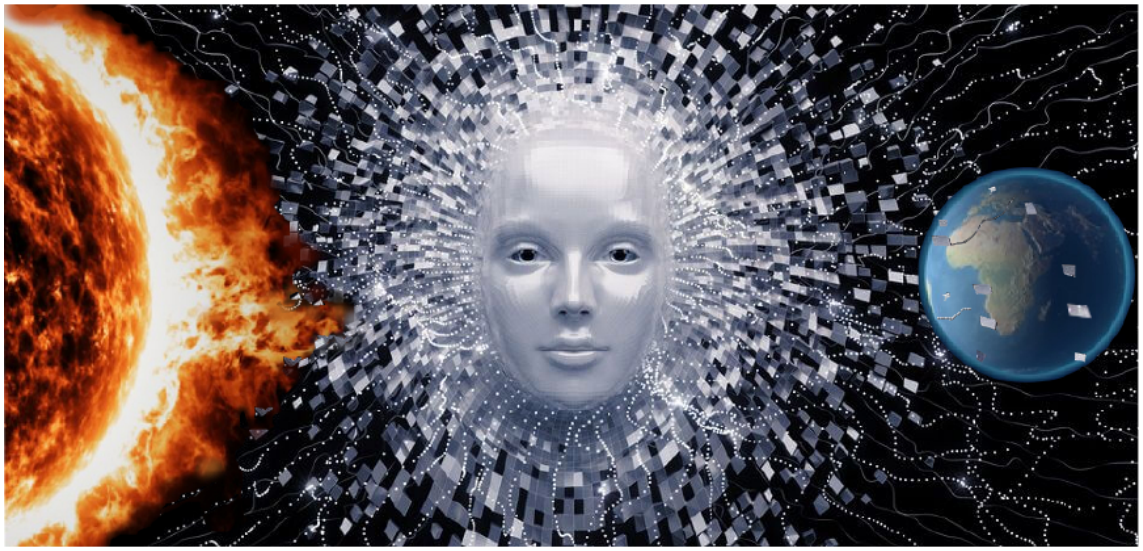
KU LEUVEN

FACULTY OF SCIENCE

Processing Solar Images to forecast Coronal Mass Ejections using Artificial Intelligence



Savvas Raptis

Supervisor: Prof. Giovanni Lapenta
Centre for Mathematical
Plasma-Astrophysics (KU Leuven)

Co-supervisor: *Dr. Jorge Amaya*
Centre for Mathematical
Plasma-Astrophysics(KU Leuven)

Thesis presented in
fulfillment of the requirements
for the degree of Master of Science
in Astronomy and Astrophysics

Academic year 2017-2018

Processing Solar Images to Forecast Coronal Mass Ejections using Artificial Intelligence

By

Savvas Raptis

Centre for mathematical Plasma Astrophysics (CmPA)
University of Leuven

Thesis supervised by:
Prof. Giovanni Lapenta (CmPA KU - Leuven)
Dr. Jorge Amaya (CmPA - KU Leuven)

June 2018

Abstract

The subject of this project is the process of solar images used to forecast possible developments of Coronal Mass Ejections (CMEs). CMEs are energetic events originating from the Sun leading to disturbances of the magnetic field of the Earth, named geomagnetic storms. These storms can then produce several disastrous effects to satellites, electrical devices and grid networks.

This thesis aims to contribute in the development of a functional Artificial Intelligence (A.I.) model used to forecast CME events. These events are categorized in the LASCO and CACTUS catalogs. The reason why A.I. models are used is the lack of separately positioned satellites providing data, for the creation of a fully functioning physics based model.

To attain its main objective, this project has been divided in three major components.

First, an extensive data cleaning procedure was implemented. SDO data is downloaded, sorted and organized to be used as input for the A.I. model LASCO and CACTUS CME catalogs where enhanced to be used as output of the model.

Second, a Convolution Neural Network (CNN) was designed to process the enhanced data, showing promising results. In specific, an accuracy of 76.6% in the prediction of the CMEs as described in CACTUS catalog was obtained. A similar CNN model was applied to distinguish between halo and non-halo CMEs as shown in LASCO catalog. Its accuracy reached a peak of 83.5%. Feature selection of the model was carefully done from a wide range of observations, that included different wavelengths, resolution etc. For the training of the CNN and for the image processing, I implemented HPC techniques using the Flemish Supercomputer Centre (VSC).

Third, a new processing tool, written in python, was implemented to recover SDO solar images and generate "History Maps" that contain, in a 3D matrix, information about the Suns temporal and spatial evolution.

These three components show that a smart use of the available data from the SDO satellite in combination with artificial intelligence algorithms, like CNNs, can be used to forecast CME emergence on the solar corona.

General Public Summary

The constant flow of particles ejected from the Sun is at the origin of the Northern Lights, one of the most fascinating natural phenomena on Earth. However, it can also endanger our technological infrastructure and the health of astronauts and airline passengers.

In this project, a new modern approach, called machine learning is used. Machine learning is a computational technique used to find patterns and correlations among large data collections. The goal of this work is to find a connection between the activity of the lower layers of the solar atmosphere, and the observations of plasma ejections on the solar corona.

This approach has several steps, of which the most time-consuming is the creation, cleaning and organization of the data that the computer will use to train its code. This step is done by processing data from the SDO satellite of NASA/ESA.

The second step is the implementation of the machine learning model. The model was able to predict 76.6 % of the phenomena that originated from the Sun. Later, it was also used to identify potentially hazardous Earth-directed energetic events. The result was again promising, distinguishing between events that are directed to Earth and events that do not with a 83.5 % success rate.

Finally, a program was created and discussed. This tool creates new data that may be useful for similar forecasting tries. The tool follows the rotation of the Sun in order to catch energetic phenomena that form in different areas. The idea behind this approach is to assist the machine learning model in finding the energetic events with greater ease.

Acknowledgements and dedication

I would like to acknowledge the following people for the valuable help that they gave me while working on this thesis.

Prof. Giovanni Lapenta has always been extremely supportive not only during this thesis but throughout the last two years that I have been studying in KU Leuven. His guidance and the thoughts he shared with me were very important for completing this work.

Dr. Jorge Amaya has always been there to provide feedback, thoughts and motivation when it was needed. He was a great person to discuss and sometimes even argue with, but also a great instructor to listen and gain vital knowledge.

Furthermore, I would like to acknowledge the assistance that was provided by the frequent meetings that took place with the Machine Learning Project group of CmPA. Dr. Francesco Pucci and Dr. Diego Gonzalez were always there to share their thoughts on the projects and provide insightful comments.

A special thanks to my friends Adam Shamash and Gilles Depypere for supporting me by having extensive discussions regarding each other's Master thesis. We all worked in different parts of the same project and therefore our discussions were more than insightful.

This thesis is dedicated to my parents (Vasilis, Eleftheria) and my brother Omiros that supported me throughout my academic years. It is because of them that I have managed to reach this far and they are the reason I keep trying to accomplish greater goals every day.

TABLE OF CONTENTS

Energetic activity on the Sun influences space missions, satellites, airplanes, electrical/electronic devices and grid networks. As a result, there is a strong need for accurate predictions of solar activity to implement effective safety measurements and prevent a major economic loss. The ability to predict the effects of solar energetic events on the Earth's environment has always been of great interest to scientists of various disciplines. Several research groups work for many years in order to provide a satisfactory prediction model to avoid these disasters. This work is motivated from this effort and tries to contribute in the development of heliospheric prediction modeling.

Overview of the Project

In this project, an investigation of a solar energetic phenomenon called "Coronal Mass Ejection" (CME) is being conducted. The goal of this project is to work towards the creation of an artificial intelligence model capable of providing accurate results predicting when the phenomena of CME is going to happen and with what characteristics. The reason why A.I. models are used is the lack of separately positioned satellites providing data, for the creation of a fully functioning physics based model.

The work of this project can be separated into three components. The first is the data cleaning of SDO measurements and the enhancement of CME associated catalogs (LASCO/CACTUS). After this procedure, an implementation of the SDO data as input and of the catalogs as output is being done in a Machine learning model. This resulted in a fairly significant and promising result. The third and final component, the central development in this work, is the creation of a new Pre-processing tool. This tool uses a model of the differential rotation of the Sun and creates a new kind of solar data that is called "History Maps" (HM). This work closes with the hope that, research in the future will combine the obtained tool and implement it in a similar forecasting model to hopefully derive even more fascinating results.

This master thesis work is a significant part of the project that is now developed into the AIDA H2020 (`www.aida-space.eu`) project funded by the European Commission (EC).

Structure of Thesis

The first chapter is an introduction to solar physics and space weather, which consists of the physical **Problem** that is challenged. The second chapter introduces the **Tool** that is used in the present project, describing the basic concepts of machine learning and neural networks. Finally, the **Analysis** section, in the third chapter, presents all the computational work that was done for this project. At the end of the report, appendices with information regarding space physics, space weather and code implementation can be found.

The structure of this report follows the diagram shown in Fig. 1.

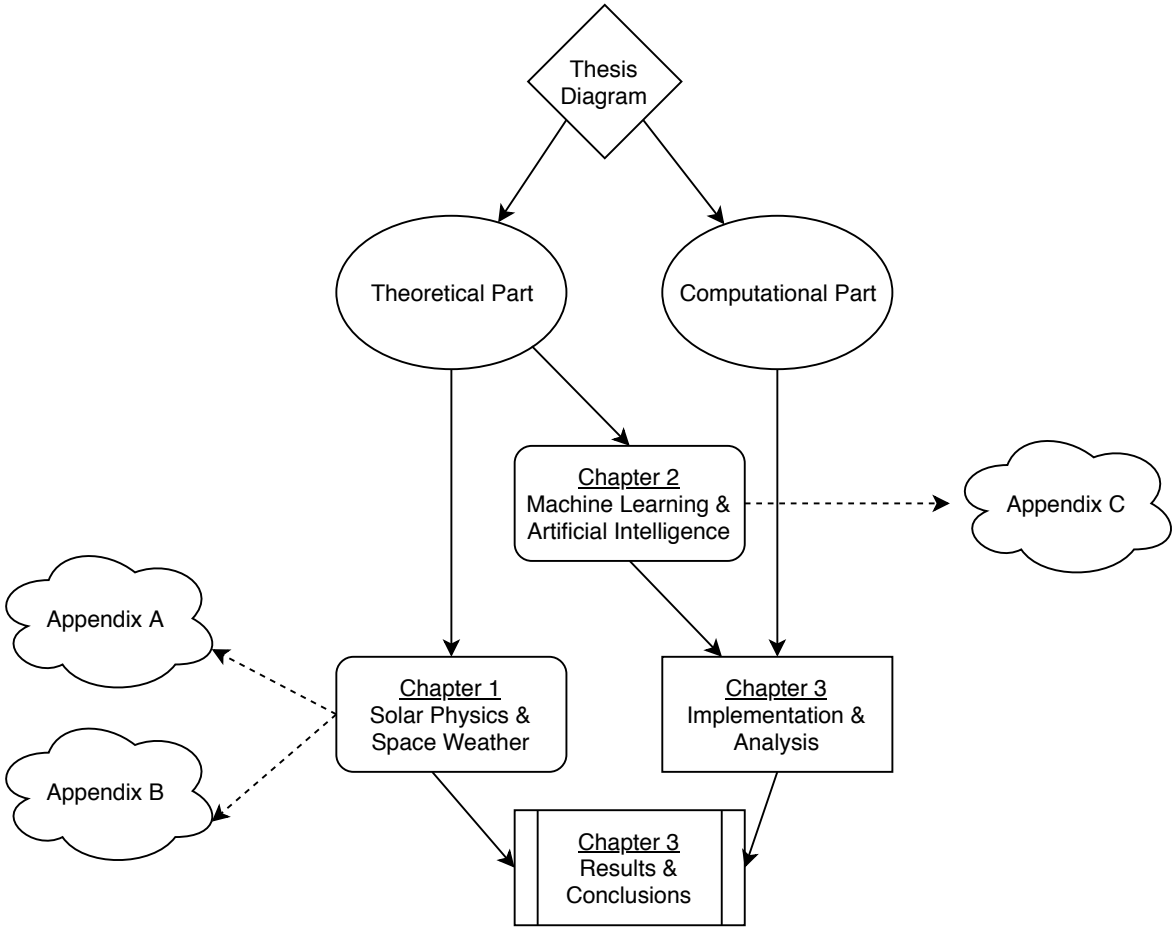


Figure 1: Schematic of Thesis' structure

Abbreviations - Acronyms

Acronyms	Definitions
ANN	Artificial Neural Network
AI	Artificial Intelligence
API	Application Programming Interface
CACTUS	Computer Aided CME Tracking
CME	Coronal Mass Ejection
CNN	Convolution Neural Network
CR	Cosmic Ray/Radiation
DL	Deep Learning
GCR	Galactic Cosmic Rays
GD	Gradient Descent
GIC	Geomagnetically Induced Current
GOES	Geostationary Operational Environmental Satellite
GS	Geomagnetic Storm
HPC	High Performance Computing
ICME	Interplanetary Coronal Mass Ejection
FWO	Fonds Wetenschappelijk Onderzoek
ISS	International Space Station
LASCO	Large Angle Spectrometric Chronograph
LEO	Low - Earth Orbit
LSTM	Long Short Term Memory
MC	Magnetic Cloud
MHD	Magnetohydrodynamics
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
NOAA	National Oceanic and Atmospheric Administration
RNN	Recurrent Neural Network
SDO	Solar Dynamics Observatory
SEB	Single Event Burnout
SEL	Single Event Latch up
SEP	Solar Energetic Particle
SEU	Single Event Upset
SLP	Single Layer Perceptron
SOHO	Solar and Heliospheric Observatory
SPE	Solar Proton Event
SREM	Standard Radiation Environment Monitor
STEREO	Solar Terrestrial Relations Observatory
SVM	Support Vector Machine
SW	Solar Wind
VSC	Vlaams Supercomputing Centrum
VSO	Virtual Solar Observatory

Table 1: Acronyms that are used throughout the Thesis

CHAPTER 1

SOLAR PHYSICS & SPACE WEATHER

Space weather is a branch of space and solar physics where multiple topics, including astrophysics, physics, computer science and aerospace engineering are employed. In this chapter, the reader will be introduced to the basic concepts of space weather and solar physics.

1.1 Introduction

The Sun (Fig.2) is the closest star to Earth and the main source of energy that sustains life on this planet. It is a main sequence star of G2V spectral type and it contains roughly 99.9 % of the solar system mass. The basic characteristics of the Sun are presented in Table 2.

Characteristic	Value
Mean distance to Earth	$150 \cdot 10^6$ [km]
Radius	$696 \cdot 10^3$ [km]
Mass	$2 \cdot 10^{33}$ [gr]
Luminosity	$3.82 \cdot 10^{33}$ [erg/s]
Composition	74% H, 25% He, 1% Others

Table 2: Basic characteristics of the Sun

1.2 Energetic Phenomena of the Sun

The Sun has a very complex structure. Energetic activity that manifest in the Photosphere or the Corona manifest also in other layers of the Sun. Sunspots and granulations can be correlated to Coronal Mass Ejections (CMEs) and solar flares. However, in this report I do not describe

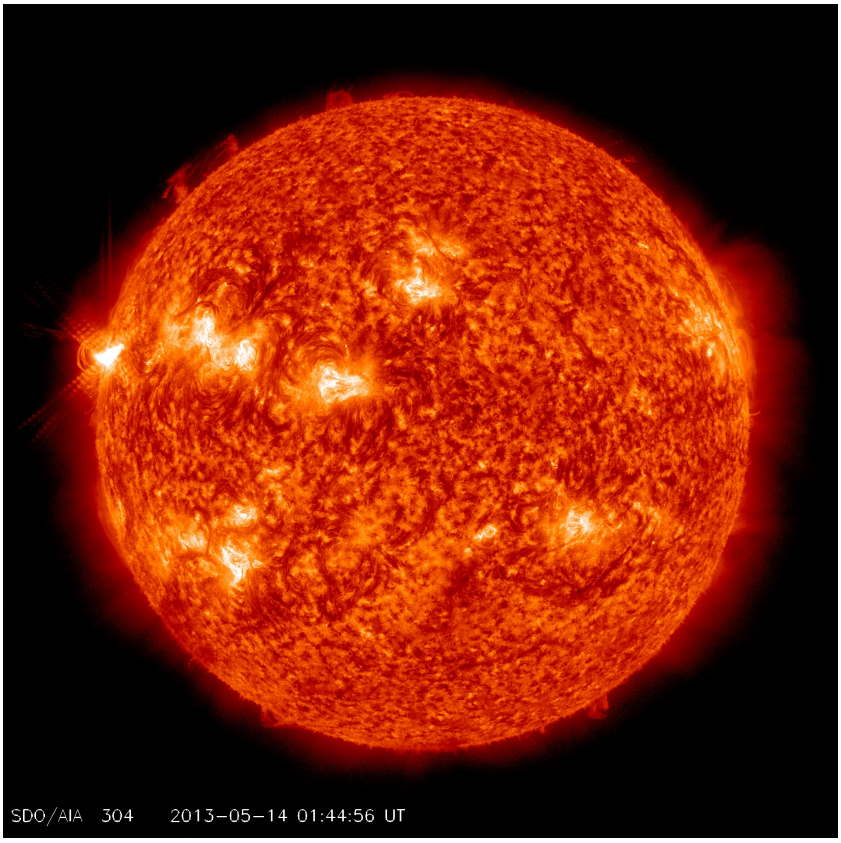


Figure 2: The Sun as seen by SDO/AIA at 15/5/2013 01:44 for wavelength λ = 304[Å]. Figure courtesy: NASA/SDO/jhelioviewer.

every phenomenon in detail. The reader can refer to Table 3 for the basic properties of these phenomena.

1.2.1 Solar Cycle

"Solar Cycle" (SC) is the periodic 11-year change in the Sun's activity (radiation, solar wind enhancement, magnetic field etc.) and appearance (sunspots, solar flares, CMEs etc.)

Solar maximum refers to the period in which the maximum number of sunspots is observed and **solar minimum** is when the sunspot number is at its minimum. A cycle last from one solar minimum to the next. Although the period is around 11 years, in the past there have been periods with different duration. This has been caused by unknown reasons that affected vastly the activity of the Sun (Fig.3).

Glasstone et al. (1997) [46] showed that energetic phenomena correlate well with the 11 year solar cycle. During the solar maximum, more activity is observed compared to the solar minimum. Several phenomena are associated with solar cycle periodicity.

Phenomenon	Characteristics	Description
Faculae	$t \approx 15$ [min] $B \approx 800$ [Gauss]	Inhomogeneity of the magnetic field of the Photosphere, active region at the photosphere of the Sun
Sunspots	$t \approx 1-100$ [days] $d \approx 10^{3-4}$ [km] $B \approx 10^{3}$ [Gauss]	Inhomogeneity of magnetic field, lower temperature, also visible in the Chromosphere
Prominence Filaments	$L \approx 10^{4-5}$ [km] $D \approx 10^{2-4}$ [km]	Filamentary formations of magnetic loops forming above a sunspot pair and extending up to the Corona
Coronal holes	Dark areas on the Corona	Open magnetic field lines, origins of of several space weather phenomena such as the fast solar wind.

Table 3: Short summary of phenomena that happen when the Sun is active. t shows duration, d diameter, L length, B magnetic Field and D depth

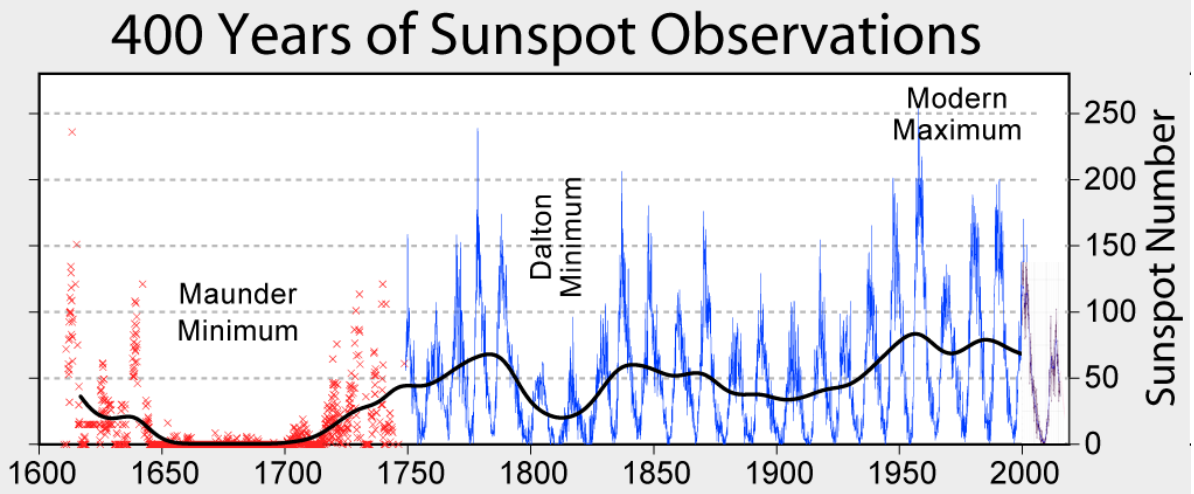


Figure 3: Sunspot number versus time. This figure summarizes sunspot number observations since 1749. Figure courtesy of Dr. Robert A. Rohde (`robert@globalwarmingart.com`).

In specific:

1. The number, the area and the position of Sunspots. (Fig.3 & Fig.4)
2. The number, the area, the position and the morphology of coronal holes.
3. The frequency and the energy output of energetic activity. (Solar flares, CMEs etc.)

At the beginning of each cycle, sunspots appear around mid-latitudes and slowly move closer to the equator of the Sun until the cycle ends and a new one begins. This pattern is called "The Butterfly Diagram" (Fig.4), first discovered by Richard Christopher Carrington[1] and improved by Gustav Spörer[2] [103].

[1] English Astronomer [1826 - 1875]
[2] German Astronomer [1822 - 1895]

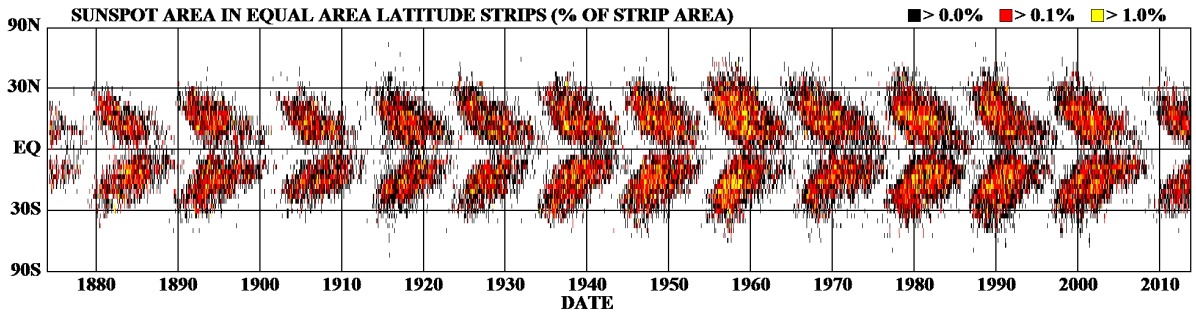


Figure 4: The Butterfly diagram shown in latitude against time plot. Figure courtesy of NASA Marshall Space Flight Center.

The SC is attributed to the rotation of the magnetic field of the Sun, which is explained by Babcock's theory (Appendix A) [6]. Scientists suspect that other phenomena such as tidal forces may have a significant impact on the existence and the variation of the solar cycle [113].

1.2.2 Solar wind

The Solar Wind (SW) is a stream of charged particles in plasma state, originating from the Sun's Corona. It consists mainly of electrons, protons and alpha particles [3]. The SW can be thought as the outflow of material from high pressure to low pressure. Its analogy with winds of Earth that blow from high to low pressure is the origin of its name [82]. The Interplanetary Magnetic Field (IMF) is being dragged along with the outflow, which follows the rotation of the sun [89].

There is a strong correlation between certain characteristics of the SW (speed, energy, composition) and solar cycle. The SW properties change during a solar cycle but also varies from one cycle to the next [79].

1.2.2.1 Fast Solar Wind

Fast wind is known to originate from Coronal holes. These regions show a darker intensity in X-ray images, due to plasma density being substantially lower ($< 50\%$) than the plasma in surrounding regions [128]. These are regions where magnetic field lines are open, and the surrounding plasma can rapidly swipe to regions of space with significantly lower pressure.

Basic characteristics of fast SW can be seen in Table 4. Fig.5 shows measurements describing its development during solar minimum and maximum.

1.2.2.2 Slow Solar Wind

Slow solar wind mainly originates from Sun's equator [5]. During solar minimum, slow SW mainly originates at latitudes $0° - 35°$, during solar maximum, slow SW is also coming from the

[3] Also called "doubly ionized helium nuclei" He^{+2}

poles [40]. The slow SW is much denser than the fast SW and holds a more complex structure with extensive regions of turbulence [31].

Basic characteristics of slow SW can be seen in Table 4. Fig.5 shows measurements describing its development during solar minimum and maximum.

Characteristic	Slow Wind	Fast Wind
Flow speed (v_p)	$250-400\cdot10^6$ [km/s]	$400-800\cdot10^6$ [km/s]
Proton density (n_p)	10.7 [cm^{-3}]	3.0 [cm^{-3}]
Proton temperature (T_p)	$3.4\cdot10^4$ [K]	$2.3\cdot10^5$ [K]
Total energy flux density	1.55 [$erg\cdot cd^{-2}\cdot s^{-1}$]	1.43 [$erg\cdot cd^{-2}\cdot s^{-1}$]
Main sources	Streamer belt	Coronal hole

Table 4: Basic characteristics of the SW during solar minimum as observed at 1 AU. Data obtained from Schwenn, (2001) [107].

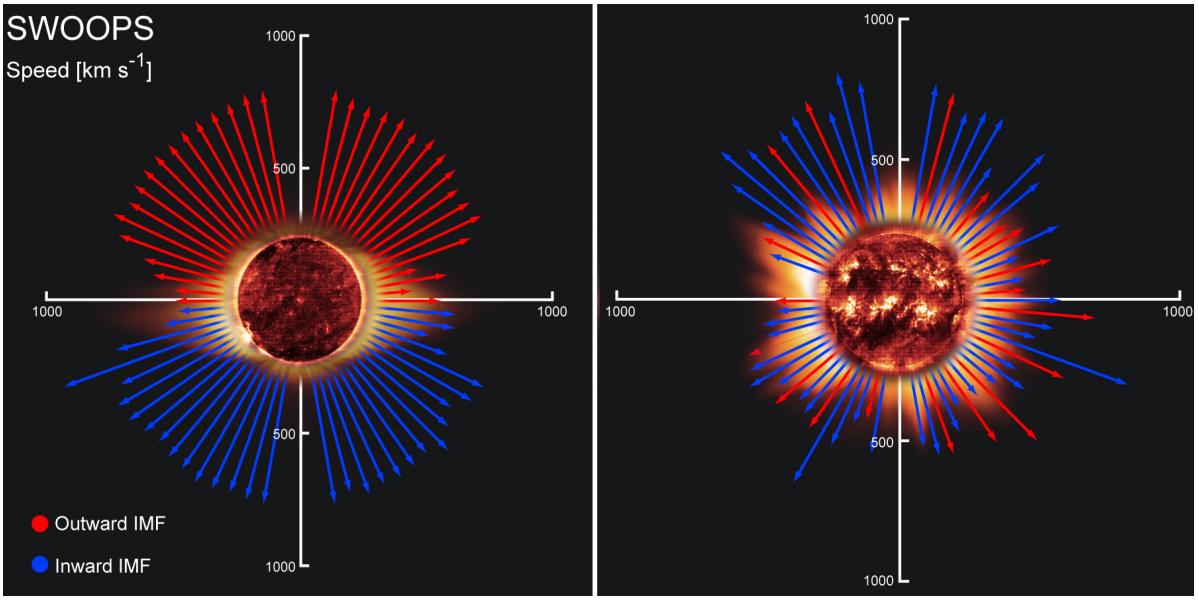


Figure 5: [**Left**]: During solar minimum fast SW originates from the poles filling 2/3 of the heliosphere and holding an average speed of 750 [km/s]. Whereas, Solar wind from the Sun's equatorial zone travels at 350 [km/s]. [**Right**]: During solar maximum, the solar wind is experiencing turbulence and irregularities, making slow and fast SW almost indistinguishable. Figure courtesy of ESA, Ulysses [4].

1.2.3 Solar flares & CMEs

The two most relevant events for space weather are solar flares and CMEs.

[4] http://sci.esa.int/ulysses/

1.2.3.1 Solar Flares

Solar flares are a rapid increase in Sun's brightness especially in the X-ray band. They are usually accompanied by CMEs, and their median energy output is roughly $\sim 10^{20}$ [J] [68].

Two types of classification models of solar flares can be seen in Table 5. GOES classification is the one that is currently used while the second is considered obsolete. An example of a solar flare, as captured by the Solar Dynamics Observatory (SDO) in multiple wavelengths, can be seen in Fig.6.

GOES Class	**Flux on (1-8Å)**	$H\alpha$ **Class**	$H\alpha$ **Surface**
X10	10^{-3} [Wm^{-2}]	4	24.7 [deg^2]
X	10^{-4} [Wm^{-2}]	3	12.4 [deg^2]
M	10^{-5} [Wm^{-2}]	2	5.1 [deg^2]
C	10^{-6} [Wm^{-2}]	1	2.0 [deg^2]
B	10^{-7} [Wm^{-2}]	S	<2.0 [deg^2]
A	$< 10^{-7}$ [Wm^{-2}]	S	<2.0 [deg^2]

Table 5: Solar flare classification based on GOES X-ray measurements and on $H\alpha$ surface.

Syrovatskii, (1972) [115] separated solar flares, from an energetic point of view, in three distinct phases:

- **Initial phase**. Before the impulsive release of energy, there is an accumulation of energy in the active region. An increase of the magnetic field and in temperature can be observed.
- **Explosive phase**. Lasts about 1 minute and can be seen throughout all wavelengths.
- **Recovery phase**. Takes roughly 30 minutes, temperature is decreasing and X-ray radiation is also gradually declining.

1.2.3.2 Coronal Mass Ejection (CME)

Coronal Mass Ejections were first observed back in 1971 by R.Tousey [55]. Dayeh (2015) defines CMEs as an extremely energetic solar phenomenon that can be described as extensive structure of magnetized plasma which propagates away from the Sun into space, driven by magnetic forces [32]. They originate from the Sun's active and filament regions [50]. An example of a CME observed by the Solar and Heliospheric Observatory (SOHO) is shown in Fig.7.

The basic characteristics of a standard CME are the following [57]:

- $M = 10^{11-12}$ [kg]
- $v = 400 - 1000$ [km/s] (Extremes: $20 - 3200$ [km/s])
- $F \leq 1$ / day (solar minimum), $F \approx 4 - 5$ / day (solar maximum)
- Transit time = 13 [h] - 86 [days]

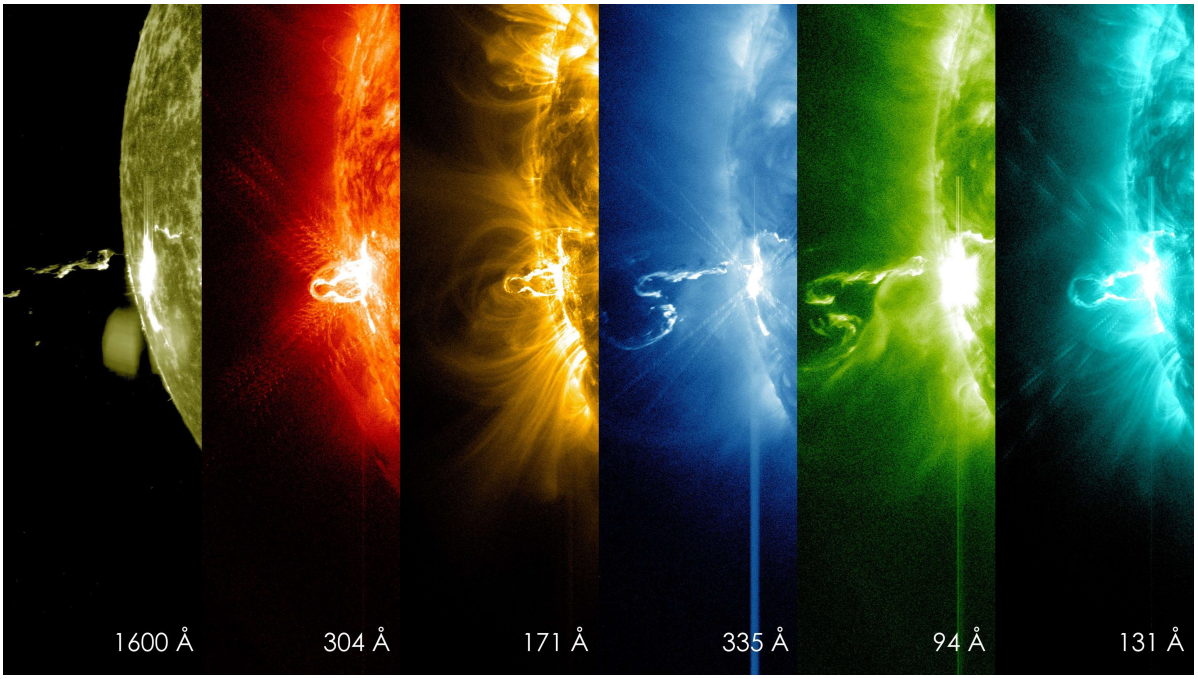


Figure 6: Image taken from SDO, showing an X-class flare in different wavelengths (2014/2/24 19:49). Figure courtesy: NASA/SDO [5].

When referring to CMEs that can reach the Earth and are detected through in-situ measurements, researches use the term Interplanetary CME (ICME). These events can be geoeffective by injecting energy into the inner magnetosphere environment of the planet [85]. The main difference between CMEs and ICMES is the location and the method they are observed. CMEs are detected directly, and are related to a specific solar eruption. Howard (2011) [57] states on the other hand, that ICMEs are observed when they have already achieved a substantial distance from the Sun.

Other important subcategories of CMEs are Magnetic Clouds (MC) and halo CMEs. Magnetic Clouds are defined as magnetic structures in which an enhanced strength and a smooth rotation of the magnetic field vector can be observed, while proton temperature is decreasing [19]. Halo CMEs, as shown in Fig.8, propagate in the direction of the Earth, following the Sun-Earth line. They are observed with coronagraphs and since they are moving towards the Earth, it is essential to study and detect them [75].

A list of the CME events can be found in the literature [20]. These list were originally compiled by visual inspection of each image from different coronagraphs, including the Large Angle Spectrometric Chronograph (LASCO)[7] [18]. More recently, there has been progress to autonomously detect CMEs in image sequence from LASCO. The result of this process can be found in the

[5] https://www.nasa.gov/content/first-moments-of-a-solar-flare-in-different-wavelengths-of-light
[6] http://sci.esa.int/soho/
[7] List available at https://cdaw.gsfc.nasa.gov/CME_list/

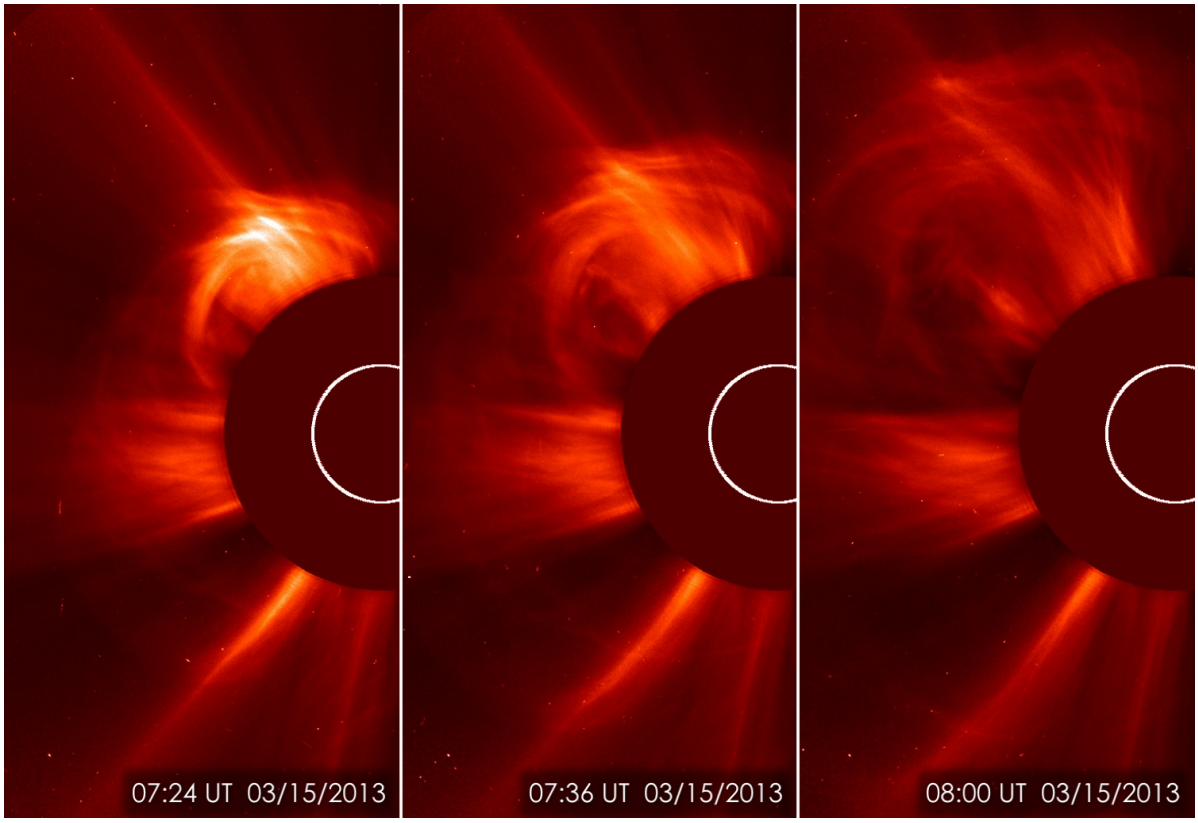


Figure 7: Images of the Sun showing a Coronal Mass Ejection (CME) on 15/3/2013. This type of image is called a coronagraph, where a disk is placed over the Sun to achieve better visualization of Sun's Corona. Figure Courtesy: ESA & NASA SOHO.

"Computer Aided CME Tracking" (CACTUS) [8] database, introduced by Robbrecht and Berghmans in 2014 [98].

The main characteristics of these catalogs are the starting time (t_0), the principal angle (θ), the angular width (D) and the velocity (v).

Traditionally, CMEs are categorized in two classes according to data from LASCO [4] :

- **Gradual CME**: not correlated with solar flares, $V \sim 400 - 600$ [km/s], developing acceleration towards the coronagraph.
- **Explosive CME**: correlated with solar flares, $V > 700$ [km/s], decelerating as they travel towards the coronagraph.

Due to flares and CMEs, energetic particles can achieve high velocities as they travel towards the Earth. But how much time do these particles take to arrive to Earth? One can calculate the

[8] http://sidc.oma.be/cactus/

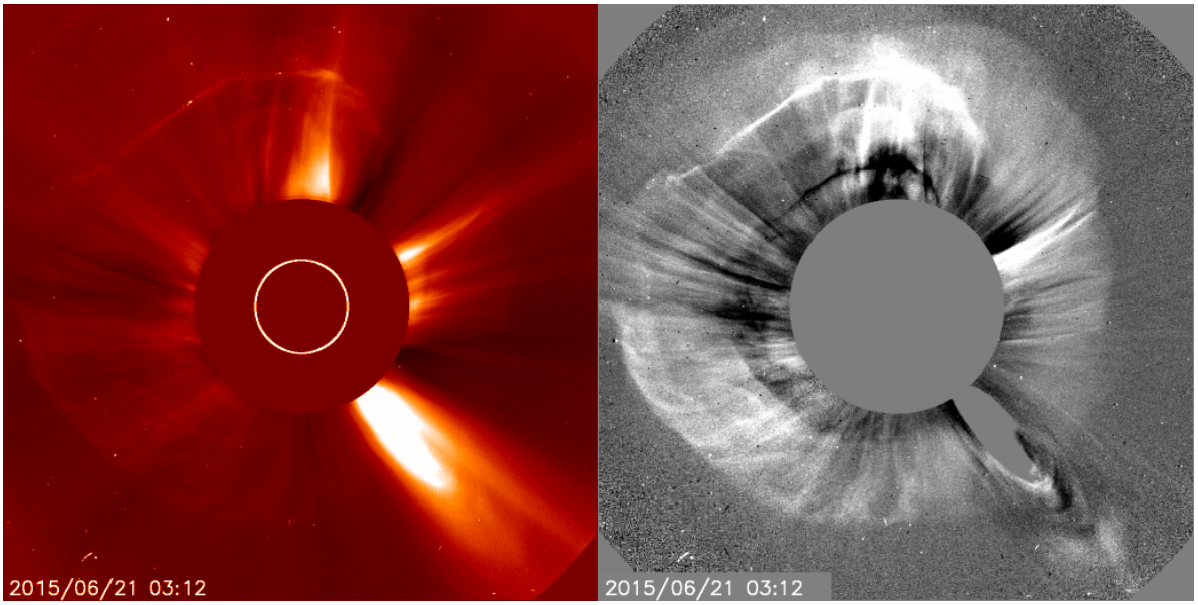


Figure 8: Images of the Sun showing a halo CME on 20/6/2015. This can be seen more clearly in the right-hand image, called a difference image, which is created by subtracting two consecutive frames. Figure Courtesy: ESA & NASA SOHO [6].

time photons take to propagate to Earth as following:

$$t = \frac{D}{u} \tag{1.1}$$

Where, $D = 1\,[AU] \sim 1.5 \times 10^{11}\,[m]$ and $u = 3 \times 10^{8}\,[m/s]$. As a result, the time it takes for a photon is:

$$t_\gamma \approx 500\,[sec]$$

For ions (protons and electrons), the following can be obtained:

$$E = \gamma mc^2 = \frac{mc^2}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{1.2}$$

For an electron with energy of $10\,[KeV]$ and a proton of $10\,[MeV]$ one can find that:

$$t_{p^+} \approx 42\,[min]$$
$$t_{e^-} \approx 57\,[min]$$

The mean propagation time for a CME depends on its mean velocity, given, $v_{\mathrm{CME}} = 1000$ [km/s] is:

$$t_{CME} \approx 2\,[days]$$

This value is not absolute, since the actual velocity of a CME varies a lot from one event to another (13 [h] - 86 [days]) [57].

1.2.3.3 Relationship between CME and Solar Flares

Solar flares and CMEs are well correlated with each other; however, there is no causality between them. A statistical analysis by Youssef et al. (2012), showed that from 1996 to 2010 there have been 12.433 CMEs and 22.688 solar flares as seen by SOHO. Their analysis concluded that, there are cases where solar flares occur and there is no associated CME [127]. On the other hand, CMEs without a flare is also a possibility and has been studied by many researchers like MacQueen and Fisher in 1983 [77].

Both solar flares and CMEs are the result of extensive magnetic activity in the Sun. Solar flares are local events that happen in the surface of the sun, whereas, CMEs are much more extensive ejection that occur in Sun's corona. Many models of flare-associated CMEs are based on magnetic reconnection models. An introduction to magnetic reconnection is presented in Appendix A. In reconnection models, as described by Yashiro et al. (2008), a flare develops just below an erupting filament which in the end becomes the center of a Coronal Mass Ejection [126]. An example of such a flare associated CME is shown on Fig.9.

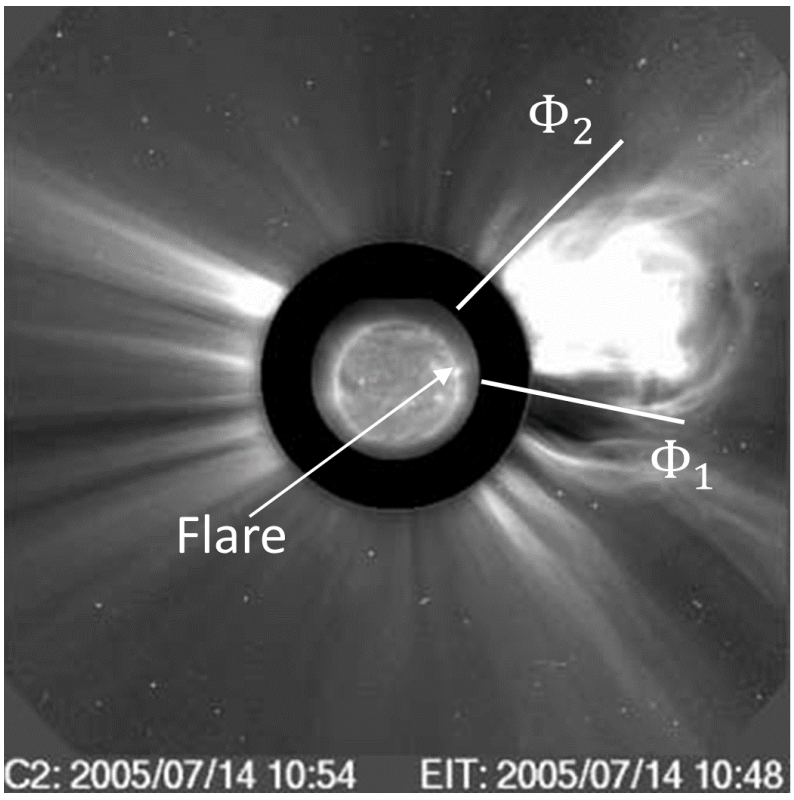


Figure 9: A Flare associated CME observed by SOHO, LASCO. $\Phi_{1,2}$ indicates the positional angles [9]of the edges of the main CME. An arrow is also pointing to the location of the associated flare. Figure Courtesy: ESA & NASA SOHO & Yashiro et al. (2008) [126].

[9]Position angle [^{o}] is measured counterclockwise starting from north

In Appendix A, the reader can find a list with some models that explain how solar flares and CMEs may be created.

1.2.3.4 Solar Energetic Particles (SEPs)

Reames (2013) [96], describes Solar Energetic Particles (SEPs) as energetic ions and electrons found in the SW that have energies from the suprathermal regime (~ 10 [KeV]) up to tens of GeV.

Kallernode (2003) [63] proposed a simple model for classifying SEP events. He showed that SEPs can be either accelerated at the solar flare sites via the release of magnetic energy (flare-associated impulsive events) or by the Inter Planetary (IP)-driven shock waves in the corona and in the IP medium via diffusive acceleration (CME-associated gradual events). This offers another classification of solar flares depending on how they act on SEPS (Table 6). If another separation based on the abundance of $^3He/^4He$ is included, a more complex and accurate model which consists of 3 SEP categories can be derived as described in Table 7.

Flare Characteristic	**Impulsive**	**Gradual**
Duration in soft X-rays	< 1[h]	> 1[h]
Height in corona	$\leq 10^4$[km]	$\sim 5 \times 10^4$[km]
Volume	$10^{26} - 10^{27}$[cm^3]	$10^{28} - 10^{29}$[cm^3]
Energy density	High	low
Coronal Mass Ejection	Rare	Always

Table 6: Classes of solar flares based on how they affect SEPS [63].

SEP Characteristic	**^{3}He-rich**	**Impulsive**	**Gradual**
Particles	Electron rich	Electron rich	Proton rich
$^3He/^4He$	~ 1	~ 0.2	~ 0.0005
H/He	~ 10	~ 10	~ 100
Duration	Hours	Hours-days	Days
CMEs	No	Sometimes	Yes
Event rate	$\sim 1000/y$	$> 10/y$	$\sim 10/y$

Table 7: Classes of SEPs and their characteristics [63].

A more recent and extensive work in 2015 by Kim et al. [65], categorized 42 SPEs[10] that were observed in 1997 - 2012, in four different groups based on possible acceleration mechanisms that resulted in the observed characteristics.

[10] Most of the times, in literature, the terms SPE (Solar Particle Event) and SEP (Solar Energetic Particle) are identical

1.3 Geomagnetic storms

All the previously mentioned events can manifest on the Earth's plasma environment. We call these manifestations "geomagnetic storms".

The plasma environment around the Earth affected by its magnetic field is called "The Magnetosphere" and the term was first mentioned by T. Gold in 1959 [74]. Its exact shape and characteristic is heavily influenced by the solar wind. The term Geomagnetic Storm (GS) refers to the disturbance of the Earth's magnetosphere due to its interaction with the solar wind [80]. Eruptions of plasma carrying the magnetic field from the Sun (CMEs) are at the origin of these storms. In a GS the magnetic field's disturbance is caused by an enhancement of the ring current caused by magnetic reconnection (More information can be found in Appendix A) at various places of the magnetosphere such as the magnetopause and the magnetotail. The structure of the magnetosphere is shown in Fig.10

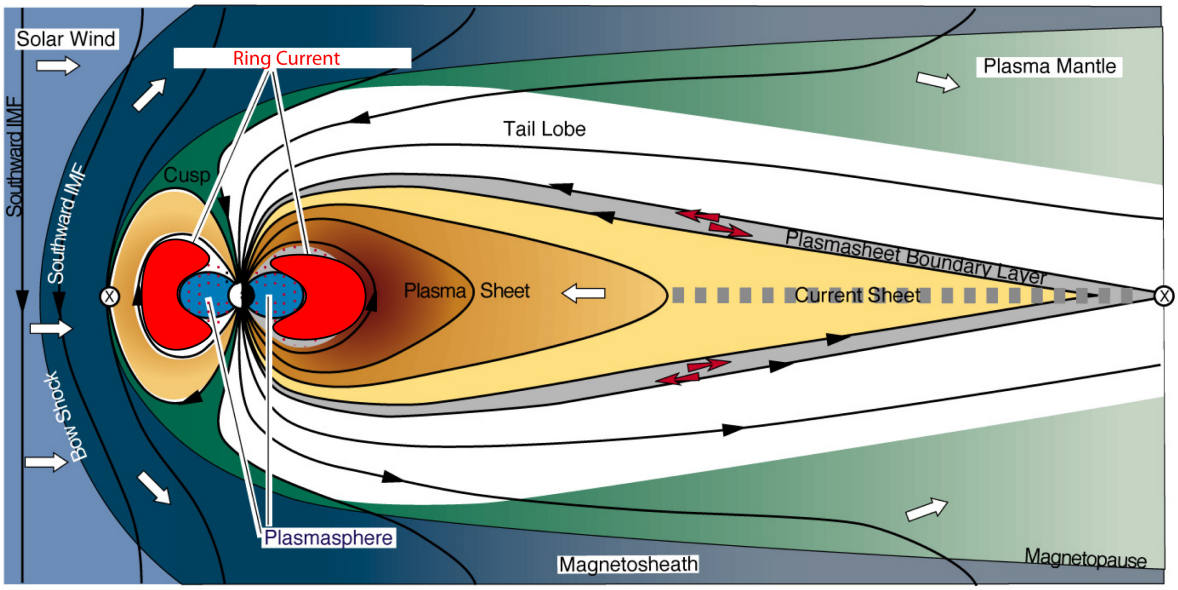


Figure 10: Interaction of solar wind with Earth's magnetosphere. Indications are made for the occurrence of the ring current and other areas of the Magnetosphere. Figure Courtesy & more information regarding the figure: Chih-Ping Wang [11].

Magnetic reconnection is a process where the topology of the magnetic field changes due to non-ideal (in a Magneto-Hydro Dynamical (MHD) sense) diffusive interactions in the plasma [101]. An important consequence of this magnetic phenomena is the fact that solar particles can enter the Earth's inner magnetosphere. As described by Gonzalez et al. (1994), during magnetic reconnection particles transform magnetic energy into kinetic energy [47].

[11] http://people.atmos.ucla.edu/cat/

Since the SW particles are injected into the magnetosphere, the majority are trapped by the planet's magnetic field. The magnetic field lines are closing on the poles, so particles make a "bounce" motion towards the equator. This phenomenon is occurring due to the conservation of the magnetic moment. More information on the motion that is called "magnetic mirror" can be found in the Appendix. A (Fig.11). Daglis et al. (1999), [29] showed that due to their charge and drift forces caused by their gyroradius, electrons will travel to the east and protons to the west. This phenomenon results in a ring current that generates a magnetic field opposing the Earth's and therefore, effectively weakening it.

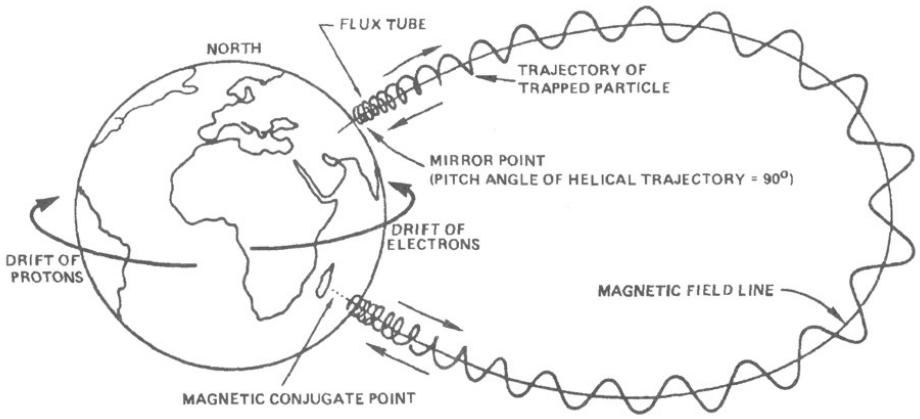


Figure 11: Trapped energetic particles and their movement inside the magnetosphere of the Earth. Their drift motion and the bounce that occurs in the mirror point can be visualized. Figure Courtesy: Adolph Jursa et al. [62].

The difference between the perturbed and unperturbed magnetic field is measured using the Dst index. The Dst is zero when the magnetic field of the Earth is undisturbed. The disturbance produces a negative value of the Dst, showing the "weakening" of the magnetic field. The value of Dst along with its pattern (sharpness, duration, etc.) provide a profile for the geomagnetic storm [28].

For the categorization and the research of geomagnetic storms, there are many additional indexes used, such as, the PC (Polar Cap) index and the indexes of k and kp, both openly available through NOAA websites[12].

Geomagnetic storms are usually characterized by their intensity and the phases that they go through. Table 8 shows the standard classification based on the storm's intensity. An example of a storm profile and its classification as shown with the Dst index can be seen in Fig.12.

We can roughly divide the storm into three phases: initial, main and recovery [48]:

[12]National Oceanic and Atmospheric Administration, `https://www.swpc.noaa.gov/`

Storm strength	Dst [nT]
Quiet	>-50
Moderate	[-50,-100]
Intense	[-100,-250]
Super-storm	<-200

Table 8: Classification of different geomagnetic storms according to the Dst index as proposed by Gonazelez et al. (1999) [48].

- The **initial** phase lasts in the order of minutes and is not always present in a storm. It consists of a sudden increase of Dst by roughly $20-50$ $[nT]$.
- The **main** phase is defined as the decrease of Dst. It depends on the intensity of the storm and can generally last between 1 and 8 hours.
- The **last** phase is the gradual recovery of the Dst back to zero value, and can take up to a week.

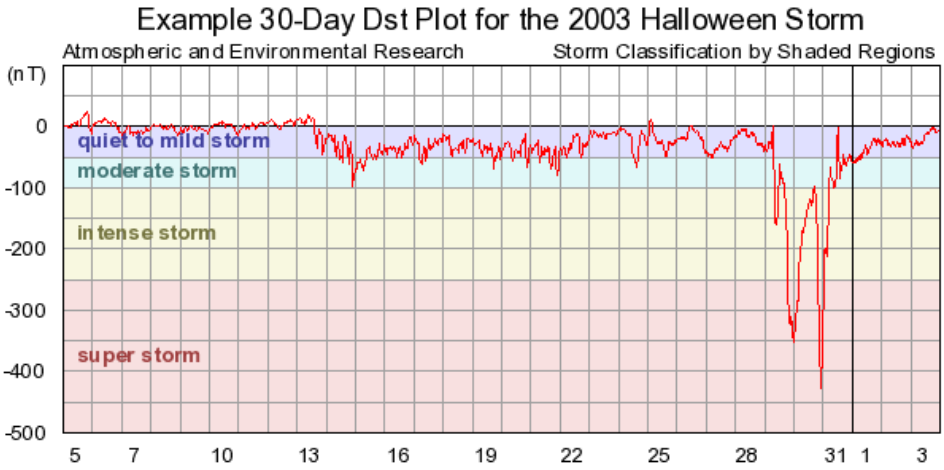


Figure 12: An example of geomagnetic storm characterization and classification using the Dst index. This storm in particular is known as "The 2003 Halloween super storm". Figure Courtesy: AER (Atmospheric and Environmental Research) [13].

A superposed epoch analysis by Hutchinson et al. (2011) [58] using the SYM-H index[14] showed clearly the general phases and the characteristics of the storms at different strengths. These phases can be seen in Fig.13.

There is no correlation between the initial phase duration and the storm size. The duration of the main phase of the storm decreases with the increase of the storm size. Intense storms are associated with enhancements of the solar wind. Hutchinson et al. (2011), [58] proposed that

[13] `http://www.aer.com/science-research/space/space-weather/space-weather-index`

[14] The Symmetric (SYM) Disturbance Index - 1min resolution, is essentially the same as the Dst, but it uses 1 minute values from different sets of stations and a different coordinate system.

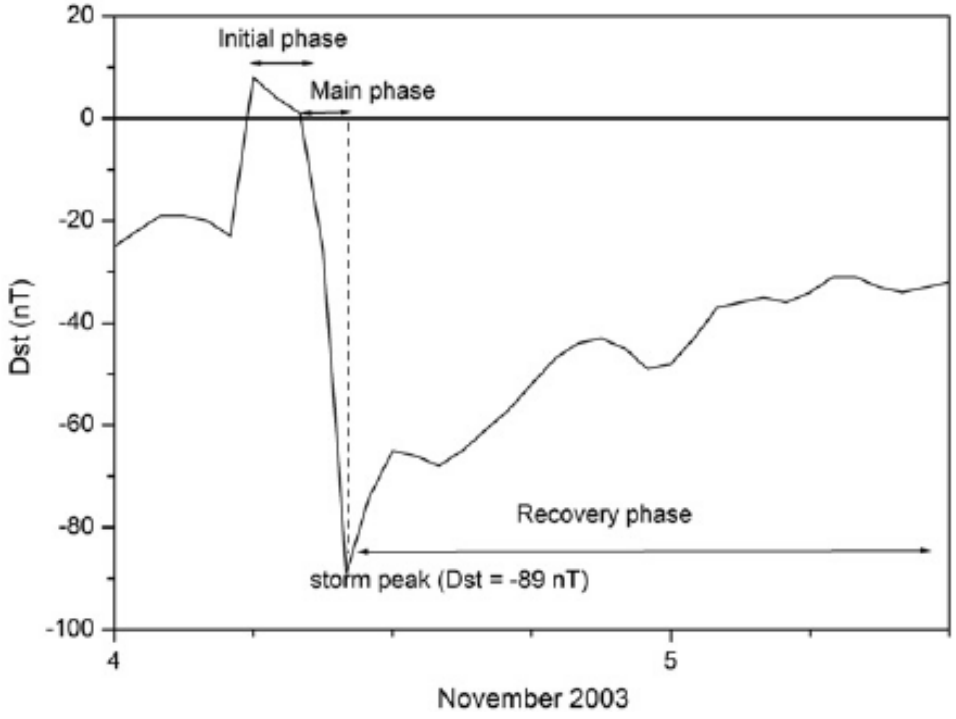


Figure 13: An example of the phases of a geomagnetic storm as shown by the Dst measurement. Figure courtesy of Echer et al. (2011) [36].

the main phase duration increases with the storm size until the corresponding recovery rate maximizes; after this point, solar wind enhancements are so intense that they force the main phase to its minimum at a faster rate.

1.4 Effects of Space Weather

Probably the most spectacular effect of space weather are the auroras at the North and the South pole of the Earth. They are caused by the injection of high energetic particles at the poles, following the magnetic field lines. However, space weather related phenomena can be dangerous to humans and technology in space and on Earth [7].

Space weather effects, include catastrophic problems in infrastructures (e.g. Electric grid network), destruction of satellite equipment and an increase of the cost of aviation in certain areas due to an increase of radiation from particles of the SW [17].

More information on these topics, explaining in details the effect of space weather phenomena can be found in Appendix B.

CHAPTER 2

MACHINE LEARNING & ARTIFICIAL INTELLIGENCE

Artificial Intelligence is a computer science technique where, a computer system is capable of performing tasks that were traditionally believed to require human intelligence. Examples of such tasks are visual perception, speech recognition, decision-making, and translation between languages. Under the umbrella term of A.I. there are many fields that are going to be described in this chapter, such as Machine Learning (ML), Neural Networks (NN) and Deep Learning (DL).

2.1 Introduction

A visualization of the differences between the previously mentioned terms is shown in Fig.14. Artificial Intelligence is a broad field that includes philosophical as well as computational aspects, describing how a computer program may sense, act and adapt to various circumstances similarly to Humans. On the other hand, machine learning refers to the computer science part of A.I. in which by utilizing statistics methods, a computer algorithm can improve its performance using historical data without being specifically programmed [105]. Deep learning describes a specific ML technique. This technique implements Neural Networks that use enormous amount of data trying to imitate the way humans acquire knowledge.

2.1.1 Classification & Regression

When machine learning is used for prediction tasks, problems are separated into two broad categories. *Classification* (qualitative) and *Regression* (quantitative) problems.

In **classification** problems, a prediction of discrete number of values is attempted. The out-

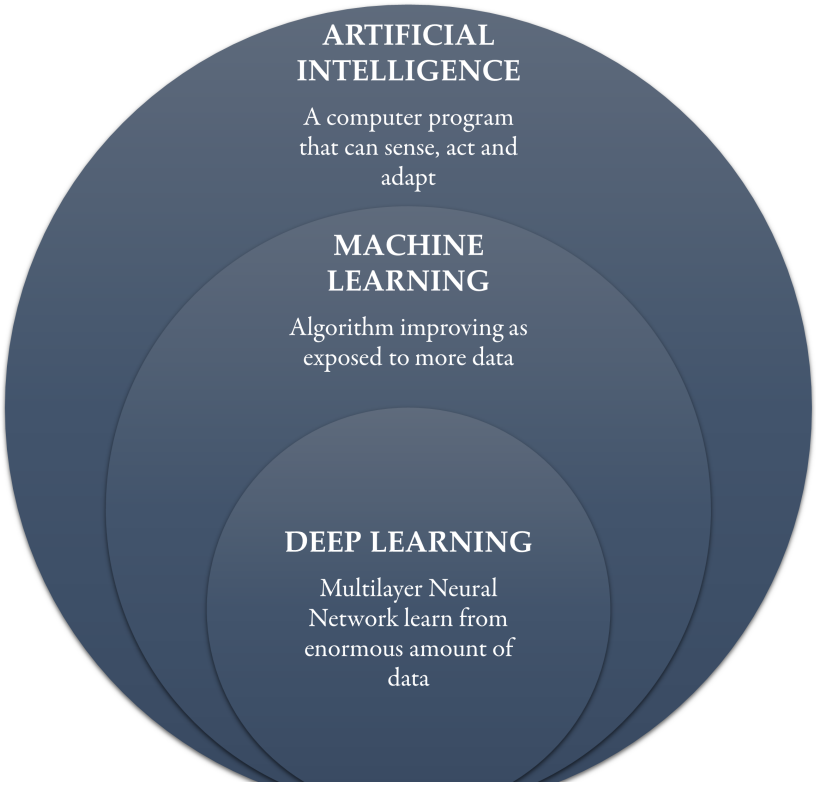


Figure 14: Visualization of A.I. and its main subcategories.

put variable takes class labels (e.g. $a \in [0,1] \subset \mathcal{N}$). When there are two classes, we call the classification "binary". When there are more we describe it as "multiclass" classification.

In **regression** problems, a prediction of continuous values is attempted. Therefore, the output variable takes values of non-infinitesimal gap (e.g. $a \in \mathcal{R}$).

The differences between the two types of problems can be seen in Fig.15. In this figure there are two examples.

In Fig.15 [Left], a binary classification problem in which the model finds the boundary between "Healthy" and "Unhealthy" genes is presented. Fig.15 [Right] shows a regression problem where the output is a continuous value. In this case the amount of survival years for the patient that has a specific gene.

2.2 Neural Networks & Deep Learning

2.2.1 Neurons & Activation Functions

Historically, research in biological neurons stimulated the creation and the development of Artificial Neural Networks (ANN).

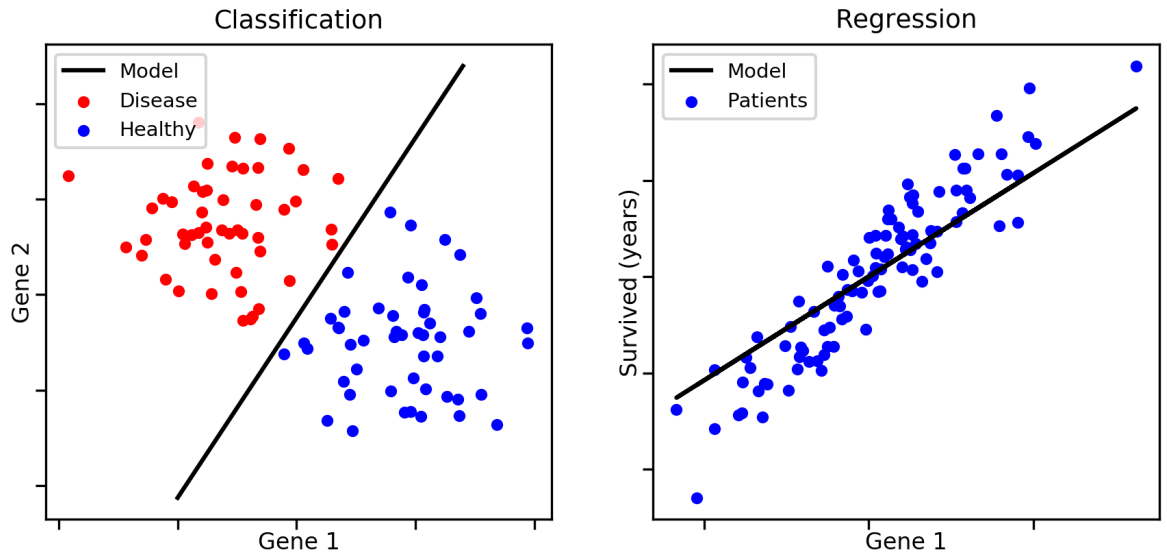


Figure 15: [**Left**]: Example of a classification problem as represented in a graph [**Right**]: Example of a regression problem as represented in a graph. Figure courtesy: Alexandre Drouin [1].

Assuming a very simplistic model of the human brain, different neurons can be activated depending on a signal received from other neurons. Through the learning process the activation threshold and the synaptic weights of the neurons change. As a result, new connections are made among the brain's neuron, and we are capable of learning new information.

According to this idea, Artificial Neural Networks (ANNs) develop functions that work in a similar way, activating the neurons of a network depending on the input signal. The similarities between the concept of biological and artificial neurons can be seen in Fig.16. The initial idea of the perceptron (artificial neuron) goes back to 1943, when it was introduced by Warren McCulloch[2] and Walter Pitts [3] [78].

2.2.2 Single Layer Perceptron (SLP)

A Single Layer Perceptron (SLP) with one node is shown in Fig.17. This structure is called a Rosenblatt's perceptron and was first introduced in 1957 by Frank Rosenblatt [4] [100]. A SLP contains a node with a value and is connected to n inputs with weights $w_i \in \{w_1, ..., w_n\}$. The output value of the node of the SLP is:

$$\alpha = f(z)$$

[1] `https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/`
[2] American neurophysiologist (1898 - 1969)
[3] American logician (1923 - 1969)
[4] American psychologist (1928 - 1971)

Where,

$$z = \sum_{k=1}^{n} w_k x_k + b \tag{2.1}$$

Where, b is a constant named "bias", f is the activation function, x is the input data and α is the output.

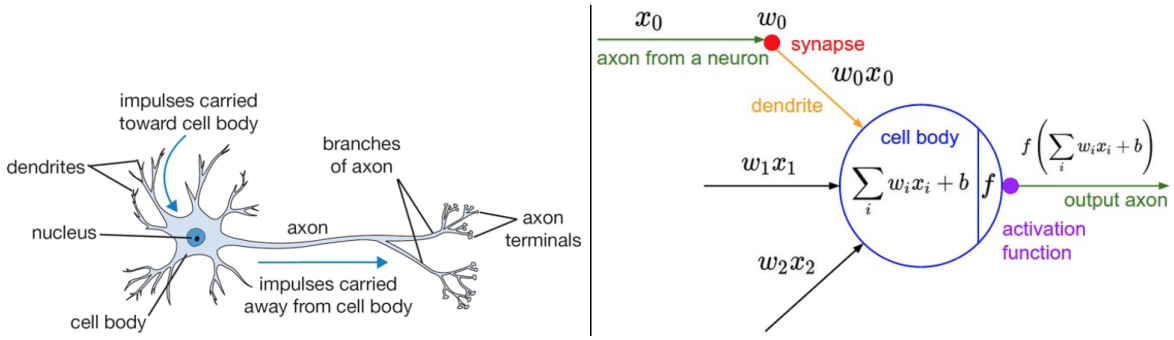


Figure 16: [**Left**]: Representation of a biological neuron. [**Right**]: Mathematical model of an artificial neuron. Figure Courtesy: Isaac Changhau [5].

2.2.3 Weights, Biases and Activation Functions

Weights and biases are the learnable parameters of the SLP. The values of these parameters are randomly initialized. The activation function produces the output of a specific node given an input [49].

Weights: are numbers showing how active the synapse between 2 neurons is. If $w_i \approx 0$ it would mean that this input would not affect the output. On the other hand, a $w_i \gg 1$ would mean that the specific input is dominating and is almost fully responsible for the resulted output.

Biases: to understand biases, it is useful to think of the simple yet vital definition of a straight line.

$$y = wx + b \tag{2.2}$$

In the equation of a line, b allows to shift the crossing point of the line over the coordinate axis. In a similar sense, bias in neural networks allows to shift the activation function and therefore increasing the region of possible outputs that originate from one input.

Activation functions: Returning to the analogy of biological neurons, an activation function is ideally acting like a switch, to indicate whether a neuron is active or not. In the case of a sigmoid

[5] `https://isaacchanghau.github.io/post/activation_functions/`

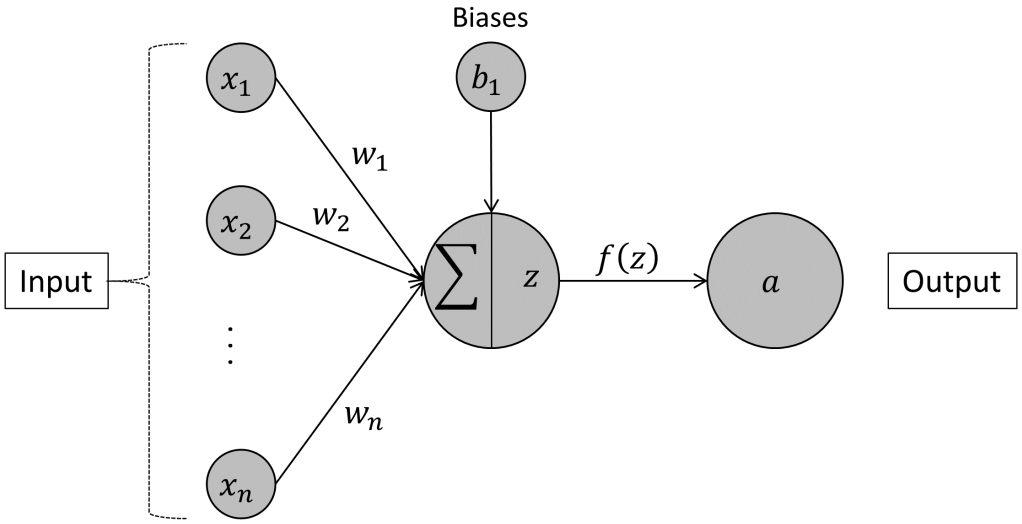


Figure 17: A Single Layer Perceptron (SLP). In this example, there are n inputs that are summed with their weights w_i, and a bias term b. The result of the sum named z is put through an activation function f to produce an output value a.

function (Fig.18 [Left]):

$$f(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}$$

The output may be between 0 and 1. Therefore, it can be easily seen that this function can be used in classifications problems since it provides a "yes or no" answer. For a regression problem a continuous value derived from the input must be found. In that case, a linear activation function (Fig.18 [Right]) is preferred, providing a continuous value according to the given input.

Some of the most advanced activation functions that provide the best results so far are shown in Table 9 and visualized in Fig.19.

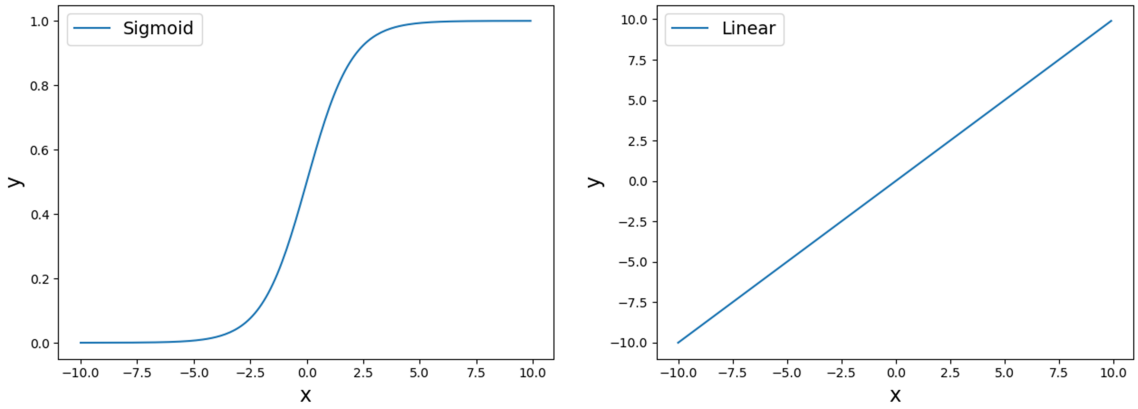


Figure 18: Basic activation functions used in NNs. [**Left**]: A sigmoid function - classification problems and [**Right**]: A linear function - regression problems

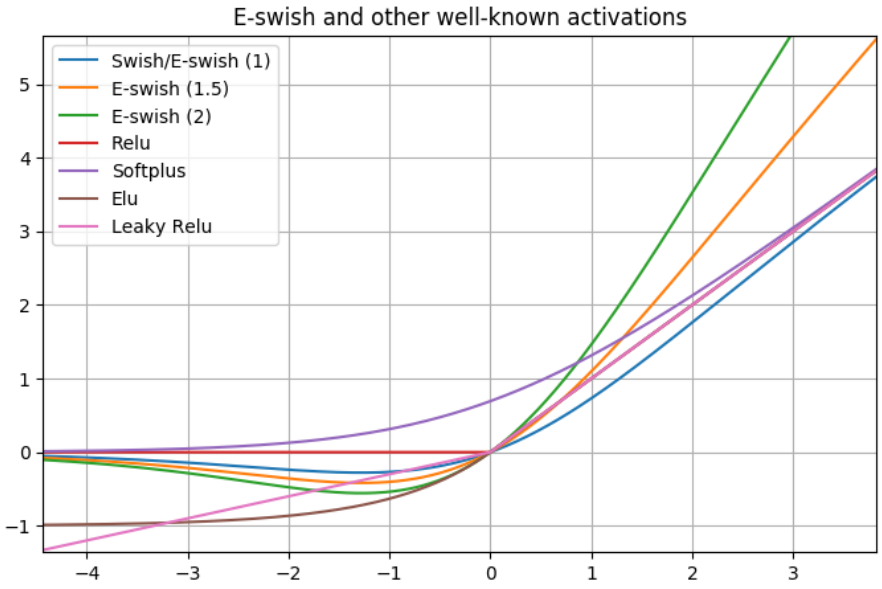


Figure 19: E-Swish with $\beta = 1.5$ and $\beta = 2$ compared with other commonly used activation functions. Figure Courtesy: Eric Alcaide, (2018) [3].

Name	Equation	Description
Rectified linear unit (ReLU)	$f(x) = x^+ = \max(0, x)$	Introduced in 2000 by Hahnloser and Seung, based on a direct link between biological neurons and their mathematical modeling inspired its creation [51]. After its first demonstration in 2011, it became the standard method for deep learning due to its superior results.
Leaky ReLU	$f(x) = \begin{cases} x & \text{if x > 0} \\ \alpha x & \text{Otherwise} \end{cases}$	Compared to ReLu, Leaky ReLu can have a small, positive value ($\alpha \in [0.001 - 1]$) when the node is inactive.
Parametric ReLU (PReLU)	$f(x) = \begin{cases} x & \text{if x > 0} \\ \alpha x & \text{Otherwise} \end{cases}$	Parametric ReLU (PReLU) is an advanced form of Leaky ReLU in which the small positive value α is actually a learnable parameter.
Exponential Linear Units (ELUs)	$f(x) = \begin{cases} x & \text{if x} \geq 0 \\ \alpha(e^x - 1) & \text{Otherwise} \end{cases}$	A relatively new (2016) activation function, which decreases the computational time of learning in deep neural networks and achieved amazing classification accuracies. In research, ELUs appear to have significantly better performance than ReLUs and LReLUs on networks with more than 5 layers [23].
Swish	$f(x) = x \cdot \text{sigmoid}(\beta x)$	Swish (2017) is basically a modification of the sigmoid function. β is a constant or trainable parameter [94]. If β = 1 Swish is equivalent to the Sigmoid-weighted Linear Unit that was proposed for reinforcement learning [38].
E-Swish	$f(x) = x\beta \cdot \text{sigmoid}(x)$	A generalization of Swish (2018). Experimentally it has been shown that a great choice of the parameter β might be $1 \leq \beta \leq 2$ [3].

Table 9: Advanced activation functions & their characteristics

2.2.4 Multi layer Perceptron (MLP)

It is possible to combine SLPs by stacking them in a single layer and by using the output of one perceptron as an input of the next. The resulting structure is named Multilayer Perceptron (MLP). All the layers between the input and the output are called "hidden" layers.

To discuss Neural Networks a simple case of an MLP is going to be introduced as shown in Fig.20. More complex models follow the same rules but they tend to be too complex and limit any visualization capability. In the present case, the input a_i along with the weights w_{ij} and the bias b are connecting the input layer with the hidden. This connection forms an activation signal for the hidden layer:

$$z_j = b_j + \sum_{i \in I} a_i w_{ij} \tag{2.3}$$

This signal is then passed through the activation function of the hidden layer g_j leaving the hidden layer as a new signal a_j. Similarly, a_j is multiplied with the weights that connect the hidden layer with the output, w_{jk} and finally, a bias term b_k is added. Once again, the signal that arrives passes through the activation function of the output layer, g_k and a final signal a_k is created.

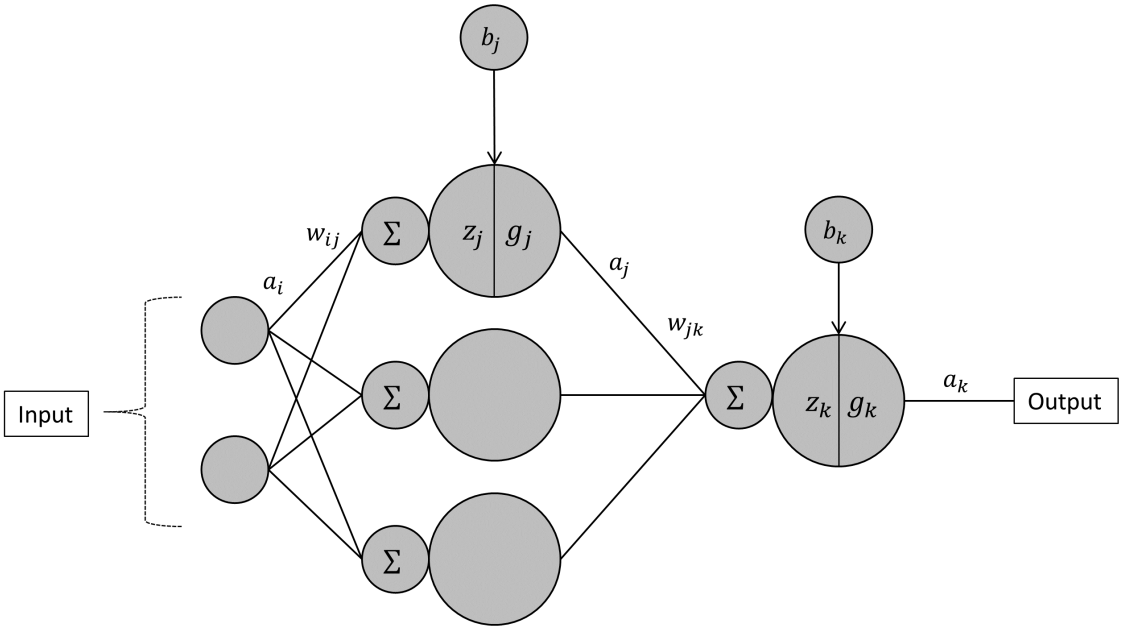


Figure 20: A Multi Layer Perceptron. In this example there is just one hidden layer and a single output.

2.2.5 Loss Function

The structure of the neural network and the production of output based on the given input has already been introduced. In order to measure the accuracy of the model, a function called cost function (Also called "Loss function" and "objective function") E is used. The loss function takes the average difference of all the results of the hypothesis function and the desired output t_k that ideally should be derived. For the example shown in Fig.20, the desired output is named t_k and the difference between the two can be calculated as [104]:

$$E = \frac{1}{2} \sum_{k \in K} (a_k - t_k)^2 \tag{2.4}$$

The goal of the model is to minimize the cost function E. This is accomplished using a training procedure. During this procedure the model is adjusted using historical examples provided by the user in order to derive the optimal parameters. The adjustment is being done by using a set of values for the input and the desired output that we call "data".

2.2.6 Data, Input & Output

2.2.6.1 Normalization

All the data that are used in NN models, should undergo normalization (Feature scaling) in order for the result to be optimal. There are several reasons why this process is essential for a machine learning code to work:

1. Different inputs might be in different scales.
2. Normalized data are affected more effectively by activation functions.
3. Ioffe and Szegedy in 2015 [59] showed that the training process, which is developed in the next section, works much faster when normalization is applied.

Regarding the different scaling, if the input has different range of values, the determination of the error will be biased towards the quantities that have the wider range and therefore, can reach higher absolute values. This phenomenon can be visualized in the data provided in Table 10. In that case, two different physical parameters measured by ACE spacecraft are a different scale. Without a normalization, there will be a bias on the neural network towards temperature values.

Normalization of data is vital because by definition the activation functions are more "active" on normalized data. Normalized data are centered around zero and so are the functions that are being used. A visualization of that phenomenon is shown in Fig.21.

Temperature $[K]$	**Density** $[N/cm^3]$
94347	13.7
86954	16.8
77863	17.4

Table 10: Example of data showing hour average values of temperature and density of the SW as measured by ACE during 1/1/2018. Data obtained from OMNIWeb site: `https://omniweb.gsfc.nasa.gov/form/dx1.html`

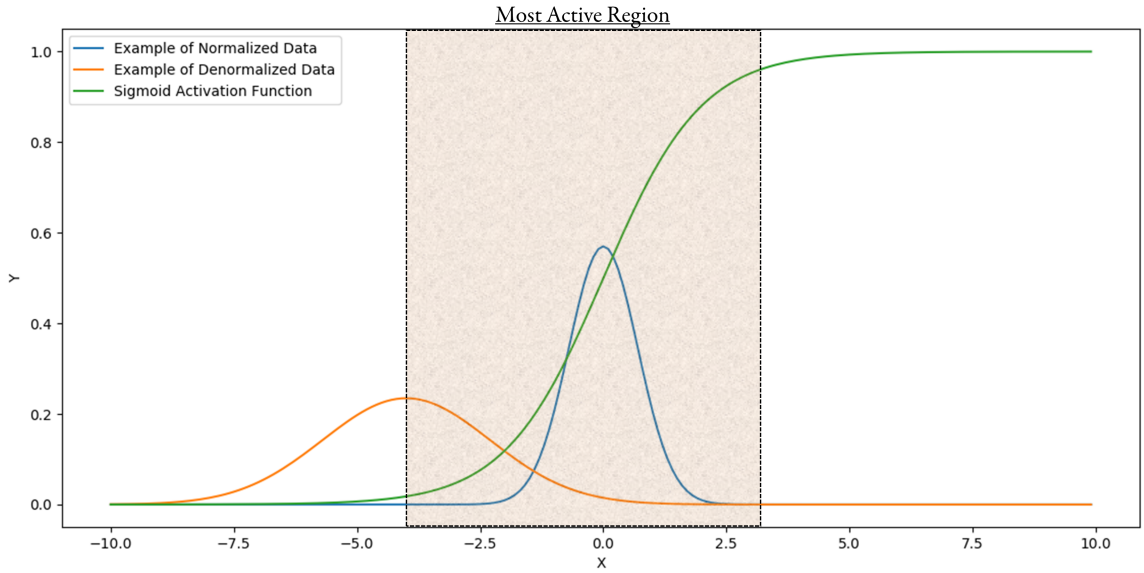


Figure 21: Normalized and de-normalized example of data plotted against a Sigmoid activation function. Special indication has been made to the region where the activation function is changing faster.

Some standard normalization methods are:

$$\text{Rescaling: } x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.5}$$

$$\text{Mean normalization: } x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)} \tag{2.6}$$

$$\text{Standardization: } x' = \frac{x - \text{mean}(x)}{\sigma} \tag{2.7}$$

where, x' is the normalized value, x is the original value and σ is the standard deviation.

2.2.6.2 Train, Validation & Test Set

Dataset is generally separated in three distinct sets.

Training set is the data provided to the model in order to learn and therefore, adjust its parameters (weights, biases etc.) in order to accurately predict the output [97].

Test set is a part of the dataset, independent of the training set. It is used to test the model in data that were not introduced in the training process and therefore can show how well the model behaves when provided with new data [14].

Validation is a separate data set that is used to tune the architecture of the NN and find its optimal form [14].

Traditionally, the original dataset is divided into training, test and validation set by taking 60%, 20% and 20% of the total data, respectively. However, the distribution of data among these sets can influence the testing and modeling potential [109]. In practice, the percentage of each set usually depends on the problem and the availability of data.

2.2.7 Training Process - Backpropagation

The weights and biases of the NN, when presented with data, are adjusted to predict correct output in such a way that the error/loss function is minimized. The method that allows the training of the NN and the determination of the optimal parameters is called "Backpropagation" or "Gradient Descent".

The idea behind backpropagation is that the learning algorithm will change the weights of the synapses to improve its prediction accuracy.

The rule for changing the weights during the back-propagation is given by the Newton-Raphson [6] iterative method. In practice, it is a root-finding algorithm that is using Taylor expansion on a function $f(x)$. The process is repeated until an approximate root (x_m) of the function is found. The process can be described as [121]:

$$x_{n+1} = x_n - \alpha \frac{f(x_n)}{f'(x_n)} \tag{2.8}$$

Where, α is called "learning rate". In practice, α a small constant value $\alpha \in [0.001 - 0.1]$ initially provided by the user. The gradient descent algorithm takes its name from the fact that, at each Newton iteration, the change in the weights of the neuron is given by the gradient of the cost function E multiplied by a constant factor α [104]. A visualization of how gradient descent algorithm works is shown in Fig.22.

The factor α, known as the *learning parameter*, represents the length of the step taken every time towards the direction of the steepest descent. Careful attention has to be put on the choice of the

[6] Named after Isaac Newton [1642 - 1727] and Joseph Raphson [1648 - 1715].

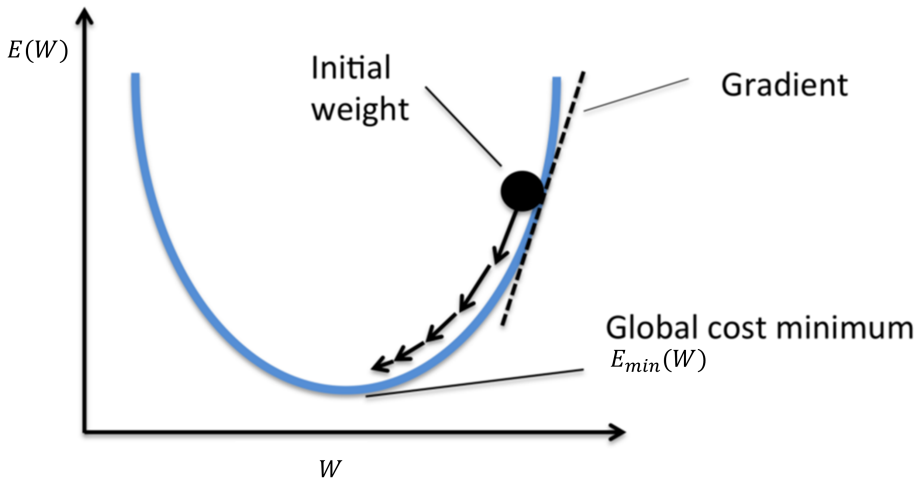


Figure 22: Visualization of the gradient descent algorithm. To minimize the error $E(W)$, the weights need to be changed to values where the error is resides in global minimum. Weight change in the opposite direction of the gradient $\frac{\partial E}{\partial w}$. Edited version of Figure Courtesy: `https://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/`

learning parameter. A small value would result in a small step, which might increase the training time drastically. A big step would accelerate the training of the model, but it might overshoot the minimum and start to "climb up" the cost function and thus fail to converge (Fig.23).

The value of α is typically in the range of 0.001 and 10 and the choice depends on the problem. It is considered easier to find an optimal value through trial and error. Automatic adaptation of this step can also be implemented. The learning rate is essentially the fraction of the vector that is going to be moved in space in order to change the NN's parameters (weight/biases). Having a variable or adaptive learning rate can greatly increase the performance of the neural network while decreasing the computation time [49].

Before explaining how the training procedure is being done mathematically, a formal definition of the notation that is going to be used should be stated:

- z_j is the input to the node j for layer l.
- g_j is the activation function for the node j again at layer l which is applied to z_j.
- $a_j = g_j(z_j)$ is the output of the activation function for the same node and layer.
- w_{ij} are the weights connecting the node i of the layer $l-1$ to the node j of the next layer.
- b_j is the bias term for node j that resides in the layer l
- t_k is the desired output of the output layer for the node k

[7] `https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0`

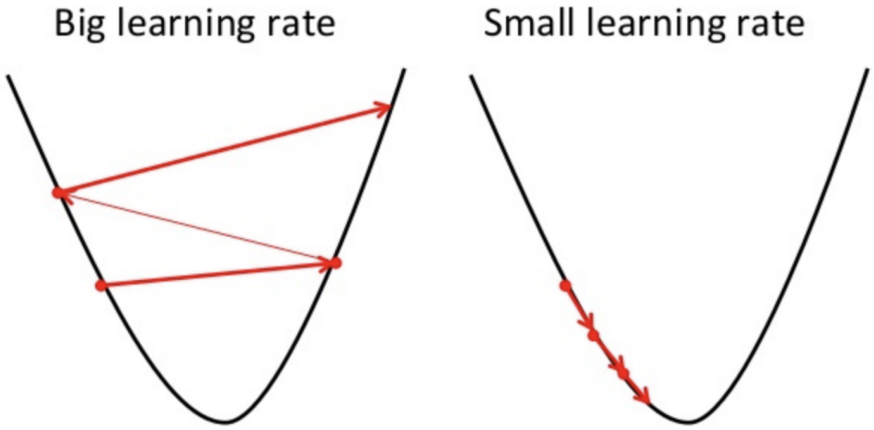


Figure 23: Visualization different learning parameter (α). On the **[Left]** we can see what may happen if we have a very large α, compared to a scenario shown on the **[Right]** where although converging, the procedure takes a very long time. Figure courtesy: Niklas Donge [7]

For E given by equation 2.4, if we consider only 1 training element, for the output layer as shown in Fig.20:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \frac{1}{2} \sum_{k \in K} (a_k - t_k)^2 = (a_k - t_k) \frac{\partial}{\partial w_{jk}} (a_k - t_k) \tag{2.9}$$

For the above expression, the chain rule is used and the summation term disappears because it is derived for a specific weight. The derivative regarding t_k is zero since it is not a function of w_{jk} and from the introduced notation $a_k = g(z_k)$, one can derive:

$$\begin{aligned} \frac{\partial E}{\partial w_{jk}} &= (a_k - t_k) \frac{\partial}{\partial w_{jk}} a_k \\ &= (a_k - t_k) \frac{\partial}{\partial w_{jk}} g_k(z_k) \\ &= (a_k - t_k) g'_k(z_k) \frac{\partial}{\partial w_{jk}} z_k \end{aligned} \tag{2.10}$$

However, $z_k = b_k + \sum_j g_j(z_j) w_{jk}$ and as a result, the above can be re-written:

$$\frac{\partial E}{\partial w_{jk}} = (a_k - t_k) g'_k(z_k) a_j \tag{2.11}$$

Therefore, the gradient of the loss function for the output layer has three terms. The first is the difference between the output of the NN and the desired value t_k. The second is the derivative of the activation function that is used in the output layer and the third is the output of the hidden layer node j. For simplicity, we define:

$$\delta_k \equiv (a_k - t_k) g'_k(z_k) \tag{2.12}$$

Therefore, equation 2.11 can be rewritten as:

$$\frac{\partial E}{\partial w_{jk}} = \delta_k a_j \tag{2.13}$$

In this case δ_k is it the error "signal" of the NN . Therefore, equation 2.13 may be interpreted as the contribution of every weight w_{jk} to the error signal. The output weights are then to be updated as:

$$w_{jk} \leftarrow w_{jk} - \alpha \frac{\partial E}{\partial w_{jk}} \tag{2.14}$$

Where, α is the learning rate we introduced in the previous sections.

This error is propagated back to the input layer by projecting δ_k through the activation function g_j to derive the error signal δ_j. This process continues until we reach the input layer at which the reconfiguration the weights is conducted.

The biases terms are also changing during this process in a similar way:

$$\frac{\partial}{\partial b_k} z_k = \frac{\partial}{\partial b_k} \left[b_k \sum_j g_j(z_j) \right] = 1 \tag{2.15}$$

And as a result,

$$\frac{\partial E}{\partial b_k} (a_k - t_k) g'_k(z_k) \cdot (1) = \delta_k \tag{2.16}$$

Moving to the hidden layer, there are a few more calculations to be done. However, the basic principle remains the same. In specific,

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{1}{2} \sum_{k \in K} (a_k - t_k)^2 \\ &= \sum_{k \in K} (a_k - t_k) \frac{\partial}{\partial w_{ij}} a_k \end{aligned} \tag{2.17}$$

Which by using $a_k = g_k(z_k)$ can be written as:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \sum_{k \in K} (a_k - t_k) \frac{\partial}{\partial w_{ij}} g_k(z_k) \\ &= \sum_{k \in K} (a_k - t_k) g'_k(z_k) \frac{\partial}{\partial w_{ij}} z_k \end{aligned} \tag{2.18}$$

At this point, we need to expand z_k and write it as :

$$\begin{aligned} z_k &= b_k + \sum_j a_j w_{jk} \\ &= b_k + \sum_j g_j(z_j) w_{jk} \\ &= b_k + \sum_j g_j(b_i + \sum_i a_i w_{ij}) w_{jk} \end{aligned} \tag{2.19}$$

Therefore, one can calculate:

$$\begin{aligned} \frac{\partial z_k}{\partial w_{ij}} &= \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \\ &= \frac{\partial}{\partial a_j} a_j w_{jk} \frac{\partial a_j}{\partial w_{ij}} \\ &= w_{jk} \frac{\partial a_j}{\partial w_{ij}} \\ &= w_{jk} \frac{\partial g_j(z_j)}{\partial w_{ij}} \\ &= w_{jk} g'_j(z_j) \frac{\partial z_j}{\partial w_{ij}} \\ &= w_{jk} g'_j(z_j) \frac{\partial}{\partial w_{ij}} (b_i + \sum_i a_i w_{ij}) \\ &= w_{jk} g'_j(z_j) a_i \end{aligned} \tag{2.20}$$

And by using that result in equation 2.18, one can derive:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \sum_{k \in K} (a_k - t_k) g'_k(z_k) w_{jk} g'_j(z_j) a_i \\ &= g'_j(z_j) a_i \sum_{k \in K} (a_k - t_k) g'_k(z_k) w_{jk} \\ &= a_i g'_j(z_j) \sum_{k \in K} \delta_k w_{jk} \end{aligned} \tag{2.21}$$

As expected, the gradient term for the hidden layer as a similar form to the one calculated for the output layer. Using the same definitions and notation as the output layer calculations, it can be shown that:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= a_i g'_j(z_j) \sum_{k \in K} \delta_k w_{jk} \\ &= \delta_j a_i \end{aligned} \tag{2.22}$$

Where,

$$\delta_j = g'_j(z_j) \sum_{k \in K} \delta_k w_{jk}$$

Finally, similarly to the output layer, the bias terms can be calculated for the hidden layers as:

$$\begin{aligned}\frac{\partial z_k}{\partial b_i} &= w_{jk} g'_j(z_j) \frac{\partial z_j}{\partial b_i} \\ &= w_{jk} g'_j(z_j) \frac{\partial}{\partial b_i}(b_i + \sum_i a_i w_{ij}) \\ &= w_{jk} g'_j(z_j)(1)\end{aligned} \tag{2.23}$$

Providing a final result of:

$$\frac{\partial E}{\partial b_i} g'_j(z_j) \sum_{k \in K} \delta_K w_{jk} = \delta_j \tag{2.24}$$

To wrap up, the computations that are done during backpropagation can be summarized by the following steps:

1. Calculating the signal from input to output layer
2. Calculating the error by comparing the prediction a_k to the output t_k
3. Back-propagating the error signal in previous layers
4. Calculating $\frac{\partial E}{\partial w, \beta}$ based on the backpropagated error signal.
5. Updating the parameters (W, β) using the calculated gradients as $w \leftarrow w - \alpha \frac{\partial}{\partial w}$.

2.2.8 Methods of Gradient Descent

There are several ways to implement Gradient Descent (GD). The most commonly used are: batch, stochastic and mini-batch methods.

Batch GD uses all the data for one update of the weights & biases. As a result, it can be slow and may have problems with large databases that can not be kept in memory.

Stochastic GD performs an update of parameters for each example of input & output that is given. As a result, this method performs many updates in a short time but with a high uncertainty.

Mini-Batch GD lies somewhere in between the previous two methods. It performs an update for small subsets of the dataset and therefore, reduces the variance compared to stochastic GD and reduces the memory use compared to batch-gradient descent.

Usually the choice of the backpropagation method falls under the term "optimizer". Optimizers are algorithms that try to use the gradient descent method in the most efficient way.

Generally, the best choice for the optimizer depends on the problem. A more extensive analysis of optimizers can be found in the overview paper of Sebastian Ruder (2016) [102].

2.2.9 Overfitting & Undefitting

Overfitting happens when the model has too many degrees of freedom and fits too well the training set to the expense of generalization. This problem can occur when the amount of features (elements of the input vectors) is too high compared to the amount of data. In practice, the model is "trained too well" to be realistic. The model is storing all the training examples in the NN weights, creating an internal representation of each specific example and therefore failing in the testing data that the NN has not seen before.

Underfitting, describes a model that does not accurately represent the training data nor generalize to any new data provided because it has too few degrees of freedom.

A visualization of the described problems can be seen in Fig.24

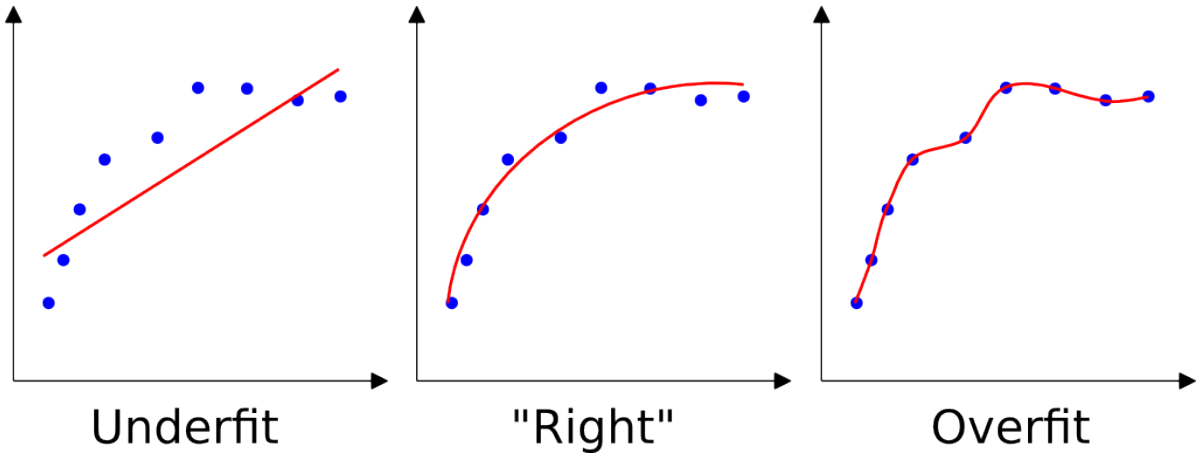


Figure 24: [**Left**]: underfit situation, [**Middle**]: well-behaved model representation. [**Right**]: overfit situation.

Underfiting is not usually a big issue since it can be easily solved by using more features. On the other hand, overfiting can be problematic in machine learning algorithms. The solution to overfit is usually constraining the degrees of freedom. This can be done with several methods. To name a few, having a better architecture on the network, increasing the amount of data or decreasing the amount of neurons [49].

The methods to avoid overfiting are categorized under the term "regularization techniques" that are summarized in the following subsection.

2.2.10 Regularization Techniques

Traditionally there have been several regularization techniques, namely called "L1 Lasso Regression" and "L2 - Ridge Regression" but their principle is similar. The idea is to introduce a penalty term to the error function that allows the code to minimize the relevance of particular weights of

the cost function preventing them from dominating the result. In this section, an analysis of the three most popular indirect methods of regularization is presented. These methods are named "dropout" "dropconnect" and "batch normalization" [112].

Dropout is a technique where a randomly selected subset of neurons has their activation set to zero. During the training process, a different set of neurons is randomly chosen. This method prevents co-adaptation, and therefore, allows neurons to detect features that are more helpful to produce a general answer. Dropout is shown to provide significant improvements on several benchmark tasks and is used in several state of the art [52].

Dropconnect is a generalization of the dropout method. Although it is based on the same principle, this time it is the weights of each neuron that are set to zero and not the node itself. It is considered a generalization because it produces more possible scenarios, since there are, in principle, more connections than nodes within a neural network [119].

A visualization of the dropout and dropconnect methods compared to a non-regularized network is shown in Fig. 25.

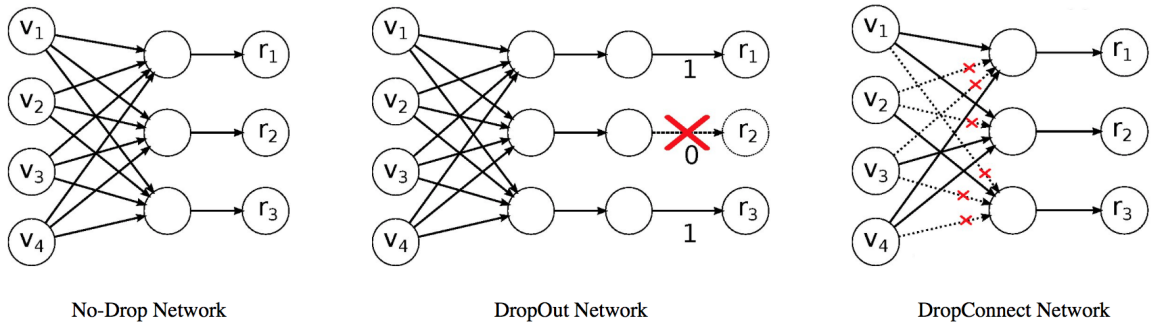


Figure 25: Illustration of dropout and dropconnect neural net model. Simplistic representation of original Figure Courtesy: Li Wan et al. (2013) [119].

A recent indirect regularization technique that provide amazing results is **Batch Normalization** introduced by Ioffe et al. (2015) [59]. This method is based on making normalization a component of the neural network's architecture and carrying it out for every mini-batch of the training procedure [59].

2.2.11 Hyper-parameters

In most neural network architectures, the following hyperparameters have to be set:

1. Learning parameter α.
2. Activation function g.
3. Number of iterations (batch size & number of epochs).
4. Number of layers.

5. Number of nodes.
6. Regularization techniques (e.g. dropout & dropconnect).

Features are the configurations that are determined using data that we use as input in our problem. Hyperparameters are specifications that are hand-picked by the user/programmer and are often used to assist in estimating the actual parameters of our model [70].

2.3 Convolution Neural Network (CNN)

2.3.1 Motivation

After neural networks were introduced, it was quickly realized that the way input of images is treated in dense[8] layers is not ideal. The reason is that through the transformation of the image into a flatten vector that is used in these layers, the "spatial" information of the original image is lost. MLPs ignore any nearby pixel correlations. This means that in NN there is no difference between nearby pixels and distant ones due to their full connect nature [49].

One of the main reasons CNN were created was to develop a neural network that is robust in certain changes of an image, such as translation invariance. Translational symmetry is when an object has experienced a shift, movement or slide, in a certain direction and distance without any other effects acting on it. For example, a human can easily detect an object (car, house etc.) no matter where it is on a picture. However, a feed-forward neural network, since it has different weights and biases everywhere, it might not be capable of determining the characteristics of an object when it is not located in the same spot as the training images used[9].

CNNs were first introduced in the early 90s by Lecun et al. [71] and they demonstrated excellent performance on well-known machine learning tasks such as face detection [69]. Their popularity rose greatly in 2012 after their impressive performance on ImageNet [10] benchmark [69].

CNN also has a direct relationship to biology. Cox et al. (2014) [26] showed that a lot of the ideas used in their architecture are driven from visual neuroscience and the way our eyes transmit information to our brain when we look at images.

2.3.2 Description of CNN & Layers

2.3.2.1 Convolution

Convolution is a mathematical operation acting on two functions. It shows the amount of overlap of a function x when shifted over another function w [122]. For 1D temporal signals it is defined

[8] Dense or fully connected layer are different names of the standard layers that we analyzed and explained so far in this chapter.

[9] The viewer can see the difference in the accuracy between NN and CNN when dealing with images as input at Appendix. C.

[10] ImageNet is an image database commonly used in Machine Learning testing. Can be freely accessed through: `https://www.kaggle.com/c/imagenet-object-localization-challenge`.

as:

$$(x * w)(t) = x(t) * w(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau = \int_{-\infty}^{\infty} w(t)x(t-\tau)d\tau \tag{2.25}$$

Or, in discrete space that we treat numerically objects such as images:

$$(x * w)[n] = x[n] * w[n] = \sum_{m=-\infty}^{\infty} x[m]w[n-m] = \sum_{m=-\infty}^{\infty} W[m]x[n-m] \tag{2.26}$$

The first argument x is called input, and the second w "kernel". The output is sometimes also called "feature map" (Can be also described as units that share the same weights and biases) [49].

When discussing 2D images it is useful to define convolution in discrete space as:

$$(f * g)(x,y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f(u,v) \cdot g(x-u, y-v) \tag{2.27}$$

As a mathematical operation, convolution is commutative, therefore the above can be re-written as:

$$(g * f)(x,y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f(x-u, y-v) \cdot g(u,v) \tag{2.28}$$

In this case, g is a 2D kernel, f is a two-dimensional image and the summation is applied to both dimensions (x,y).

In principle, images can be represented by an array, $I[x,y,c]$ where x, y show the amount of pixels in each direction and c is the channel. Generally, I takes a value between 0 and 255 which shows the intensity of each pixel. In most cases, where we have a single image input, $c = 1,2,3$ and it represents RGB[11] values.

Channel is usually the intensity of one of the primary colors (E.g. for RBG model, Red Blue or Green). Channel however can describe any component of the original image. In practice, channels offer an extra dimension in which we can align several 2D images to create a "bunch" of grayscale images (3D structure) as input .

Convolution layer. These layers are used to apply the convolution procedure to a given image x. Multiple kernels w can be used resulting in an output of multiple convoluted images, one per kernel. An example can be visualized in Fig. 26.

[11]RGB is a color model where by adding Red, Green and Blue together, one can produce a broad range of colors.

Stride is the number of horizontal or vertical spaces that the kernel moves during the convolution operation. For example, one can see Fig.26 to visualize how a stride equal to unity looks like. A stride equal to 2 would result into a $3x3$ matrix and not a $5x5$ as shown in Fig.26.

Padding is the extra pixels that are added around the images so that it is possible to move the kernel around and reach the edge of the image without going outside of it causing an error to the procedure.

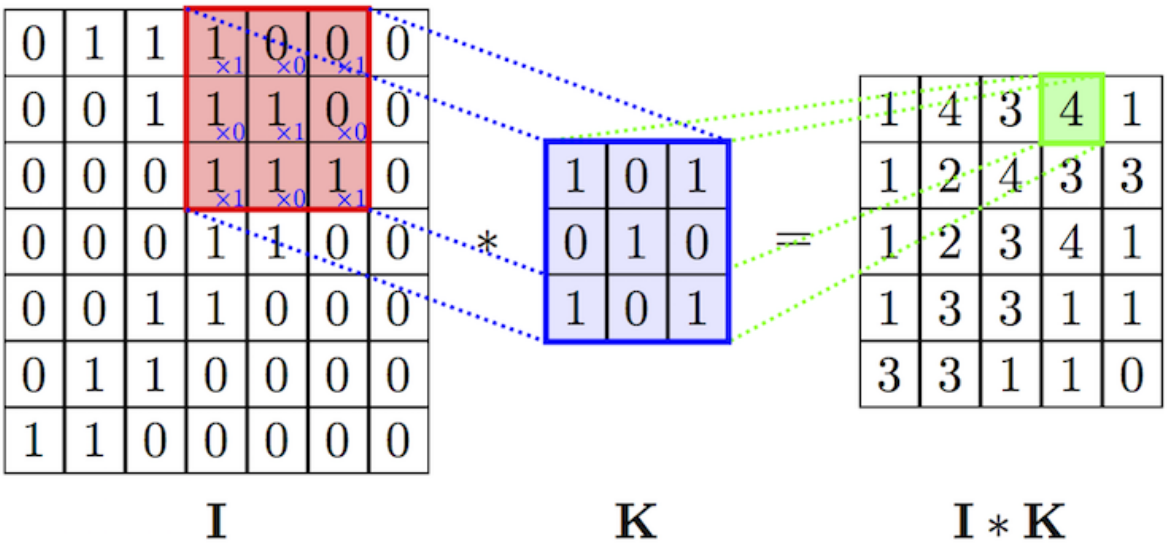


Figure 26: Visualization of the convolution procedure (Equation 2.27). Here we can see the kernel (K) as it goes through an image, acting like a feature detector. Figure Courtesy: Cambridge Spark Ltd (`https://cambridgespark.com`)

Convolution layer improves a NN system mainly because of three reasons. The first is that CNN have sparse weights. This is because the number of kernel weights to train is much smaller than the input of a fully connected NN. As a result, compared to the traditional NN, there are fewer parameters and the memory requirements are reduced. Efficiency is increased and the actual training requires fewer operations. Another vital advantage is the parameter sharing. In a traditional NN all the neurons are tied to each other. On the other hand in CNN, each part of the same kernel is applied everywhere on the layer. This parameter sharing means that there is only one set of parameters that needs to be learned. Once more, this provides a drastic decrease in memory and computational time requirements. Finally, due to this parameter sharing, a new property arises that is called equivariance to translation. This means that if the input of the CNN changes, the output change as well. To simply put it, when we "move" an object that resides in the input image, the convolution procedure will move the output by the same "distance" [49].

2.3.2.2 Pooling & Subsampling

There are two more extremely important layers that are implemented in a traditional CNN architecture.

Pooling layer. These layers are trying to reduce the dimensionality of the feature map while retaining most of its information. It effectively down-samples the output of the previous layer. In this layer, a kernel scans across the image, but rather than extracting features, it tries to efficiently compress information. Again, it is required to define a window size and a stride parameter for this layer. A visualization of a "max" pooling layer is shown in Fig.27. Pooling layers are usually implemented after applying several convolution layers. They are extremely helpful in making the model invariant to small translations of the input. This means that if the goal of the model is to find whether some feature is presented than exactly where, pooling layer is a vital addition.

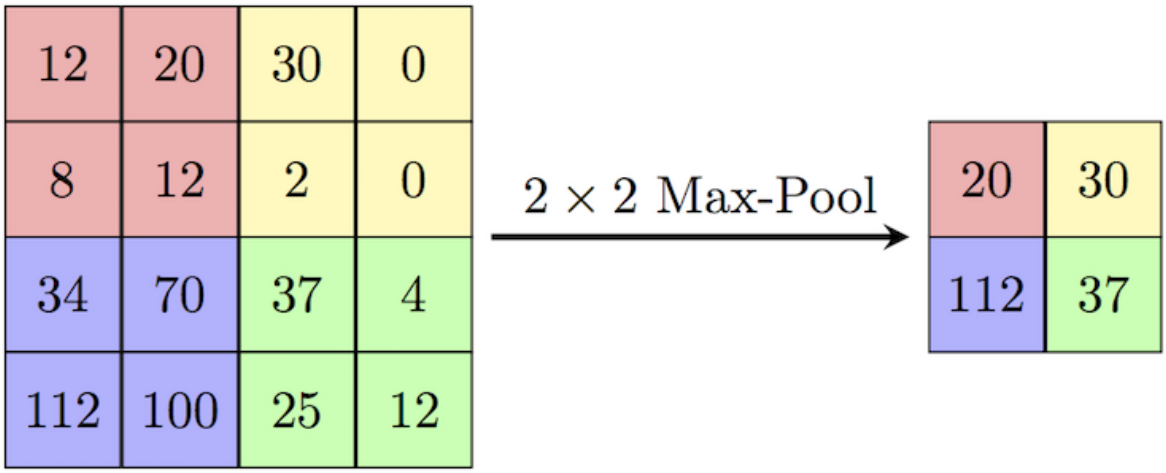


Figure 27: Visualization of the pooling procedure for an example of 2x2 max pooling. Figure Courtesy: Cambridge Spark Ltd `https://cambridgespark.com`

Subsampling layer. A more advanced layer based on the idea of pooling layer is called "sub-sampling". In practice, it is an average pooling layer that has its own trainable parameters (weights) per feature map.

A standard architecture of a CNN that utilizes the previous mentioned layers can be seen in Fig.28.

What was discussed in this chapter also applies in CNN, meaning that applying non-linear activation functions such as ReLu and Leaky-ReLu is necessary after each convolution or pooling layer. Different regularization techniques such as dropout and drop-connect may also be applied on each layer and as always the normalization of the data is vital when searching for the optimal solution and to decrease the computational time of the training procedure.

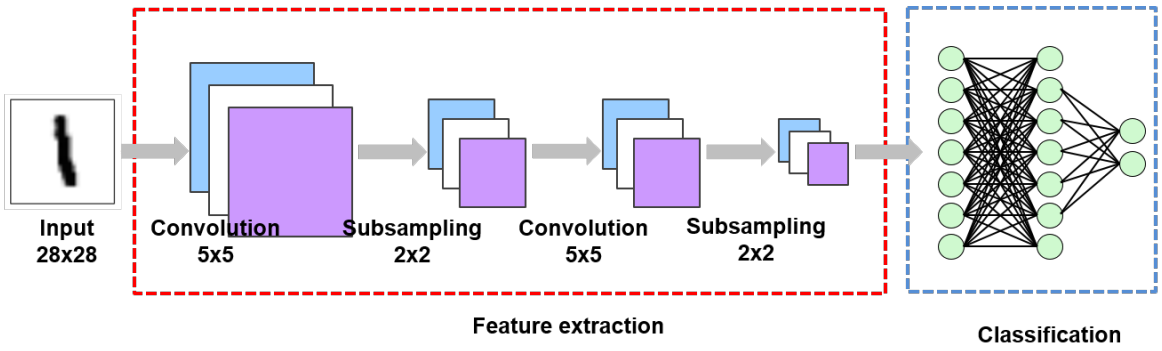


Figure 28: Architecture of CNN used to classify the number "1" shown in the input (28x28) image. It represents a topology with two convolution and two subsampling steps followed by a classification part that uses classic NN fully connected layers. Figure Courtesy: Suhyun Kim iSystems Design Lab [12].

2.4 Other Topologies

Another NN architecture that is widely used is RNN (Recurrent Neural Network). In practice, it is a NN that can process sequential data and its full input consists of the initially provided input, the current timestep and the output that was computed in a previous timestep. As a result, we have some kind of re-injection of the result as input again and therefore our input changes over time.

Long Short Term Memory (LSTM) networks, were proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [53]. Compared to the classic RNN, LSTM is designed to remember information for long periods of time [13].

More information regarding the topologies of RNN and LSTM alongside with a more extensive analysis on the previous sections that were discussed can be found in the freely available book of Goodfellow et al. [49] and in the well-known textbook of Aurélien Géron [44]

[12] http://isystems.unist.ac.kr

[13] A very well organized explanation of RNN and LSTM can be found in http://colah.github.io/posts/2015-08-Understanding-LSTMs/

CHAPTER 3

ANALYSIS & RESULTS

In the previous chapters, the reader was introduced to various concepts of space weather and artificial intelligence. In this chapter, there is a step by step description of the implementation and analysis that was done for this research project

At the beginning, there is an introduction to the actual problem that was confronted. Moving on, there is a full description of the work that was done during this project. Finally, the results along with a discussion section are presented.

3.1 Introduction - Description of the Problem

The ultimate goal of this project is to forecast the emerged CMEs using solar images taken from SDO.

The work that is being done can be separated into three components as visualized in Fig.29.

The **first** project consists of an extensive data cleaning and enhancement procedure. This approach although time-consuming, is vital for the implementation of physical data in machine learning models. Contrary to machine learning applications that rely on data that are pre-cleaned or human derived (e.g. stock market prices, questionnaires etc.), space data are usually heavily contaminated and require a special process to be done before they can be successful implemented.

In the **second** part of this work, I attempt a CME forecasting, deploying a CNN and using as input SDO images from the Helioviewer client [1]. The output of the model is the enhanced version

[1] https://www.helioviewer.org/

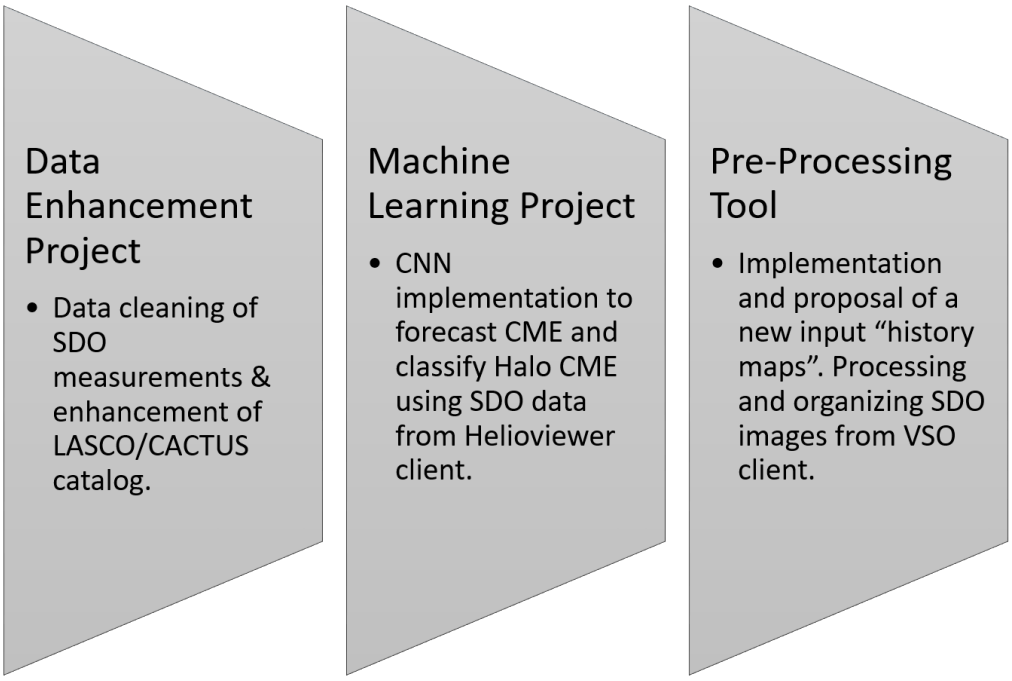


Figure 29: Project description as separated into three distinct components.

of CACTUS and LASCO versions. During this stage, multiple CNN forecasting models have been tested, using data of different resolution and wavelength.

The **third** and final topic of this work is the proposal, explanation and implementation of a Pre-processing tool. This tool derives a new type of solar data called History Maps (HM) that may be implemented as input for machine learning models.

3.2 The Data

Before explaining every part of the project, it is important to present the data that were used as Input and Output in all parts.

3.2.1 Input

For the actual input that was applied in the CNN model, images of SDO were used. Their format is:

$$I[C,x,y] = \text{Value} \tag{3.1}$$

Where, C is the channel, and x,y is the resolution of the image in each dimension.

Every input consists of an array I that holds the intensity value of C number of images showing the history of the Sun for a total time T. For example,

If given, $T = 2$ [h] and a sampling of $dt = 10$ [min]. There are 13 images that are shown in the channels of the array (I).

$$N_C = T/dt + 1 = 13$$

where, N_C is the number of channels.

As a result $C \in [1, 13]$ where C_1 represents the Sun 10 minutes before the event, C_2 is 20 minutes before, and so on until the last image shows the Sun 2 hours before the CME.

3.2.2 Output

For the output of this problem, I selected the CACTUS and LASCO catalogs. These catalogs have dates of CME events along with their characteristics.

The LASCO CME catalog is a manually derived catalog from SOHO/LASCO and is publicly available[2].

Another catalog of automatic detection can be accessed from CACTUS [3]. CACTUS catalog is considered to be accurate and possibly even better than human detection. As shown by Robbrecht et al. (2009) [99], the events that are only found in the CACTUS catalog are in principle smaller scale outflows related to other previous CME activities. However, a significant percentage (~ 15%) of the events are found to be independent.

The differences of the two catalogs should be taken under careful consideration. Taking as example the year 2014, The automatic CACTUS catalog consist of 1855 events, whereas, the LASCO catalog describes 2478 distinct events.

More information about the characteristics of these catalogs is given in the theory section, in Chapter 1.

3.3 The Data Enhancement Project

In this part of the work, I use SDO images of multiple resolution (512x512, 1024x1024, 4096x4096) and wavelengths (171 Å, 193 Å, 1700 Å).

For all the testing and analysis, I used images from the period 2014-2015. This is due to the fact that both the pre-process and the training of the CNN are extremely expensive in compute power and memory use. As a result, certain constrains had to be implemented on the data. 2014 was chosen as the test year because it is part of the solar maximum period of the 24th solar cycle and is expected to contain several CMEs.

[2] https://cdaw.gsfc.nasa.gov/CME_list
[3] http://sidc.oma.be/cactus/

3.3.1 Data Cleaning - Multiple CMEs

Both catalogs had multiple CMEs happening on the exact same date and time windows. In this project, it was decided to train the neural network to predict single occasions of CMEs since it supposes to act as a protection tool rather than a multi-CME modeling software. It was therefore necessary to clean the dataset from multiple CMEs occurrences and leave only the most intense and energetic events. It was decided to keep the CME with the highest angular width since a higher value indicates a partial or full halo CME that in principle have a higher geomagnetic effect [118].

From the LASCO catalog during 2014, 37 cases of CMEs happening at the exact same time were found. However, in CACTUS catalog due to its automatic nature this phenomenon is much more frequent. It was found that in total, 183 cases had the exact same date and time.

3.3.2 Data Integration - Halo CME distinction

Apart from the forecasting of CMEs, during this work I also tried to classify between halo and Non-halo CMEs. To create an appropriate catalog for this classification, it was necessary to have an output that clearly distinguishes between the two phenomena.

In the case of CACTUS catalog to get all possible partial and full halo CMEs, I labeled all CMEs that had a principle angle $\theta > 90$ as halo.

In the case of LASCO catalog, LASCO research time had provided a detailed description for each of the event. It was decided to accept as halo cases these tagged with the keyword "partial halo" or "halo". Furthermore, we neglected from the database all events that were characterised as "very poor events".

3.3.3 Data Integration - No CME phenomenon

A neuron network prediction model for CMEs needs to be trained also for non-CME events in order to be accurate. This is done to train the network to distinguish between a signal that pre-exists of a CME and a signal that does not.

As a result, it was necessary to integrate non-CME dates into the CACTUS and LASCO catalogs. To derive the non-CME dates, it was decided that the non-CME cases should be added at least 1 hour after a CME and 1 hour before the next one. This was done in order to ensure that during the prediction time window of 2 [h], the input data is not contaminated with previous CME signals. The uncertainty of the CME onset time is quite significant in both catalogs [99], making this procedure an even greater priority.

For the prediction window that I used, it was decided to take a time window of $T_D = 3$ [h] in order to avoid any incorrect signal.

This resulted in the creation of 779 cases for the CACTUS catalog and 935 cases for the LASCO that indicate a "safe" non CME existence during 2014.

After adding these dates of the non CME cases, the dataset of CACTUS and LASCO catalog have the format that is shown in Table 11. In the final enhanced dataset, both CACTUS and LASCO catalog have their CME dates labeled either "1", showing the existence of an event or "0" showing the absence.

A visualization of the CACTUS and LASCO catalog enhancement procedure is shown in Fig.30.

Date	**CME** [0/1]	**Halo CME** [0/1]
2014/01/01 00:12:05	1	0
2014/01/04 23:12:05	1	1
2014/05/03 20:42:00	0	0

Table 11: Example of the modified version of the CACTUS and LASCO catalogs, used for both the CNN model and the Pre-processing tool.

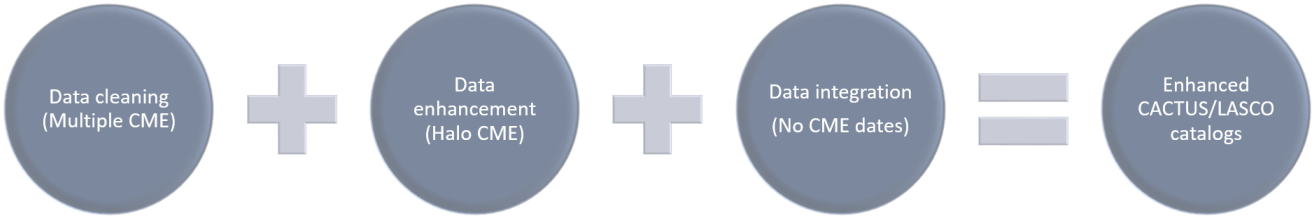


Figure 30: Visualization of the steps taken for the enhancement of the CACTUS and LASCO catalogs to use them as outputs for the CNN.

3.3.4 Data Cleaning - Corrupted Data

Corrupted data can be found in the enriched catalogs. To avoid any errors on the training procedure I had to delete certain parts of the dataset. Some of the major corrupted data cases that were found during the project are enlisted.

From the LASCO catalog:

- 2014/05/14 00:42:05 - Corrupted data found during pre-processing
- 2014/02/12 12:12:05 - Corrupted data found during pre-processing
- 2014/02/12 14:00:05 - Fatal error during pre-processing
- 2014/02/12 18:46:46 - Fatal error during pre-processing
- 2014/03/12 22:18:58 - Corrupted FITS file during pre-processing
- 2014/03/17 09:54:05 - Unknown fatal error
- 2014/07/18 06:55:26 - Unknown fatal error
- 2014/07/20 06:24:07 - Corrupted data found during pre-processing

- 2014/07/23 18:00:57 - Fatal error, corrupted SDO image

From the CACTUS Catalog:

- 2014/01/02 11:36:00 - No data available to use
- 2014/02/10 21:36:00 - Corrupted data found during pre-processing
- 2014/05/04 00:12:00 - Corrupted FITS file during pre-processing

There have been more cases of corrupted files during 2014. This time-consuming task showed the necessity of a full project focusing on the data-cleaning process of input and output. This procedure is essential especially if a fully functioning machine learning model for the heliosphere is to be created.

In summary, the following steps have been performed on the data

For the **Input**:

- I created a routine that downloads 512x512 or 1024x1024 images of the Sun taken from SDO at various wavelengths. In specific, 171 Å (Region Corona), 193 Å (Corona/Flare Plasma), 1700 Å (Photosphere).
- The SDO images were processed slightly using the helioviewer tool as it is implemented in the SunPy library (Appendix.C).
- The starting date of the script was set to start a few minutes before the event's starting point and reached up to 2 hours before the event.
- The sampling frequency of the images can be directly adjusted from the script. In this work data were taken with 10 min intervals.

As a result, the input that was tested in our machine learning model was:

1. 512x512 SDO images (λ = 171 [Å] - dt = 10 [min] - T = 2 [h])
2. 1024x1024 SDO images (λ = 171 [Å] - dt = 10 [min] - T = 2 [h])
3. 512x512 SDO images (λ = 193 [Å] - dt = 10 [min] - T = 2 [h])
4. 512x512 SDO images (λ = 1700 [Å] - dt = 10 [min] - T = 2 [h])

For the **Output**:

- I created a routine that downloads, processes and organizes the output files (CACTUS & LASCO catalogs)
- The processing of the catalogs was done in a dataframe format using the library Pandas[4] while adjusting their content to be used in the CNN.
- Several data cleaning and integrating algorithms were implemented to derive the final form.

[4] `https://pandas.pydata.org/`

As a result, the output that was used in the ML model was:

1. LASCO catalog (Date, CME, Halo = String, 0/1, 0/1)
2. CACTUS catalog (Date, CME, Halo = String, 0/1, 0/1)

3.4 The Machine Learning Project

3.4.1 Architecture of CNN

For the architecture of the CNN, I use a similar approach to the classic CNN examples that have been used for various problems such as classifying handwritten numbers (MNIST) [69].

The basic idea is to use 2-3 convolution layers in order to capture the most important features then include a max pooling layer to reduce dimensionality. The process is repeated until the length of the final input to the dense layers is minimized efficiently to be handled by a GPU for the training.

For the implementation of the CNN, I used Keras API with tensorflow library as back-end. More information regarding these libraries and how they can be implemented can be found in Appendix. C. During the implementation of the CNN models, I worked with state-of-the-art regularization techniques that included dropout, drop-connect and batch normalization.

The input of the network is a series of images derived from the helioviewer client. For the 2 [h] prediction window, 13 images taken every 10 minutes and starting from the date of the CME are grouped together as input. An example of the images used in the input array can be seen in Fig.31.

In Table 12 the architecture of the model that best predicted the CME occurrences of CACTUS catalog is shown.

The test loss of the model was calculated using binary cross entropy. In binary classification the formula is:

$$E = -(y\log(p) + (1-y)\log(1-p)) \tag{3.2}$$

Where, y is a binary indication (0/1) whether the class label (c) is correctly classified for the observation (o). p is the predicted probability that observation o is classified as class label c. In practice, the closer the value of the error function to zero, the better the result/model.

Apart from the architecture shown in Table 12, for the following result, 12 epochs with a batch size of 20 inputs per iteration were used. A total 80 % of the data was used for training, a 10 % was used for validating the model and the Adam optimizer was used with $\alpha = 0.02$.

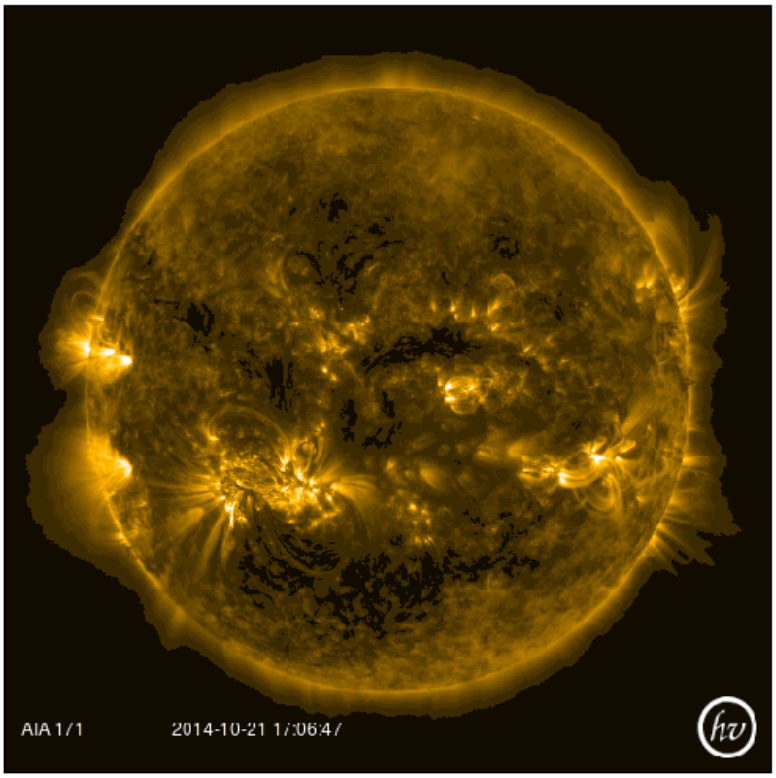


Figure 31: The Sun as it looks in 512x512 resolution at λ = 171 Å, downloaded and slightly processed by the helioviewer client.

The result after using 10% of the dataset as test set for the CNN was:

$$\text{Accuracy} = 76.6\%$$

$$\text{Test loss} = 0.6$$

A similar model was used to classify between halo and non-halo CMEs using LASCO catalog this time.

Apart from the architecture shown in Table 12, for the following result, 15 epochs with a batch size of 20 inputs per iteration were used. A total 80 % of the data was used for training, a 10 % was used for validating the model and the Adam optimizer was used with $\alpha = 0.02$.

The result after using 10% of the dataset as test set for the CNN was:

$$\text{Accuracy} = 83.5\%$$

$$\text{Test loss} = 0.7$$

Layer	Details & Operations	Output shape
Input	-	[512,512,13]
Convolution	Convolution [14] & 3x3 Kernel	[510,510,14]
Convolution	Convolution [16] & 3x3 Kernel	[508,508,16]
Convolution	Convolution [18] & 3x3 Kernel	[506,506,18]
Max Pooling	Max Pooling with 2x2 Kernel	[253,253,18]
Dropout	20 % Dropout	[253,253,18]
Convolution	Convolution [20] & 3x3 Kernel	[251,251,20]
Convolution	Convolution [28] & 3x3 Kernel	[249,249,28]
Convolution	Convolution [36] & 3x3 Kernel	[247,247,36]
Max Pooling	Max Pooling with 2x2 Kernel	[247,247,36]
Dropout	20 % Dropout	[123,123,36]
Convolution	Convolution [40] & 3x3 Kernel	[121,121,40]
Convolution	Convolution [56] & 3x3 Kernel	[119,119,56]
Convolution	Convolution [72] & 3x3 Kernel	[117,117,72]
Max Pooling	Max Pooling with 2x2 Kernel	[58,58,72]
Dropout	40 % Dropout	[253,253,18]
Convolution	Convolution [80] & 3x3 Kernel	[56,56,80]
Convolution	Convolution [112] & 3x3 Kernel	[54,54,112]
Convolution	Convolution [144] & 3x3 Kernel	[52,52,144]
Max Pooling	Max Pooling with 2x2 Kernel	[26,26,144]
Flatten	Flattening of the input	97344
Fully Connected	400 Neuron - Dense layer	400
Fully Connected	200 Neuron - Dense layer	200
Fully Connected	2 Neuron - Dense layer	2
Output	Classifier, 0.5 Threshold Sigmoid	2

Table 12: Basic architecture of the CNN model used for predicting CMEs as they are shown in the enhanced version of the CACTUS catalog.

In all models, it was found that the best results were obtained using a LeakyReLu activation function on every unit apart from the output. The a parameter of LeakyReLu was set to be $a = 0.02$. The output layer uses a sigmoid activation function with a threshold of 50% for the classification.

Training evolution for the epochs of the CME prediction model is shown on Fig.32. Every epoch consists of 182 batches with 20 data points each. Increasing the epoch number resulted in an increased computational time without having any significant difference in the accuracy.

The most vital implementation was batch normalization. Slight changes between different gradient descent techniques and between activation functions were also tested but the most essential element was by far the batch normalization layer. To provide the best result, it was aplied after each convolution that operated on the provided data. Two more batch normalization layers were added after the fully connected layers that are applied at the end of the CNN

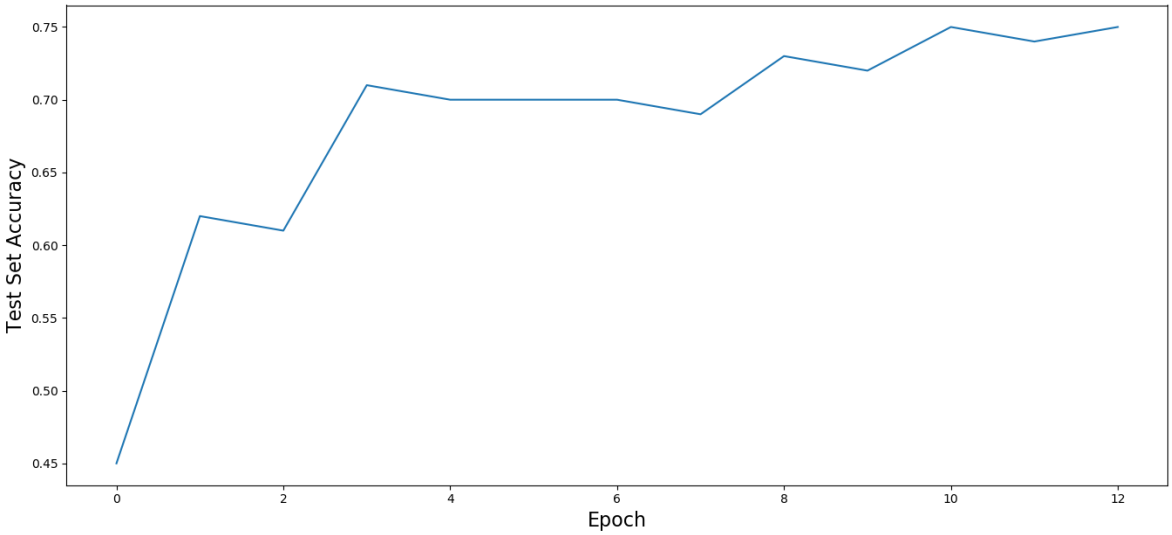


Figure 32: Accuracy of test set versus epoch. Accuracy is provided as a percentage of correct prediction, $y \in [0, 1]$. The peak is reached at roughly 12 epochs.

architecture. All the models with the above architecture were tested for the year 2014. Each training and validation run had a computational time of $\sim 2-3\,[h]$ using two Tesla K40c graphic cards in parallel configuration.

A diagram of the CME model is shown in Fig.33.

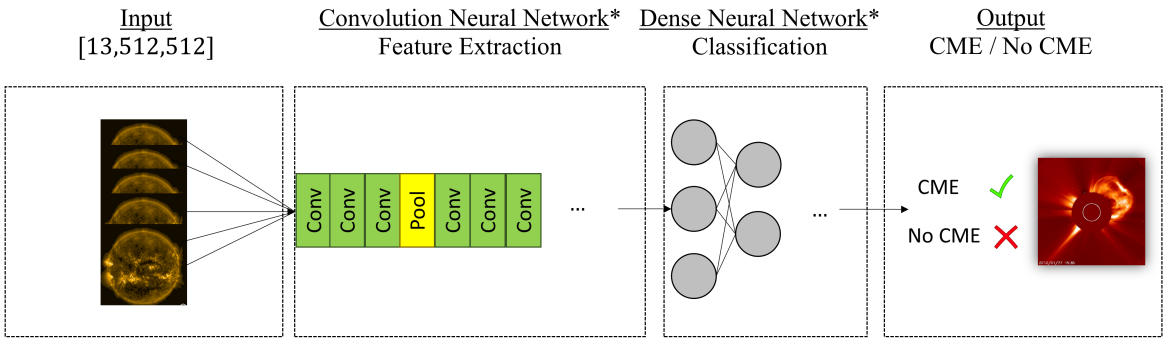


Figure 33: Diagram of the CNN implementation done for the machine learning project.

3.4.2 High Performance Computing (HPC) & VSC implementation

A very common problem met during the training of neural networks and specifically of CNNs is the computational time required for the training and the enormous amount of data. These problems are treated by the fields of HPC and "big data algorithms", which describe codes that utilize data sets too extensive and complex to be processed with conventional methods. In practice, these fields are trying to do one thing. Get the most information out of data while minimizing the computational time, cost and storage.

For this work, space and access on the VSC [5] supercomputer was provided. VSC is managed by FWO and collaborates with five Flemish universities including KU Leuven. To access the cluster SSH (Secure Shell) was used and both CPU and GPU nodes were utilized. In specific, CPU nodes were used for the download and process of the input and output, while GPU nodes were used for the training of the CNN models.

3.5 The Pre-Processing Tool

On the following sections, we are going to describe in details the pre-proecessing tool that was created as the third and final component of this project.

3.5.1 The Pre-processing Tool - Overview

The most important step to create an optimal machine learning model is to find and pre-process the data used as input. Usually this procedure can be more complex and time-consuming than the implementation of the model itself and this project is no exception.

After investigating several research papers on both solar flare prediction and CME forecasting using machine learning ([42], [15]), it was decided that a new strategy for the use of SDO images has to be created. This idea was further enhanced after utilizing a Convolution Neural Network and realized that there might be a better alternative than predictions with pure SDO data. Therefore, it was decided to follow a more risky and experimental procedure, deriving a new type of solar data, called "History Maps" (HM).

The processing tool is a module, built in python, that downloads, processes and organizes images from SDO in order to be fully used as inputs to an A.I. model. The idea behind the processing tool is to create an automatic procedure that uses the Sun's differential rotation and images from SDO to derive new features (HM). These history maps show how the intensity of a specific area of the Sun develops during a time period. The tool takes the intensity of multiple longitudes at a given date, and then tracks their position back in time ([t] hours), creating one HM per longitude. A representation of the intensity ([time vs latitude vs longitude]) keeps a recollection

[5] `https://www.vscentrum.be/`

of the evolution of fast evolving features; like CMEs; in the atmosphere of the Sun. In practice, after applying a differential model, depending on the input of the user, the output is an array $I[dt, x, y]_T$.

I shows the intensity values for a specific time (T), where, x is the Sun's latitude, y is the longitude and dt the timestep that was used for applying Sun's differential rotation.

By the time this report is written, there are 2 fully functioning versions. One that takes as input the dates of the CACTUS and LASCO catalog and one that takes as input a date by the user. In the following section an explanation of the tool in the most general case is being presented.

3.5.2 Pre-processing tool - Motivation

Additionally to what has been already stated, there is deeper reasoning behind this approach. I believe that, capturing the time information in aligned images in a similar way that color is captured in everyday photos can be of great use for a CNN network.

To be precise, likewise our original SDO input, the final result of our pre-processing tool is similarly:

$$I[C, x, y] = \text{Value} \tag{3.3}$$

However, compared to equation 3.1, in this case, x is the amount of points along a specific longitude line y that is propagated back in time following a differential rotation model of the Sun.

There are two important reasons that this idea is appealing. The first is that it lowers the resolution of the input, since in principle one can take a much smaller amount of y longitude lines than the resolution of SDO data. For the space scales of CMEs a $y \leq 512$ [pix] can be used.

The second reason is that it is a big step towards multi-CME forecasting, since it shows different areas of the Sun depending on the initial longitude line. In practice, a specific area of the Sun is being tracked back in time, giving the possibility to spot all kind of MHD phenomena that may arise such as sunspots, CMEs, flares etc. To visualize this, a **"History Map" (HM)** of a sunspot is shown in Fig.34. On this figure, there is a visualization of how a longitude line ($-2\,[^\circ]$) looks like as tracked back in time for $T = 2$ [days].

This image is a visualization of $I[C, x, y]$, if one fixes a specific value in the dimension $y = -2\,[^\circ]$) while, $C \in [0, 288]$ and $x \in [0, 2000]$. It can be viewed as a representation of the time and space evolution of the -2 degrees longitude line of the Sun. The actual Sunspot as it looked at t_0 and at $t_0 - 2$ [days] back in time is shown in Fig.35.

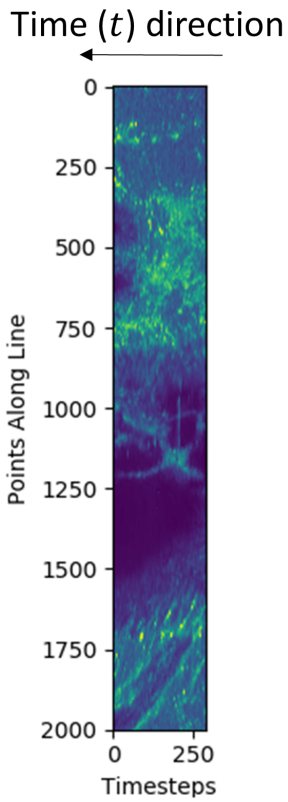


Figure 34: Visualization of a sunspot development as shown in the HM format (x versus C). The t_0 moment is captured in the last time step (dt). Zero timestep shows the longitude line at *2014-10-23 18:39:42* (left edge). There is a total propagation of 2 days back in time with a timestep of $dt = 10$ [min], leading to the last timestep (right edge) at *2014-10-21 18:39:42*

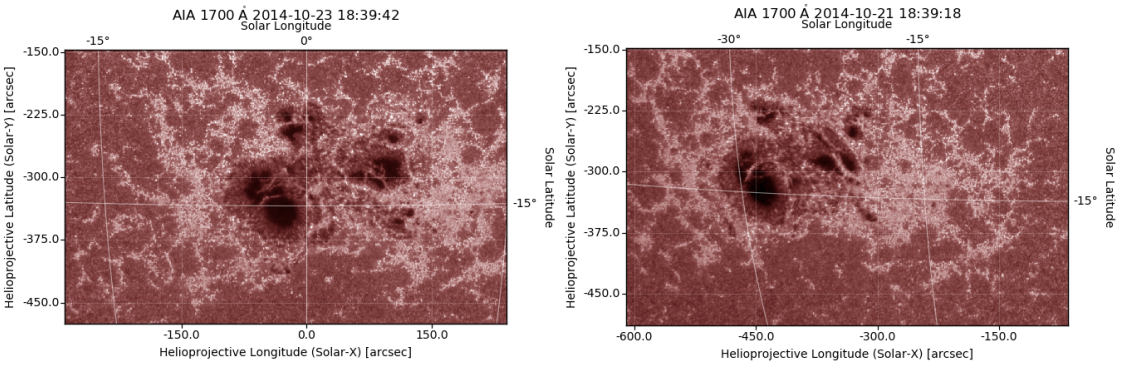


Figure 35: **[Left]**: Visualization of the sunspot at *2014-10-23 18:39:42* **[Right]**: Visualization of the same sunspot at *2014-10-21 18:39:42*. Both the latitude (y) and longitude (x) are in [arcseconds]. The History Map (HM), of this structure as seen by following the $-2°$ longitude line can be seen in Fig.34.

3.5.3 Pre-processing tool - Differential Rotation

For the differential rotation of the Sun, several modules can be applied. The ones that are already implemented in the SunPy library and that were used, are :

1. Howard's
2. Allen's
3. Snodgrass's

The first model by Howard et al. (1970) [56], is a widely used model describing rotation law as derived from observations of spectroscopy. This model was later analyzed and further developed by Robert Howard and therefore was named after him [54]. The model can be summarized in the following equation:

$$\omega = A + B\sin^2(\varphi) + C\sin^4(\varphi) \tag{3.4}$$

where, ω is the angular velocity of the Sun, ϕ describes Sun's latitude and A, B, C are constants. These constants as implemented in SunPy's library[6] are:

$$A = 2.894\,[\text{rad/sec}]$$
$$B = -0.428\,[\text{rad/sec}]$$
$$C = -0.370\,[\text{rad/sec}]$$

The second model by Allen is a much simpler model, shown in the textbook he originally wrote and developed by Arthur N. Cox [25]. His approach is much simpler and when dealing with models that require high precision, its use is not advisable.

The third model by Snodgrass et al. (1990) [111] is similar to Howard's model. However, the constants of the equations are derived differently and are considered to be more suited for magnetic structures of the Sun. The differential rotation follows the expected form of:

$$\omega = A + B\sin^2(\varphi) + C\sin^4(\varphi) \tag{3.5}$$

With constants:

$$A = 14.713 \pm 0.0491\,[\text{deg/day}]$$
$$B = -2.396 \pm 0.188\,[\text{deg/day}]$$
$$C = -1.787 \pm 0.253\,[\text{deg/day}]$$

[6] http://docs.sunpy.org/en/v0.9.0/_modules/sunpy/physics/differential_rotation.html#diff_rot

It was decided after careful examination and by following the movement of magnetic structures, that the most accurate model for the pre-processing method is the one developed by Snodgrass et al. [111]. As a result it is the differential rotation model that was implemented in our tool.

A detailed comparison between the differential models can be found in the excellent review paper of Beck, J. (2000) [10].

3.5.4 Pre-processing tool - Procedure & Results

The module takes as **input** the following parameters:

- Amount of points (x) for each longitude line.
- Date (either "string" or "LASCO" / "CACTUS" keyword for direct import of dates from the catalogs).
- Amount of longitude lines (y) and degree range (e.g. [-60,60,100] means 100 lines from -60 ° to 60 °).
- Timestep (dt) to apply differential rotation module into all the coordinates of the given (x, y) pairs.
- Total duration (T), showing the total propagation back in time.
- Wavelength (λ) to select the corresponding instrument of the SDO satellite.
- Number of loops, meaning how many dates we want to propagate in time totally. For example, loops = 10 means that 10 dates that are either provided by the user or derived from LASCO/CACTUS catalogs are going to be used.

In the case of the CACTUS/LASCO specific module, input for specifying the starting point of the catalog can be given. Starting point shows the origin date from which the differential module is applied.

The **output** of the module for every loop (date) are:

- A database file (.sql) that holds all the information of the images of SDO used along with their path to the local folder where they reside.
- Scientific SDO data (4096 X 4096) for every timestep, saved as .FITS files and automatically distributed according to the wavelength that is given by the user. Files are downloaded using the Virtual Solar Observatory (VSO) [7] client.
- An array file for the neural network input ($I[C, x, y]$ = Value) saved as NumPy array (.npy) locally.
- (*Optional*) Providing a keyword to the script, a history map (2D) visualization of every imported longitudinal line can be derived.
- (*Optional*) Providing a keyword to the script, an animated GIF, including all frames that were used during the procedure can be derived.

[7] `https://sdac.virtualsolar.org`

- (*Optional*) Providing a keyword to the script, diagnostics that include a printout of all the steps while the script is running can be derived. Furthermore, the dates of the .FITS files that are being used as the procedure goes on are also printed.

A diagram of the procedure can be visualized in Fig.36.

Am important step is the choice of coordinate frame of reference before, during and after the differential rotation module is applied. If a specific longitude line is to be propagated back in time, the choice of the coordinate system for the degree initialization should be the Stonyhurst [8] heliographic system. Considering that the location of SDO changes over time, it is important to transform the coordinate system every time based on its current position. Fortunately, this is already implemented in the SunPy library [9], allowing to transform the coordinate system to be centered to the current SDO position every time an operation is being done.

Both the differential rotation module and the coordinate system transformation has used a lot of the properties that are implemented in the .maps format that is provided by the SunPy library [10]. These variables keep metadata, meaning data that provide information about other data that are also included in the file of the actual measurements. Metadata include the position of the satellite, its resolution, its exact date of measurements and much more information.

An example of a longitude line being propagated back in time is shown in Fig.37 and Fig.38. Another example of a multiple line propagation can be seen in Fig.39 and Fig.40.

In these figures, the evolution of a sunspot on the southern hemisphere of the Sun can be observed. As expected, the generated HM tracks the space and time evolution of this active region.

Fig.37 and Fig.38 are generated using the slightly edited data provided by the Helioviewer client at 2048x2048 resolution and $\lambda = 1700$ Å

Fig.39 and Fig.40 are generated using the full resolution scientific images provided by the VSO client at 4096x4096 resolution in $\lambda = 1700$ Å

[8] Named after the village Stonyhurst in which the observatory where the development of this coordinate system resides.

[9] `http://docs.sunpy.org/en/v0.9.0/code_ref/coordinates.html`

[10] `http://docs.sunpy.org/en/v0.9.0/code_ref/map.html`

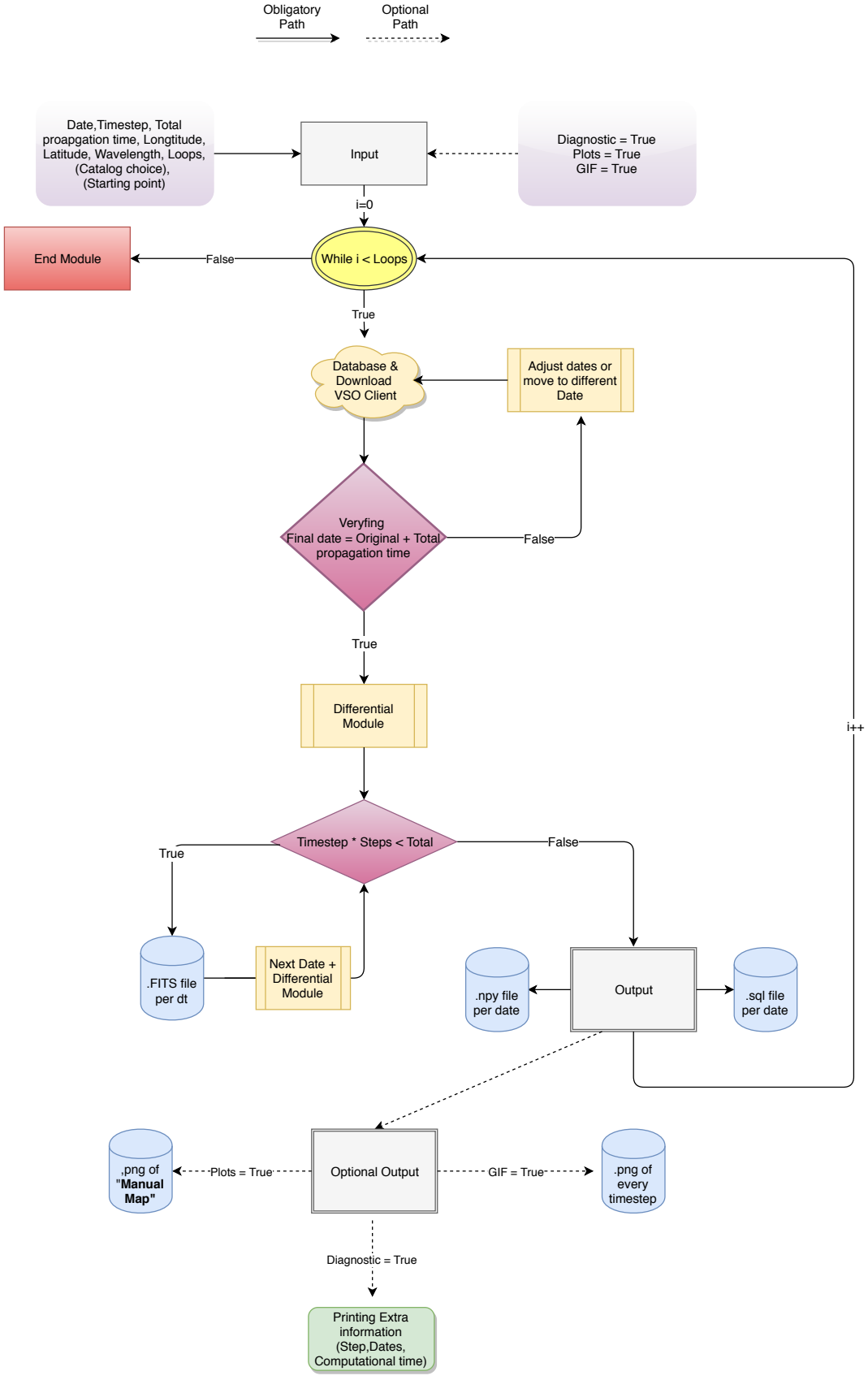


Figure 36: Diagram of the pre-processing tool as described in the main text.

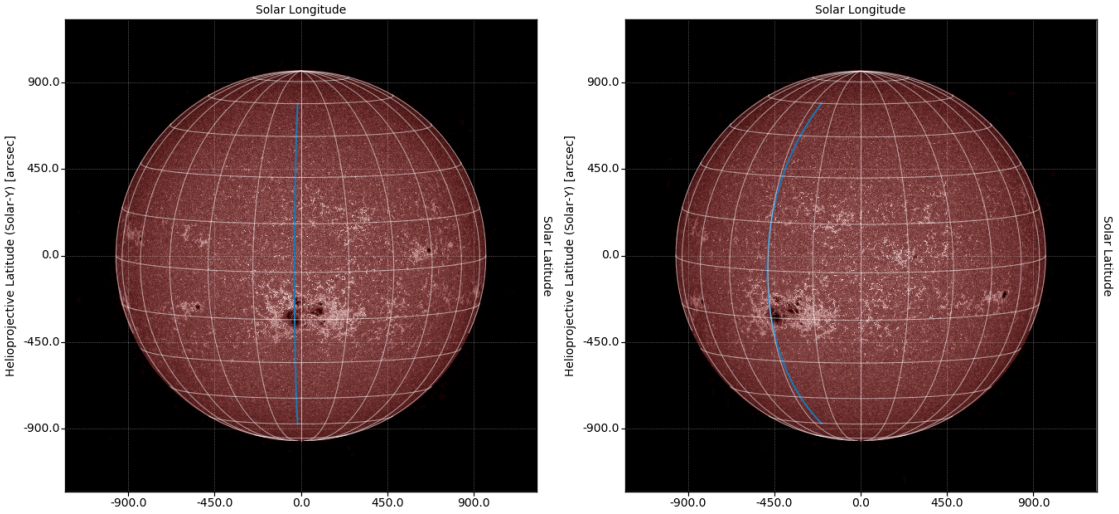


Figure 37: **[Left]**: Visualization of the Sun at *2014-10-23 18:39:42* including a longitude line ($-2°$). **[Right]**: Visualization of the Sun at *2014-10-21 18:39:42* including the same line propagated back in time. Both the latitude (y) and longitude (x) are in [arcseconds].The history of the line can be seen in Fig.38.

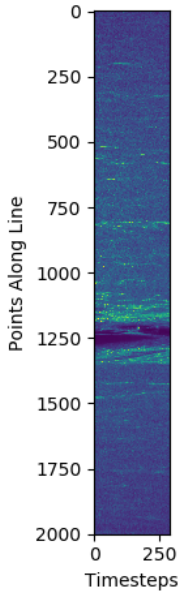


Figure 38: History Map (HM) derived for the longitude line that is shown in Fig.37. A sunspot is clearly visible in the southern hemisphere near the equator of the Sun. The time t_0 (*2014-10-23 18:39:42*) is plotted in the last time-step (dt), therefore in the left-most column of the figure. The right-most part of the figure shows the final moment of the back in time propagation (*2014-10-21 18:39:42*).

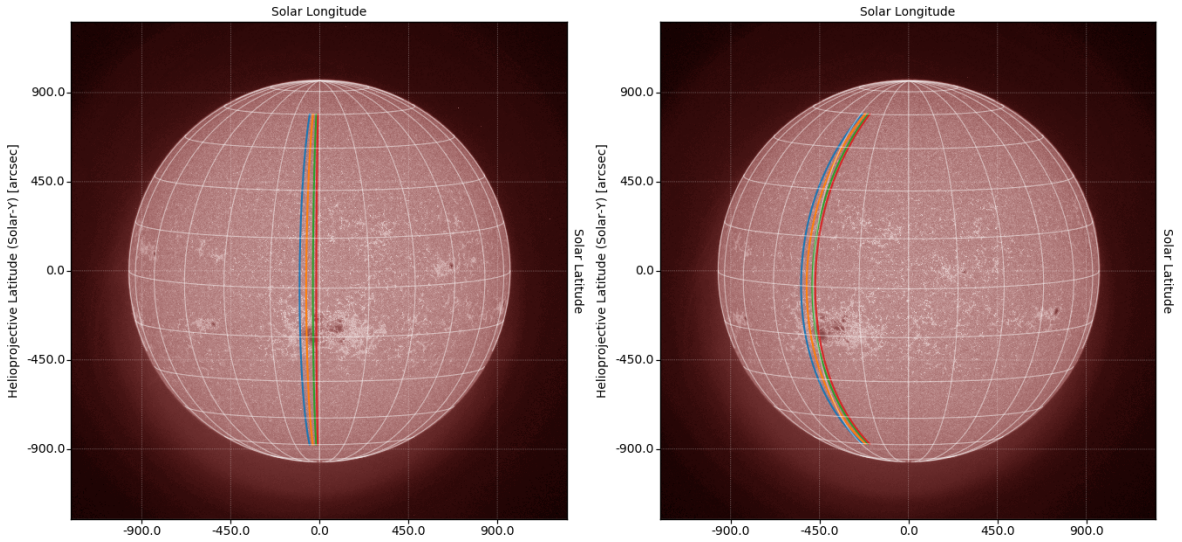


Figure 39: **[Left]**: Visualization of the Sun at *2014-10-23 18:39:42* including five longitudinal lines ($-2°$, $-4°$, $-5°$, $-6°$). **[Right]**: Visualization of the Sun at *2014-10-21 18:39:42* including the same back in time propagated lines. Both the latitude (y) and longitude (x) are in [arcseconds]. The history of the lines can be seen in Fig.40.

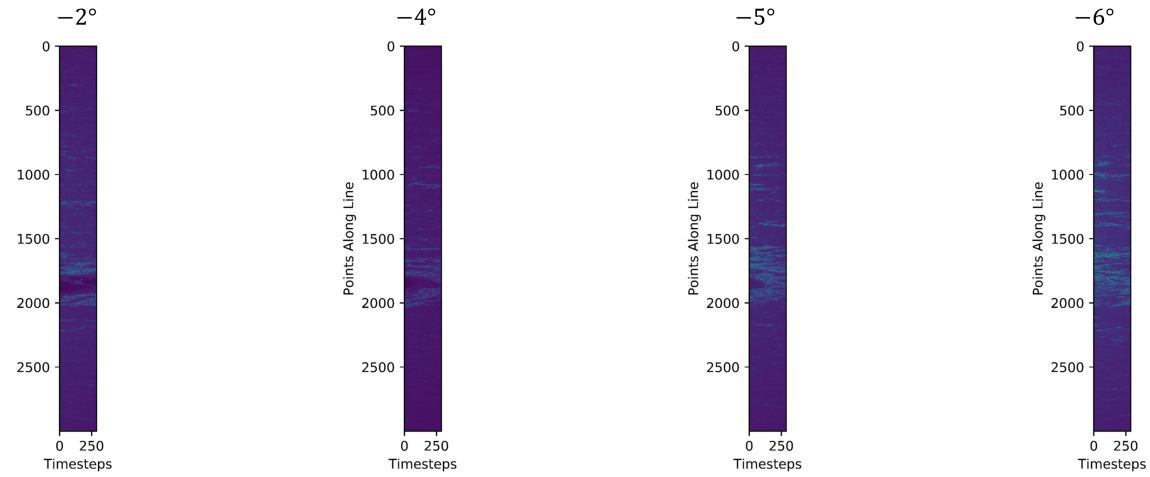


Figure 40: History maps derived for the longitude lines that are shown in Fig.39. The Sunspots structural evolution in time is visible in all the provided lines.

3.6 The Results

The scientific output of this work can be summarized as follows. In this research work:

- A python code that downloads and organizes SDO data, accessed from Helioviewer client, was implemented. This tool was used to create the input of the CNN model.
- A series of data enhancing, cleaning and integrating routines were used to adjust the CACTUS and LASCO catalogs in order to use them as output for the presented pre-processing tool and for the A.I. model.
- Multiple CNN models that use state-of-the-art techniques were deployed for the prediction of the enhanced catalogs. The best models derived original and promising results, predicting 76.6% of the CMEs in CACTUS catalog.
- A similar CNN model was able to distinguish between halo and non-halo CMEs with an accuracy of 83.5% as described in LASCO catalog.
- A new type of input ("history maps") was conceptualized and implemented by a pre-processing tool that uses the VSO client to download SDO data, and the SunPy library to operate on them.

I tested both helioviewer and VSO client for downloading and processing SDO data. Both clients provide many options regarding the instrument choice and the resolution options. Helioviewer however, only provides access to slightly pre-processed .jpeg files. On the other hand, VSO provides access to the full scientific .FITS files that include the highest possible resolution (4096x4096).

The best machine learning prediction results were derived when using solar images at $\lambda = 171$ [Å]. Using $\lambda = 191$ [Å] resulted in a slightly lower accuracy ($\sim 1-5\%$ less accuracy depending on the model).

There were no significant differences between the CNN results using LASCO and CACTUS catalog. Both catalogs performed similarly when used as output files in a CNN that uses pure SDO images as input.

The Adam optimizer showed the best results in all model architectures when compared with other known optimizers such as Stochastic Gradient descent and RMS. This result is in direct agreement with results that were shown in recent research as described by Kingma et al. (2014), [66].

Batch normalization is an extremely useful and vital tool for a model that wants to reach the highest possible accuracy. The implementation on every single layer was the main reason that this modeling was possible. The result of this work is once more in full agreement with most neural network applications that are implementing state-of-the-art techniques as shown by Ioffe et al. (2015), [59].

Optimal batch size and epoch number varies a lot from experiment to experiment. In the case that was described, a batch size of $10-30$ inputs and $15-25$ epochs seemed to provide the best result.

The prediction accuracies derived are promising and may contribute to the already steadily increasing literature on machine learning space weather forecasting models. So far most of the models focus on solar flares ([61], [42]) or on geomagnetic indices prediction ([124], [125]). The CNN that was implemented in this project filed a missing gap of preliminary CME prediction results.

For the pre-processing tool, a wavelength of $\lambda = 1700$ [Å], was used for the motivation, validation and visualization. These images, taken from the Helioseismic and Magnetic Imager (HMI) of SDO are important to validate the accuracy in following certain magnetic structures such as sunspots.

The output of the pre-processing tool, apart from its standalone scientific value has not been tested in combination with the CNN model. However, after deriving optimistic results from using pure SDO data, it is speculated that it may provide even more spectacular results if fully utilized. Of course, such implementation is extremely complex and far above the level of a Master thesis and its strict finishing time schedule. The proposed CNN that uses the pre-processing tool as the input generator of the network can be visualized in Fig.41.

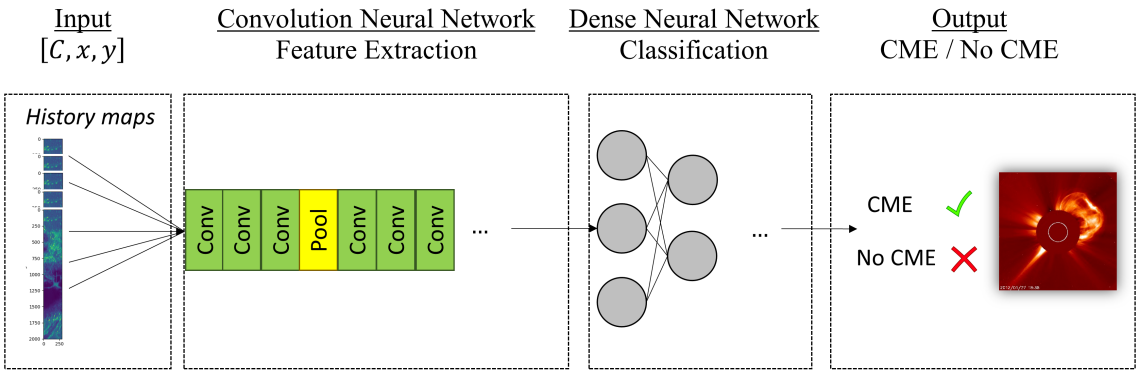


Figure 41: Diagram of the CNN that is proposed to be combined with the pre-processing tool of this work.

Currently the pre-processing tool requires more than 10 days to download, process, validate and organize 1 year of SDO data, for the use of CACTUS/LASCO catalog as output, for a 2 [h] prediction window.

3.7 Conclusions & Further research

One of my suggestions for a future work is splitting the Sun into different areas. This procedure is already available in the pre-processing tool but there is space for improvement. By separating the Sun into distinct areas and training a neural network to each area individually, it may be possible to derive multiple CME predictions. This technique might be important to find several CME occurrences at the same time but also to increase the performance of the CNN by providing a more distinct spatial pattern for each CME event.

On the data enhancement process of the output, even though the method of cleaning and optimizing the pre-existed databases of CACTUS and LASCO was successful, a lot of improvement can take place. It is shown throughout this project that a more in depth analysis of the CME occurrences and a side by side comparison of the differences between these two catalogs is required. Machine learning algorithms can be as good as the data that is provided. An inexact definition and classification of CMEs can limit significantly the possibility of creating a fully functioning machine learning prediction model. On the choice between the LASCO and CACTUS catalog, although both had several issues as previously discussed, CACTUS seems to be more consistent. I believe that with the correct pre-process and data cleaning procedure CACTUS might be more successful than LASCO as a possible output of an A.I. forecasting model.

Regarding the choice between helioviewer and VSO client, even though helioviewer might be easier to use for image processing, providing data at .jpeg. It should not be thought as the best choice. The not explicitly explained pre-process done by the team behind helioviewer might have an effect on the prediction capability of a Neural Network therefore careful investigation is advised. VSO providing the original scientific data should be a more straightforward choice for similar kinds of work, assuming an extensive procedure for the pre-process of the data is done.

It may also be beneficial to combine the CACTUS and LASCO catalogs with other measurements (e.g. GOES x-rays). The result might provide an interesting output dataset that may prove important if the model deals with a regression problem, for example, predicting the characteristics of a CME.

Regarding the neural network implementation, more work can be done to further optimize the architecture of the CNN. However, it should be noted that all the training of the neural networks was done using extremely expensive equipment that is provided by the VSC supercomputer facilities. Any further optimization that include more complex architecture will most likely require a similar facility for the training and the validation of the network.

An interesting addition might be to implement specific pooling layers that act on some channels of the input but not on all. This process can ensure that both highly invariant and low invariant features will not under-fit and it might be extremely useful if multi-area forecasting is to take place. This approach is theorized in the work of Szegedy et al. (2015) [116].

Regarding the choice of CNN for forecasting, although it is a fully justified choice, there is evidence that other topologies might be more suitable. To be more precise, I believe that the combination of CNN and LSTM neural networks may have better results, especially if it is decided to use data of several hours or even days before the CME event.

The pre-processing tool might prove to be an important addition not only in the case of multi-CME prediction but also in significantly decreasing the memory and computation requirements of the model. The fact that every dimension of the input is adjustable might assist in making the input of the model more flexible. Furthermore, the ability to adapt the input dimensionality may be a valuable asset in predicting phenomena of different time and space scales such as solar flares, sunspots, etc.

Regarding the advancement and the further improvement of the proposed pre-processing tool, there are a few additions that can increase its performance and scientific output. In specific, a more strict debugging procedure that is capable of spotting contaminated data while they are being downloaded or processed should be implemented. A multi process implementation will also be of great use in order to decrease the computational time required so that the tool may be applicable for even longer periods of time or for a longer prediction window.

Finally, if anyone is interested in either re-producing these results or further developing any of the mentioned codes (download/machine learning/pre-processing tool), they may request the latest version by sending an email [11] to the author of this work.

To conclude, I hope that this study will stimulate further investigation in the field of CME prediction and classification. This problem remains open and unsolved but also challenging and fascinating. It is with collaboration and determination that similar problems have been solved in the past and I expect this topic to be no exception.

[11] SavvasRaptis@gmail.com

APPENDIX A

SOLAR & SPACE PHYSICS MATERIAL

In this Appendix, there are some information about solar and space physics that are either too technical to be included in the main text or not directly related to the main topic. This Appendix acts as a supplementary material to Chapter 1

Sun's Magnetic Field & Babcock's Theory

The Babcock's model is the standard model explaining the magnetic field rotation and the sunspot pattern that is observed on the Sun every 11 years. The theory was first proposed by George Ellery Hale[1] and was further developed by Horace W. Babcock [2] and Robert B. Leighton[3] [6]. A summary of the Babcock-Leighton dynamo model can be visualized in Fig.42 and Fig.43

Although Babcock theory is successful at explaining observed phenomena, there have been several more complex models that try to produce even more accurate results. Some example that the reader can find in the bibliography are the model developed by Dikpati et al. (2006) [35] and the Sun's kinetic model developed By Wang et al. (2005) [120].

The magnetic field lines observed in the Sun can be grouped in 3 different components, as shown in Fig.44.

[1] American Solar Astronomer [1868 - 1938]
[2] American Astronomer [1912 - 2003]
[3] American experimental physicist [1919 - 1997]

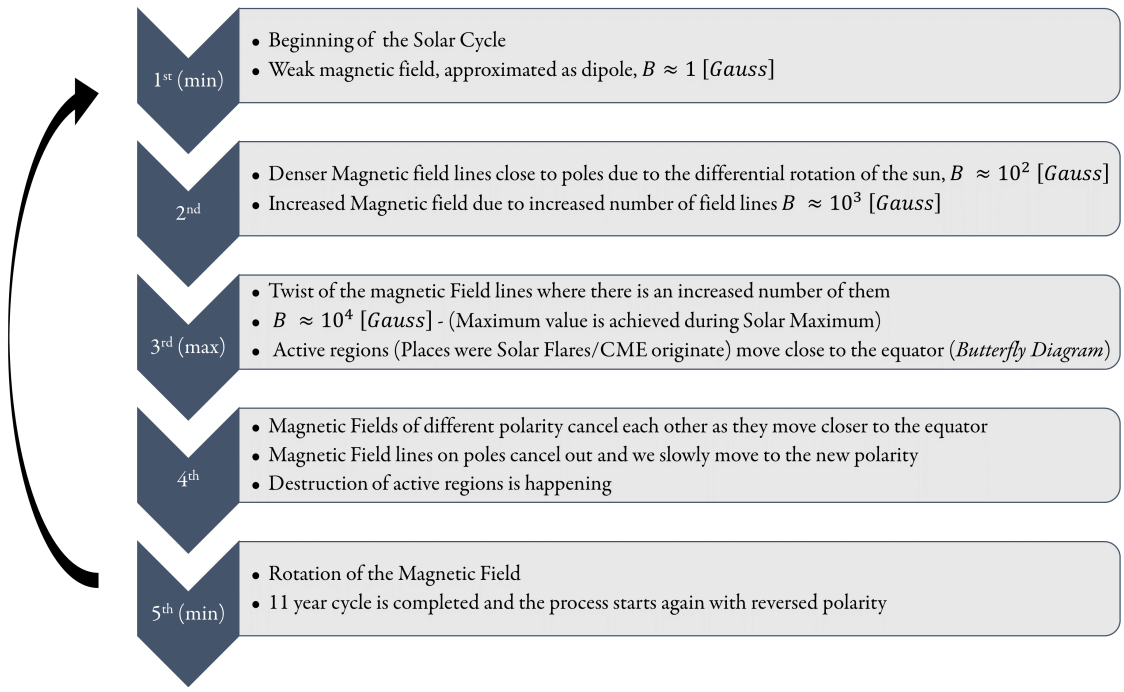


Figure 42: Summary of the steps describing Babcock-Leighton dynamo model

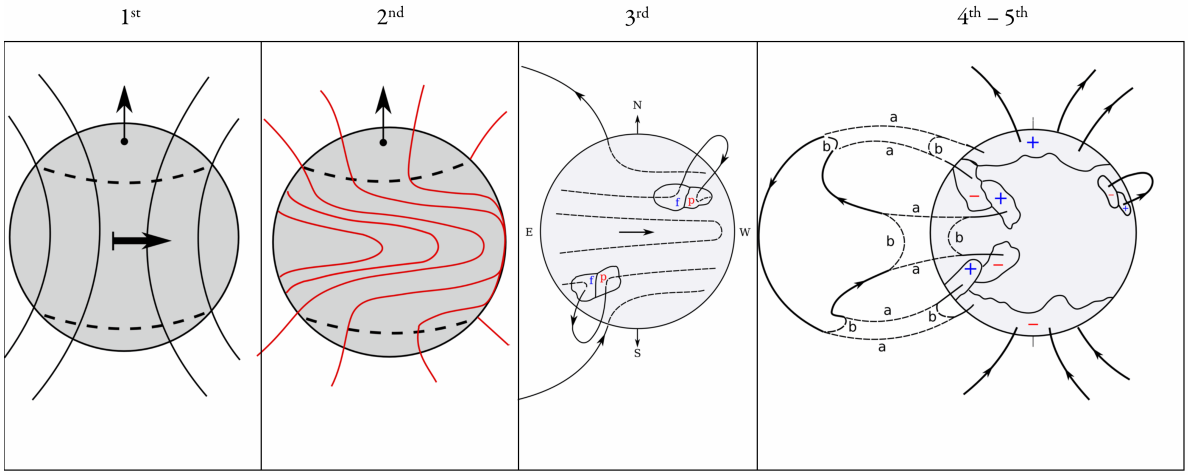


Figure 43: Sun's magnetic field during every stage of the Babcock theory as described in Fig.42. The pictures of the Sun are a compilation from the work done by Costas Alissandrakis et al. (2015)

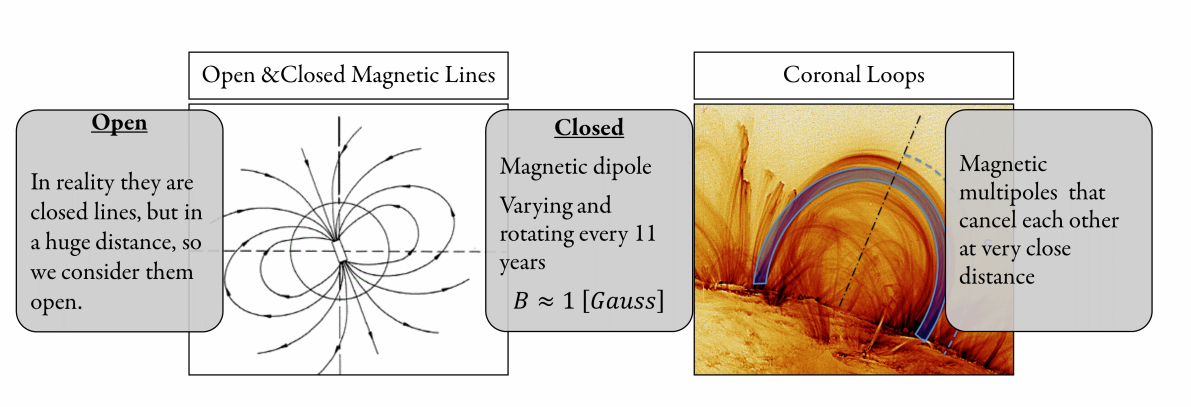


Figure 44: The three components of Sun's magnetic field. [Left], Figure Courtesy: `http://solar.physics.montana.edu`. [Right], Figure courtesy: Fabio Reale (2014), [95].

Magnetospheres & Planets

A very interesting magnetic construct that is vital for space weather is the Magnetosphere of a planet. Almost all planets (apart from Mars and Venus) have a magnetosphere around them [4].

A whole analysis on the relevance of the magnetosphere for the phenomena of space weather has been given in Chapter1. The magnetic fields of other planets are tilted in their own unique way [67]. The basic characteristics of each magnetosphere are shown in Table 13.

Characteristic	Mercury	Earth	Jupiter	Saturn	Uranus
Distance [AU]	0.31 - 0.47	1	5.2	9.5	19
Solar wind density [amu cm^{-3}]	35-80	8	0.3	0.1	0.02
Radius [km]	2.439	6.373	71.398	60.330	25.559
Surface magnetic field [Gauss]	3×10^{-3}	0.31	4.28	0.22	0.23
Magnetosphere size [R_{Planet}]	1.4-1.6	8-12	50-100	16-22	18

Table 13: Characteristics of some planets and their magnetosphere. Data were taken from Kivelson et al. (2014) [67].

The SW's interaction with every celestial object may cause a magnetic field with similar structure like the magnetosphere that forms on Earth. Research on the structure of the magnetic field and its variation due to solar wind include satellites of planets and comets [67].

Cosmic Radiation in Space Weather

A significant contribution to space weather phenomena originate from Cosmic Radiation (CR). **Cosmic Rays** consist of highly energetic particles originating from sources that are not in

[4] A very nice collection of artistic representations can be found in the following link : `http://lasp.colorado.edu/home/mop/resources/graphics/graphics/`

our solar system. The energetic particles have a gyro radius, which shows their origin to be from another galaxy. When referring to cosmic radiation, Solar Energetic Particles (SEPs) that originate from the sun and Galactic Cosmic Rays (GCR) that originate from other sources are included [11].

CRs are divided into two major groups. The primary and the secondary CR. **Primary** is the radiation that comes directly from a source. **Secondary** is the resulted radiation of the interaction between the primary cosmic rays and Earth's elementary particles that reside in its atmosphere. The result of these interactions cause the effect known as "Cosmic/Electromagnetic Shower" (Fig.45), named after the highly energetic photons that are produced [87].

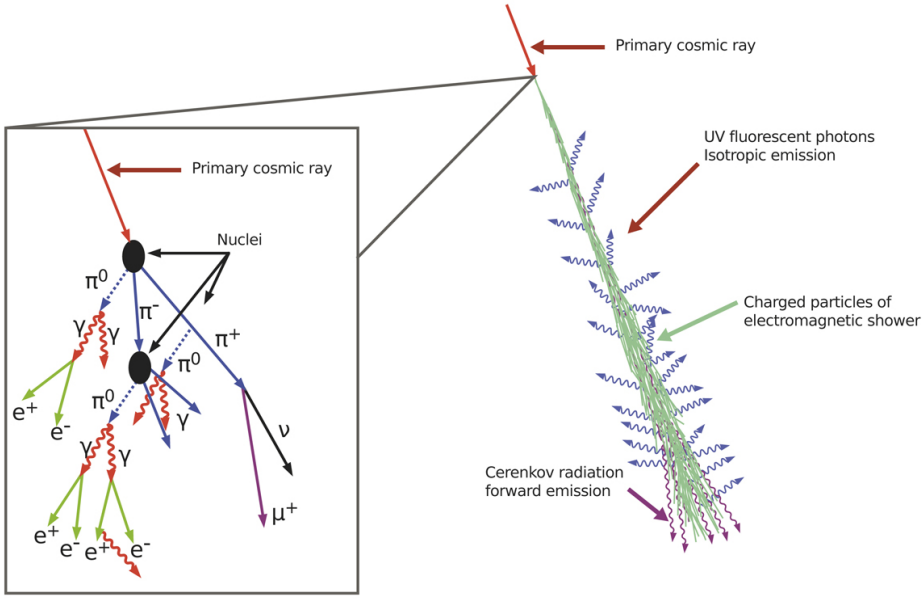


Figure 45: Illustration of the creation of secondary CR. A visualization of pions (π) that later on generate other particles along with highly energetic photons (γ) is shown. Figure Courtesy: Bietenholz, Wolfgang (2013) [13].

A very interesting phenomenon that can be observed is the relationship between the solar cycle and the observed cosmic rays. High sunspot number (solar maximum) correlates with low cosmic ray intensity. This phenomena is happening because during solar maximum, the enhanced solar wind that propagates outwards of the heliosphere is weakening the CRs that arrive from sources outside of our solar system [93]. CRs are studied in the framework of space weather due to their enhancement of the SW but also due to the consequences that they can directly have (Appendix B). Svensmark et al. (2000) has shown that CRs have a direct effect on Earth's climate and in cloud formation [114].

Particles in a magnetic field

Assuming a particle of mass m in an electromagnetic field, its equation of motion can be written as:

$$m\frac{du}{dt} = q(\vec{E} + \vec{u} \times \vec{B}) \tag{A.1}$$

Now, under the assumption that there is no electric field one can re-write the above as:

$$m\frac{du}{dt} = q(\vec{u} \times \vec{B}) = \vec{F} \tag{A.2}$$

Investigating both examples of parallel and perpendicular velocity to magnetic field, one can easily derive that:

For a velocity perpendicular to the magnetic field:

$$r = \frac{\mu_{\perp}}{qB}$$
$$\omega = \frac{q}{m}B$$

Which shows that the force acts as a centrifugal force, and therefore creates an equation of motion:

$$\vec{\omega} = \frac{q}{m}\vec{B}$$

Now if there is a velocity parallel to the magnetic field, one can simply [5] derive:

$$u_{\parallel} = cst.$$

By combining the two components, the gyromotion of the particle is derived, which can be seen in Fig.46 (a). It basically is a spiral motion around the magnetic field line with the direction depending on the charge of the particle (q).

Considering now the case were the magnetic field is increasing along the propagating axis, the following aspects arise:

- As the magnetic field grows the gyroradius is getting smaller
- The parallel velocity is getting smaller as $u = \frac{F}{q\sin(\theta)}$

The resulted trajectory is the so called "magnetic mirror" (Fig.46 (b). It can be proven that there is a critical point in which the magnetic field is so extreme that the particle bounces and does the opposite movement [86]. This motion was shown in Chapter 1 when discussing the movement of particles inside the magnetosphere of the Earth as they move towards the poles.

[5] Using the fact that the cross product of two parallel vectors is always zero.

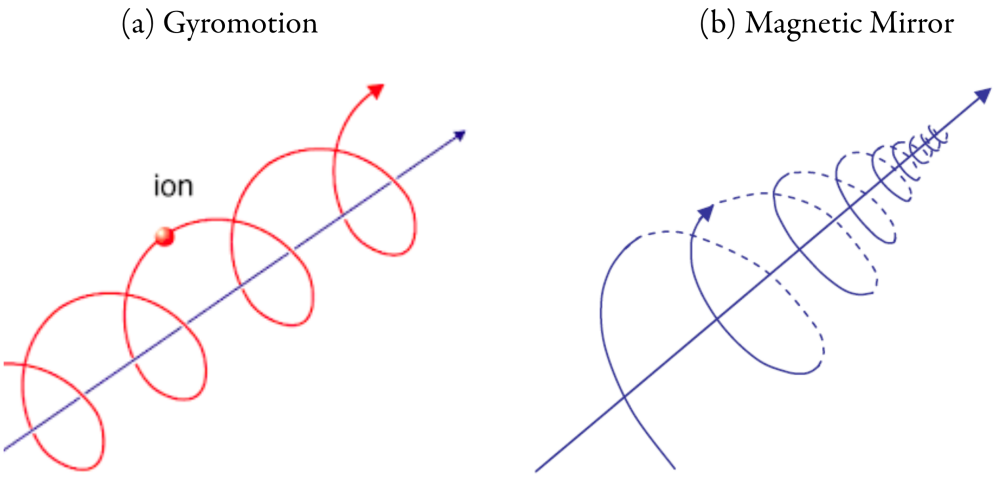


Figure 46: [**Left (a)**] Visualization of gyromotion movement. [**Right (b)**]: Visualization of the magnetic mirror phenomenon. Figure Courtesy: Panagiota Preka-Papadema

Overview of Coronal Loops & Magnetic Reconnection

In this section, the phenomena of magnetic reconnection & coronal loop are introduced. Their importance lies to the evidence that CMEs and solar flares originate from them [81].

Coronal Loops

A visualization of how coronal loops look like was shown in Fig.44. But what exactly is a coronal loop?

Coronal loops are magnetic structures, they are flux tubes formed in Sun's Corona, where extremely hot plasma is confined. It is believed that these magnetic structures are the result of twisted magnetic flux originating from the Sun. They usually form above sunspots and are considered to be the main magnetic construct that is connected to solar flares and CMEs. It is a magnetic construct in which magnetic reconnection phenomenon takes place, producing the energetic phenomena that are observed at the Sun [110].

Magnetic Reconnection

Magnetic reconnection [6] can be considered as a type of electromagnetic acceleration. It is the resulted phenomenon of Lorentz force acting on a specific magnetic topology.

[6]Magnetic reconnection, exists on various places, such as in the interaction of solar wind with Earth's magnetosphere and in fusion experiments.

The magnetic topology that most magnetic reconnection models are using is called (non-symmetric) "X-point" due to its shape (Fig.47).

In this topology, there is a magnetic field:

$$\vec{B} = y\hat{x} + \alpha^2 x\hat{y}$$

Where, $\alpha^2 > 1$. The non-symmetric hyperbolas create field lines of:

$$y^2 - \alpha^2 x^2 = C$$

The Lorentz force is described then as :

$$\frac{\mathbf{J}\times\mathbf{B}}{c} = \frac{1}{4\pi}(\nabla\times\mathbf{B})\times\mathbf{B} \tag{A.3}$$

Which can be re-written as:

$$\frac{\mathbf{J}\times\mathbf{B}}{c} = -\nabla\frac{B^2}{8\pi} + \frac{\mathbf{B}\cdot\nabla\mathbf{B}}{4\pi} \tag{A.4}$$

As a result, using the above, for the magnetic topology of "X-point":

$$\frac{\mathbf{J}\times\mathbf{B}}{c} = \frac{\alpha^2-1}{4\pi}(-\alpha^2 x\hat{x} + y\hat{y})$$

Therefore, there are two components of the Lorentz force. An inward for x axis and an outwards for y.

$$y = 0 \Rightarrow \frac{\mathbf{J}\times\mathbf{B}}{c} = -\frac{\alpha^2(\alpha^2-1)}{4\pi}x\hat{x}$$
$$x = 0 \Rightarrow \frac{\mathbf{J}\times\mathbf{B}}{c} = \frac{\alpha^2-1}{4\pi}y\hat{y}$$

It has been shown therefore, that this magnetic topology can accelerate the plasma, due to Lorentz force, inwards in the x direction and outwards in the y.

If the Lorentz power is not neglected by the plasma pressure, the creation of a current sheet might occur, which is described as a flow of plasma and a pile up of magnetic energy. This pile up is followed up by the transformation of the magnetic energy into kinetic and thermal.

Qualitatively speaking, let's consider plasma with little resistance, enough to make diffusion phenomena significant. When such a plasma area is interacting with a magnetic field line, the areas split and the area outside of the diffusion region move outwards of the structure. When these areas move outwards they will meet other field lines and then, the "cut" and "reconnect" phenomena occurs from which the whole structure was named after. A visualization of this phenomenon is shown in Fig.48.

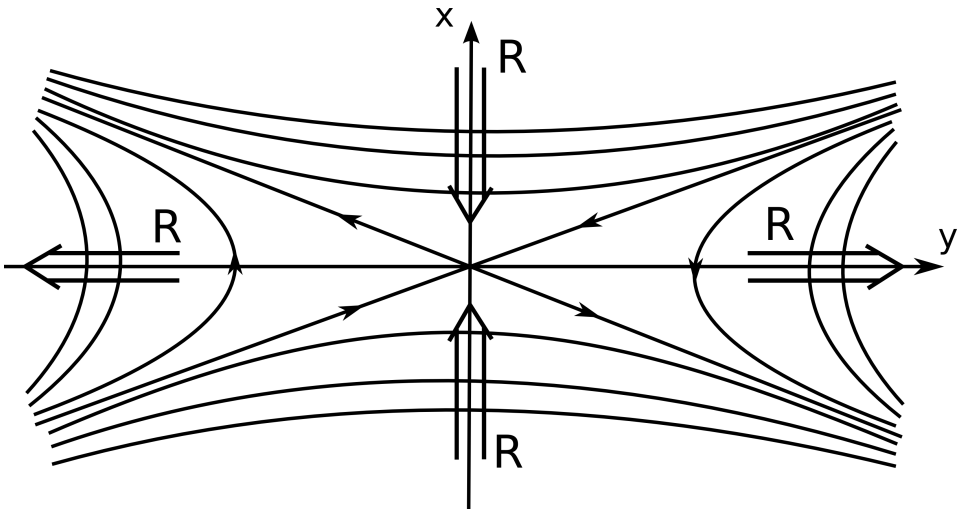


Figure 47: Visualization of the magnetic topology "Non symmetric x-point". Figure Courtesy: Costas Alissandrakis et al. (2015)

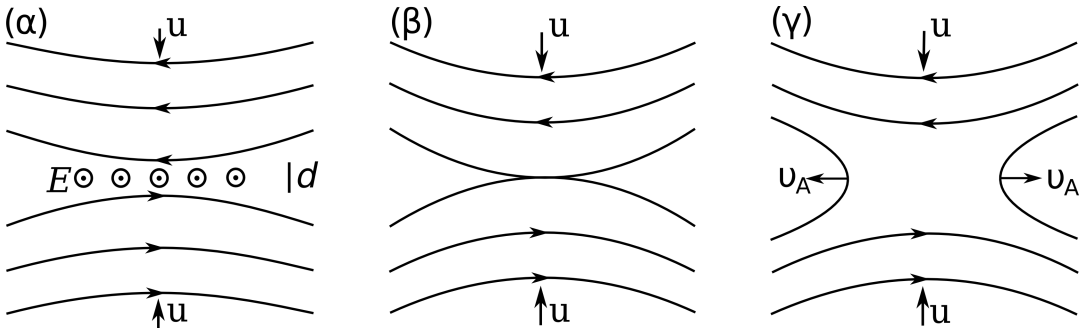


Figure 48: Schematic representation of the magnetic reconnection standard model. Figure Courtesy: Costas Alissandrakis et al. (2015)

Sweet - Parker model

In this example, the topology of the magnetic field lines is that of the "X-point", proposed by Peter Sweet [7](1956) and later developed by Eugene Parker, and therefore named "Sweet-Parker model".

It can be proven, as shown in the textbook of Fitzpatrick (2014) [41], that the speed of the plasma going inside the diffusion zone is:

$$u = u_a \sqrt{\sqrt{2}/R}$$

Where, u_a is the Alfven speed in that area, and R is the Reynold magnetic number:

$$R = \frac{4\pi u_a L}{\eta c^2}$$

[7]British Astronomer, (1921–2005)

Where, L is half the length of the diffusion area, η is the conductivity parameter and c the speed of light.

Furthermore, it can be shown that the magnetic reconnection is happening at a rate [41]:

$$r \sim \frac{1}{\sqrt{R}}$$

In the case of Sun's corona, the rate of magnetic reconnection can be found to be $\sim 10^{-4} - 10^{-6}$ which provides:

$$u \approx 0.01\,[\text{km/s}]$$
$$u_a \approx 1000\,[\text{km/s}]$$

This means that if the Sweet-Parker model was fully accurate, phenomena such as solar flares would have a time scale of several weeks. However, as discussed in Chapter 1, they can have a time scales of minutes.

Petschek model

Another model[8] was proposed by Petschek in 1964 [90]. Its main idea is that there is a shorter length (L). This result in a much more rapid rate of magnetic reconnection (three times faster):

$$r \approx \frac{\pi}{8 \ln R}$$

Similar calculations as before provide:

$$u \approx 10 - 20\,[km/s]$$
$$u_a \approx 1000\,[km/s]$$

Finally, there have been tries to develop 3D models of magnetic reconnection that compared to the 2D version, require a more complex magnetic field. Some works on 3D magnetic reconnection has been done by Anna G Frank (1999) [43] and more recently by Miho Janvier (2017) [60].

Example of Flare and CME models

In principle, flares & CME models try to address the following questions:

1. How does the magnetic energy pile up.
2. How does the energy gets transformed and released in the observed way.
3. In the CME case, how magnetic field lines open in such a way that they let material escape.

[8] Also called "Fast magnetic reconnection" model

The basic idea of most CME and solar flare models is that the magnetic field lines can somehow get distorted and create a pile up of magnetic energy. In these models, magnetic reconnection is utilized to simulate the conversion of magnetic energy to kinetic and thermal.

A typical model for the combined phenomenon of solar flare and CME was developed back in 1960 - 1970 and was modernized by several scientists in 1990 [55].

The model (Fig.49) can be summarized as [73]:

1. The geometry has some kind of co-rotated magnetic field (e.g. flux rope)
2. For some reason (e.g. Magnetic instability) flux rope is going upwards in such a way that some magnetic field lines are antiparallel.
3. As the flux rope is going upwards, some field lines close each other and a current sheet[9] is developed.
4. Instabilities caused in the current sheet begin the phenomenon of magnetic reconnection
5. A solar flare and a CME is developed at the place where the magnetic reconnection occurred. Plasma and magnetic flux is ejected and kinetic energy is added to the ions that reside in the hot propagating plasma.

Common criticisms of this model is that the magnetic field is treated as dipole which is not true for the Sun. The fact that it was made in two dimensions and its requirement of enormous current sheets to be developed in order to generate the amount of energy that is observed.

Another model, developed by Lynch et al. (2008) [76], in which the magnetic reconnection phenomenon has a more leading role can be seen in Fig.50. The steps of this model are:

(a) The main magnetic topology is a series of distorted magnetic loops.

(b) Due to secessional motion from the east to the west, the magnetic pressure in the central magnetic field lines is increasing.

(c) Magnetic reconnection is occurring.

(d) Magnetic reconnection will "open" the magnetic field lines and therefore assist the main magnetic structure to have a considerable increase in magnetic pressure, leading to the explosion that cause the flare/CME phenomenon.

[9] Dense magnetic field regions with an exceed of magnetic energy.

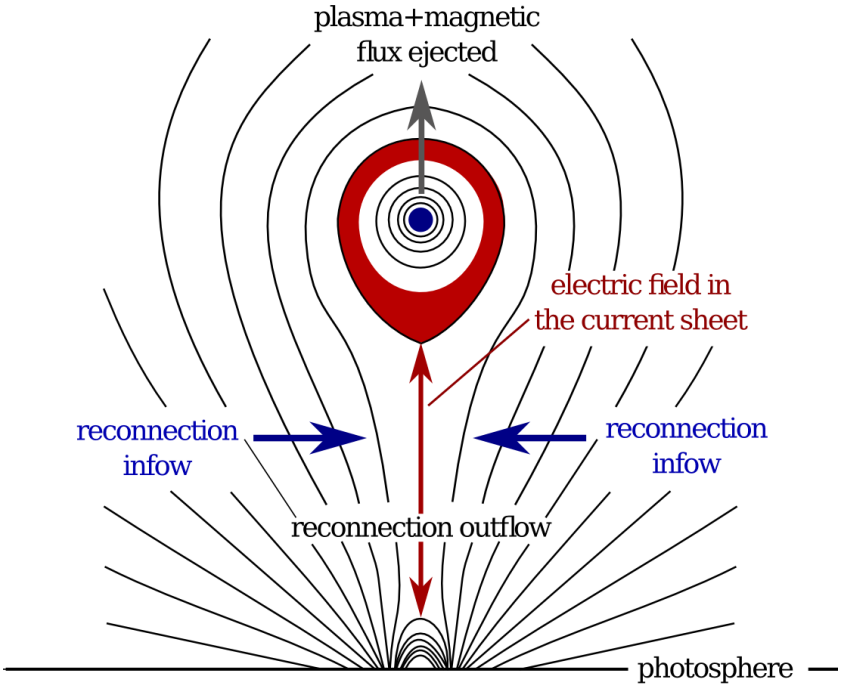


Figure 49: The typical model of Solar Flare-CME. Figure courtesy: Lin et al. (2000) [73].

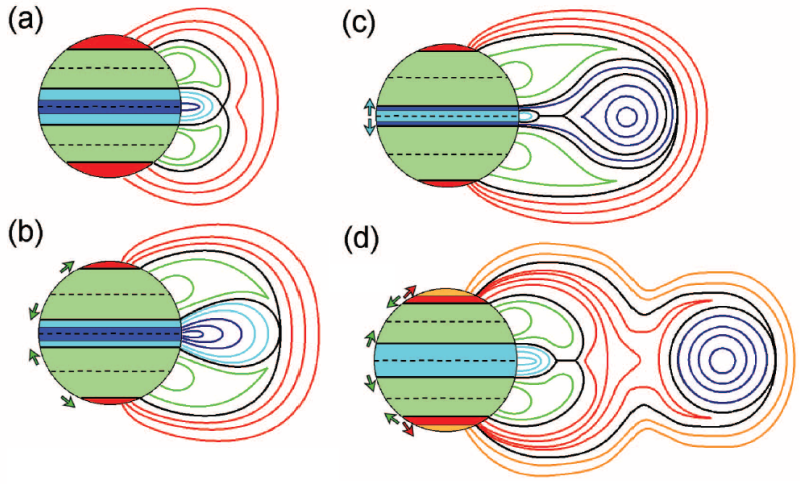


Figure 50: Alternative model for the CME creation. Figure courtesy: Lynch et al. (2008) [76].

APPENDIX B

EFFECTS OF SPACE WEATHER & PREVENTION

In this Appendix the reader can find more information regarding the effects of space weather and the methods that are implemented to prevent the disastrous effects caused by Sun's activity.

Effects of Space Weather

Effects on infrastructures

A very important effect of geomagnetic disturbances is the impact on groundbased infrastructures, like power grids and railway systems. Since the power grids hold a high voltage network, the currents produced by geomagnetic storms (GICs) can vastly damage the network possibly leading to its full destruction. Another infrastructure that can be affected is the pipeline network, which can experience corrosion from the produced GICs [91].

As a historical overview, there are two very interesting events that happened recently. The "1989 Quebec Blackout" and the "2003 Halloween Solar Storms". The first caused the whole Hydro-Québec transmission system to breakdown [16]. During this event a blackout that affected more than 6 million people occurred and it took more than 9 hours for a full recovery of the system. The second event, was a series of major solar flares happening during the solar maximum period that had a severe effect on satellites, grids and airlines routes [91].

Space weather phenomena that produce GICs can result in the following infrastructure related hazards [91]:

- Transformer saturation
- Reactive power losses

- Transformer & Generator overheating
- Pipeline degradation
- False signals on railways

Effects of Space Missions

In space, intense SW can damage satellites and their equipment. In specific, energetic particles can cause "Single Event Effects" (SEE) that can be seen as frozen bits, NaN values of measurements or even the total destruction of MOSFET [1] transistors [30]. SEEs can be classified in three categories [9].

1. **Single Event Upset (SEU)** is defined as the error caused by a highly energetic SPE or Cosmic Ray particles passing through microelectronic circuits producing electron-hole pairs during their propagation. Necessary action to recover the equipment to either reset or switch-off the device is required.

2. **Single Event Latch up (SEL)** is a phenomenon triggered by protons, neutrons and heavy ions. It can be destructive and the device may be recovered by a power off-on reset. Compared to SEU, it is strongly dependent on temperature and if the power is not shut down quickly, the equipment may receive a fatal failure.

3. **Single Event Burnout (SEB)** is a phenomenon triggered by protons and neutrons causing a permanent failure of a device.

SELs and SEBs are the most disastrous phenomena that is necessary to avoid. These events require careful design both at operation and design phases since there are no techniques available to correct their effect in the scientific equipment [9].

For a spacecraft or a space application, one or more of the following requirements must be fulfilled [30]:

1. Derive accurate characteristics of the space environment using correct statistic measurements.
2. Long-Term prediction of Space environment in order to design the spacecraft.
3. Short and Medium term forecasting of the Space environment for arranging the necessary operations.
4. Evaluating in real time if the original specifications are sufficient or reactions are required.
5. Data gathering of possible abnormalities and categorizing carefully the data for in-depth analysis.

[1] metal-oxide-semiconductor field-effect transistor

Effect on Humans' Health

Apart from the effects on space missions and on electrical devices, energetic events in space also have an immediate effect on biological matter.

Exposure to energetic particles and CR may also be harmful to humans. The radiation received even inside the magnetosphere of the Earth can be hazardous since there is a direct correlation between ionizing radiation from high energy particles and deadly health problems such as cardiovascular problems, mutation and cancer cell growth [2]. This is at the moment, one of the biggest problem regarding the inhabitation of Mars and other long duration space missions that include human transportation [33].

To calculate the dose of radiation, researchers use the unit Gray[2] : [Gy] = $\frac{[\text{Joule}]}{[\text{kg}]}$. However, when referring to the effect on biological matter another unit is usually used, Sievert[3] [Sv]. Sievert is called "equivalent dose" and even though it has the same physical unit, it takes under consideration the effects that radiation has on different tissue.

For example, 1 [Gy] of gamma rays is 1 [Sv]. However, 1 [Gy] of alpha rays is 20 [Sv] since the alpha radiation is much more dangerous if the source has already penetrated the target. The reason alpha particles are more dangerous is because they can be more easily absorbed by humans cells [21].

Even in an LEO (Low Earth Orbit), like the one in the International Space Station (ISS), a crew member can receive up to 1[mSv] per day [27], which is approximately 10 chest X-rays or 1/5th of maximum recommended yearly dose, taken daily [72].

An example of the amount of [mSv] received by humans in different situations is shown in Fig.51. It should be noted that a dose of 100 [mSv] increases the possibility of cancer development significantly [92], while a dose close to 1000 [mSv] is fatal in most cases [45].

Aircraft Safety

The crew and the passengers of an aircraft are also in immediate danger of space weather effects. The closer to the upper atmosphere the less protection there is from all the various phenomena that contribute to radiation.

The cosmic ray radiation dose increases until up to ~ 22 [km] (66.000 [ft]). Furthermore, the closer to the poles, the more radiation received since these are the least protected areas of Earth's magnetosphere. The crew of an airplane flight may receive an annual dose of several [mSv], which puts them in a similar occupational exposure to nuclear workers and miners of radiative environments [8].

[2]Named after the English Physicist, Gray, Louis Harold (1905–1965)
[3]Named after the Swedish Physicist, Rolf Maximilian Sievert (1896 – 1966)

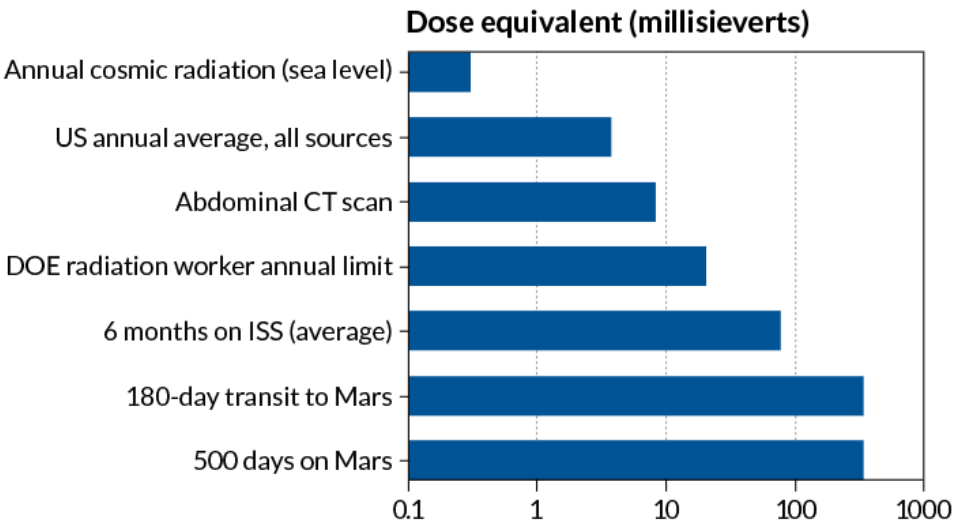


Figure 51: Radiation received by Humans in various circumstances. Figure Courtesy: JPL-CALTECH/NASA, SWRI.

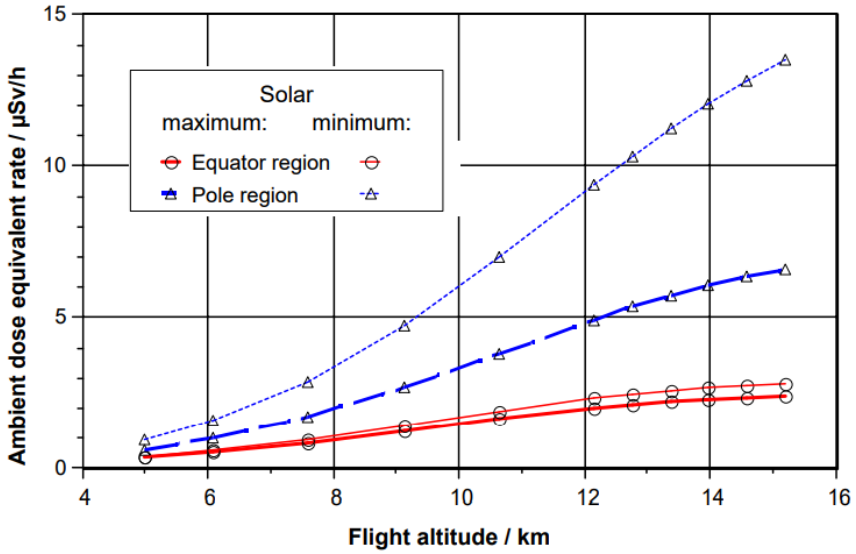


Figure 52: Ambient dose calculated for different inputs. Solar maximum (1990) is represented using thick lines and solar minimum (1998) using thin ones. The calculations that produced this graph were made with EPCARD software and they show results for both aviations close to the equator and to the poles. Figure Courtesy: CORDIS (Community Research and Development Information Service), European Commission.

As shown in Fig.54, the dose received is significantly more during solar minimum. This is due to the relationship between the solar cycle and the cosmic rays, explained in Appendix A.

Traveling at high altitudes offer many advantages especially for long flights. When a plane is

going higher in latitude, the air becomes thinner. As a result, the airplanes can travel faster using less fuel and therefore costing less money per flight. Furthermore, to avoid turbulence the airplane should preferably reside above the stratosphere (> 33.000 [ft]) since there are no clouds there or strong instabilities that cause turbulent effects [37].

Monitoring of Energetic Particles

To avoid all the previously mentioned possible disasters, several methods have been developed as counter-measurements. One of the simplest yet effective method is the use of small and cheap monitors, that act as a safety measurement for the more expensive scientific instruments.

An example of such device is SREM (Standard Radiation Environment Monitor), developed by ESA, which is on board of the missions PROBA – I, Rosetta, Planck, INTEGRAL and Herschel. Although SREM is also used for scientific research, it mainly acts as an alarm for other valuable devices included in these missions. Its main goal is to signal an alarm when it detects radiation above a certain threshold so that the other devices can be switched off [106].

SREM is a detector that was designed to measure e^- with energies $E > 0.5$ [MeV] and p^+ with energies $E > 10$ [MeV]. SREM is able to provide consistently good in quality measurements to describe the radiation environment in space while being minimal intrusive for the rest of the space equipment [39].

The advantage of using such a monitor is that it is generally a small addition that takes little to no space on the spacecraft while at the same time its installation is straightforward and requires little effort [106].

Other monitors that are used are:

1. REM (Radiation Environment Monitor, 1994 - 1998)
2. XMM-Newton Radiation Monitor (2000)
3. NGRM (Next Generation Radiation Monitor, 2014)
4. HMRM (Highly Miniaturized Radiation Monitor, 2015)
5. EMU (Environment Monitor Unit, 2017)

Of course, these monitors although extremely useful. They are still susceptible to rapid changes in radiation and to malfunctions that may occur.

Forecasting Models

Another more elegant method to prevent the hazardous effects of space weather is the development of forecasting methods. This is a much more complicated and extensive work compared to

the use of monitors. However, it is by far the most valuable tool since it can help predicting a disaster before it even happens.

In order to be able to fully protect satellites, systems and humans, it is vital to develop forecasting capabilities for both solar events (flares/CMEs) and SEPs. Even though from a modeling point of view, solar flares, CMEs and SEPs are similar to each other. For a prediction and warning viewpoint, these phenomena are different due to the different time scales that they require to develop and reach manmade objects.

Physical Models

Physics based forecasting models are based on physical laws, observations and well-established theories. These models utilize data driven computer simulation to predict future events.

A widely used heliosphere physical forecasting tool is the WSA-ENLIL. It is a global 3D ideal MHD code which tries to model and forecast the evolution and the characteristics of solar wind.

Another model called ENVIL +cone[4] is also used for the forecasting of CMEs and their effect on solar wind characteristics. The model is quite accurate in finding out when a phenomenon is happening. However, there is usually a significant error in the parameters that are found. Taktakishvili et al. (2009) found the arrival time to be of an average absolute error: ~ 6 [h] [117].

ENLIL+cone model forecasts CME propagation and takes the cone model input parameters at its inner boundary. The cone model is based on the idea that close to the Sun CME propagates with constant angular and radial velocity, and therefore, has the shape of a cone [88].

A result from a ENLIL cone model can be seen in Fig.53. There, one can see a simulation that was created to simulate a CME [5].

Statistical Models

Statistic based modeling is another type of model that appeared in recent years due to the availability of satellite data. Depending on the forecasting phenomena, there is a different justification for its use. In the case of solar flare prediction, the reasoning is that as a global phenomenon, flare event is a time-dependent Poisson process [6] and can be therefore modeled by a statistic process [123].

[4]Developed by D. Odstrcil, University of Colorado, Boulder.

[5]The results of the simulation can be accessed at `https://ccmc.gsfc.nasa.gov/database_SH/Savvas_Raptis_032617_SH_1.php`.

[6]Poisson process uses variables that are useful when counting the occurrence of a specific event that even though it happens at a specific rate, it is in principle random.

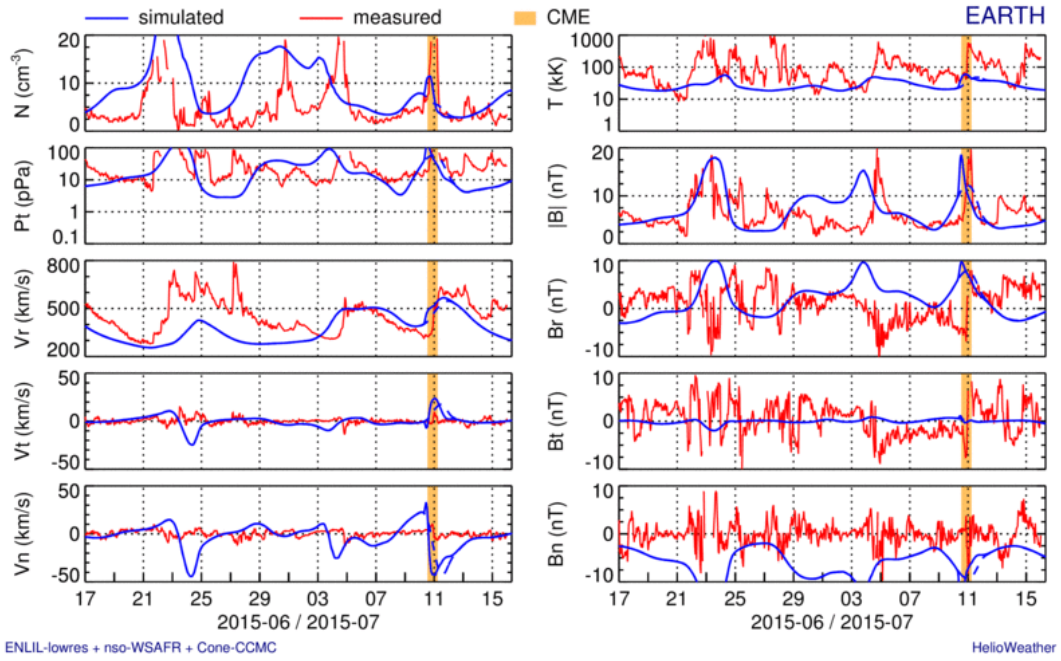


Figure 53: Simulation result using Envil Cone Model showing various parameters shown for 2165 CR. A special reference is done to the period when the CME hits earth to show the fluctuations in the shown parameters

Machine Learning Models

In the last few years, more and more researchers are working towards the creation of a fully functioning machine learning model that can predict and characterize all solar related phenomena.

Some initial tries that were developed by Colak et al. (2009) achieved results that were significantly better than NOAA Space Weather Prediction Center (SWPC) for the period 1999 - 2002 [24].

In space weather people use ML techniques to forecast all kind of phenomena. Most of the research describes tries to accurately predict solar flares ([61], [42]). Some tries have been made to predict if CMEs will originate from active regions [15]), while other works focus on the forecast of various geomagnetic indexes such as DsT and KP [124], [125].

Other Models

Apart from the "forecasting" models, there are two more type of models that are extremely vital for space weather. These being the *"nowcasting"* models and the *"climate* models".

Nowcasting models are models that are used in order to resolve anomalies that were not possible to forecast. These models are used to readjust parameters during a space mission if an emergency happens.

Climate models are models with long forecasting capabilities showing an estimation of the worst case scenario. They are used for the design phase and the mission planning of a spacecraft mission [7]. The field that deals with these models is often called "space climate" since it is the long-term average of space weather. Similarly, to how the climate of Earth's atmosphere is treated [84].

APPENDIX C

SOFTWARE & A.I. LIBRARIES

In this Appendix, there are descriptions and examples on libraries that were used in this research work. All the libraries that are going to be discussed are implemented in the Python language.

It is assumed that certain prerequisite libraries that are used in scientific programming in python such as Scipy, Numpy and Matplotlib are already known and installed.

SunPy

SunPy[1], is a community-developed Python library mainly used in space and solar physics. It provides a data-analysis environment accessible by everyone, consisting all the necessary tools to analyze and visualize heliospheric and solar data. Sunpy is heavily dependent on other libraries such as Astropy, Scipy, etc. [83]. In our project, it was used for various tasks as explained in Chapter 3.

Code Example

An example of basic SunPy code that used for the project can be seen below. This code was the first code that was created (version 0.1) which was later developed into the main pre-process tool that is described in Chapter 3.

[1] http://sunpy.org/

CODE:

```
"""
@author: Savvas Raptis, SavvasRaptis@gmail.com
Machine Learning Project, KU Leuven, Centre of Plasma-Astrophysics
"""
from __future__ import print_function, division
from sunpy.time import parse_time
from datetime import timedelta
from sunpy.net.helioviewer import HelioviewerClient
import matplotlib.pyplot as plt
import astropy.units as u
from astropy.coordinates import SkyCoord
import sunpy.map
import sunpy.data.sample
from sunpy.physics.differential_rotation import solar_rotate_coordinate


hv = HelioviewerClient() # initialize the helioviwer client
def GetAllDataOfSun(y,x,date,total):

    date = parse_time(date)
    total=timedelta(days=total)

    hpc_y = u.Quantity(y, u.arcsec)
    hpc_x = u.Quantity(x, u.arcsec)
    past_date = date - total
    # Download the image of the initial date
    filepath = hv.download_jp2(date, observatory='SDO',
    instrument='AIA', detector='AIA', measurement='171')
    manual_map = sunpy.map.Map((filepath))
    # Download the new image  of the date that we want to propagate the position
    filepath2 = hv.download_jp2(past_date, observatory='SDO',
    instrument='AIA', detector='AIA', measurement='171')
    manual_map2 = sunpy.map.Map((filepath2))

     #Initializing the Solar_rotated_Coordinate function to move the spot to its new location
    start_coord = SkyCoord(hpc_x, hpc_y, frame=manual_map.coordinate_frame)
    coord = SkyCoord([start_coord.Tx], [start_coord.Ty], frame=manual_map.coordinate_frame)
    rotated_coord_Final = solar_rotate_coordinate(start_coord, past_date)
    coord_rotated_Final = SkyCoord([rotated_coord_Final.Tx],
    [rotated_coord_Final.Ty], frame=manual_map2.coordinate_frame)

    #plot the differences
    fig = plt.figure()
    fig.suptitle('The effect of {0} days of differential rotation'.format(total.days))
    ax1 = fig.add_subplot(1, 2, 1, projection=manual_map)
```

```
    manual_map.plot()
    ax1.plot_coord(coord, 'x')
    ax2 = fig.add_subplot(1, 2, 2, projection=manual_map2)
    manual_map2.plot()
    ax2.plot_coord(coord_rotated_Final, 'x')
    plt.show()

    return

dt=3600 #[Seconds] Frequency
total= 1 #[Days]
GetAllDataOfSun(-450,-450,'2013/02/13 12:40:00',total)
```

OUTPUT:

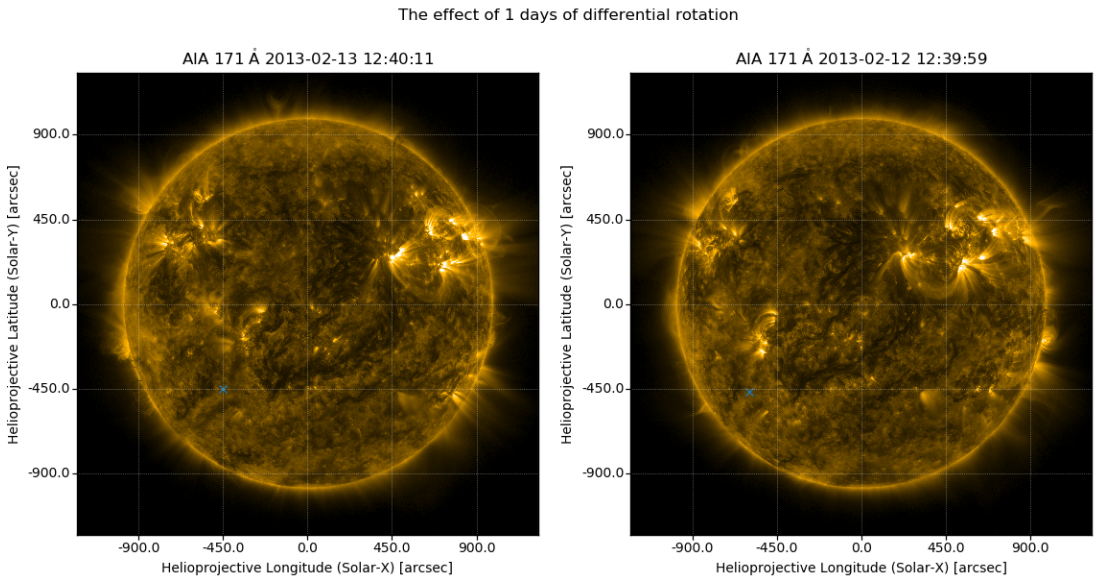


Figure 54: **[Right]**: How the Sun looks like in the given date. **[Left]**: How the Sun looks like 1 day before the date that was initialized as input. Special indication has been made to a specific spot on the sun that is rotated according to the Sun's differential rotation.

Tensorflow

Tensorflow is an open-source library in Python, mainly used for machine learning applications. It is produced and developed by Google and was initially released in 2015. It uses dataflow graphs across many machines in a cluster and therefore utilizing multiple CPUs and GPUs. As a result, it was build for and is widely in distributed systems [1].

Tensorflow as a library can be used as a GPU or a CPU Package. In our project a GPU version of Tensorflow was used as the back-end, processing library of the Neural Network.

For the coding example, a general database that is commonly used in ML benchmark is utilized. This database is called MNIST[2] and is an immense database of handwritten digits that is used for training image processing systems [34]. A visualization of the MNIST database is shown in Fig.55.

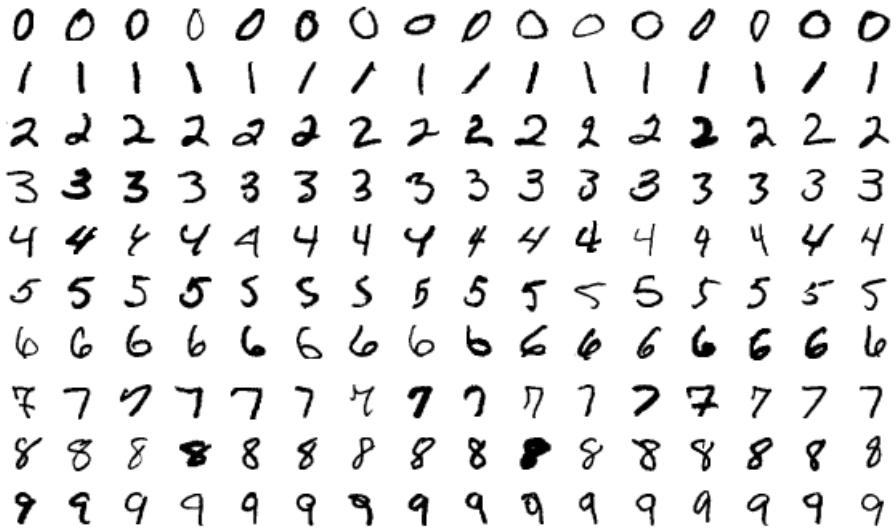


Figure 55: An example of a few samples from the MNIST database. Figure Courtesy: Josef Steppan, Source: `https://en.wikipedia.org/wiki/MNIST_database`

Code Example

An example of a code [3] can be seen below. On this example, the MNIST database is used and a NN is implemented for its classification.

The NN in this example has:

- 4 Dense Layers (400 - 200 - 150 - 10 Neurons)

[2]Openly accessible via `https://www.kaggle.com/c/digit-recognizer/data`, a more massive version called EMNIST may also be found online.

[3] The code is motivated from the work of Andy Thomas. More of his work and tutorials can be found in `https://github.com/adventuresinML/adventures-in-ml-code`.

CODE:

```
"""
@author: Savvas Raptis, SavvasRaptis@gmail.com
Original idea modified from  Andy Thomas (http://adventuresinmachinelearning.com)
Machine Learning Project, KU Leuven, Centre of Plasma-Astrophysics
"""

from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf

# Downloading and inputing MNIST database
#one_hot= True makes the input binary. Eg 5 = 0000010000
mnist = input_data.read_data_sets("data/", one_hot=True)

# # Determine basic parameters of the dataset/model
learning_rate = 0.25 # a learning parameter
batch_size = 128 # samples that going to be propagated through the network.
epochs = 5 #How many times we read through the data

#Creating input and output placeholders for tensorflow

# input = 28 x 28 pixels = 784
x = tf.placeholder(tf.float32, [None, 784])
# output = 10 digits = 10
y = tf.placeholder(tf.float32, [None, 10])

# our model will be a simple 3 hiddel layer deep neural network
L= 400 # Our hidden layer in that case has 400 neurons
M= 200 # Our hidden layer in that case has 200 neurons
N = 150# Our hidden layer in that case has 150 neurons

# Decalting weight and biases from the input to the 1st  hiddel layer
#When using reou,  biases should have small *positive* values for example 0.01
w1 = tf.Variable(tf.random_normal([784, L], stddev=0.01), name='w1')
b1 = tf.Variable(tf.random_normal([L]), name='b1')
# Decalting weight and biases from the 1st hidden to the 2nd layer
w2 = tf.Variable(tf.random_normal([L, M], stddev=0.01), name='w2')
b2 = tf.Variable(tf.random_normal([M]), name='b2')
# Decalting weight and biases from the 2nd hidden to the 3rd layer
w3 = tf.Variable(tf.random_normal([M, N], stddev=0.01), name='w3')
b3 = tf.Variable(tf.random_normal([N]), name='b3')
# Decalting weight and biases from the 2nd hidden to the output layer
w4 = tf.Variable(tf.random_normal([N, 10], stddev=0.01), name='w3')
b4 = tf.Variable(tf.random_normal([10]), name='b3')

# calculate the output of the 1st hidden layer
```

```
hidden_out = tf.add(tf.matmul(x, w1), b1)
hidden_out = tf.nn.relu(hidden_out)  #Using a Relu activation function
# calculate the output of the 2nd hidden layer
hidden_out2 = tf.add(tf.matmul(hidden_out, w2), b2)
hidden_out2 = tf.nn.relu(hidden_out2)  #Using a Relu activation function

# calculate the output of the 2nd hidden layer
hidden_out3 = tf.add(tf.matmul(hidden_out2, w3), b3)
hidden_out3 = tf.nn.relu(hidden_out3)  #Using a Relu activation function


#since we have a classification problem we use as the last activation function
#a softmax to classify between the last 10 neurons/classes
y_result = tf.nn.softmax(tf.add(tf.matmul(hidden_out3, w4), b4))

yresult_clipped = tf.clip_by_value(y_result, 1e-10, 0.9999999)
#ensuring that we dont' get nan values

#Cross Entrpy calculation
cross_entropy = -tf.reduce_mean(tf.reduce_sum(y * tf.log(yresult_clipped)
                        + (1 - y) * tf.log(1 - yresult_clipped), axis=1))

# including an optimizer for the training process
optimiser = tf.train.GradientDescentOptimizer(
learning_rate=learning_rate).minimize(cross_entropy)

# using an initializer
initializer = tf.global_variables_initializer()

# deifning how the accuracy is calculated
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(yresult_clipped, 1))
Total_accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

'''
TRAINING PROCESS
'''

# starting the training session
with tf.Session() as sess:
   # initialise the variables
   sess.run(initializer)
   total_batch = int(len(mnist.train.labels) / batch_size)
   for epoch in range(0,epochs):
        average = 0
        for i in range(total_batch):
            #using batches for x and y
            bx, by = mnist.train.next_batch(batch_size=batch_size)
```

```
            #Running the session with optimizer/cross_entropy/data parameters set
            g, c = sess.run([optimiser, cross_entropy],
                            feed_dict={x: bx, y: by})
            #Calculating the average result on each epoch
            average += c / total_batch
        print("Epoch, Number", (epoch + 1), ":average cost =", "{:.4f}".format(average))
    print(sess.run(Total_accuracy, feed_dict={x: mnist.test.images, y: mnist.test.labels}))
```

OUTPUT:

```
Epoch, Number 1 :average cost = 2.8782
Epoch, Number 2 :average cost = 0.8372
Epoch, Number 3 :average cost = 0.3230
Epoch, Number 4 :average cost = 0.2121
Epoch, Number 5 :average cost = 0.1680
0.9703
```

In this example, by using just 5 epoch, a 97% accuracy with a deep neural network is achieved, which is already a very satisfactory result.

Keras

Keras[4] is a high-leve API (Application Programming Interface), written in Python, capable of running on top of TensorFlow. it is mainly designed to empower fast experimentation with deep neural networks with using simple syntax [22].

A common criticism of Keras is that it makes things "too" easy. On the other hand, Keras provides the user with all the building blocks they need to test different neural network architecture on their dataset.

Code Example

An example of a code can be seen below. On this example, using again the MNIST database, a CNN is implemented.

The CNN in this example has:

- 2 Convolution Layers

[4]Keras was initially developed as part of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System)

- 1 Max Pooling Layer
- 2 Dense Layers (128 & 10 Neurons)
- Dropout Regularization technique

CODE:

```
"""
@author: Savvas Raptis, SavvasRaptis@gmail.com
Machine Learning Project, KU Leuven, Centre of Plasma-Astrophysics
"""

import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import MaxPooling2D, Conv2D
from keras.layers.advanced_activations import LeakyReLU
from keras import backend as K

# Determine basic parameters of the dataset
batch_size = 128 # samples that going to be propagated through the network.
num_classes = 10 #How many classes we are classifying in our problem
epochs = 5 #How many times we read through the data
verbose = 1 #Method of showing training process 1 takes more time

# input image dimensions (MNIST database is normalized to 28x28 pixels)
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

'''
Data Transformation, checking if the shapes of the matrix is correct to
impliment the Neural Network
'''
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
```

```
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)


'''
Model Creation: In this example we have 2 Convolution layer and 2 dense layers.
The first of the last two dense layers works as a normal network and the last
one is using softmax activation to derive one of the 10 classification classes.

There is also a dropout method implementation to increase accuracy and
regularize our data.
'''
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 input_shape=input_shape))
model.add(LeakyReLU(alpha=.01))
model.add(Conv2D(64, (3, 3)))
model.add(LeakyReLU(alpha=.01))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128))
model.add(LeakyReLU(alpha=.01))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))


'''
Model compilation and training proccedure
'''
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=verbose,
          validation_data=(x_test, y_test))
```

```
score = model.evaluate(x_test, y_test, verbose=0)

'''
Printing the result
'''
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

OUTPUT:

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 166s 3ms/step - loss: 0.2485 - acc: 0.9226 -
val_loss: 0.0557 - val_acc: 0.9814
Epoch 2/5
60000/60000 [==============================] - 152s 3ms/step - loss: 0.0863 - acc: 0.9740 -
val_loss: 0.0387 - val_acc: 0.9869
Epoch 3/5
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0667 - acc: 0.9799 -
val_loss: 0.0371 - val_acc: 0.9871
Epoch 4/5
60000/60000 [==============================] - 167s 3ms/step - loss: 0.0531 - acc: 0.9841 -
val_loss: 0.0325 - val_acc: 0.9889
Epoch 5/5
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0452 - acc: 0.9862 -
val_loss: 0.0290 - val_acc: 0.9907

Test loss: 0.028962729370724263
Test accuracy: 0.9907
```

By using just 5 epoch, An accuracy, bigger than 99% was acvieved. The network utilized, LeakyReLU activation functions and the dropout method. Comparing this result with the 97 % that was received in the tensorflow example code, it is immediately shown that the advantage of CNN in image related ML problems is significant.

Other Machine Learning & A.I. Libraries

Theano, Pytorch & CNTK are three libraries that were not studied nor used during this work however, they are also widely used in the Machine learning community.

Theano, originally developed at the Université de Montréal,[5] is one of the oldest and most stable libraries for ML algorithms used in Python. It is an open source project that is widely used throughout academia and industry [12]. Although recently Theano lost its momentum because major development stopped, it is still a very stable and safe option. Theano is a low level library that can be used amazingly well for education purpose and for deriving a deeper understanding of the concepts and the mathematics behind NN.

Pytorch is an open-source ML library for Python, based on the previously made, Torch library and primary developed by Facebook. Torch was originally designed for lua language but was later dropped and transformed to Pytorch [64]. Pytorch is gaining a lot of popularity in the last year (2017) because compared to the other popular libraries, it supports dynamic computation graphs [64].

CNTK[6] (Microsoft Cognitive Toolkit) is an open source toolkit developed by Microsoft, that uses directed graphs in order to describe neural networks [108].

[5] Link: `http://deeplearning.net/software/theano`
[6] `https://www.microsoft.com/en-us/cognitive-toolkit/`

BIBLIOGRAPHY

[1] M. ABADI, P. BARHAM, J. CHEN, Z. CHEN, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, G. IRVING, M. ISARD, ET AL., *Tensorflow: A system for large-scale machine learning.*, in OSDI, vol. 16, 2016, pp. 265–283.

[2] P. ALAEI, *Introduction to health physics*, Medical Physics, 35 (2008), pp. 5959–5959.

[3] E. ALCAIDE, *E-swish: Adjusting Activations to Different Network Depths*, ArXiv e-prints, (2018).

[4] M. ANDREWS AND R. HOWARD, *A two-type classification of lasco coronal mass ejection*, Space Science Reviews, 95 (2001), pp. 147–163.

[5] S. ANTIOCHOS, Z. MIKIĆ, V. TITOV, R. LIONELLO, AND J. LINKER, *A model for the sources of the slow solar wind*, The Astrophysical Journal, 731 (2011), p. 112.

[6] H. BABCOCK, *The topology of the sun's magnetic field and the 22-year cycle.*, The Astrophysical Journal, 133 (1961), p. 572.

[7] J. L. BARTH, C. DYER, AND E. STASSINOPOULOS, *Space, atmospheric, and terrestrial radiation environments*, IEEE Transactions on Nuclear Science, 50 (2003), pp. 466–482.

[8] D. BARTLETT, *Radiation protection aspects of the cosmic radiation exposure of aircraft crew*, Radiation protection dosimetry, 109 (2004), pp. 349–355.

[9] I. BAYLAKOGLU AND M. HUDAVERDI, *Reliability concerns of radiation effects on space electronics*, Space Technologies Research Institute, Ankara, Turkey, (2009).

[10] J. G. BECK, *A comparison of differential rotation measurements–(invited review)*, Solar Physics, 191 (2000), pp. 47–70.

[11] V. BEREZINSKII, S. BULANOV, V. GINZBURG, V. DOGEL, AND V. PTUSKIN, *The astrophysics of cosmic rays*, Moscow, Izdatel'stvo Nauka, 1984, 360 p. In Russian., (1984).

[12] J. BERGSTRA, O. BREULEUX, F. BASTIEN, P. LAMBLIN, R. PASCANU, G. DESJARDINS, J. TURIAN, D. WARDE-FARLEY, AND Y. BENGIO, *Theano: A cpu and gpu math compiler in python*, in Proc. 9th Python in Science Conf, 2010, pp. 1–7.

[13] W. BIETENHOLZ, *The most powerful particles in the universe: a cosmic smash*, arXiv preprint arXiv:1305.1346, (2013).

[14] C. M. BISHOP, *Neural networks for pattern recognition*, Oxford university press, 1995.

[15] M. G. BOBRA AND S. ILONIDIS, *Predicting coronal mass ejections using machine learning methods*, The Astrophysical Journal, 821 (2016), p. 127.

[16] D. BOTELER, *Assessment of geomagnetic hazard to power systems in canada*, Natural Hazards, 23 (2001), pp. 101–120.

[17] V. BOTHMER AND I. A. DAGLIS, *Space weather: physics and effects*, Springer Science & Business Media, 2007.

[18] G. BRUECKNER, R. HOWARD, M. KOOMEN, C. KORENDYKE, D. MICHELS, J. MOSES, D. SOCKER, K. DERE, P. LAMY, A. LLEBARIA, ET AL., *The large angle spectroscopic coronagraph (lasco)*, in The SOHO Mission, Springer, 1995, pp. 357–402.

[19] L. F. BURLAGA, *Magnetic clouds*, in Physics of the Inner Heliosphere II, Springer, 1991, pp. 1–22.

[20] H. CANE AND I. RICHARDSON, *Interplanetary coronal mass ejections in the near-earth solar wind during 1996–2002*, Journal of Geophysical Research: Space Physics, 108 (2003).

[21] V. CHAUHAN, M. HOWLAND, B. KUTZNER, J. P. MCNAMEE, P. V. BELLIER, AND R. C. WILKINS, *Biological effects of alpha particle radiation exposure on human monocytic cells*, International journal of hygiene and environmental health, 215 (2012), pp. 339–344.

[22] F. CHOLLET, *Keras*.
`https://github.com/fchollet/keras`, 2015.

[23] D.-A. CLEVERT, T. UNTERTHINER, AND S. HOCHREITER, *Fast and accurate deep network learning by exponential linear units (elus)*, arXiv preprint arXiv:1511.07289, (2015).

[24] T. COLAK AND R. QAHWAJI, *Automated solar activity prediction: A hybrid computer platform using machine learning and solar imaging for automated prediction of solar flares*, Space Weather, 7 (2009).

[25] A. N. COX, *Allen's astrophysical quantities*, Springer, 2015.

[26] D. D. COX AND T. DEAN, *Neural networks and neuroscience-inspired computer vision*, Current Biology, 24 (2014), pp. R921–R929.

[27] F. A. CUCINOTTA, *Space radiation risks for astronauts on multiple international space station missions*, PloS one, 9 (2014), p. e96099.

[28] I. A. DAGLIS, *The role of magnetosphere-ionosphere coupling in magnetic storm dynamics*, Magnetic Storms, (1997), pp. 107–116.

[29] I. A. DAGLIS, R. M. THORNE, W. BAUMJOHANN, AND S. ORSINI, *The terrestrial ring current: Origin, formation, and decay*, Reviews of Geophysics, 37 (1999), pp. 407–438.

[30] E. DALY, A. GLOVER, AND A. HILGERS, *Effects on spacecraft hardware and operations*, in Space Weather-Physics and Effects, Springer, 2007, pp. 353–381.

[31] S. DASSO, L. MILANO, W. MATTHAEUS, AND C. SMITH, *Anisotropy in fast and slow solar wind fluctuations*, The Astrophysical Journal Letters, 635 (2005), p. L181.

[32] M. A. DAYEH, *Coronal mass ejections*, Handbook of Cosmic Hazards and Planetary Defense, (2015), pp. 81–98.

[33] M. D. DELP, J. M. CHARVAT, C. L. LIMOLI, R. K. GLOBUS, AND P. GHOSH, *Apollo lunar astronauts show higher cardiovascular disease mortality: Possible deep space radiation effects on the vascular endothelium*, Scientific Reports, 6 (2016).

[34] L. DENG, *The mnist database of handwritten digit images for machine learning research [best of the web]*, IEEE Signal Processing Magazine, 29 (2012), pp. 141–142.

[35] M. DIKPATI AND P. A. GILMAN, *Simulating and predicting solar cycles using a flux-transport dynamo*, The Astrophysical Journal, 649 (2006), p. 498.

[36] E. ECHER, W. GONZALEZ, AND B. TSURUTANI, *Statistical studies of geomagnetic storms with peak dst≤- 50nt from 1957 to 2008*, Journal of Atmospheric and Solar-Terrestrial Physics, 73 (2011), pp. 1454–1459.

[37] L. EHERNBERGER, *Stratospheric turbulence measurements and models for aerospace planedesign*, in 4th Symposium on Multidisciplinary Analysis and Optimization, 1992, p. 5072.

[38] S. ELFWING, E. UCHIBE, AND K. DOYA, *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning*, Neural Networks, (2018).

[39] H. EVANS, P. BÜHLER, W. HAJDAS, E. DALY, P. NIEMINEN, AND A. MOHAMMADZADEH, *Results from the esa srem monitors and comparison with existing radiation belt models*, Advances in space research, 42 (2008), pp. 1527–1537.

[40] L. FISK, *Acceleration of the solar wind as a result of the reconnection of open magnetic flux with coronal loops*, Journal of Geophysical Research: Space Physics, 108 (2003).

[41] R. FITZPATRICK, *Plasma physics: an introduction*, Crc Press, 2014.

[42] K. FLORIOS, I. KONTOGIANNIS, S.-H. PARK, J. A. GUERRA, F. BENVENUTO, D. S. BLOOMFIELD, AND M. K. GEORGOULIS, *Forecasting solar flares using magnetogram-based predictors and machine learning*, arXiv preprint arXiv:1801.05744, (2018).

[43] A. G. FRANK, *Magnetic reconnection and current sheet formation in 3d magnetic configurations*, Plasma physics and controlled fusion, 41 (1999), p. A687.

[44] A. GERON, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, Sebastopol, CA, 2017.

[45] S. GLASSTONE AND P. J. DOLAN, *The effects of nuclear weapons*, tech. rep., Department of Defense, Washington, DC (USA); Department of Energy, Washington, DC (USA), 1977.

[46] M. GNEVYSHEV, *Essential features of the 11-year solar cycle*, Solar Physics, 51 (1977), pp. 175–183.

[47] W. GONZALEZ, J. JOSELYN, Y. KAMIDE, H. KROEHL, G. ROSTOKER, B. TSURUTANI, AND V. VASYLIUNAS, *What is a geomagnetic storm?*, Journal of Geophysical Research: Space Physics, 99 (1994), pp. 5771–5792.

[48] W. D. GONZALEZ, B. T. TSURUTANI, AND A. L. C. DE GONZALEZ, *Interplanetary origin of geomagnetic storms*, Space Science Reviews, 88 (1999), pp. 529–562.

[49] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[50] N. GOPALSWAMY, *Coronal mass ejections of solar cycle 23*, Journal of Astrophysics and Astronomy, 27 (2006), pp. 243–254.

[51] R. H. HAHNLOSER AND H. S. SEUNG, *Permitted and forbidden sets in symmetric threshold-linear networks*, in Advances in Neural Information Processing Systems, 2001, pp. 217–223.

[52] G. E. HINTON, N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER, AND R. R. SALAKHUTDINOV, *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv preprint arXiv:1207.0580, (2012).

[53] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[54] R. HOWARD, *Solar rotation*, Annual review of astronomy and astrophysics, 22 (1984), pp. 131–155.

[55] R. HOWARD, *A historical perspective on coronal mass ejections*, Solar Eruptions and Energetic Particles, (2006), pp. 7–13.

[56] R. HOWARD AND J. HARVEY, *Spectroscopic determinations of solar rotation*, Solar Physics, 12 (1970), pp. 23–51.

[57] T. HOWARD, *Coronal mass ejections : an introduction*, Springer, New York, 2011.

[58] J. A. HUTCHINSON, D. WRIGHT, AND S. MILAN, *Geomagnetic storms over the last solar cycle: A superposed epoch analysis*, Journal of Geophysical Research: Space Physics, 116 (2011).

[59] S. IOFFE AND C. SZEGEDY, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, 2015, pp. 448–456.

[60] M. JANVIER, *Three-dimensional magnetic reconnection and its application to solar flares*, Journal of Plasma Physics, 83 (2017).

[61] E. JONAS, V. SHANKAR, M. BOBRA, AND B. RECHT, *Flare Prediction Using Photospheric and Coronal Image Data*, AGU Fall Meeting Abstracts, (2016), pp. SH34A–02.

[62] A. S. JURSA ET AL., *Handbook of geophysics and the space environment*, vol. 1, Air Force Geophysics Laboratory, Air Force Systems Command, United States Air Force, 1985.

[63] M. KALLENRODE, *Current views on impulsive and gradual solar energetic particle events*, Journal of Physics G: Nuclear and Particle Physics, 29 (2003), p. 965.

[64] N. KETKAR, *Introduction to pytorch*, in Deep Learning with Python, Springer, 2017, pp. 195–208.

[65] R.-S. KIM, K.-S. CHO, J. LEE, S.-C. BONG, A. JOSHI, AND Y.-D. PARK, *Characteristics of four spe groups with different origins and acceleration processes*, Journal of Geophysical Research: Space Physics, 120 (2015), pp. 7083–7093.

[66] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[67] M. G. KIVELSON AND F. BAGENAL, *Planetary magnetospheres*, in Encyclopedia of the Solar System (Third Edition), Elsevier, 2014, pp. 137–157.

[68] G. KOPP, G. LAWRENCE, AND G. ROTTMAN, *The total irradiance monitor (tim): science results*, in The Solar Radiation and Climate Experiment (SORCE), Springer, 2005, pp. 129–139.

[69] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[70] M. KUHN AND K. JOHNSON, *Applied predictive modeling*, vol. 26, Springer, 2013.

[71] Y. LECUN, Y. BENGIO, ET AL., *Convolutional networks for images, speech, and time series*, The handbook of brain theory and neural networks, 3361 (1995), p. 1995.

[72] E. C. LIN, *Radiation risk from medical imaging*, in Mayo Clinic Proceedings, vol. 85, Elsevier, 2010, pp. 1142–1146.

[73] J. LIN AND T. FORBES, *Effects of reconnection on the coronal mass ejection process*, Journal of Geophysical Research: Space Physics, 105 (2000), pp. 2375–2392.

[74] M. LOCKWOOD AND P. MURDIN, *Magnetosphere of earth*, Encyclopedia of Astronomy and Astrophysics, (2001).

[75] L. LU, B. INHESTER, L. FENG, S. LIU, AND X. ZHAO, *Measure the propagation of a halo cme and its driven shock with the observations from a single perspective at earth*, The Astrophysical Journal, 835 (2017), p. 188.

[76] B. LYNCH, S. ANTIOCHOS, C. DEVORE, J. LUHMANN, AND T. ZURBUCHEN, *Topological evolution of a fast magnetic breakout cme in three dimensions*, The Astrophysical Journal, 683 (2008), p. 1192.

[77] R. MACQUEEN AND R. FISHER, *The kinematics of solar inner coronal transients*, Solar Physics, 89 (1983), pp. 89–102.

[78] W. S. MCCULLOCH AND W. PITTS, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115–133.

[79] S. W. MCINTOSH, K. K. KIEFER, R. J. LEAMON, J. C. KASPER, AND M. L. STEVENS, *Solar cycle variations in the elemental abundance of helium and fractionation of iron in the fast solar wind: indicators of an evolving energetic release of mass from the lower solar atmosphere*, The Astrophysical Journal Letters, 740 (2011), p. L23.

[80] M. MOLDWIN, *An introduction to space weather*, Cambridge University Press, 2008.

[81] R. L. MOORE, A. C. STERLING, H. S. HUDSON, AND J. R. LEMEN, *Onset of the magnetic explosion in solar flares and coronal mass ejections*, The Astrophysical Journal, 552 (2001), p. 833.

[82] D. MULLAN, *Physics of the sun : a first course*, CRC Press, Boca Raton, 2010.

[83] S. J. MUMFORD, S. CHRISTE, D. PÉREZ-SUÁREZ, J. IRELAND, A. Y. SHIH, A. R. INGLIS, S. LIEDTKE, R. J. HEWETT, F. MAYER, K. HUGHITT, ET AL., *Sunpy—python for solar physics*, Computational Science & Discovery, 8 (2015), p. 014009.

[84] K. MURSULA, I. G. USOSKIN, AND G. MARIS, *Introduction to space climate*, Advances in Space Research, 40 (2007), pp. 885–887.

[85] G. NAT, *Properties of interplanetary coronal mass ejections*, Space Science Reviews, 124 (2006), pp. 145–168.

[86] E. NIKLASSON AND D. MITRA, *Modeling the path of a charged particle in a magnetic mirror*, Research Academy for Young Scientists July, 11 (2012).

[87] A. OBERMEIER, M. AVE, P. BOYLE, C. HÖPPNER, J. HÖRANDEL, AND D. MÜLLER, *Energy spectra of primary and secondary cosmic-ray nuclei measured with tracer*, The Astrophysical Journal, 742 (2011), p. 14.

[88] D. ODSTRČIL AND V. PIZZO, *Three-dimensional propagation of coronal mass ejections (cmes) in a structured solar wind flow: 1. cme launched within the streamer belt*, Journal of Geophysical Research: Space Physics, 104 (1999), pp. 483–492.

[89] M. J. OWENS AND R. J. FORSYTH, *The heliospheric magnetic field*, Living Reviews in Solar Physics, 10 (2013), p. 5.

[90] H. E. PETSCHEK, *Magnetic field annihilation*, NASA Special Publication, 50 (1964), p. 425.

[91] R. PICCINELLI AND E. KRAUSMANN, *Space weather and power grids–a vulnerability assessment*, JRC Scientific and Policy Reports, European Commission, Luxemburg, (2014).

[92] D. A. PIERCE AND D. L. PRESTON, *Radiation-related cancer risks at low doses among atomic bomb survivors*, Radiation research, 154 (2000), pp. 178–186.

[93] M. POTGIETER, *Solar cycle variations and cosmic rays*, Journal of Atmospheric and Solar-Terrestrial Physics, 70 (2008), pp. 207–218.

[94] P. RAMACHANDRAN, B. ZOPH, AND Q. V. LE, *Swish: a self-gated activation function*, arXiv preprint arXiv:1710.05941, (2017).

[95] F. REALE, *Coronal loops: observations and modeling of confined plasma*, Living Reviews in Solar Physics, 11 (2014), p. 4.

[96] D. V. REAMES, *The two sources of solar energetic particles*, Space Science Reviews, 175 (2013), pp. 53–92.

[97] B. D. RIPLEY, *Pattern recognition and neural networks*, Cambridge university press, 2007.

[98] E. ROBBRECHT AND D. BERGHMANS, *Automated recognition of coronal mass ejections (cmes) in near-real-time data*, Astronomy & Astrophysics, 425 (2004), pp. 1097–1106.

[99] E. ROBBRECHT, D. BERGHMANS, AND R. VAN DER LINDEN, *Automated lasco cme catalog for solar cycle 23: are cmes scale invariant?*, The Astrophysical Journal, 691 (2009), p. 1222.

[100] F. ROSENBLATT, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), p. 386.

[101] L. ROSENQVIST, *Energy Transfer and Conversion in the Magnetosphere-Ionosphere System*, PhD thesis, Acta Universitatis Upsaliensis, 2008.

[102] S. RUDER, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747, (2016).

[103] G. RÜDIGER AND A. BRANDENBURG, *A solar dynamo in the overshoot layer: cycle period and butterfly diagram.*, Astronomy and Astrophysics, 296 (1995), p. 557.

[104] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning internal representations by error propagation*, tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[105] A. L. SAMUEL, *Some studies in machine learning using the game of checkers*, IBM Journal of research and development, 3 (1959), pp. 210–229.

[106] I. SANDBERG, I. DAGLIS, A. ANASTASIADIS, P. BUHLER, P. NIEMINEN, AND H. EVANS, *Unfolding and validation of srem fluxes*, IEEE Transactions on Nuclear Science, 59 (2012), pp. 1105–1112.

[107] R. SCHWENN, *Solar wind: Global properties*, Encyclopedia of Astronomy and Astrophysics (Nature Publishing Group, Bristol, 2001), (2001).

[108] F. SEIDE AND A. AGARWAL, *Cntk: Microsoft's open-source deep-learning toolkit*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 2135–2135.

[109] G. SENI AND J. F. ELDER, *Ensemble methods in data mining: improving accuracy through combining predictions*, Synthesis Lectures on Data Mining and Knowledge Discovery, 2 (2010), pp. 1–126.

[110] P. SIMÕES, L. FLETCHER, H. HUDSON, AND A. RUSSELL, *Implosion of coronal loops during the impulsive phase of a solar flare*, The Astrophysical Journal, 777 (2013), p. 152.

[111] H. B. SNODGRASS AND R. K. ULRICH, *Rotation of doppler features in the solar photosphere*, The Astrophysical Journal, 351 (1990), pp. 309–316.

[112] N. SRIVASTAVA, G. HINTON, A. KRIZHEVSKY, I. SUTSKEVER, AND R. SALAKHUTDINOV, *Dropout: A simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research, 15 (2014), pp. 1929–1958.

[113] F. STEFANI, A. GIESECKE, N. WEBER, AND T. WEIER, *Synchronized helicity oscillations: A link between planetary tides and the solar cycle?*, Solar Physics, 291 (2016), pp. 2197–2212.

[114] H. SVENSMARK, *Cosmic rays and earth's climate*, in Cosmic Rays and Earth, Springer, 2000, pp. 175–185.

[115] S. SYROVATSKII, *Solar flare time development: Three phases*, Comments on Astrophysics and Space Physics, 4 (1972), p. 65.

[116] C. SZEGEDY, W. LIU, Y. JIA, P. SERMANET, S. REED, D. ANGUELOV, D. ERHAN, V. VANHOUCKE, A. RABINOVICH, ET AL., *Going deeper with convolutions*, Cvpr, 2015.

[117] A. TAKTAKISHVILI, M. KUZNETSOVA, P. MACNEICE, M. HESSE, L. RASTÄTTER, A. PULKKINEN, A. CHULAKI, AND D. ODSTRCIL, *Validation of the coronal mass ejection predictions at the earth orbit estimated by enlil heliosphere cone model*, Space Weather, 7 (2009).

[118] J. UWAMAHORO, L. MCKINNELL, AND J. HABARULEMA, *Estimating the geoeffectiveness of halo cmes from associated solar and ip parameters using neural networks*, in Annales Geophysicae, vol. 30, Copernicus GmbH, 2012, p. 963.

[119] L. WAN, M. ZEILER, S. ZHANG, Y. LE CUN, AND R. FERGUS, *Regularization of neural networks using dropconnect*, in International Conference on Machine Learning, 2013, pp. 1058–1066.

[120] Y.-M. WANG, J. LEAN, AND N. SHEELEY JR, *Modeling the sun's magnetic field and irradiance since 1713*, The Astrophysical Journal, 625 (2005), p. 522.

[121] E. W. WEISSTEIN, *Newton's method*, (2002).

[122] ——, *Convolution*. `http://mathworld.wolfram.com/Convolution.html`, 2003.

[123] M. WHEATLAND AND Y. E. LITVINENKO, *Understanding solar flare waiting-time distributions*, Solar Physics, 211 (2002), pp. 255–274.

[124] P. WINTOFT, M. WIK, J. MATZKA, AND Y. SHPRITS, *Forecasting kp from solar wind data: input parameter study using 3-hour averages and 3-hour range values*, Journal of Space Weather and Space Climate, 7 (2017), p. A29.

[125] WINTOFT, PETER, WIK, MAGNUS, MATZKA, JÜRGEN, AND SHPRITS, YURI, *Forecasting kp from solar wind data: input parameter study using 3-hour averages and 3-hour range values*, J. Space Weather Space Clim., 7 (2017), p. A29.

[126] S. YASHIRO, G. MICHALEK, S. AKIYAMA, N. GOPALSWAMY, AND R. HOWARD, *Spatial relationship between solar flares and coronal mass ejections*, The Astrophysical Journal, 673 (2008), p. 1174.

[127] M. YOUSSEF, *On the relation between the cmes and the solar flares*, NRIAG Journal of Astronomy and Geophysics, 1 (2012), pp. 172–178.

[128] J. B. ZIRKER, *Coronal holes and high-speed wind streams*, Reviews of Geophysics, 15 (1977), pp. 257–269.

DEPARTEMENT OF PHYSICS AND ASTRONOMY
Celestijnenlaan 200d bus 2412
3001 HEVERLEE, BELGIË
tel. + 32 16 32 71 24
fys.kuleuven.be