power grids, telecommunication networks, orbiting satellites, and exposing astronauts to harmful doses of radiation.

1.1.2 Adversarial effects

In recent centuries, solar storms caused several malfunctions, affecting electronic devices or structures.
The Carrington Event in 1859 provoked worldwide telegraph system failures. According to some historical reports, operators were receiving electric shocks and sparks showering from telegraph machines, setting papers ablaze.
"In 1989, a CME accompanied a solar flare that hit Earth, plunging the entire province of Quebec, Canada, into an electrical blackout that lasted 12 hours" state NASA [17]. The event cost Quebec's utility company Hydro-Quebec at least $ 10 million in damages.
CMEs can cause swells in electrical currents, overloading power grids and causing widespread blackouts. Also, CMEs can jostle Earth's magnetic field and impair radio transmissions, increasing radio static in Earth's ionosphere.
GPS systems are particularly vulnerable to disturbances in the ionosphere, with coordinates strayed by tens of feet during a CME event. The disruption occurs because GPS uses radio signals to relay information between a satellite and a ground receiver. The radio signal passes through the ionosphere layer containing charged plasma that bends the path of the GPS signal similarly to lens-bending light.
Normally, GPS systems can compensate for this bending of the radio signal, leaving the accuracy of GPS unaffected. However, during a CME event, the ionosphere can be so severely disturbed that the GPS models cannot keep track of such changes, disenabling receivers to evaluate an accurate position.
Earth-orbiting satellites are vulnerable to CMEs, particularly those in high geosynchronous orbits, where most communications satellites are found. When a CME triggers a geomagnetic storm, satellites can be struck by a high current discharged or damaged when high-energy particles penetrate them. As such, vulnerable satellites can be placed in "safe mode" to prevent damage to electronics.
A direct hit of a colossal geomagnetic storm like the one observed in 1859 -the Carrington Event- could take a heavy toll on our satellite fleet according to the research described in [18].
"There are more than 900 working satellites with an estimated replacement value of $170 billion to $230 billion, supporting a $90 billion-per-year industry. One scenario showed a 'superstorm' costing as much as $70 billion due to a combination of lost satellites, service loss, and profit loss" [18]. "A worst-case solar storm could

have an economic impact similar to a category 5 hurricane or a tsunami," said Steigerwald et al. [19].
SpaceX has already witnessed firsthand the damage space weather can create when a geomagnetic storm destroyed up to 40 Starlink satellites worth over $50 million, in Feb. 2022.
"In low-Earth orbit, astronauts receive higher doses of radiation than in case of Earth surface but they are still mostly protected by the magnetosphere" according to S. Frazier [20]. Furthermore [20], "the real danger to astronauts comes if they roam from the safety of the magnetosphere, for example, to explore the surface of the moon or Mars. Upon such an expedition -outside Earth's "protective shields"- they are vulnerable to dangerous space weather events such as CMEs." According to B. Mendez [21], if a CME-driven shock wave were to hit an unprepared astronaut exploring the lunar or martian surface, they would be hit with as much radiation as 300,000 simultaneous chest X-rays. This would have lethal consequences as it would only take 45,000 simultaneous chest X-rays to be considered lethal.
Luckily, CMEs take several hours or even days to reach Earth, thus giving the community some time to prepare for their arrival. Various organizations keep relevant attention to the sun and report any changes in surface characteristics that could imply a CME ejection such as an increase in solar activity and solar flare ejections. If a strong M or X-class solar flare is detected it will likely be accompanied by a CME, but not always as already outlined.
Forecasters use various parameters -size, speed, and direction- inferred by orbital satellites' measurements or coronograph images to determine the likelihood of a CME hitting Earth.
For the latter, scientists are equipped with a specific instrument -the coronagraph- which can block out the light of the sun, allowing the display of the outermost layer (the corona). It mimics the natural phenomenon of a solar eclipse when the moon's shadow covers the bright center allowing the corona to be observed.
On the CME detection frontline is the Deep Space Climate Observatory (DSCOVR) satellite that is stationed at the first Lagrange point L1 between Earth and the sun at about 1 million miles (1.6 million km) from the terrestrial surface. DSCOVR [22] monitors any changes in the interplanetary magnetic field (IMF) strength and solar wind speed which are vital to the accuracy and responsiveness of NOAA's space weather alerts and forecasts.
From its parking spot at L1, the DSCOVR [22] satellite can provide between 15 to 60 minutes of advanced warning before a CME reaches Earth. When an Earth-bound CME is detected, the SWPC alerts vulnerable groups such as power companies, satellite companies, and airlines to take appropriate measures. With advanced warning, utility companies can redirect power loads to protect the grids from being overloaded when the CME hits, satellites can be placed into "safe" mode and planes can be redirected.

Yet, "Vigil" ESA mission [23] is going to monitor the sun from Lagrange 5, approximately 93 million miles (150 million kilometers) from Earth. The spacecraft will be positioned in a way to keep an eye on the "side" of the sun. It will observe solar conditions before they rotate around to face Earth in a bid to give us advanced warning of possibly hazardous solar activity.

1.1.3 Forecasting methods

As previously highlighted 1.1.2, Solar weather can have drastic consequences; thus, monitoring, understanding, and forecasting such events become crucial. A beneficial factor is given by the relatively slower travel time of such ejections, useful to give those concerned more time to prepare for such an arrival.
In this sense, adopting a statistical approach for forecasting Space Weather phenomena could result in effective, giving particular consideration to predicting the geomagnetic response to the impact of geo-effective solar structures, the relativistic electron flux, the occurrence of solar flares, the propagation time of CMEs, the transit of high-speed streams to Earth, and the crossing of the heliospheric current sheet.
Forecasting methods used so far are based on remote-sensing observations of solar phenomena, like CMEs, causing geomagnetic storms, providing expectations of CME arrival times. Those models are generally categorized into three classes: physics-based, event-based, and drag-based.
Physics-based models rely on photospheric magnetic field observations to initiate numerical MagnetoHydroDynamic (MHD) simulations of the eruption of the CME and its propagation from the Sun to Earth. These numerical codes require the use of supercomputers to run efficiently. In addition, their reliability depends on a correct representation of the physical processes within the models, i.e., the understanding of the physics of the corona and the solar wind, which is unfortunately not yet total.
The most used MHD models for such predictions are the WSA-Enlil [24], the prediction model of the heliosphere, able to provide warnings of solar wind structures and Earth-directed CMEs 1-4 days in advance, and the EUropean Heliospheric FORecasting Information Asset -EUHFORIA- [25], consisting of two major components like a coronal model and a heliosphere model including coronal mass ejections.
in the last 30 years, different studies have been carried out [26], [27], [28],[29], [30], [31], [32], [33], [34], [35].
Simpler and less computationally expensive, yet equally reliable, event-based models count on statistical studies of past CMEs, so resulting in empirical nature, and essentially relate the CME Sun-Earth transit times to their propagation speeds, as inferred from coronagraphic images. Such models allow the establishment of

empirical laws, say analytical functions, which, assuming that past observations are analogous to future ones, allow prediction of the impact time on Earth of a new CME.
This category's most popular works are related to the analysis of coronagraphic speed measurements, like in [36], [37], and [38].
Another example is highlighted in [39], where a probability distribution of the geomagnetic Dst index as a function of the CME and solar flare parameters is determined. Other empirical works are given by [40], [41], [42], [43], [44], [45].
Observational evidence for an adjustment of the CME propagation speed to the background solar wind and its interpretation in terms of aerodynamic drag encouraged the development of the so-called drag-based models, which assume that the CME propagation in the heliosphere is governed by aerodynamic drag, as presented in Gopalswamy et al.[46] and Vršnak et al. [47].
Among the others, the 3D COronal Rope Ejection (3DCORE) model introduced by Möstl et al. [48] is one of the most accurate: it analytically describes the dynamics of the CME through an easy equation of motion, thus providing real-time prediction of the CME arrival time and impact speed with the terrestrial surface, at the cost of considering intrinsic approximations.
The three-class methods can forecast geomagnetic disorders 1–4 days in advance; however, the predictions are significantly model-dependent and affected by large uncertainties.
One of the biggest deficiencies characterizing these approaches is the negligible usage of in-situ solar wind data measured at the Lagrangian point L1, whose deployment could potentially provide more accurate warnings concerning CME arrival and storminess degree. This deficit is essentially due to the challenging work of locally identifying CMEs within in-situ observations.
In this regard, in [10], where Telloni summarizes results of previous works, statistically based, a technological readiness in implementing Machine Learning tools for real-time prediction of geomagnetic events is highlighted. Even E. Camporeale [49] gives a significant boost, remarking how "the numerous recent breakthroughs in Machine Learning make imperative to carefully ponder how the scientific community can benefit from a technology that, although not necessarily new, is today living its golden age".

1.2 The role of Deep learning in space weather forecasting

As previously outlined 1.1.3, Deep Learning tools can be deployed for different use cases, with an increase in the usage of these techniques for forecasting purposes mainly based on the massive amount of observations collected at an impressive

frequency, helping forecasters to predict more accurately.
Citing again [49], "Today, machine learning poses both a challenge and an opportunity for the space weather community. The challenge is that the current data science revolution has not been fully embraced, possibly because space physicists remain skeptical of the gains achievable with Machine Learning.
The clearest opportunity lies in creating space weather forecasting models that can respond in real-time, built both on physics predictions and observed data".
In the last years, multiple works have been presented, applying Deep Learning tools on various nature datasets to predict specific geomagnetic disturbances, mainly solar flares, and CMEs.
In their work, Dhuri et al. [50] use ML to understand the underlying mechanisms governing flares.
Li Rong and Zhu [51] use sequential sunspot data to implement solar active regions information and give them as input to a Multi-Layer Perceptron and learning vector quantization to predict the flare level within 48 hours. Pandey et al. [52] present a solution to full-disk flare prediction using compressed magnetogram images, performed by training a set of Convolutional Neural Networks to perform operations-ready flare forecast.
Again, starting from a solar flare magnetic map data, Jun Chen et al. [53] define a two-stage solar flare early warning system: in the first phase, unsupervised clustering algorithm implementations like k-means detect sunspot group in which positive sample account for the majority, furthermore for these groups, an ensemble model integrating boosting and CNN predict whether solar flares will occur in the next 48 hours.
D. Sudar et al. [54] analyze transit times of coronal mass ejection using a simple feed-forward NN, providing as input only the CME velocity V and the CMD of its associated flare, and observing that transit time dependence on V is showing a typical drag-like pattern in the solar wind. Delouille et al. [55] demonstrate that coronal holes and filaments could be distinguished in solar EUV images through the usage of an ML algorithm combined with segmentation techniques.
A new approach is proposed in [56], with B. Dhuri et al. building a prediction engine for CME arrival time forecasting, equipped with an SVM algorithm and based on partial-/full halo CME.
Yimin Wang et al. [57] make use of a CNN regression model, giving as input only the instances of the white-light observation (images) of CMEs to predict their arrival time, while Yurong Shi et al. [58] utilize logistic regression on a set of CME parameters like central position angle, angular width and linear velocity derived from LASCO coronagraph images, to predict whether a CME will hit or miss the Earth surface, and in case of a hit, their expected arrival time.
A new tool for CME detection and tracking is presented in [59], composed of three

modules. Firstly, a LeNEt solves a supervised image classification problem, considering images illustrating CME structures. CME regions are then spotted using a deep descriptor transforming. In the end, a graph-cut technique is applied to finely tune the detected CME region.
In [60], Yanru Sun et al. propose a multimodality solar wind prediction method that jointly learns vision and sequence information in a unified end-to-end framework. Specifically, their prediction tool consists of three modules: Vmodule, Tmodule, and Fusion module. Vmodule uses pre-trained GoogLeNet to learn visual representation from the extreme ultraviolet (EUV) images, Tmodule applies a combination of a one-dimensional CNN and a BLSTM for learning sequence representation on a multivariate time series, and a Fusion module to improve the overall performance.

1.3 Purposes of Thesis

The goal of this thesis work is that of using Deep Learning tools like Neural Networks for the prediction of geomagnetic events. The solar phenomena are described by a dataset composed of solar wind and geomagnetic index observations, collected in situ L1 over multiple years. By leveraging this data, the networks should be able to forecast future evolution related to solar ejections, to alert in advance in case critical events might result harmful to the Earth.
Different prediction tasks have been proposed, from a Machine Learning point of view treated as classification problems, and disclosing information concerning the criticality degree of such CME and their temporal extension.
Such thesis work has been deployed in collaboration with the Osservatorio Astoronico di Torino -interested in developing Deep Learning techniques able to predict geomagnetic events- that provided the dataset for the study.
Moreover, the High computation capacities have been provided by the hpc@polito, Academic Computing project of Dipartimento di Automatica e Informatica at Politecnico di Torino [61].
The thesis structure is the following:

- In chapter 2, the Deep Learning world has been briefly explained, here intended as a milestone in the evolution of artificial intelligence. Then, the characteristics of modern learning systems have been highlighted, followed by their general use cases. An introduction of the most popular DL tools -the Neural Network- took place, providing basic concepts and characteristics behind their working principle.
  All the steps aiming to correctly implement a Neural Network algorithm are exposed, offering a direct linkage with the ongoing study. The background

behind each architecture category has been briefly explained, examining the pros and cons, and their hypothetic suitability for the desired goals.

- In chapter 3, the offered dataset is explored, understanding from different perspectives its composition and the feature contribution in geomagnetic disturbances.
  A readiness examination of the dataset has been carried out, explaining the reason for its direct unusability, then introducing the sequential data preprocessing steps aiming to transform it and achieve readiness.

- Chapter 4 presented the neural network architectures proposed for the study, examining their strengths and limitations. The original intention of adopting a heterogeneous set of networks led to the construction of structures with different complexity levels, multiple subblock configurations, and the implementation of extra solutions.

- In chapter 5, an analysis of the conducted experiments is performed, outlining network performances concerning different challenges. Multiple scenarios have been defined, each consisting of a classification type.
  The multiple assignments deliver the possibility to obtain specific CME information concerning the desired degree of severity and the (future) temporal extension to investigate.

- Finally, conclusions are given in chapter 6, remarking on the great support Deep Learning can offer for such complex problems the space weather field is constantly dealing with. Suggestions for future enhancements of the work are offered, making up for the main limitations the project faced.

Chapter 2

Deep Learning

As introduced in 1, the ongoing digital transformation is enhancing the technologies helpful in guaranteeing modern society better life conditions.
Multiple solutions are reshaping the job environment and daily routine, with a relevant breakthrough given by the capabilities of computing systems, nowadays more capable of performing intellectual tasks as human beings. As David Petersson [62] claims, their developed smartness -commonly known as Artificial Intelligence- started becoming more fascinating when they stopped taking rule-based decisions, moving to a different approach, oriented to the examination of data and surroundings, to solve or foresee problems without any explicit programming, but only relying on a learning process being self-teaching. In this way, AI can not only relieve humans of various repetitive tasks but also support them when it comes to facing more challenging and dynamic tasks.
Over the years, different milestones have been achieved in AI's evolution, principally due to the increased computing capabilities and the growth of data availability.
The following chapter intends to go through such evolution, analyzing the peculiarities of each chequered flag. The developed techniques are then examined, defining their suitability concerning future challenges.

2.1 From Artificial Intelligence to Deep Learning

As previously claimed, Machine and Deep Learning refers to the ability of a computer system to learn without being explicitly programmed but completely relying on its computing capabilities to infer information from data.
Over the years, multiple learning algorithms have been developed, each with its mathematical expression; yet the high-level process through which the model undergoes is the same, conceivable as composed of the following three main elements:

1. The decision process, where the system receives input variables and try to estimate the output. A decision rule consists of an analytical function, completely different among algorithms and based on the nature of data and their expected outcome.

2. The error function, so an analytical function used to evaluate the goodness of a machine in estimating outcomes. Again, the implementation of the function may vary according to the problem and data nature. On its base, a re-adjustment of the model is executed.

3. A model optimization process, concerning the learning model re-assessment, to understand if data are better fitted. Considering the model as described by parameters, the objective is to update them to decrease inconsistencies between the known instances and the model's evaluation. This optimization procedure has an iterative nature, meaning that as more data incomes, it autonomously repeats so to improve the overall performances.

Algorithms differently implement these sequential steps, according to the given task and data available; however, the data nature and expected estimations lead to the definition of three main learning categories: supervised, unsupervised, and reinforcement learning. There are also other techniques in literature, here out of the discussion due to their irrelevance for the work.
Supervised algorithms consist in using learning techniques supervision-oriented, implying in the training process the usage of labeled data so to estimate the output. The term "labeled" means that, during the training, the input given to the algorithm is provided together with its expected output. In this way, the estimation generated by the model can be straight compared with the true one, so to acknowledge the degree of correctness the machine is operating, and as a consequence the direction in which estimation could be further improved.
An example of this type of problem is the recognition of an object in a set of photos, here forming the dataset. Given an image representing just one object for sake of simplicity, the corresponding label will be the description of the object contained. The role of the algorithm is that of coming up with features illustrated in the images, which could be simply graphical ones like colors, shapes, etc to infer the represented object. The level of success of the training process will be then evaluated by the algorithm's correctness in categorizing images according to their objects.
Supervised learning algorithms may be further divided, distinguishing classification and regression tools. The difference is given by the output information provided with the data, which is a class label in the first case, or a continuous value in the other. Such characterization has consequences on the working principle of the

algorithm for all the three steps beforementioned. In particular, the decision of the error function is strictly related to this problem property: while the continuousness nature of the output implies different mistake severity that may be committed by the algorithm (in forecasting a temperature value equal to 15 $°C$, estimating 30 $°C$ is much worse than 16 $°C$), a wrong label classification has generally the same incorrectness level, whichever has been the class provided by the model as the outcome.
Even if out of this work's scope to describe and analyze the different model structures and characteristics, it is worth mentioning at least the famous ones and their classical use cases. Most typical classification problems refer to the prediction of data sample categorization and for them, algorithms like the Decision Tree, Logistic regression, random forest, and Support vector machine are exploited.
For regression, classical use cases are the prediction of values like in the weather or market field, generally, via techniques like linear or simple Regression, o again decision tree. In the end, applications carried out thanks to supervised Learning are Fraud Detection, Image segmentation, medical diagnosis, and Spam detection.
Unlike supervised, unsupervised learning algorithms lack supervision while training, meaning that data here are unlabeled, in most cases for the absence of such prior knowledge. Therefore, the machine is exposed to a kind of input that needs to be recognized to be then associated with a specific cluster of estimations. Association and clustering are the two types of unsupervised ML existing. As regards the first, the goal is to find relations between variables in a large dataset, discovering data dependent on the other, to maximize returns. Here, the most notable algorithms are the Apriori, Eclat, and FP-growth algorithms.
Clustering is used instead as a method for grouping similar objects into a cluster, so to derive groups from the original dataset. The most common algorithms are the K-means, BSCAN, ICA, FCA, and Mean-shift.
Typical unsupervised applications are Anomaly detection, Network analysis, Singular-Value decomposition, and Recommendation Systems. Those kinds of learning techniques find applications in robotics, video games, and self-driving cars.
The last learning category is represented by reinforcement learning, most similar in terms of the learning process if compared to humans; this is based on a trial and error approach. Reinforcement learning is played by three factors: the agent, the environment, and the actions. The agent is the one learning by communicating with its environment through actions, getting a reward in return, either positive or negative.
In such a scenario, no data labeling is present, and the agent only learns from interactions with the environment. Reinforcement learning is based on feedback, with the agent performing multiple actions to maximize the positive rewards. In

this case, positive and negative reinforcement learning represent the classical approaches. The first refers to an event that occurs as a result of a particular behavior. This type of reinforcement strengthens the behavior and increases its frequency, positively affecting the actions taken by the agent. Positive reinforcement maximizes the performance and sustainability of change over an extended period.
In the opposite direction, the negative reinforcement decreases the frequency of the occurrence of a behavior.

2.2 The complete Machine Learning procedure

Previous sections provided a high-level description of the learning algorithm procedure, bringing out similarities with the same human behavior. However, the given explanation refers only to a phase of a wider scheme, where data input has been already transformed to become usable by the machine, and where the respective output will be analyzed for evaluating the algorithm's performance. Inspired by the work carried out by Iniesta et al [63] and Haque et al. [64], the entire process can be displayed through a scheme -conceived as a sequential set of steps- visible in 2.1:

Problem definition refers to the formulation of the study to perform, defining the set of tools to be deployed for achieving the designed goals; **Data collection** regards the act of collecting data information for composing the dataset; **Data preprocessing** represents the set of techniques implemented to make the dataset ready for the algorithm; **Model training and validation** consists in the training procedure previously explained; **Model testing** test the created model onto a new set of data, unknown by the machine; finally, **Performance evaluation** evaluates the performances of the algorithm on top of the outcomes returned by the previous block, implementing evaluation metric.
For the project under study, the next section will define different aspects related to scheme steps, highlighting for the correct work deployment the negligibility of some and the rigid necessity of others.

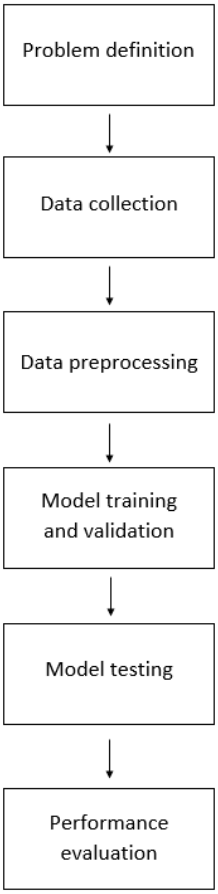

Figure 2.1: Machine Learning schematic process

2.3 Neural Network

The achieved technological maturity in the artificial intelligence field has allowed the deployment of several ML techniques, each more suitable for a specific application domain, dataset nature, and prediction outcomes.
Recently, neural networks showed the most incredible outbreak, justifying their spreading as a direct consequence of the Deep Learning boom. Working with complex algorithms like NNs requires necessarily a huge amount of data and computing capacity, aspects equally crucial in the deep learning paradigm.
Their structure appears intrinsically inclined for a deep learning approach, thus giving the possibility to learn complex patterns that in comparison other tools would struggle with.
Neural Networks are very versatile, adopting different architectures and configurations according to the peculiarities of the problem. Moreover, neural networks are famous for their not-necessity to dispose of knowledge about the phenomena of interest, as their analytical implementation consists in constructing mathematical relationships not necessarily meaningful from the physical perspective. Such aspects may result beneficial for the proposed study, since establishing physical relationships in space weather fields may result challenging. Such premises motivated the Neural Network choice for the task proposed in the following work.
The next section will investigate the main aspects of such deep learning tool architecture, explaining the overall functioning and the appropriate deployment, highlighting the analytical procedure behind the learning process, and the proper way to construct and set all its parameters. Afterward, a description will explain the general performance evaluation step, highlighting which metrics should be more suited for such work.
Analyzing exhaustively the network architecture and behavior is out of this study's scope; however, recalling the main elements is beneficial for expressing a critical judgment and extracting meaningful considerations from the results.

2.3.1 Definition

As the name suggests, Neural networks were born attempting to imitate the human being's brain network, where biological neurons are connected through links, establishing a distinctive architecture. In the Artificial domain, these neurons, also called perceptrons, create "weighted" connections, giving each of them a different relevance.
From the mathematical point of view, neural networks implement a huge multidimensional nonlinear function, whose final expression will be the result of an iterative procedure during which an adjustment fit more and more the input composition with the expected outcome. As said, the primary element of a neural

network is the perceptron, the unit in charge of weighing the incoming input and giving a combination of them as output. Fig. 2.2 shows the main working principle:

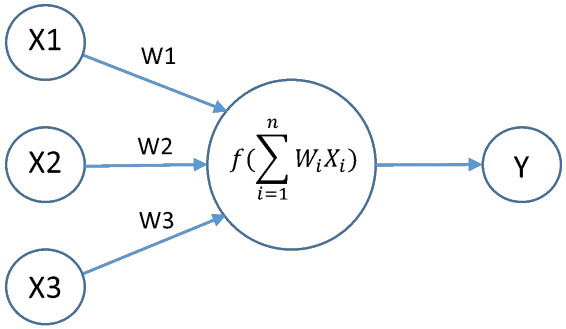

Figure 2.2: Perceptron working principle [3]

The input-output connection is formally expressed with the mathematical formula,

$$y = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{if} w \cdot x + b < 0 \end{cases} \tag{2.1}$$

where w and x are vectors representing respectively the weights and incoming inputs to the neuron, while the operation is a dot product, defined as:

$$w \cdot x = \sum_j w_j x_j \tag{2.2}$$

The neuron compares the product with a threshold to return as output either 1 or 0. The architecture of a neural network is structured over layers, describable as a set of neurons. A fundamental requirement demands the first layer match the shape of the input data point. Differently, the last layer shape is designed according to the task. In the middle, different layers may be present, defining the specific network structure.
Different structure types may be implemented, each with proper characteristics and a certain level of fittingness for a specific task. The most famous categories comprise the feedforward, the convolutional, and the recurrent networks.
In the first class of networks, data moves in one direction between the input and output node, going potentially through multiple layers, and without cycling backward in the crossed ones. Although the complexity of such networks may vary a lot, with structure presenting different layers of multiple nodes, the one-way movement of data makes Feedforward neural networks relatively simple. For these reasons, Feedforward models are mainly used for simplistic classification problems.

Convolutional networks correspond to a class designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. It has become dominant in various computer vision tasks, where the spatial correlations among pixels are well interpreted and extracted to recognize objects or actions.
Finally, recurrent networks are powerful when a model is designed to process sequential data. The model will move data forward and loop it back to previous steps to best achieve tasks and improve predictions. Looping back dataflow helps in retaining relevant information, acting as a cell of memory, and improving the process for the next input. In this way, models can relate the context of input with already processed ones, inspecting if any dependencies among them are present. For example, a predictive text system may use the memory of a previous word in a string of words to better predict the outcome of the next word. A recurrent neural network would be better suited to understand the belief behind a whole sentence compared to more traditional machine learning models.
In general, the learning process of a network is mathematically expressed by the correct tuning of weights and bias parameters: starting from an initial value, these parameters are refined with a Δw proportional to the delta present between the generated and the actual final output. Such an update is defined according to the decision rule used by the neuron. However, the activation function described in 2.1 is not able to guarantee this educated upgrade, due to a lack of continuity in the codomain function. A different activation function should be implemented, more capable of "smoothing out", intended as the capability of generating slightly different output for slightly different inputs. This is the case of the "sigmoid" function, defined as in the formula 2.3:

$$y = \sum \sigma(w \cdot x + b) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

Multiple activation functions exist, each with a different behavior according to the criteria needed to choose the output. In regard of this study, working with classification problems leads to the implementation of a Softmax function in the output layer. Defined as in the formula 2.4:

$$y = \sum \sigma(w \cdot x + b) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.4}$$

While a graphical representation is offered in fig. 2.3:

Its usage lies in the interpretation given to such function: Softmax assigns decimal values to each class, visible as the belonging probability for a multi-class problem. The summation of such probabilities is surely equal to 1, resulting in an additional constraint for the network, which helps the training procedure converge

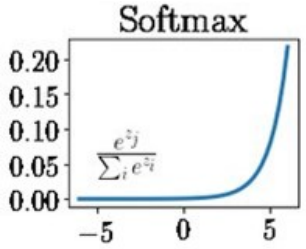

Figure 2.3: Softmax trend [4]

more quickly than it otherwise would.
A further important choice concerns the decision rule to be implemented for updating weights, aiming to have the outcome of the network as much as possible equal to the real one for each input. Considering the differences between the two quantities, an intuitive criterion is represented by the minimization of their summation. In literature, this concept is expressed as a cost function, and the objective would be minimizing it. A cost function type is represented by the quadratic one, defined as follows:

$$C(w,b) = \frac{1}{2n}\sum_x ||y(x) - a(x,w,b)||^2 \tag{2.5}$$

At this point, it seems reasonable to update the cost function value following each weight's update. This approach is deployed by the gradient descent algorithm, whose idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point because this is the direction of the steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function, characterizing the procedure known as gradient ascent. According to this algorithm, to minimize the cost function (by definition positive quantity), the needed negative Δc is proportional to the gradient of C for all the weights and their Δ, thus expressed as:

$$\Delta c \approx \nabla C \cdot \Delta v \tag{2.6}$$

Supposing $\Delta v = -\eta \nabla C$, with η defined as the learning rate, the 2.6 becomes:

$$\Delta c \approx -\eta ||\nabla C||^2 \tag{2.7}$$

To minimize $C(v)$, the update will be $v = v - \eta \nabla C$, converging eventually to a local minimum of the cost function. The algorithm may be applied with a batch approach, estimating ∇C only on a random set of training inputs and not on all of them, so to speed up the cost gradient evaluation. Exhausted the training input,

an epoch is completed; the network will be considered trained once it has gone through a sufficient number of epochs. Defined the criteria for minimizing the loss function, the absence of an algorithm performing the procedure for all the weights of each layer neuron is still present. Here the Backpropagation algorithm comes into help, computing the gradient of the loss function for a single weight by the chain rule.
The algorithm computes how the final neuron's output would have been affected by a small change in each input value, and whether the change would have pushed the result closer to the right answer, or away from it.
The result is a set of error values for each neuron in the second-to-last layer; essentially, a signal estimating whether each neuron's value was too high or too low. The algorithm then repeats the adjustment process for these new neurons in the second layer. For each neuron, it makes small changes to the input weights to encourage the network to get closer to the correct answer.
Then, once again, the algorithm uses partial derivatives to compute how the value of each input to the second-to-last layer contributed to the errors in that layer's output—and propagates the errors back to the third-to-last layer, where the process repeats once more.
The strategy strength is given by the property of efficiently treating one layer at a time, unlike a native direct computation. The algorithm computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule. A high-level explanation of the algorithm is the following:

1. Inputs X, arrive through the preconnected path.
2. X is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs.
5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

This procedure is supposed to be iterative, so repeated until the desired output is achieved. Backpropagation is claimed to be the most implemented algorithm for adjusting network parameters due to its velocity, simplicity, and ease of programming. Moreover, it has no parameters to tune apart from the numbers of input, and it is a flexible method as it does not require prior knowledge about the network. It is a standard form that generally works well, not needing any special mention of the features of the function to be learned. As a drawback, the algorithm is input-dependent and suffers from noisy data.

2.3.2 Tuning a Neural Network

As pointed out in 2.2, the neural network architecture will be subject to the training process first, and the model testing then. In this regard, there is a need of defining how the whole dataset will be employed for these two stages.
When working with ML algorithms, the dataset is split into different parts, called Train, Validation and Test: first two represent respectively the 70% and 20% of the overall dataset and are implemented during the **Model training and validation**; Test composes the 10% and is used in **Model testing**.
Train and Validation sets are necessary to train the model and verify the level of success of such training stage, suggesting ongoing modifications; Test is instead referring to data that will be employed after the training procedure to test the algorithm's effectiveness.
While the samples composing the Test set are fixed at the beginnig of the study, Train and Validation compositions can be subject to modifications along the training procedure.
In this regard, the most famous techniques are the Fixed split ratio, also known as Hold Out, and the cross-validation. In the first case, a fixed partitioning is applied, meaning that before starting the training procedure deployed by the backpropagation algorithm, both the portion and the samples of the dataset devoted to train and validation are decided. Along all the epochs, the two sets will be always constituted by the same samples. The static choice of train and validation is generally

appropriate for large datasets, as the generalization is more likely to be guaranteed.
Conversely, the other approach proposes training the model multiple times, at each turn with a different train-validation combo. In the specific, the k-fold cross-validation consists in partitioning the dataset into k sets and devoting k-1 for the training and one for the validation. In this way, the procedure can be repeated K times, each time using different samples for the training and validation step. Such a method is mostly used when working with a very small dataset, as the obtained dynamicity makes the model more robust, accounting for more variance between possible splits in training and validation points.

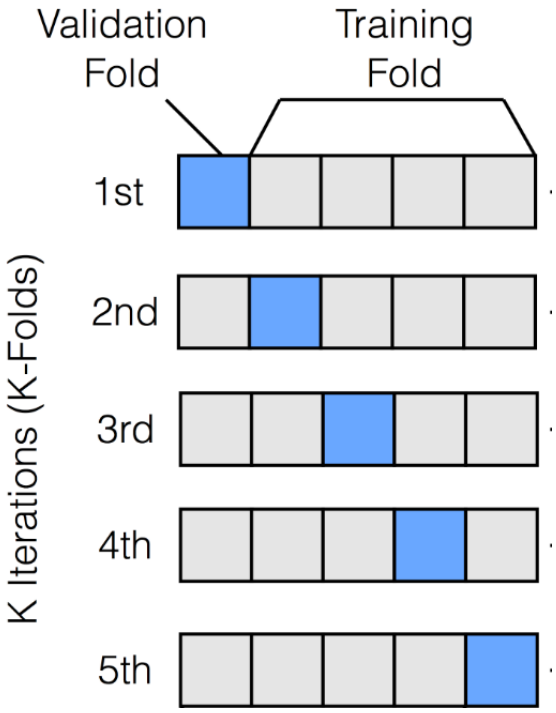

Figure 2.4: K-folds cross validation scheme [5]

Proceeding with the training model discussion, the next decision concerns the choice of its optimal hyperparameters, used to update weights and biases composing the analytical function. Given a specific architecture, it is not apriori clear which configuration could let the model express at its best; therefore, the network has to go through a tuning process to discover the parameter values for the highest learning capability.
The approach used is the classical "trial and error", training the network several

times -with a different set of hyperparameters at each turn- so treating the tuning procedure as a grid search.
During this process, a simplification appears unavoidable, otherwise, the attempts to take would be exponentially growing. In practice, decisions about parameters to fix are taken, delineating then in which direction is worth making further attempts.
Among all the NN parameters, the most important to be tuned are two -the Learning Rate and the Batch Size- both referring to the model training and validation step mentioned in 2.2.
Learning rate, encountered in section 2.3.1 gives a measure of the step size taken by the optimizer to descend the error curve. Looking at its correction tuning, when a slow convergence toward the minimum is present, the LR is considered too small; on the other hand, if an oscillation around the minimum value is present, the LR is considered too big.
Batch size is a gradient descent parameter controlling instead the number of training samples to work through before the model's internal parameters are updated. In general, using smaller batch size values shows faster convergence to a "good" solution, as the model start learning "immediately". However, using those values doe not guarantee convergence to the global minima. On the other hand, using a bigger batch size may guarantee the convergence to the local minima, but paying with lower convergence or poor generalization.
Other parameters related to the training procedure worth to be mentioned are the number of epochs and the momentum.
Moreover, when constructing a network structure there are other decisions to take: the number of neurons within a layer, the activation function used in each of it, the weight initialization, and the weight for each class, etc.
Other techniques are then implemented to prevent the most common problems affecting neural networks, like overfitting. Again, the most popular concerns the usage of early stopping conditions, or the implementation of dropout or regularization.

2.3.3 Model performance evaluation

Another important step for an ML tool -and so a neural network- is represented by the performance evaluation. Evaluating a model construction means examining its efficiency, analyzing the model building time or the classification time, or yet the scalability, or the robustness, intended as the capacity to deal with noisy or missing data.
However, the most important analysis is generally oriented toward to quality of the prediction. As stated before 2.3.1 , at the end of a prediction problem, the output

should be as much as possible equal to the real one. Once the model training terminates, the focus moves to understand its behavior when new data income, evaluating the level of success of its estimation, and the achieved reliability.
To do so, different techniques are put in place, taking into consideration various factors like the composition of the dataset or the type of the problem, here of a classification nature.
For sake of simplicity, the following discussion will treat the problem as a binary classification; however, the explanation could be easily generalized to whichever number of classes.
By definition, in a binary classification, each data point is associated with one out of two possible classes, with the ML in charge of estimating the correct one. Therefore, the situations that could occur at each prediction attempt are four. Supposing for example that the classes are represented by 1 and 0, the classifier could get right, guessing the correct class in two cases, when the actual and the estimated class are both 1 or 0, or it could get wrong in the other two where estimated and actual labels do not coincide. The four combinations create the so-called Confusion Matrix, shown in fig. 2.5:

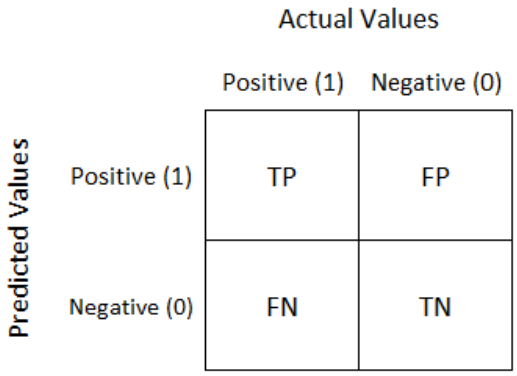

Figure 2.5: Confusion Matrix composition [6]

The diagonal of the such matrix refers to the cases in which the classifier gets right -indicated with the textbfTrue annotation- while the off-diagonal elements characterize the error taken by it (so textbfFalse estimation). textbfPositive and textbfNegative are simply referring the two classes.
Looking at the classifier indices inside the matrix, the goal is to have TP and TN as high as possible and vice versa for FP and FN. This leads to the definition of the most widely-used metric for model evaluation, i. e. the Accuracy, expressed by the mathematical formula in 2.8:

$$Accuracy = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} = \frac{\text{TP + TN}}{\text{TP + FN + TN + FP}} \quad (2.8)$$

However, accuracy is not always able to reflect the total correctness of classification estimations. There are phenomena described by the dataset whose composition represents a limitation for the usage of this index.
The classical example is represented by a highly unbalanced dataset. Supposing to work with a binary problem with 99% of the overall Test samples belonging to class 1. Disregarding the goodness of the training procedure it went through, the classifier is prone to return prediction labels as if all samples are belonging to the overwhelming class, obtaining around 99% of accuracy. In this scenario, accuracy is a strongly misleading metric, as the model presents a very high value even if it will not detect any class 0 at all.
Another limitation is generally represented by datasets whose classes have different relevance, perhaps because the misclassification of the object of a given class is more important than the other, as often occurs in the medical field (ill patients classified as a healthy result more dangerous than vice-versa). Therefore, there are scenarios where other evaluation metrics should be taken into consideration, perhaps providing assessments separately for each class. In this sense, Precision and Recall are the most popular, mainly in case of ill-conditioned accuracy. Their analytical definition is expresses in 2.9 and 2.10:

$$Precision_C = \frac{\text{Number of samples correctly assigned to C}}{\text{Number of samples belonging to C}} = \frac{\text{TP}}{\text{TP + FP}} \quad (2.9)$$

$$Recall_C = \frac{\text{Number of samples correctly assigned to C}}{\text{Number of samples assigned to C}} = \frac{\text{TP}}{\text{TP + FN}} \quad (2.10)$$

Given a class C, Precision measures the level of correctness of the predictions the classifier associated with C, while Recall returns the level of success concerning all the actual samples belonging to C.
The two metrics present an inverse relationship: considering a class C, the classifier may increase its precision score by reducing the number of predictions related to that class (changing the labels to the ones most "debatable"), thus assuming a more conservative behavior but at the cost of finding out fewer samples belonging to that class, resulting in a lower recall score. It is up to the problem nature to understand whether a higher precision or recall will pay off.
In case the objective is instead that of maximizing on average their contributions, a solution could be represented by combining somehow the two into one metric

and maximizing it. This is what occurs for the F1 score, defined as the harmonic mean of precision and recall. Its expression is given in 2.11:

$$F1_C = \frac{2\ \text{Precision}_C\ \text{Recall}_C}{\text{Precision}_C + \text{Recall}_C} \tag{2.11}$$

The next chapter 3 will point out a high imbalance characterizing the dataset provided for this study, therefore, standing out the necessity of using these last three scores for the performance evaluation.
Moreover, a new metric will be considered, representing a variant of the classical accuracy here useless for the before mentioned reasons: the "balanced" accuracy. Defined in the equation 2.12, such an index has been just devised for situations characterized by a target class that strongly overwhelms the other. Its measure intends to give the same relevance to the two contributions, despite their different composition on the overall dataset.

$$Accuracy_{balanced} = \frac{\text{Specificity} + \text{Sensitivity}}{2} = \frac{1}{2} * \left(\frac{\text{TP}}{\text{TP} + \text{TN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \tag{2.12}$$

Chapter 3

Dataset

The dataset represents the most important element for an algorithm, in charge of describing the phenomena it intends to learn about.
The perfect scenario has been depicted in the last years, thanks to the exponential growth of collected data (asserted as the BigData era), allowing those tools -considered data hungry- to perform at their best.
It is then straightforward to imagine that the higher the amount of data, the better the description of the environment where the algorithm will take place.
On the other hand, the creation of an appropriate dataset is strictly linked to the quality of the collected data, whose nature must be in accordance with the desired use case.
Working with a high dataset cardinality appears a necessary step, yet not sufficient for the correct algorithm training: collected data are raw, thus, often not immediately usable. In most cases, they have to go through a preprocessing phase, during which they are analyzed, to catch their peculiarities and strengths, then transformed to results suitable for the training process. With a more formal definition given by [65], "Data preprocessing for Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. It refers to the application of techniques (cleaning and organizing) to the raw data to make it suitable for building and training Machine Learning models."
For this study, the forecasting purposes suppose the implementation of a multivariate time series, leading to the application of specific techniques. In the following sections, a brief explanation of the feature composing the set is provided, analyzing the given features their relationships with the targeted label.
Statistical properties of those characteristics are then inspected, to infer which of them could be more informative for the ultimate prediction. A discussion related to the dataset readiness is then provided, highlighting the reason behind the implemented transformation techniques, which are finally illustrated.

3.1 Dataset of this work

Forecasting future events leveraging on the previous history requires an indispensable characteristic for the given dataset: each sample needs to be timestamped, so associated with a time instant representing when the collection occurred. In this way, respecting the sequence order offered by the timestamp, a temporal series of samples is determined: fixing any data point, all the others preceding it will constitute the past, and the subsequent ones the future. Datasets equipped with this property are called time series.
Generally, the examination of a dataset consists of two main aspects. To kick off, the comprehension of its technical composition, to achieve an understanding of the features' meaning, and their contribution to the geomagnetic event.
Then, a statistical investigation to show analytical properties, which is sometimes useful for inferring the most appropriate transformation technique. Next subsections will present respectively the technical and the statistical analysis.

Index	Name	Measure unit
0	Time	[s]
1	Time shift	[s]
2	Magnetic intensity B	[nT]
3	Bx	[nT]
4	By	[nT]
5	Bz	[nT]
6	Solar wind bulk speed	[km/s]
7	Vx	[km/s]
8	Vy	[km/s]
9	Vz	[km/s]
10	Proton density	$[cm^{-3}]$
11	Proton temperature	[K]
12	Flow pressure	[nPa]
13	Plasma pressure	[nPa]
14	Magnetic pressure	[nPa]
15	Total pressure	[nPa]
16	Plasma beta	A-dimentional
17	Alfvenicvity	[s]
18	AE	[nT]
19	SYM-H	[nT]

Table 3.1: Features composing the dataset

3.1.1 Technical analysis

As expressed also by ESA [66], most of the datasets are mainly composed of solar measurements - forming a time series- or of coronographic and solar magnetogram photos (imageset).
For this work, the dataset consists of a collection of solar wind and geomagnetic indices measured from the 1st of January 2005 up to the 31st of December 2019 in situ L1. The total number of observations is 7.888.320, taken with a 1-minute resolution, resulting in a collection of 20 features, depicted in Table 3.1.
A particular that stands out immediately is the presence of two temporal variables: **Time** is the time taken by the terrestrial magnetosphere to react to the Coronal Mass Ejection disturbance, while **Time shift** is the time needed by the CME disturbance to cover the L1-Earth surface distance, thus dependent from the CME velocity. The reason behind this distinction is promptly explained. **Time** represents the instant at which the spacecraft situ in L1 measured the CME quantities, so highlighting when the event impacted the Lagrangian point, far from the Earth about 1.6 million km. Differently, the SYM-H quantity is measured on the terrestrial surface, thus leading to potential inconsistencies given by the different spacial places where collections occurred. In this sense, combining the two offers the possibility to "relate" measurements to the same spatial point. For example, subtracting **Time shift** to **Time** can be seen as a translation of measurements from L1 to the Earth. However, no combination of those quantities has been considered, supposing that the Machine Learning tool can catch the correlation even in those circumstances. To conclude, the used time instants are given by **Time**.
Going on, **Magnetic intensity B** refers to the magnetic field intensity expressed by the CME, with **Bx**, **By**, and **Bz** representing respectively the x, y, and z coordinate. **Solar Wind Bulk Speed** represents the ejection travelling speed and **Vx**, **Vy** and **Vz** its x, y and z coordinate. **Proton density** indicates the proton number density, while **Proton temperature** its temperature. Again, **Flow pressure** considers the solar wind flux pressure, **Magnetic pressure** refers to the magnetic one, while **Plasma pressure** concerns the thermal pressure of the plasma ejection. As highlighted by Burlaga and Ogilvie in [67], an increase of the **Plasma pressure** generally corresponds to a decrease of **Magnetic pressure** and vice versa. The sum of the thermal and magnetic pressure returns **Total pressure**, while their ratio results in **Plasma beta**. **Alfvenicity** aims at offering a correlation measure between magnetic field and speed vector, while in the end **AE** and **SYM-H** are respectively the expressions of the geomagnetic disturbance severity at low and high latitudes.
Notice that in the provided dataset there are no Magnetohydrodynamics (MHD) parameters: even if those quantities would be really helpful, they are reflecting phenomena characteristics that are not occurring "punctually", but also to previous

and subsequent periods. This counteracts the spirit of the prediction task, where punctual measurements are used to forecast future values, with any possibility to exploit parameters containing information related to other periods.

3.1.2 Statistical analysis

As anticipated, the understanding of a dataset passes also for statistical analysis: this may highlight relationships among variables on one hand, and reveal information for future decisions on the other hand. As highlighted, the dataset here corresponds to a time series, formally defined as "a sequential set of data points, measured typically over successive times", whose analysis "comprises methods for analyzing time series data to extract meaningful statistics." In particular, the time series here is multivariate -multiple features composing the set- and discrete, this latter referring to the fact that measurements are taken at a discrete point in time (1 minute).
A general characterization consists of treating the quantities as random variables, thus evaluating their statistical properties, resumed in Table 3.2:

Name	mean	std	min	max
Time [s]	6.350841e+10	1.366297e+08	6.327176e+10	6.374506e+10
Time shift[s]	3.457890e+03	1.233186e+03	-4.010000e+03	1.079400e+04
Magn field [nT]	5.191268e+00	2.729287e+00	1.300000e-01	5.889000e+01
Bx [nT]	3.394247e-02	3.267459e+00	-4.452000e+01	2.819000e+01
By [nT]	-1.076916e-02	3.673980e+00	-3.435000e+01	5.730000e+01
Bz [nT]	1.698139e-02	2.926935e+00	-5.251000e+01	4.376000e+01
SWB speed [Km/s]	4.218924e+02	9.897865e+01	2.308000e+02	1.114800e+03
Vx [Km/s]	-4.207202e+02	9.855801e+01	-1.104900e+03	-2.308000e+02
Vy [Km/s]	1.073117e+00	2.346353e+01	-2.613000e+02	5.834000e+02
Vz [Km/s]	-5.559315e+00	2.112286e+01	-2.293000e+02	2.017000e+02
Proton density [cm^{-3}]	6.037258e+00	4.783631e+00	1.4000e-01	8.095000e+01
Proton temperature [k]	8.642249e+04	8.593112e+04	9.200000e+02	4.831295e+06
Flow pressure [nPA]	1.960801e+12	1.505612e+12	3.122123e+10	9.620146e+13
Plasma pressure [nPA]	1.264986e-10	1.867450e-10	2.398771e-13	1.726334e-08
Magnetic pressure [nPA]	1.368664e-10	2.171831e-10	6.724296e-14	1.379886e-08
Total pressure [nPA]	1.960801e+20	1.505612e+20	3.122123e+18	9.620146e+21
Plasma beta	1.425756e+00	2.767306e+00	7.781150e-04	7.430067e+02
Alfvenicity	6.295395e-03	6.784865e-01	-1.000000e+00	1.000000e+00
AE [nT]	3.552084e+05	1.116324e+06	1.000000e+00	5.797498e+06
SYM-H [nT]	-9.645804e+00	1.527413e+01	-3.050000e+02	1.510000e+02

Table 3.2: Statistical properties

Three main observations stand out by looking at those outcomes. First, the **count** value should be equal to the dataset cardinality whichever the feature. This is not the case, as **Alfvenicity** presents 1545 samples less, thus suggesting treating this feature carefully, as missing values present in the dataset can impact the model performance by creating a bias in the dataset. This bias can create a lack of relatability and trustworthiness in the dataset. In this regard, the next section will inspect different solutions to tackle the problem.
Going on, an interesting element is outlined by the **Standard Deviation**: defined as the index of dispersion of a set of values, it somehow expresses how much information may potentially be extracted from a characteristic. In general, the higher the value, the higher the amount of information. In this sense, it may offer a preliminary evaluation for choosing a subset of features that most affect the targeted value. However, such analysis is just preliminary, with a better counterproof offered by a correlation assessment.

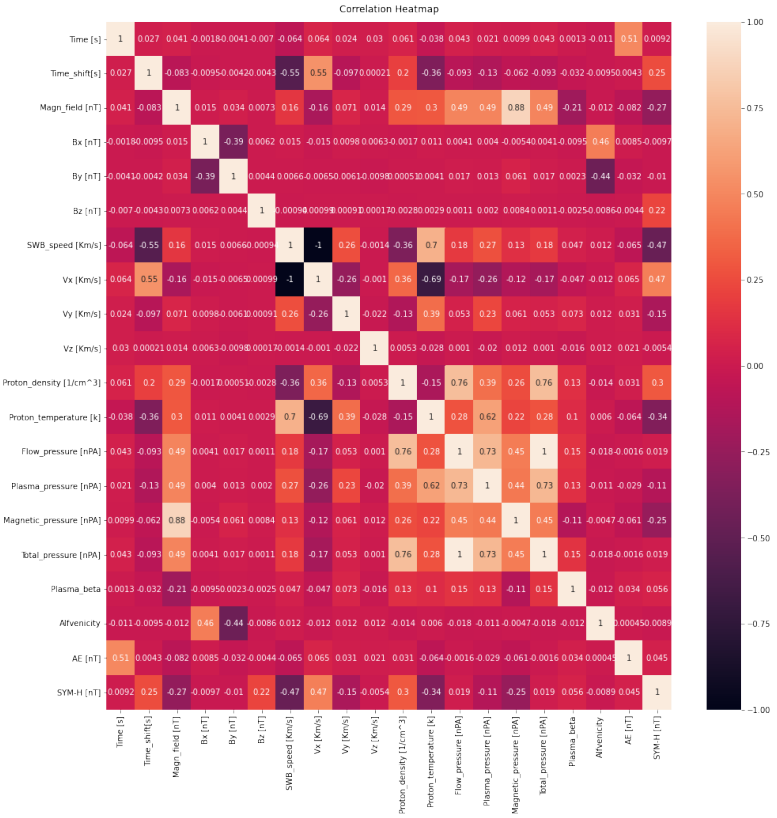

Figure 3.1: Correlation matrix

The last emerging aspect points out the need for normalization before implementing the ML models: the high magnitude of features like pressures, proton temperature, and AE may lead to the exploding gradient problem, a phenomenon often occurring when working with Neural Networks. Moreover, having features with similar scales helps to stabilize the gradient descent step, allowing the adoption of bigger learning rates or helping models converge faster for a given learning rate.
A further interesting analysis is offered by a correlation study, aiming at examining the relationship among each couple of features. As mentioned before, the significant ones could be those influencing the targeted value, SYM-H, suggesting a dependence degree, thus their potential forecasting support. In fig. 3.1, the complete correlation matrix is presented:
Generally, autocorrelation is useful to explain relationships among variables, supporting physical expression also applicable in this case. However, it is out of this work's scope to highlight those dependencies for strengthening such relationships; on the contrary, catching the ones presenting a high correlation with SYM-H could be more meaningful. A better overview of this is offered in fig. 3.2

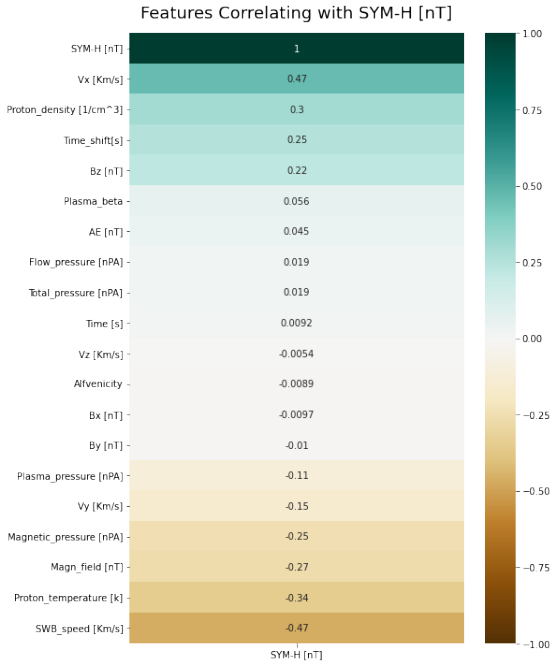

Figure 3.2: Features correlating with SYM-H

Here, a relevant contribution seems to derive from features whose absolute correlation value is higher than 0.2 -proton density, speed, and magnetic field- finding similitudes with different studies offered in 1.1.
The next chapter will treat a detailed discussion regarding the most pertinent features; however, keeping in mind those correlations may help in case of a feature selection step.
The last examination consists in evaluating the distribution of the occurrences of the targeted variable. The such examination helps in understanding whether the given dataset presents disturbances with heterogeneous properties, visible from the ML perspective as a sufficient number of samples describing the different event severities. In this case, dealing with a classification problem leads to the disposal of a balanced-class dataset, meaning that the number of samples belonging to the different classes should be equivalent, or at least none of the classes should overwhelm the others.
The distribution is illustrated in Fig. 3.3, underlining that the property is not respected at all, with a substantial imbalance leaning in favor of Not Critical events. In the specific, the number of events classifiable as Critical (presenting an

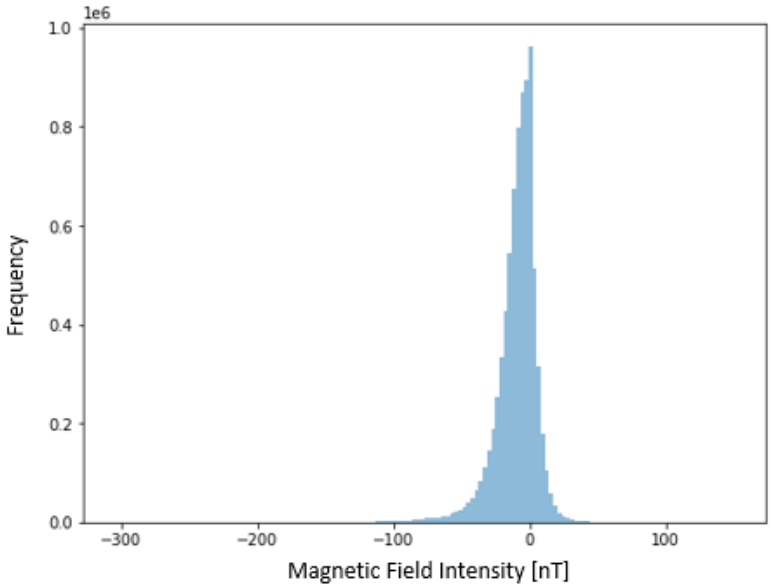

Figure 3.3: SYM-H distribution

SYM-H value lower than -50nT [10]) is 159249, corresponding to the 2,019% of the overall dataset. Thus, a transformation of the dataset for rebalancing it appears indispensable.
Such investigation will be performed in the next section, describing the problem exhaustively and employing the solutions to tackle it.

3.2 Dataset transformations

The analysis previously executed highlighted something already anticipated in 2.2: the need for a preprocessing step, whit the purpose of transforming the dataset to make it ready for the prediction step. The following subsections will go through the entire manipulation, aimed at cleaning, reshaping, and augmenting the original dataset to obtain in the end a more appropriate version for the neural network architectures.

3.2.1 Data cleaning

The first operation consists of a normalization: one of the most popular formulas expects each sample to be subtracted from the mean value (centering) and divided by the standard deviation (standardizing the scale).
The next problem regards the missing values which affect Alfvenicity. In this regard, different techniques may be implemented, generally categorized into two: Drop the record with the missing value and Impute the missing information. Dropping the missing value is however an inappropriate solution, as the correlation of adjacent observations could be lost. On the contrary, estimating or imputing the missing values can be an excellent approach to facing the problem.
The most common solutions for filling are "Last Observation Carried Forward (LOCF)", "Next Observation Carried Backward (NOCB)", "Rolling Statistics" and "Interpolation". In LOCF and NOCB, the replacement is performed by considering respectively the previous or the next non-missing value. In Rolling statistics, statistical properties can be used to impute missing values by aggregating the previous non-missing ones. An example is given by considering the type of averaging of previous samples, like the simplex or the moving one. Finally, the interpolation technique estimates the missing value by using past and future known data points. However, a simpler decision could be that of removing completely this feature, as excluding an ill-conditioned quantity -characterized also by a weak correlation with the targeted value- does not seem so despicable.
For this case study, the choice of interpolating seemed to be the best one; however, in case of a filtering step for the features to be provided as input, Alfvenicity will likely be discarded.

3.2.2 Data windowing

The next step consists of a reorganization of the time series measurements, forming a more suitable dataset for the neural network. Reshaping the data structure is generally an operation performed before the algorithm training; however, the need of compensating for the unbalance leads to taking the decision as soon as possible. In this case, reorganizing the dataset means rearranging the input samples present in the whole time series. As the algorithm will leverage the previous history for forecasting future behavior, a first decision concerns the number of samples it should look back at to perform then the estimation. Given the whole dataset, the neural network should be provided with a set of consecutive samples representing the trend of the phenomena of interest and then return a forecasting estimation. To be more precise, the definition of a window is necessary, where a bigger fraction of samples will contribute to the input values, on top of which the network will train and adjust the belief, while a smaller part will represent the output, so the value(s) for which to estimate the class severity.
The parameters which should be defined for a window are essentially 3, which are the size, the resolution among adjacent samples, and the fraction to devote as input (and as output). The size is generally chosen according to some periodicity encountered in the time series; however, no significant trend, seasonal or daily behavior was detected for this study. As a consequence, an informative contribution could be provided by the nature of such ejection: their traveling speed, which allows them to cover the distance with the lagrangian point at most in some hours, allow them to give importance only to the most recent history of the phenomena. Thus, an educated guess corresponding to a daily period seems to be adequate. Regarding the time resolution between adjacent samples, the dataset resolution of 1 minute gives total freedom for this decision; however, choosing a too-low resolution among samples would lead to having windows composed of too many samples. Again, a ponderate choice could be based on the time taken by the disturbance to cover the L1-earth distance, suggesting to consider as adjacent window samples those for which the occurrence differs of 1 hour. In the end, the decision could be that of fixing 24 samples for the input -1 hour spaced- and plugging then a variable part representing the prediction output, consisting of just one sample for the SSM or multiple ones for MSM.
All of the features will be inserted in the windows input, while the output will be composed only of SYM-H, taking the role of the label to be predicted. The creation of a set of windows from the time series is then performed, obtaining at the end (7888296 - *output_length*) elements. In the case of SSM, there will be 7888296 windows, while in MSM, if i.e. *output_length* = 8, then there will be 7888288.
Notice that such reshaping served only for reorganizing the samples of the time

series dataset, without impacting its composition. This means that the dataset still presents high imbalances, which will be tackled in the next section.

3.2.3 Data Augmentation

Facing the imbalance of this dataset's work represented the main task required in the data preprocessing step, involving techniques aiming to change its composition, yet respecting the correct description of the phenomena under study. However, a necessary remark has to be pointed out before the deployment of such a transformation. As anticipated in 2.3.2, the overall dataset is divided into three different subsets -Train, validation, and test- each with a different role in the ML prediction step: first two are implemented in the **Model training and validation** stage to train the network and verify the level of success of the such process, suggesting appropriate adjustment; the test is treated as data that will be revealed after the training procedure and will be employed for testing the effectiveness of the algorithm, so useful for the **Model testing** step.
Since altering the dataset composition helps the model become more robust and able to generalize during the training process, the only subsets which have to be subject to modifications are the training and validation. On the contrary, there is no reason in modifying the test component, which should preserve the original composition to properly reproduce the ejection occurrences. Since the only Train and Validation sets have been subject to changes, the following discussion will involve only these two sets' contributions.
Given the high dataset cardinality, in the beginning, an undersampling procedure to "thin out" the dataset has been performed, removing random windows associated with non-critical events.
On the other side, the minority of the relevant events suggested increasing the dataset by adding samples belonging to the interested classes.
One of the most popular oversampling options is the acquisition of new data by generating synthetic samples. Such a solution is called Data Augmentation: this expression refers to the implementation of a set of techniques aiming to increase the amount of data by generating new points starting from the existing ones.
Citing [7], "data augmentation is a universal model-independent data side solution attempting to increase the generalization ability of trained models by reducing overfitting and expanding the decision boundary of the model [68]. The need for generalization is essential for real-world data and can help networks overcome small datasets or datasets with imbalanced classes [69], [70]."

Continuing [7], data augmentation techniques can be divided in four main categories, as summarized in fig. 3.4:

Disregarding the explanation of these technique classes, out of scope for this work, the evaluation done here consisted in choosing which of those techniques

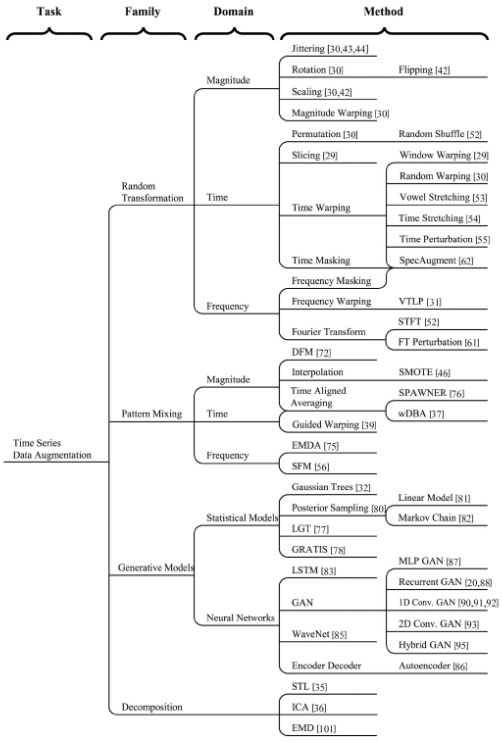

Figure 3.4: Time Series Data Augmentation Techniques [7]

could have guaranteed acceptable results, in agreement with the series nature, avoiding the implementation of complex solutions which may have impacted the structure of such events. For those reasons, a particular focus has been first given to the first two categories, considered easier to be implemented and making fewer modifications to the overall data property. Then, other options would have been taken into consideration if necessary. Looking at the random transformation and pattern mixing families and considering the nature of the time series, transformations in the Magnitude and Time domain seem to be the more suited, while those in frequency should not, given the non-periodicity of those events.
In particular, transformation methods like Jittering, Scaling, Magnitude, and Time Warping have been implemented. Defining the expression of time series may be beneficial for the explanation of those techniques. The general characterization is the following:

$$\mathbf{x} = \begin{bmatrix} x_1, & x_2, & x_3, & \dots & x_N \end{bmatrix} \tag{3.1}$$

The series $\mathbf{x}$ can be seen as a vector containing N elements, with N equal to the window input size. A transformation of such series will be indicated as $\mathbf{x}'$.

A time series window representation is shown in Fig. 3.5:

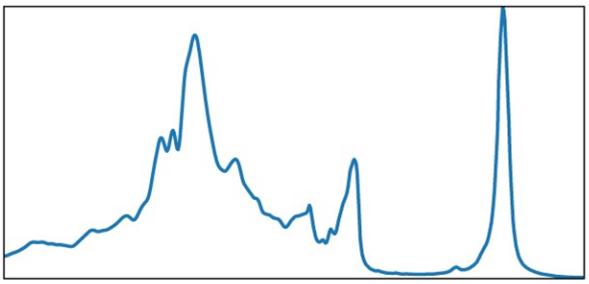

Figure 3.5: Time Series window [7]

Jittering refers to the act of adding noise to the time series. It can be mathematically defined as:

$$\mathbf{x'} = \begin{bmatrix} x_1 + \epsilon_1, & x_2 + \epsilon_2, & x_3 + \epsilon_3, & \dots & x_N + \epsilon_N \end{bmatrix} \tag{3.2}$$

where ϵ_i is typically Gaussian noise added to the i-th element. Its expression is

$$\epsilon \sim \mathcal{N}(\mu, \sigma^2)$$

, with μ equal to 0 and σ^2 equal to a random variable, proportionally set to each feature's magnitude.
A jittered version of the signal previously shown is displayed in Fig. 3.6:

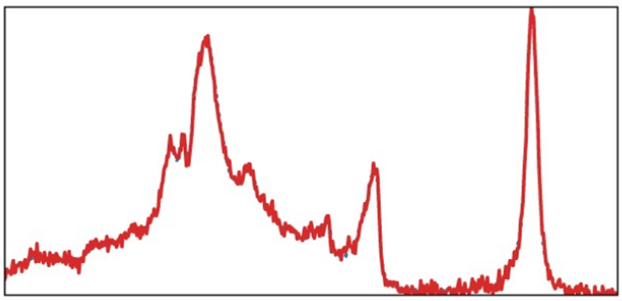

Figure 3.6: Jittered Time Series window [7]

Scaling changes the global window magnitude, or intensity, by applying a scalar factor to each element. It is defined as:

$$\mathbf{x'} = \begin{bmatrix} \alpha\ x_1, & \alpha\ x_2, & \alpha\ x_3, & \dots & \alpha\ x_N \end{bmatrix} \tag{3.3}$$

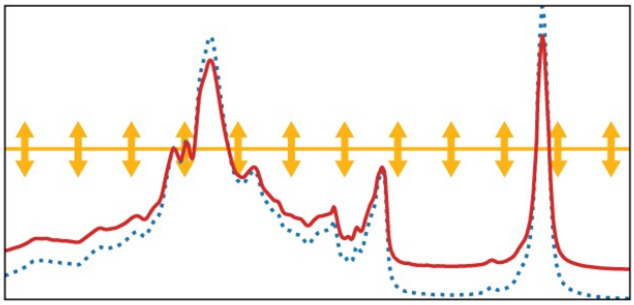

Figure 3.7: Scaled Time Series window [7]

Again, to avoid dramatic changes, a random scalar value belonging to [0.9, 1.1] has been chosen. A scaled version of the same signal is displayed in Fig. 3.7:

Magnitude warping aims at warping signal's magnitude by using a smoothed curve. Practically, it consists of dynamic scaling, in the sense that for each element, a different factor α is applied.

$$\mathbf{x}' = \begin{bmatrix} \alpha_1 \, x_1, & \alpha_2 \, x_2, & \alpha_3 \, x_3, & \dots & \alpha_N \, x_N \end{bmatrix} \tag{3.4}$$

The warping version is shown in Fig. 3.8:

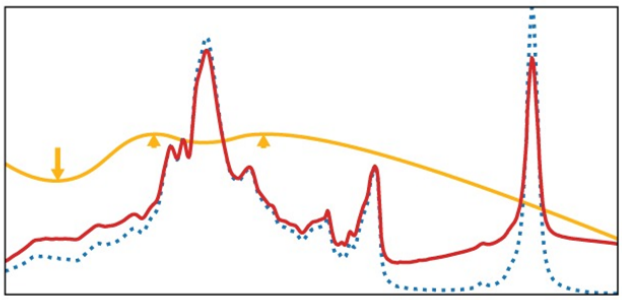

Figure 3.8: Magnitude warping applied to Time Series window [7]

Finally, Time warping adopts a different distortion approach, perturbing the trend in the temporal domain. In a sense, the act can be seen as a dynamic shrink or stretch of the given time window. The application of this method is analytically defined through the application of a smooth warping function; then expressed as:

$$\mathbf{x}' = \begin{bmatrix} x_{\tau_1}, & x_{\tau_2}, & x_{\tau_3}, & \dots & x_{\tau_N} \end{bmatrix} \tag{3.5}$$

where τ is the function that warps the time steps based on a smooth curve. Again, the temporal warping version is illustrated in Fig. 3.9:

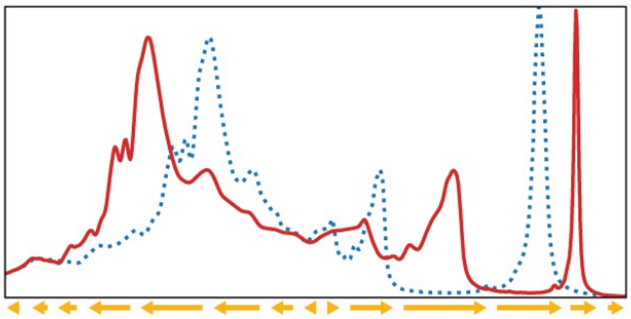

Figure 3.9: Time warping applied to Time Series window [7]

Regarding the mixing procedure, the one proposed for this work is the SMOTE (Synthetic Minority Oversampling TEchnique). Designed to combat data with an imbalanced nature, it interpolates patterns belonging to the under-represented class. In particular, for a given sample $\mathbf{x}$, another sample $\mathbf{x_{NN}}$ is selected from its nearest neighbors. Next, the difference between the two samples is multiplied by a random value λ in a range of [0, 1]. The analytical result is the following:

$$\mathbf{x'} = \mathbf{x} + \lambda|\mathbf{x} - \mathbf{x_{NN}}| \tag{3.6}$$

This work examined two SMOTE alternatives, being different in the type of distance used for evaluating neighbors: the Euclidean distance and a DTW-based distance. First, theoretical research has been performed, aiming to understand which of them could result more appropriate. In particular, citing [71], "the Euclidean distance is a widely used measure of similarity between two vectors which, by definition, enforces a lock-step one-to-one mapping between corresponding observations in any two-time series samples.
Yet, the similarity between two time series measured by the Euclidean distance can be severely distorted when minor misalignments along the time axis, for example, due to instrument measurement error, are present between the two-time series". This means that in the case of signals whose trend is similar but are misaligned in time, the euclidean approach would not be able to catch their similarity.
In those cases, DTW offers a solution to overcome the misalignment problem. It shrinks or stretches the region of one time series to best suit the other. Again [71], DTW allows a non-linear alignment between observations and is, therefore, invariant to misaligned data, as also displayed in 3.10. Since the ejection under study is characterized by a variable speed, the SMOTE augmentation DTW-based

might result more suited for identifying the most similar samples.

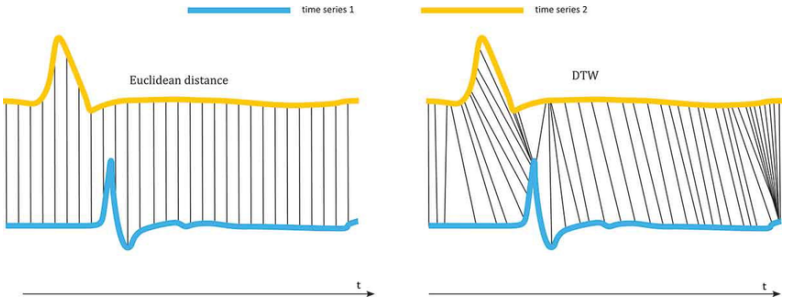

Figure 3.10: Graphical comparison between Euclidean and DTW Distance [8]

.

All the explained techniques have been equally used: for each sample within the dataset, one of them has been randomly chosen and implemented, giving a similar contribution to the augmented dataset construction.
In the end, starting with a dataset of 7888296 measurements, including about 2% of critical events, its augmented version contains 4589519, whose 32% are critical events.

Chapter 4

Neural Network

The neural network description offered in the section 2.3 highlighted the working principle behind their structures, with each generally more suited for specific assignments. Classic feedforward architectures work well with classification or regression problems; convolutional models are mostly implemented in computer vision tasks, while recurrent approaches are the favorite ones in the case of temporal data analysis.
The latter technique seems more appropriate for the nature of the problem under study, where temporal reliance among consecutive measurements represents a key aspect for forecasting purposes.
RNNs constitute a subset of neural networks whose working principle is based on the explicit handling of the order of the observations. Such capability implies that the recurrent approach can acknowledge the chronological context of input sequences, leveraging their temporal dependencies to provide better predictions.
That said, the nature of the problem could expect greater outcomes from this class of networks; however, embracing a set of diversified networks may result in more benefits, since the forecasting nature of the problem is faced as a classification challenge.
In addition, the continuous evolution of those architectures avoids the total supremacy of a structure for a specific use case.
As a consequence, the ultimate choice consisted of a heterogeneous set of networks, basing the decision both on the works presented in 1.2, but also on the composition of the dataset and some expert advice.
The next section will provide an explanation of the proposed architectures, describing both their structural composition and the reasons behind some solutions adopted to enhance performances.

4.1 Choice of Neural Network Architectures

The architectures proposed for the study are 7, constructed with different complexities and equipped with multiple solutions. The first architecture is used as a baseline reference, followed by three basic models with a similar number of parameters, each belonging to the previously mentioned different categories, and three hybrid and more complex structures.
The choice of those networks allows working on one hand with some basic architectures, to investigate the suitability of each category with the objective, and on the other hand with more complex structures, obtained by mixing basic ones or implementing additional solutions.
All the networks adopt the same input layer, whose shape is given by (batch size, number of samples, number of features), and the same output shape, defined according to the task to fulfill.
The targeted feature has been represented by SYM-H, but rather than predicting its continuous value, architectures estimate a class consisting of the belonging criticality range. By doing so, multiple classification problems have been designed, aiming to infer a different piece of information about the CME arrival time and its degree of severity. In this regard, the next chapter will explain the SCENARIOs investigated, providing from the network perspective the various output dimensions. What is left immutable regards all the middle layers, which have been constructed to let different networks have similar complexity.
Some architecture dimensions coincide with the previously mentioned parameters, strictly related to the subtask. For this reason, their structural representations will include them as well.
A list of such quantities is provided 4.1:

- **B** represents the batch size;
- **W** represents the input window size, so the number of consecutive samples describing the event behavior;
- **F** represents the number of features;
- **C** represent the number of classes, which are 2 in the binary tasks and 4 in the categorical one;
- **O** represents the number of forecasted time istants, equal to 1 in the SSM approach and up to 8 in the MSM ;

4.1.1 Linear

The Linear model is the simplest network architecture, as its composition comprises only the input shape, a "flatten" layer, and the output shape.
As the term suggests, the flatten layer flattens the previous layer's output, in this case corresponding to the input itself.
In particular, giving a 3-dimensional input (batch size, number of samples, number of features), the layer will convert it to a bi-dimensional one, with shape (batch size, number of samples*number of features). Linear is viewed as a baseline structure, acting as a reference in the project, useful to contextualize the results of trained models, although the lack of complexity may result in little predictive power.
A representation of the architecture is offered in 4.1:

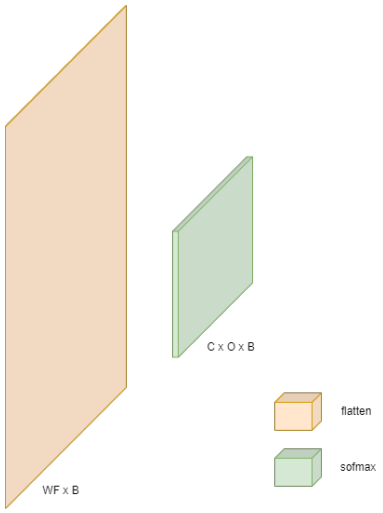

Figure 4.1: Linear model architecture

4.1.2 MLP

An immediate increase in the Linear model complexity is achievable by adding further layers, making the overall structure deeper. In literature, those types of models are called Multi-Layer Perceptron (MLP), basing their strength on the "depth" factor, following the deep learning paradigm 2.
A multilayer architecture could have a higher capacity for extracting knowledge from more complex phenomena, increasing network performances. In general, adding "depth" boosts a network's representational capability and aids in learning increasingly abstract characteristics, preventing in some cases the underfitting problem.

Taking inspiration from Wang et al.[72], an MLP has been devised by adding to the Linear model three dense middle layers, obtaining the overall the feedforward architecture proposed in fig. 4.2:

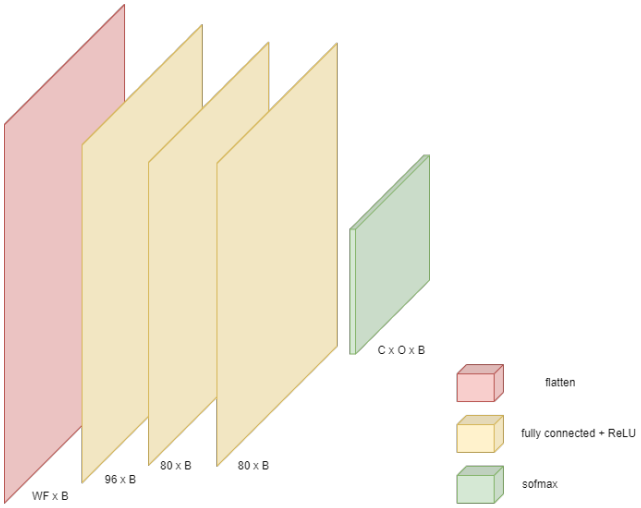

Figure 4.2: MLP model architecture

4.1.3 CNN

A convolutional architecture has the intrinsic ability to learn the position and scale patterns inside data, thus suggesting their implementation principally when dealing with image datasets, where the structure composing the data sample -a photo- contains spatial relationships.
Recently, CNN has shown a certain suitability when employed for time series forecasting tasks too. Citing [73], "CNN offers dilated convolutions, in which filters are used to compute dilations between cells. Continuing [73], "the size of the space between each cell allows the neural network to understand better the relationships between the different observations in the time series". Therefore, such work has been enlarged with a basic convolutional architecture, eager to find out whether those capacities may be put at the service of the given time series challenges.
The proposed structure is presented in fig. 4.3:

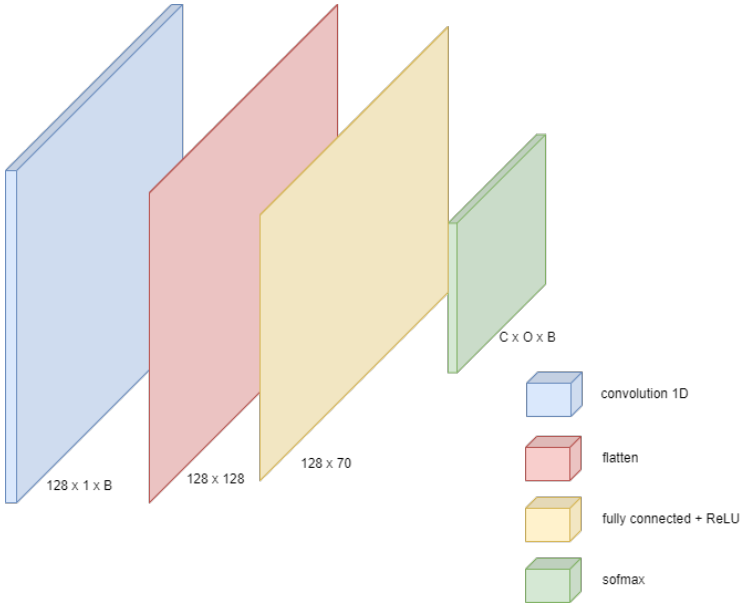

Figure 4.3: CNN model architecture

4.1.4 LSTM

For the last network categories, an LSTM structure has been taken into consideration. In general, the RNNs are considered more suited to time series tasks thanks to their structure layout, as the recurrent connections constitute an effective memory for the network, with the capacity of "storing" past trends to better estimate future ones.
As a consequence, recurring networks are used for pattern processing of variables whose length will be treated as sequences, divided into pieces, and displayed on the network at a different time step [74].
On the other hand, RNNs suffers from vanishing gradient and short-memory problem, leading to a loss of information, caused by the repeated use of the recurrent weight matrix in RNN.
Differently, in LSTM networks, the recurrent weight matrix is replaced by an identify function in the carousel and controlled by a series of gates [75]. The input gate, output gate, and forget gate act like a switch that controls the weights and creates long-term memory function.
The LSTM model presented in the work shows the following architecture:

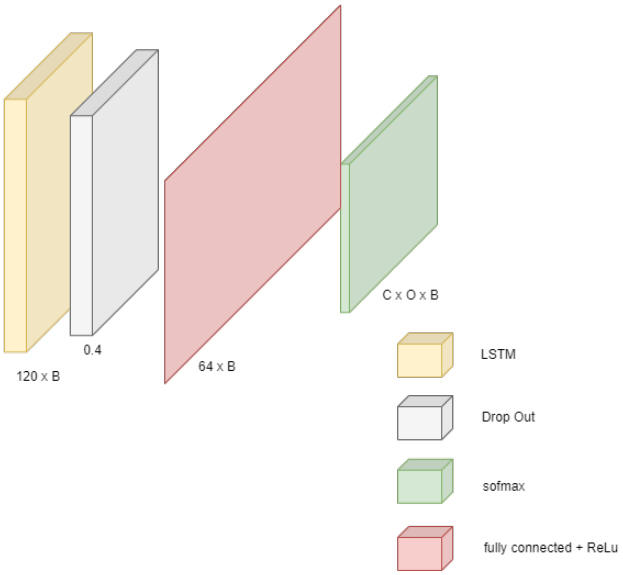

Figure 4.4: LSTM model architecture

4.1.5 Deep CNN

The three previously proposed models comprise a structure for each of the presented network categories. Despite other studies putting in place their deployment for similar use cases, it is not said that those basic models will perform well for the described challenge. Having already mentioned some of the MLP and CNN limitations, it should be noted that LSTM may arise problems as well.
Famous for their ability to learn temporal dependencies in sequences, they have difficulty with long-term dependencies in long sequences. Moreover, a critical aspect highlighted by Karim et al. [76] indicates that in some cases LSTMs by themselves cannot isolate the data into linear separable classes.
Therefore, constructing more complex architectures by combining these basic ones might lead to more appropriate structures.
In spirit with the Deep Learning paradigm 2, an immediate solution could be obtained just by replicating a basic model multiple times. As already pointed out, increasing the number of layers may lead to a better understanding of data patterns with a higher degree of complexity. Such replication is often applied for the convolutional networks, as the stacking of CNN models might lead to a better decomposition of the input. By doing so, the overall extraction is "fragmented", inferring at each step low-level features, to come up in the end with a sequential procedure potentially able to acquire more complex information.
The proposed architecture consists in replicating four times the beforementioned

CNN, leading to the structure presented in fig.4.5, called for sake of simplicity Deep CNN:

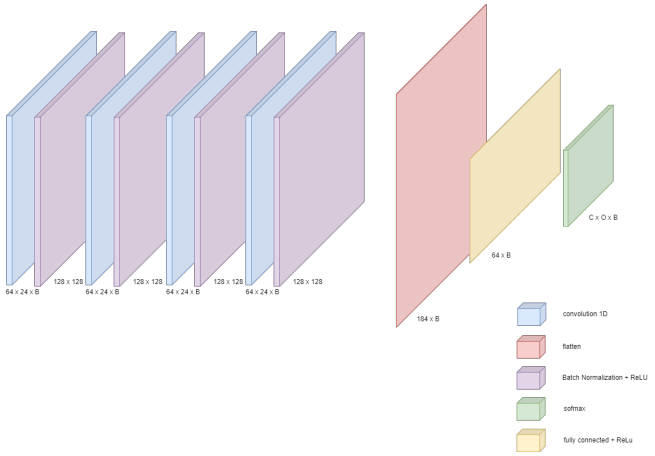

Figure 4.5: Deep CNN model architecture

4.1.6 LSTMFCN

Another solution consists in implementing multiple architectures, with each inferring its knowledge, and then combining their extraction output. In general, merging different models guarantees a better generalization, as the mixing of features extracted according to different criteria may enhance the overall understanding. In this regard, citing again [76], Karim states that the combination of an LSTM and an FCN block may transform the data into separable classes, thus leading to a construction of an LSTMFCN, with the features of recurrent and convolutional extractor converging in one model, obtaining a more robust set of features. Inspired by those researches, an LSTMFCN model has been considered, constructed by combining the previous Deep CNN with an LSTM block. The architecture is shown in the following fig 4.6:

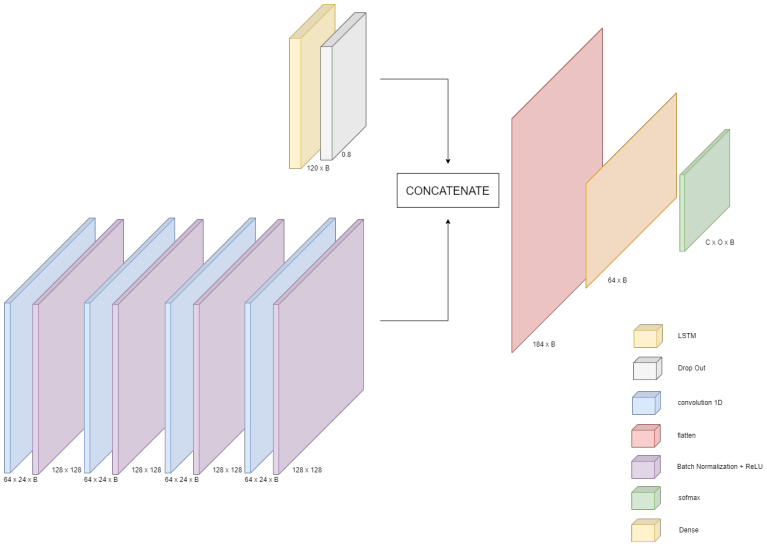

Figure 4.6: LSTMFCN model architecture

4.1.7 DEEPCNN WITH A SKIP CONNECTION

The last model implements a solution that might be beneficial for the prediction task: the skip connection. Skip connection consists of a path in charge of feeding a layer with the original input or the output of a previous layer.
Skip connections were introduced to mitigate the vanishing gradient problem and ensure reusability. One of the most popular networks equipped with this solution is the ResNet, where the output of each layer feeds both the next and the successive layer, here summed to the intended input.
As pointed out by Hao Li et al. [9], the benefits deriving from this element are reflected in the loss function, which in the case of skip connection implementation is characterized by a smoother surface, as visible in fig. 4.7, leading to an easier local minimum finding, speeding up the convergence.

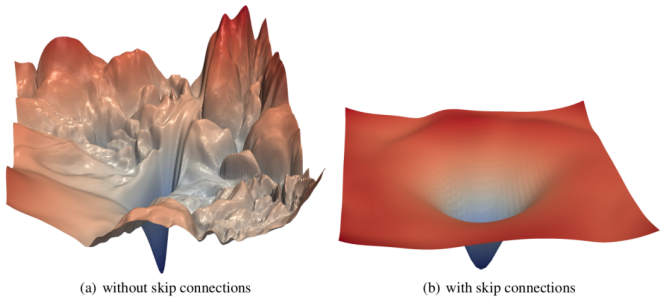

Figure 4.7: Loss Function surfance with and without skip connection [9]

Among the various existing skip connections, the one employed in this work represents the easiest solution, creating a new model by simply connecting through a link the input data with the output of the network extraction block. The outcome of these two factors is given by their summation, avoiding the introduction of further parameters and the growth of model complexity. Looking at the architectures so far proposed, such a strategy makes sense on a multi-stage model like the DEEPCNN or LSTM-FCN. However, the choice quickly fell on the first, on one hand, to slightly emulate the ResNet architecture, on the other hand, to avoid making the model unnecessarily too complex like it would happen in selecting the LSTMFCN. A diagram of the proposed model is visible in fig. 4.8:

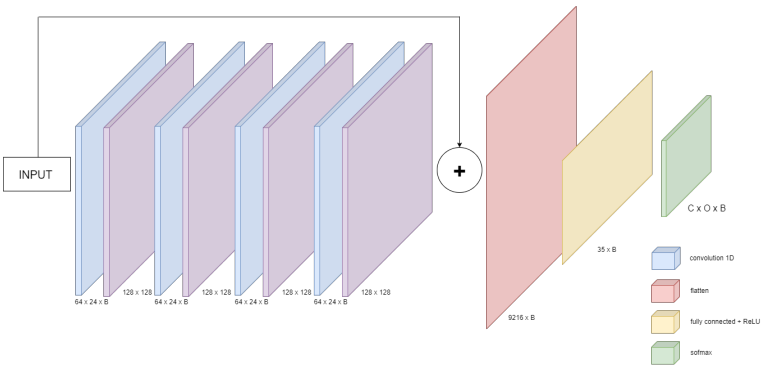

Figure 4.8: Deep CNN with a Skip Connection model architecture

Despite the differences in their structural composition, neural network classifiers present some common working principles. The main property concerns the role given to each layer/subset of layers within an architecture. In general, the

first piece corresponds to a feature extractor, in charge of extracting high-level features belonging to a low-dimensional space and not necessarily meaningful from the scientific perspective.
Such extraction feeds one or more fully connected layers, useful to learn potential non-linear combinations of these features. At the end of the stack, a classification layer receives the flow and maps it to all different classes, assigning to each of them a value corresponding to the belonging probability.
In general, the extraction step is performed by receiving in a unique block all input features; however, in some circumstances treating each input quantity disjointly may be beneficial for describing phenomena where the different contributions may conflict with each other. In this regard, supposing N input indices, the features extractor block will be then composed of N "sub_blocks", each of them in charge of extracting knowledge of only one feature. The other network layers are left immutable; therefore, the N outputs will converge to a unique fully connected block followed by a classifier layer.
This solution has been pursued in the given study, as the first attempts with all features highlighted some ill-conditioned problems, perhaps related to conflicts among them.
For all the proposed networks, two different configurations have been deployed, considering the features extractor bloc as a unique piece, treating all input together, and as a set of the subblock, each dealing with a single feature. An example in the different extraction deployment is shown in figs

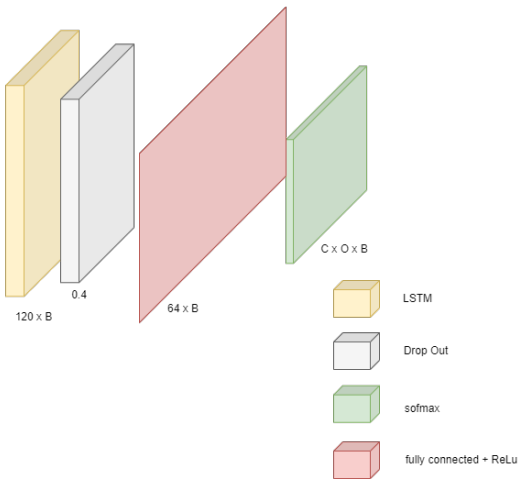

Figure 4.9: LSTM model architecture

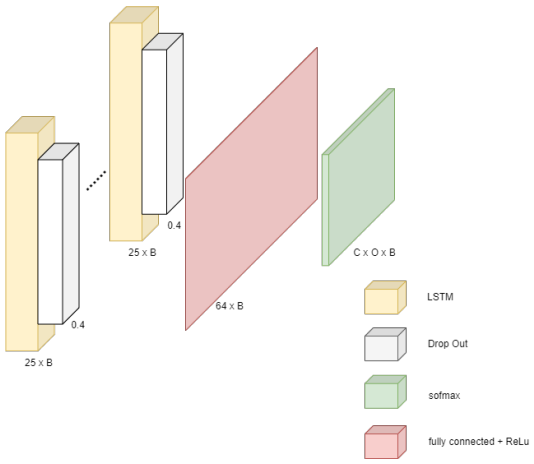

Figure 4.10: LSTM model architecture implementing single-feature extractor

Chapter 5

Experiments

As remarked multiple times, the goal of this work is predicting geomagnetic events, described through time series observations, to acknowledge whether future behavior may lead to harmful consequences.
Such forecasting has been treated as a classification, with the neural network assigning a label according to the expected degree of criticality, in single or multiple future time instants.
Dealing with a relevant number of parameters would lead to analyzing an exponential number of configurations, requiring an unfeasible amount of effort and resources. For this reason, at the end of each examined scenario, some choices have been taken, in view of the next cases to study.
The next section will present which scenarios have been investigated, explaining their peculiarities. Following, a description will provide all the parameters and set of alternatives tested along such scenarios. In the end, the experimental procedure examines the tests carried out in each scenario, the outcomes, and the decisions made for subsequent ones.

5.1 Setting the scene

Implementing neural network classifiers requires several choices, related to different factors affecting the classification construction for forecasting purposes. The definition of multiple prediction tasks supposes the continuous readaptation of the NN architecture, to properly match the input shape and to provide the outcome compliant with the challenge.
Moreover, the learning procedure of a network involves different decisions to take, aiming to optimally all those hyperparameters and additional solutions to let the model perform at its best. In this regard, this section delineates all the designed challenges -each defined as a distinct SCENARIO offering a different knowledge

shade- and the factors experimented with along those scenarios.

5.1.1 Choice of the scenarios

As mentioned in the previous chapter, the targeted feature is represented by SYM-H, but rather than predicting its continuous value, architectures estimate a class consisting of the belonging criticality range. By doing so, multiple classification problems have been designed, whose definition has been affected by the level of understanding the study intends to return.
Two aspects have been considered crucial for the event characterization: its degree of severity, and its temporal extension.
As regards the first, the definitions of multiple gravity levels coincided with the number of classes assigned. Therefore, two different cases have been inspected, supposing 2 classes first, then enlarging up to 4.
Moreover, given the variability of the CME arrival time, the same binary and classification tasks have been performed considering two different time horizons. The number of classes choice reflects the type of information to reveal: a binary classifier may alert whether the future event measurement will be critical or not; the categorical instead may return more details, pointing out the severity nuance. For the temporal extension, the model has been set to provide punctual information first, with the label reflecting the behavior in a single future point; then estimating multiple time instant labels as if to inspect the phenomena over a longer period and quantize its gravity. The first type belongs to the Single Step Model (SSM), while the second to the Multi Step Model (MSM).
The choice led to the evaluation of three situations: predicting the class 1 hour ahead, then 8 hours ahead, and finally estimating the severity up to 8 hours.
Combining different values of these two allows the creation of six scenarios, here listed:

- **SCENARIO 1**: The number of classes is 2, while the number of predicted steps is 1, one hour ahead. This results in a Single Step Binary classification model;
- **SCENARIO 2**: The number of classes is 4, while the number of predicted steps is 1, one hour ahead. This results in a Single Step Categorical classification model;
- **SCENARIO 3**: The number of classes is 2, while the number of predicted steps is 1, eight hours ahead. This results in a Single Step Binary classification model;
- **SCENARIO 4**: The number of classes is 4, while the number of predicted

steps is 1, eight hours ahead. This results in a Single Step Categorical classification model;

- **SCENARIO 5**: The number of classes is 2, while the number of predicted steps is 8. This results in a Multi Step Binary classification model;

From the Network perspective, such scenarios reflect the implementations of five distinct output layers.
Again, to avoid the exponential growth of the number of tests, the experimental procedure followed a pyramidal approach, finding the best configurations of each architecture in the Single Step SCENARIO 1 and 2, then repeating the classifications with a longer time horizon, and finally evaluating their suitability for Multi-Step challenges.

5.1.2 Choice of the parameters

Different architecture settings have been tested within each scenario; taking into consideration factors of various nature: the different network architectures, with the double feature extractor block; the provided set of input features, the hyperparameters characterizing the training procedure, etc. The following list 5.1.2 illustrates the parameters and options taken into consideration:

- **Number of input features**: At the beginning of the prediction step, the optimal set of features -potentially returning the best performances- is unknown. Useful information may be retrieved from the technical and statistical analysis carried out in 3.1.2, whose description of relationships among the features highlighted the one more correlated to the targeted SYM-H.
  That said, the first trials were performed with all features as input; however, results pointed out an ill-conditioned problem, with all classifiers trivially assigning the class label "Not Critical" -the majority- to all the samples. Such a situation may be caused by conflicts arising among features.
  The next attempt has been diametrally opposed, first feeding all models with only SYM-H as input, and then redoing the execution considering as input the targeted label coupled with each of the 19 features. The purpose was bifold: on one hand, understanding which quantity conflicts with the label; on the other hand inferring which could enhance performances compared to the initial univariate classification.
  The test highlighted that AE was creating conflict, thus demanding its exclusion. Conversely, the features offering the greater support were: Magnetic Intensity B, Bz, Solar Wind Bulk Speed, Vx, Proton Density, Proton Temperature, Total Pressure, SYM-H.
  Unfortunately, poor results came out even with the 18 features left, as all

models showed a Critical class' precision/recall value lower than 50%. In a sense, models classify as if they randomly choose the label for each sample; therefore, better results are achievable only if picking a smaller set of features. In this regard, two different feature subsets have been considered, whose composition has been inspired by both the correlation knowledge provided in 3.1.2 and support outlined from the previous executions. The subsets are :

- SET A = (Magnetic Intensity B, Bz, Solar Wind Bulk Speed, Proton Density, Total Pressure, SYM-H);
- SET B = (Magnetic Intensity B, Bx, By, Vy, Vz, Proton Density, Flow pressure, plasma pressure, magnetic pressure, plasma beta, SYM-H).

SET A composition considered which correlations among the 12 features previously mentioned count the most. SET B considered also the relevance declared by other studies analyzed for this work 1.1.3.

- **Batch size and learning rate tuning**: As for the input features, the best set of batch size values is apriori unknown. However, other studies suggested working with values considered "standard" for a tuning procedure. These are Batch size: (64, 128, 256); Learning rate: (0.001, 0.0001, 0.00001).
- **Deployment of adaptive LR and class weight solutions**: Further solutions have been investigated, leveraging mainly the phenomena' nature.
  First, the usage of a class weight approach has been considered, as suggested by the different relevance of the two classes of events. Next regarded the implementation of a variable Learning Rate during the network training: in case of a specific number of epochs elapsed without any improvement, the value would be decreased.

5.2 Experimental procedure

The conducted experimental procedure went through the presented SCENARIOs, investigating in each of them multiple model configurations composed of different parameter choices.
SCENARIO 1 and 2 were the ones in which the previously mentioned preliminary attempts took place, while the next SCENARIOs characterization can be considered as the natural consequences of results achieved in the first two.
In SCENARIO 1, the investigation involved all 7 architectures, each with two different feature extraction approaches. Every network has been tested giving as input both SET A and SET B, and performing a grid search over the batch size

and learning rate values.
A first analysis provided the best combo of BS and LR for each model equipped with each extractor block type and receiving each SET of inputs. The Comparison of the outcomes obtained with the different hyperparameter combinations does not allow the inference of meaningful information, justifying also similarity shown by the results, with the best combo better of some decimals.
Defined the best combo for each network, the next step proposed a comparison of the model performances equipped with the two different feature extractor blocks. Both in the case of SET A and SET B, simpler models return better results if adopting a feature extractor for every single input, thus confirming the sensation of conflicts presence considered before; conversely, complex models work slightly better with the JOINT approach.
The next comparison regarded the best version of each model structure in the case of SET A and SET B received as input, clearly highlighting that the smaller set returned better performances. The unknown conflicts among features do not allow the inference of any clear justification. However, the confirmation of an adversarial dataset suggests working with fewer, more relevant inputs. An additional test consisted in equipping each of the 7 best configurations with the solutions proposed in 5.1.2, showing that the adoption of different weights for the two classes improved performances; on the contrary, the variable LR did not lead to any enhancement. At the end of this batch of tests, the seven best model configurations are given in 5.1:

NETWORK	FEATURE SET	EXTRACTION	(BS, LR)	CLASS WEIGHT	ADAPTIVE LR
LINEAR	SET A	DISJOINT	(128, 0.0001)	YES	NO
MULTIDENSE	SET A	DISJOINT	(128, 0.00001)	YES	NO
CNN	SET A	DISJOINT	(256, 0.0001)	YES	NO
LSTM1	SET A	DISJOINT	(256, 0.0001)	YES	NO
DEEPCNN	SET A	JOINT	(256, 0.001)	YES	NO
LSTMFCN	SET A	JOINT	(128, 0.00001)	YES	NO
DEEPCNN_{SKIP}	SET A	JOINT	(64, 0.00001)	YES	NO

Table 5.1: Scenario1: best model configurations

Such configurations returned the results visible in Table 5.2.

NETWORK	[P_{NC}, P_C]	[R_{NC}, R_C]	[$F1_{NC}$, $F1_C$]	$Acc_{Balanced}$
LINEAR	[0.99916031, 0.64144372]	[0.99718376, 0.85738468]	[0.99817106, 0.73385855]	0.927284222274384
MLP	[0.99893408, 0.6523938]	[0.99743617, 0.81887431]	[0.99818456, 0.72621505]	0.9081552403709636
CNN	[0.99907578, 0.64769956]	[0.99730561, 0.84299619]	[0.99818991, 0.73255493]	0.9201509028626345
LSTM	[0.99926837, 0.62007491]	[0.99684681, 0.87579348]	[0.99805612, 0.73607666]	0.9363201466513011
DEEPCNN	[0.99865784, 0.62809917]	[0.99731432, 0.77190013]	[0.99798563, 0.69261439]	0.8846072224995983
LSTMFCN	[0.99885897, 0.61402095]	[0.99702213, 0.80617859]	[0.9979397, 0.69709999]	0.9016003561672015
$DEEPCNN_{SKIP}$	[0.99895988, 0.62788446]	[0.99713279, 0.82331782]	[0.9980455, 0.71244164]	0.9102253152471837

Table 5.2: Scenario1: model results

What immediately stands out is the highest suitability of simpler models, with metrics related to the Critical classes always better, suggesting that SCENARIO 1 has very low complexity, so complex models are not needed.
The such supposition is backed by the incredible LINEAR model result, which performs as well as the others, despite the little complexity. In general, all models present a Recall value higher than the Precision ones, highlighting their not-conservative behavior in predicting critical events. Such an outcome is in spirit with the different weight given to the two classes of events: obtaining a false alarm may result in less harm than missing a criticality. LSTM is the model that slightly outperforms all the others, reaching a Recall value of 0.733, reflected in a balanced accuracy of 0.936.
Conversely, MULTIDENSE is the model among the simpler ones which performs the worst; DEEPCNN performs the worst at all, highlighting that for the given task the only stacking of multiple CNN does not give benefits. However, the $DEEPCNN_{SKIP}$ seems to be the only complex one performing as well as the simpler models, thus suggesting a certain utility given by the connection skip, even better than combining the multistage CNN with an LSTM block.
Looking ahead, the first exclusion regards the input features, as SET B always returned the worst results, thus justifying its rejection.
As a consequence, the same amount of experiments took place in SCENARIO 2, but only considering SET A as input features.
Again, simpler models confirm their higher expediency towards the Single Step challenge, with LSTM1 showing the highest performances.
A particular standing out from all architecture results is the highest capacity in detecting INTENSE events, and the complete inability in spotting those SUPERs. Fortunately, the latter perturbance class represents not even 1% of the overall events, so downsizing the gravity derived from their total misdetection. Examining the feature extractor block alternatives, simpler models showed again better functioning than the one treating the quantities disjointly. This aspect may represent

a paramount outcome for the study, declaring that future network implementations should adopt such an approach. Moreover, the input features set might be widened with greater ease, as the independent extractor can straight outline the contribution of a newly inserted feature.
MLP offers again the worst metrics among the simpler structures, suggesting its exclusion for the next scenario.
As for SCENARIO 1, the most complex model shows worse results, displaying a particular deficiency in detecting MODERATE events, with DEEPCNN appearing inadequate, so suggesting its exclusion.
LSTMFCN and DEEPCNN with skip show a similar result, with the single-feature extractor block working slightly better. For the incoming Multi-Step scenarios, making an effort on both seems reasonable.
In the overall, models Combinations offering the highest performances for each model are given in 5.3:

NETWORK	FEATURE SET	EXTRACTION	(BS, LR)	CLASS WEIGHT	ADAPTIVE LR
LINEAR	SET A	DISJOINT	(64, 0.001)	YES	NO
MLP	SET A	DISJOINT	(256, 0.00001)	YES	NO
CNN	SET A	DISJOINT	(128, 0.00001)	YES	NO
LSTM	SET A	DISJOINT	(64, 0.00001)	YES	NO
DEEPCNN	SET A	JOINT	(128, 0.001)	YES	NO
LSTMFCN	SET A	DISJOINT	(256, 0.00001)	YES	NO
DEEPCNN_{SKIP}	SET A	DISJOINT	(128, 0.001)	YES	NO

Table 5.3: Scenario2: best model configurations

The such configurations returns the metrics results shown in tab. 5.4

NETWORK	[P_{NC}, $P_{Moderate}$, $P_{Intense}$, P_{Super}]	[R_{NC}, $R_{Moderate}$, $R_{Intense}$, R_{Super}]	[$F1_{NC}$, $F1_{Moderate}$, $F1_{Intense}$, $F1_{Super}$]	$Acc_{Balanced}$
LINEAR	[0.999, 0.534, 0.943, 0]	[0.996, 0.861, 0.943, 0]	[0.998, 0.659, 0.943, 0]	0.7005
MLP	[0.999, 0.534, 0.921, 0]	[0.997, 0.796, 0.878, 0]	[0.998, 0.639, 0.899, 0]	0.6674
CNN	[0.999 0.533 0.989, 0]	[0.996, 0.824, 0.920, 0]	[0.998, 0.647, 0.954, 0]	0.6852
LSTM	[0.999, 0.509, 0.932, 0]	[0.996, 0.878, 0.943, 0]	[0.998, 0.645, 0.938, 0]	0.7043
DEEPCNN	[0.998, 0.603, 0.952, 0]	[0.998, 0.646, 0.865, 0]	[0.998, 0.624, 0.906, 0]	0.6272
LSTMFCN	[0.999, 0.557, 1, 0]	[0.997, 0.752, 0.866, 0]	[0.998, 0.640, 0.928, 0]	0.6538
DEEPCNN_{SKIP}	[0.999, 0.548, 0.962, 0]	[0.997, 0.708, 0.871, 0]	[0.997, 0.618, 0.914, 0]	0.6440

Table 5.4: Scenario2: model results

As previously anticipated, the tuning procedure will not take place in the next scenarios, working only with the best configurations highlighted in SCENARIO 1 and 2, using SET A as input features.
MLP and DEEPCNN will be discarded, given their poor suitability for those simpler scenarios, leading to the implementation of the other five models, whose best

configurations are exploiting for the extraction approaches for the binary classification, while for the categorical one they are all based on the single-feature extractor block.
SCENARIO 3 and 4 are a sort of repetition of the previous tasks, this time considering a longer time horizon. In the specific, the label associated to each test sample refers to the event criticality in corresponde of 8 hour ahead. Such choice is justified by the fact that CMEs present a variable speed, resulting in a different arrival time on the Earth (up to some hours at most). Same configuration models have been considered for these two more challenging estimations. For SCENARIO 3 and 4, results are visible in Fig. 5.5 and 5.6:

NETWORK	[P_{NC}, P_C]	[R_{NC}, R_C]	[$F1_{NC}$, $F1_C$]	$Acc_{Balanced}$
LINEAR	[0.99450097, 0.54977778]	[0.99756969, 0.31122732]	[0.9972429, 0.39746193]	0.52561366
CNN	[0.99592926, 0.39661972]	[0.99733542, 0.30053362]	[0.99663185, 0.34195507]	0.64893452
LSTM	[0.99520552, 0.58802309]	[0.99928969, 0.32395945]	[0.99724342, 0.41776159]	0.66162457
LSTMFCN	[0.9964612, 0.69431818]	[0.99899612, 0.39124867]	[0.99772705, 0.50047782]	0.69512239
$DEEPCNN_{SKIP}$	[0.99583608, 0.70795515]	[0.99931955, 0.28303095]	[0.99757477, 0.40439158]	0.64117525

Table 5.5: Scenario3: model results

NETWORK	[P_{NC}, $P_{Moderate}$, $P_{Intense}$, P_{Super}]	[R_{NC}, $R_{Moderate}$, $R_{Intense}$, R_{Super}]	[$F1_{NC}$, $F1_{Moderate}$, $F1_{Intense}$, $F1_{Super}$]	$Acc_{Balanced}$
LINEAR	[0.996, 0.571, 0, 0]	[0.999, 0.252, 0, 0]	[0.998, 0.369, 0, 0]	0.4039
CNN	[0.996, 0.130, 0, 0]	[0.993, 0.227, 0, 0]	[0.994, 0.166, 0, 0]	0.4065
LSTM	[0.995, 0.406, 0, 0]	[0.999, 0.201, 0, 0]	[0.997, 0.269, 0, 0]	0.3999
LSTMFCN	[0.995, 0.253, 0.290, 0]	[0.997, 0.300, 0.015, 0]	[0.996, 0.223, 0.028, 0]	0.4400
$DEEPCNN_{SKIP}$	[0.996, 0.282, 0, 0]	[0.997, 0.232, 0, 0]	[0.996, 0.255, 0, 0]	0.4099

Table 5.6: Scenario4: model results

As expected, the higher complexity stands out performances that are worse than in SCENARIO 1. Again, the recurrent approach shows the best results, with LSTMFCN outperforming all other models, so expressing its effectiveness for long-dependent relationships.
For SCENARIO 3, It expresses a balanced accuracy of about 69%, which drops to 44% in SCENARIO 4. The higher time horizon shows also the necessity of a higher number of INTENSE events, as none of the models has been able to detect them with such a long time leg in SCENARIO 4. Also, DeepCNN with the Skip connection offers good outcomes, showing the same effectiveness as the LSTM. A last piece of information pointed out is represented by the precision values, which, unlike the previous SCENARIOs, are higher than the recall ones, thus expressing a more conservative approach adopted by the models; however, resulting in less predictive capacities for the critical events.

SCENARIO 5 introduces a new level of complexity, requiring the models to

predict several labels corresponding to the event class at i-hour ahead, with i = 1,2, ...,8. Completed the training for the five architectures, the test step is performed, resulting in the outcomes displayed in fig. 5.1.

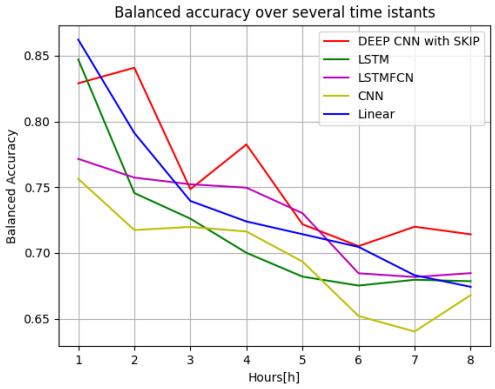

Figure 5.1: Graph showing balanced accuracy over time

As expected, all the model presents an accuracy trend going down as time goes by. The Deep CNN with the connection Skip presents the best result, maintaining an accuracy value higher than 0.7 for all eight predictions. On the contrary, LSTM does not appear as performant as in the Single Step challenges, resulting even worse than the Linear model, despite the good prediction capabilities shown for the very first value ahead. In this regard, the LSTMFCN offers an improvement for the hours in the middle of the period considered, but again not performing as good as the Deep CNN with the connection Skip.

Chapter 6

Conclusiosn

The presented work shows the implementation of Deep Learning techniques for the prediction of disturbances linked to the space weather field. Leveraging a time series dataset provided by the Osservatorio astronomico di Torino, multiple Neural Network architectures were deployed (through the HPC servers of HPC@Polito Research Team), aiming to classify the criticality of future disturbances.
Following the Machine Learning procedure, the dataset has been first analyzed, then preprocessed according to the conditions required by a NN classifier. The main challenge regarded the data augmentation, highly needed given the strong imbalance nature of such measurements. Different techniques have been executed, like jittering, scaling, time and magnitude warping, and SMOTE. By doing so, the portion of the dataset devoted to the training and validation step moved from the 2% up to the 32% of critical events. For the prediction step, multiple architectures have been proposed, choosing networks differing in composition and complexity. Simpler models consisted in creating structures like the Linear, MLP, CNN, and LSTM, while more complex compositions have been developed by combining previous structures - like Deep CNN or LSTMFCN- or adopting further solutions, like the Deep CNN with a connection skip.
Multiple classification problems have been designed, aiming to infer a different piece of information about the CME arrival time and its degree of severity. Each problem consisted of a SCENARIO, with the first 2 representing a Single Step challenge for forecasting the event criticality 1 hour ahead, adopting respectively a binary and categorical classification. In the first case, the classifier only estimates whether the event will be critical or not, in the second it may distinguish among different critical ranges. SCENARIO 3 and 4 repeated the previous two tasks, this time providing an estimation with a longer time horizon, equal to 8 hours. In the end, SCENARIO 5 represented a multi-step estimation, establishing for multiple time instants whether the event will be critical. Other choices influencing the classification tasks were given by the features to be provided as input, the composition

of the feature extraction block, the hyperparameters for the training procedure, and other options like the implementation of a class weight and adaptive LR strategy.
SCENARIO 1 highlighted the high difficulty in detecting the optimal set of input features, choosing in the end the one containing fever but more relevant ones. Considering also SCENARIO 2, the single-feature extraction block seems to offer high performances: given the conflict among the set of features, treating each of them independently at the extraction stage guaranteed high knowledge. Looking at the results of these two scenarios, LSTM slightly performed better than other models, while the MLP and Deep CNN showed a complete inadequacy, thus being excluded for the next challenges. On the other side, a high criticism is shown in the categorical classifications, as the scarcity of SUPER events, even after the augmentation, did not allow the models to classify them. SCENARIOs 3 and 4 remarked on the higher effectiveness of the recurrent approach for Single Step prediction, this time led by the LSTMFCN which outperformed all the others.
Again, the necessity of increasing the events characterized by higher intensity stood out, involving also the INTENSE ones, as fulfilling a categorical task requires a greater balancing among the multiple disturbances of gravity. Scenario 5 stood out the poor multi-prediction capabilities of the LSTM model -slightly enhanced by the implementation of the LSTMFCN- and the good performances of the Deep CNN with the skip connection, which outperformed all the others guaranteeing a balanced accuracy for the detection of criticality in at least the 70% of cases.
Further enhancement of this study could be oriented toward a deeper understanding of the features to be used for the prediction, analyzing one-by-one the features excluded and testing whether their single contribution could be constructive or disruptive.
Other augmentation techniques like generative models could be taken into consideration, mainly to increase the diversity of INTENSE and SUPER events, in case of cateogircal challenges to face.
Also, the integration different datasets could be taken into consideration, perhaps combining the constructed time series windows with coronagraph images. In the end, Deep learning tools have shown a valid alternative to the classical physics-based models, pointing out how the exponential growth of data and computing capacity asserted in society 4.0 offers valid support for very complex problems as the one studied, even in absence of meaningful knowledge regarding the phenomena of interest.

Bibliography

[1] ESA. L1, the first lagrangian point, . URL https://www.esa.int/Science_Exploration/Space_Science/L1_the_first_Lagrangian_Point#:~:text=The%20L1%20point%20is%20perhaps,the%20Sun%20and%20the%20Earth.

[2] NoAA. Coronal mass ejections. URL https://www.swpc.noaa.gov/phenomena/coronal-mass-ejections#:~:text=Coronal%20Mass%20Ejections%20(CMEs)%20are,magnetic%20field%20(IMF)%20strength.

[3] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, 2018.

[4] N. S. Johnson, P. S. Vulimiri, A. C. To, X. Zhang, C. A. Brice, B. Kappes, and A. Stebner. Machine learning for materials developments in metals additive manufacturing. 2020.

[5] Pytorch k-fold cross-validation using dataloader and sklearn. URL https://androidkt.com/pytorch-k-fold-cross-validation-using-dataloader-and-sklearn/.

[6] S. Narkhede. Understanding confusion matrix. URL https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[7] PLoS ONE. An empirical survey of data augmentation for time series classification with neural networks. 2021.

[8] Peter Ranacher and Katerina Tzavella. *How to compare movement? A review of physical movement similarity measures in geographic information science and beyond*. Taylor and Francis, 2014.

[9] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. 2018.

[10] D. Telloni. Statistical methods applied to space weather science. 2022.

[11] F. Webb and A. Howard. Coronal mass ejections: Observations. 2012.

[12] J. W. Dungey. Interplanetary magnetic field and the auroral zones. 1961.

[13] D. Telloni, F. Carbone, E. Antonucci, R. Bruno, C. Grimani, and U. Villante. Study of the influence of the solar wind energy on the geomagnetic activity for space weather science. 2020.

[14] R. K. Burton, R. L. McPherron, and C. T. Russell. An empirical relationship between interplanetary conditions and dst. 1975.

[15] R. D'Amicis, D. Telloni, and R. Bruno. The effect of solar-wind turbulence on magnetospheric activity. 2020.

[16] UCAR. The sunspot cycle, 2017. URL `https://scied.ucar.edu/learning-zone/sun-space-weather/sunspot-cycle`.

[17] NASA. The day the sun brought darkness, . URL `https://www.nasa.gov/topics/earth/features/sun_darkness.html`.

[18] NASA. Carrington-class cme narrowly misses earth, . URL `https://science.nasa.gov/science-news/science-at-nasa/2014/02may$_$superstorm`.

[19] B. Steigerwald and NASA R. A. Weintraub. Safeguarding our satellites from the sun. URL `https://www.nasa.gov/mission_pages/stereo/news/stereo_1859.html`.

[20] NASA S. Frazier. Real martians: How to protect astronauts from space radiation on mars. URL `https://www.nasa.gov/feature/goddard/real-martians-how-to-protect-astronauts-from-space-radiation-on-mars`.

[21] NASA B. Mendez. Solar flares and coronal mass ejections. URL `https://www.nasa.gov/audience/foreducators/9-12/features/F_Dangers_of_Solar_Flares_and_CME.html`.

[22] P. T. M. Loto'aniu, K. Romich, W. Rowland, S. Codrescu, D. Biesecker, J. Johnson, H. J. Singer, A. Szabo, and M. Stevens. Validation of the dscovr spacecraft mission space weather solar wind products. 2022.

[23] ESA. Introducing: Esa vigil, . URL `https://www.esa.int/Space_Safety/Vigil`.

[24] d. Odstrcil. Modeling 3-d solar wind structure. 2003.

[25] J. Pomoell and S. Poedts. Euhforia: European heliospheric forecasting information asset. 2018.

[26] Z. Smith and M. Dryer. Mhd study of temporal and spatial evolution of simulated interplanetary shocks in the ecliptic-plane within 1 au. 1990.

[27] Y. J. Moon, M. Dryer, Z. Smith, Y. D. Park, and K. S. Cho. A revised shock time of arrival (stoa) model for interplanetary shock propagation: Stoa-2. 2002.

[28] G. Tóth, I. V. Sokolov, and T. I. Gombosi. Space weather modeling framework: a new tool for the space science community. 2005.

[29] X. S. Feng and X. H. Zhao. A new prediction method for the arrival time of interplanetary shocks. 2006.

[30] J. A. Linker P. Riley, R. Lionello, , and Z. Mikic. Corotating interaction regions during the recent solar minimum: the power and limitations of global mhd modeling. 2012.

[31] P. Riley, J. A. Linker, and Z. Mikic. On the application of ensemble modeling techniques to improve ambient solar wind models. 2013.

[32] B. van der Holst, I. V. Sokolov, and X. Meng. AlfvÉn wave solar model (awsom): coronal heating. 2014.

[33] M. Jin, W. B. Manchester, and B. van der Holst. Data-constrained coronal mass ejections in a global magnetohydrodynamics model. 2017.

[34] J. Wang, X. Ao, and Y. Wang. An operational solar wind prediction system transitioning fundamental science to operations. 2018.

[35] S. Poedts, A. Lani, and C. Scolini. European heliospheric forecasting information asset 2.0. 2020.

[36] P. K. Manoharan, N. Gopalswamy, S. Yashiro, A. Lara, G. Michalek, and R. A. Howard. Influence of coronal mass ejection interaction on propagation of interplanetary shocks. 2004.

[37] R. Schwenn, A. dal Lago, E. Huttunen, and W. D. Gonzalez. The association of coronal mass ejections with their effects near the earth. 2005.

[38] B. Vršnak and T. Žic. Transit times of interplanetary coronal mass ejections and the solar wind speed. 2007.

[39] M. Dumbovic, A. Devos, B. Vrsnak, D. Sudar, L. Rodriguez, D. Ruzdjak, K. Leer, S. Vennerstrom, and A. Veronig. Geoeffectiveness of coronal mass ejections in the soho era. 2015.

[40] M. Vandas, S. Fischer, M. Dryer, Z. Smith, and T. Detman. Parametric study of loop-like magnetic cloud propagation. 1996.

[41] Y. M. Wang, P. Z. Ye, S. Wang, G. P. Zhou, and J. X. Wang. A statistical study on the geoeffectiveness of earth-directed coronal mass ejections from march 1997 to december 2000. 2002.

[42] H. Xie, L. Ofman, , and G. Lawrence. Cone model for halo cmes: application to space weather forecasting. 2004.

[43] R. Schwenn, A. Dal Lago, E. Huttunen, and W. D. Gonzalez. The association of coronal mass ejections with their effects near the earth. 2005.

[44] M. Nunez, T. Nieves-Chinchilla, and A. Pulkkinen. Prediction of shock arrival times from cme and flare data. 2016.

[45] E. Paouris and H. Mavromichalaki. Effective acceleration model for the arrival time of interplanetary shocks driven by coronal mass ejections. 2017.

[46] B. Vršnak and N. Gopalswamy. Influence of the aerodynamic drag on the motion of interplanetary ejecta. 2002.

[47] B. Vršnak, T. Žic, D. Vrbanec, M. Temmer, T. Rollett, and C. Möstl. Propagation of interplanetary coronal mass ejections: The drag-based model. 2013.

[48] C. Möstl, T. Amerstorfer, E. Palmerio, A. Isavnin, C. J. Farrugia, and C. Lowder. Forward modeling of coronal mass ejection flux ropes in the inner heliosphere with 3dcore. 2018.

[49] E. Camporeale. The challenge of machine learning in space weather nowcasting and forecasting. 2019.

[50] D. Sudar, B. Vrsnak, and M. Dumbovi. Machine learning reveals systematic accumulation of electric current in lead-up to solar flares. 2015.

[51] Rong Li and Jie Zhu. Solar flare forecasting based on sequential sunspot data. 2019.

[52] C. Pandey, R. A. Angryk, and B. Aydin. Solar flare forecasting with deep neural networks using compressed fulldisk hmi magnetograms. 2021.

[53] Jun Chen, Weifu Li, Shuxin Li, Hong Chen, Xuebin Zhao, Jiangtao Peng, Yanhong Chen, and Hao Deng. Twostage solar flare forecasting based on convolutional neural networks. 2022.

[54] D. Sudar, B. Vrsnak, and M. Dumbovi. Predicting coronal mass ejections transit times to earth with neural network. 2015.

[55] V. Delouille, S. Hofmeister, M. Reiss, B. Mampaey, M. Temmer, and A. Veronig. Coronal holes detection using supervised classification. 2018.

[56] Dattaraj B. Dhuri, Shravan M. Hanasoge, and Mark C. M. Cheung. A new tool for cme arrival time prediction using machine learning algorithms: Catpuma. 2018.

[57] Yimin Wang, Jiajia Liu1, Ye Jiang3, and Robert Erdélyi. Cme arrival time prediction using convolutional neural network. 2019.

[58] Yurong Shi, Jingjing Wang, Yanhong Chen, Siqing Liu, Yanmei Cui, , and Xianzhi Ao. Impacts of cmes on earth based on logistic regression and recommendation algorithm. 2022.

[59] Pengyu Wang, Yan Zhang, Li Feng, Hanqing Yuan, Yuan Gan, Shuting Li, Lei Lu, Beili Ying, Weiqun Gan, and Hui Li. A new automatic tool for cme detection and tracking with machine-learning techniques. 2019.

[60] Yanru Sun, Zongxia Xie, Yanhong Chen, and Qinghua Hu. Accurate solar wind speed prediction with multimodality information. 2022.

[61] URL `http://www.hpc.polito.it`.

[62] David Petersson. Ai vs. machine learning vs. deep learning: Key differences. URL `https://www.techtarget.com/searchenterpriseai/tip/AI-vs-machine-learning-vs-deep-learning-Key-differences`.

[63] Raquel Iniesta, Daniel Stahl, and Peter Mcguffin. Machine learning, statistical learning and the future of biological research in psychiatry. 2016.

[64] Safwana Haque and George Loukas. Machine learning based prediction versus human-as-a-security-sensor. 2018.

[65] A. Hasannejad. Data preprocessing in machine learning. URL `https://www.kaggle.com/code/alirezahasannejad/data-preprocessing-in-machine-learning`.

[66] ESA. Monitoring space weather, . URL `https://www.esa.int/Space_Safety/Monitoring_space_weather2`.

[67] L. F. Burlaga and K.W. Ogilvie. Magnetic and thermal pressure in the solar wind. 1970.

[68] Shorten C and Khoshgoftaar TM. A survey on image data augmentation for deep learning. 2019.

[69] Blagus R and Lusa L. Smote for high-dimensional class-imbalanced data. 2013.

[70] Hasibi R, Shokri M, and Dehghan M. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. 2019.

[71] J. Leung and R. James. A measure of distance between time series: Dynamic time warping. URL `https://www.informs.org/Publications/OR-MS-Tomorrow/A-measure-of-distance-between-time-series-Dynamic-Time-Warping`.

[72] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. 2017.

[73] Borovykh A, Bohte S, and Oosterlee CW. Dilated convolutional neural networks for time series forecasting. 2018.

[74] D. Gjylapi, A. Hyso, and E. Proko. Recurrent neural networks in time seriesprediction. 2018.

[75] Hochreiter S. and Schmidhuber J. *Long short-term memory. Neural Computation.* 1997.

[76] F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. 2019.