INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH - KOLKATA

MS PROJECT REPORT

# Sunspot Emergence Prediction: Machine Learning Based Space Weather Assessment

*Author:*
Vishal Singh
16MS007

*Supervisor:*
Prof. Dibyendu Nandi

*A MS Project Report submitted in fulfillment of the requirements for the degree of Master of Science*

*in the*

Center of Excellence in Space Sciences India
Department or Physical Sciences
July, 2021

# Declaration of Authorship

I, **Vishal Singh**, Registration No. **16MS007**, a student of Department of **PHYSICAL SCIENCES** of the **Integrated BS-MS Program** of IISER Kolkata, hereby declare that this MS Project Report is my own work and, to the best of my knowledge, it neither contains materials previously published or written by any other person, nor it has been submitted for any degree/diploma or any other academic award anywhere before. I have used the originality checking service to prevent inappropriate copying.

I also declare that all copyrighted material incorporated into this thesis is in compliance with the Indian Copyright Act, 1957 (amended in 2012) and that I have received written permission from the copyright owners for my use of their work.

I hereby grant permission to IISER Kolkata to store the MS Project Report in a database which can be accessed by others.

Signed: Vishal Singh
Vishal Singh
Department of Physical Sciences
Indian Institute of Science Education and Research Kolkata
Mohanpur, Nadia, West Bengal - 741246
India.

---

Date: July 21, 2021

---

# Certificate from the supervisor

Date: 21st July 2021

This is to certify that the MS Project Report entitled **"Sunspot Emergence Prediction: Machine Learning Based Space Weather Assessment"** submitted by Mr. Vishal Singh, Registration No. 16MS007, dated 23rd July 2021, a student of Department of **PHYSICAL SCIENCES** of the **Integrated BS-MS Program** of IISER Kolkata, is based upon his own research work under my supervision. This is also to certify that neither the report nor any part of it has been submitted for any degree/diploma or any other academic award anywhere before. In my opinion, the report fulfils the requirement for the award of the degree of Master of Science.

[illegible]

Prof. Dibyendu Nandi
Professor
Department of Physical Sciences
Center of Excellence in Space Sciences India
Indian Institute of Science Education and Research, Kolkata
Mohanpur 741246, West Bengal, India
dnandi@iiserkol.ac.in

*"The effort to understand the universe is one of the very few things that lifts human life a little above the level of farce, and gives it some of the grace of tragedy."*

— Steven Weinberg

# *Abstract*

There is a growing recognition that the environmental conditions we refer to as "space weather" have an impact on the technical infrastructure that drives the world's interconnected economies. As a result, better forecasts, environmental standards, and infrastructure design are required to better protect society from space weather. With research observatories on the ground and in space, significant progress has been done and continues to be made. However, the space weather domain is huge, spanning from deep within the Sun to well beyond planetary orbits.

With the disruptive potential of solar eruptive phenomena like solar flares and coronal mass ejections, there is a strong need for improving our capabilities in being able to predict and prepare for such events. During these events, radiation and mass ejections can cause geomagnetic storms, which can disrupt our satellites and communication systems. Being aware of these episodes beforehand allows us to be ready to deal with their effects. In this context, Machine Learning is a newly emerging valuable tool to effectively and accurately make these predictions. While substantial work has been done in predicting solar flares using machine learning based data driven models, the field of predicting sunspots remains relatively underdeveloped. It is currently not feasible to anticipate their occurrence using a physical model, and one observational prescription has not proven to be reliable.

In this study we have identified issues in previously done machine learning based studies and attempt to find possible solutions. We primarily use Dopplergams time series as the source of data for our models. Machine Learning provides with a large number of options to choose from. For this study, we work with Convolutional Neural Networks, a special kind of neural network designed for dealing with images. The dopplergram time series is used to extract relevant snapshots and used in the aforementioned models.

# *Acknowledgements*

First and foremost I am extremely grateful to my supervisor, Prof. Dibyendu Nandi for his invaluable advice, continuous support, and patience during my Masters study. His immense knowledge and plentiful experience has encouraged me in all the time of my academic research and daily life. I would also like to extend a special thanks to Dr. Dhrubaditya Mitra, Dr. Lekshmi B, Aditi Bhatnagar and Om Gupta for their aid in many different facets of this study. I would like to thank all the members in the Center of Excellence in Space Sciences India. It is their kind help and support that have made my study and life during this work a wonderful time. Finally, I would like to express my gratitude to my parents. Without their tremendous understanding and encouragement in the past few months, it would be impossible for me to complete my study.

I would like to thank the Indian Institute of Science Education and Research-Kolkata for providing me with an environment conducive for research and Scheme for Promotion of Academic and Research Collaboration (SPARC) by Ministry of Education (MoE) for sponsoring this project.

# Contents

# List of Figures

# List of Tables

*Dedicated to Saurabh, without whom I wouldn’t be where I am.*

# Chapter 1

# Introduction

## 1.1 Sunspots

During the Ming Dynasty (1368-1644), the Chinese discovered the first systematic observation of sunspot activity in 1382, when spots on the sun were noticed by peering at the sun through dense forest fire smoke. After the 18th century, sunspot levels became more than a source of amazement and fascination. Since 1834, the National Oceanic and Atmospheric Administration (NOAA) and the United States Naval Observatory have gathered reliable sunspot data.

The Sun is the central star of our Solar System. It is made up of plasma that recieves its heat through nuclear fusion at its core. Emitting energy in a number of forms, the sun is an extremely cruical source of energy for the living beings residing on the planet we call home.

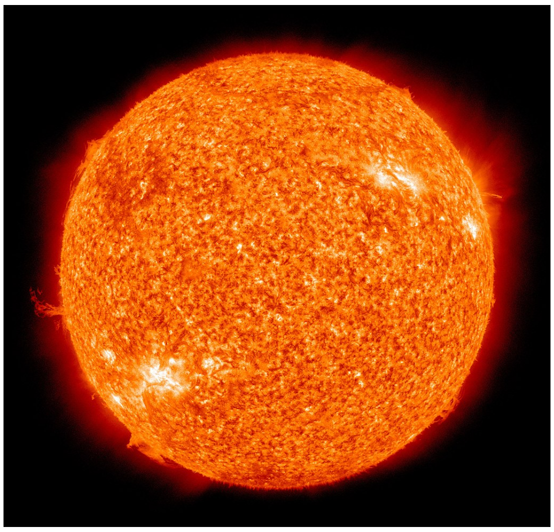


FIGURE 1.1: The Sun photographed at 304 angstroms by the Atmospheric Imaging Assembly (AIA 304) of NASA's Solar Dynamics Observatory (SDO). This is a false-color image of the Sun observed in the extreme ultraviolet region of the spectrum. [1]

[1]Image Credit: NASA/SDO/AIA/HMI/Goddard Space Flight Center.

The magnetic field of the Sun is variable across its surface. It has a polar field of 1–2 gauss, whereas sunspots have a field of 3,000 gauss and solar prominences have a field of 10–100 gauss. The magnetic field changes throughout time and space. The most noticeable fluctuation in the number and size of sunspots is the 11-year solar cycle, which is quasi-periodic.

Sunspots appear as black patches on the Sun's photosphere and correspond to magnetic field concentrations that prevent heat from convecting from the solar core to the surface. As a result, sunspots look darker because they are significantly cooler than the surrounding photosphere. Very few sunspots can be seen during a solar minimum. The ones that we do see, are at high solar latitudes. According to Spörer's law, as the solar cycle moves towards it's maximum, a higher number of sunspots are observed in proximity to the solar equator. They vary a lot in sizes and can be as big as thousands of kilometres in diameter.

There are regions on the Sun's photosphere which have high amounts of magnetic activity relative to surrounding areas. These appear as black patches and are called Sunspots (Figure 1.2). Because of the high magnetic activity, convection of the heat is limited, leading to sunspots looking darker as they are significantly cooler than the surrounding photosphere.

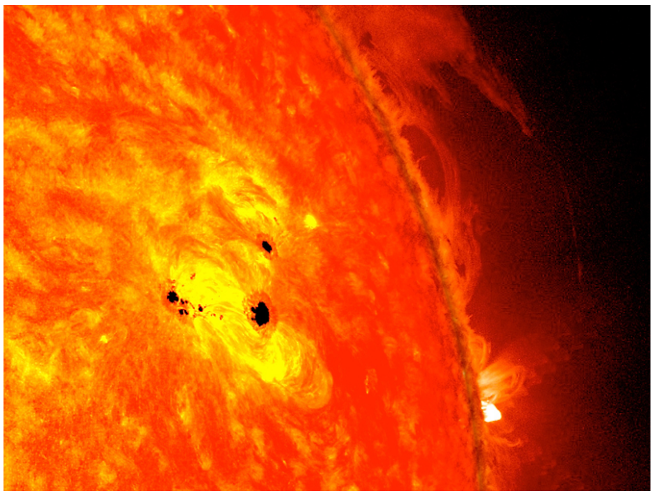


FIGURE 1.2: Over the course of Feb. 19-20, 2013, the bottom two black spots on the sun, known as sunspots, formed swiftly. These two sunspots are part of the same system. This image combines images from two instruments on NASA's Solar Dynamics Observatory (SDO): the Helioseismic and Magnetic Imager (HMI), which takes visible-light pictures of sunspots, and the Advanced Imaging Assembly (AIA), which took a 304 Angstrom wavelength picture of the sun's lower atmosphere, which is colourized in red. [2]

[2]Image Credit: NASA/SDO/AIA/HMI/Goddard Space Flight Center.

An 11-year sunspot cycle that corresponds to an oscillating exchange of energy between toroidal and poloidal solar magnetic fields, is one bisection of a 22-year dynamo cycle known as the Babcock–Leighton cycle. It has periods of solar maximum and solar minimum. During the solar maximum, the exterior poloidal dipolar magnetic field is at its lowest power but the interior toroidal quadrupolar field is near its highest power. During this time, the convective zone experiences buoyant upwelling which is the driver for the toroidal magnetic field to emerge through the photosphere, resulting in pairs of sunspots that are generally aligned east-west and with magnetic polarity footprints that are opposing.

Understanding solar magnetism is one of the most pressing issues in solar and astrophysics [14] . Sunspot areas are thought to be created by a dynamo action at the bottom of the convection zone, roughly 200 Mm below the photosphere, according to current hypotheses. However, there is no compelling evidence to support this theory, and dynamo processes working in the bulk of the convection zone or even in the near-surface shear layer have also been proposed. The depth of this process might be determined by examining emerging magnetic flux, laying the groundwork for a better understanding of sunspots and active zones. Flares and mass eruptions from active zones on the Sun can cause power outages on Earth, satellite failures, and telecommunication and navigation service disruptions. Space weather forecasts would be improved by monitoring solar subsurface processes and anticipating magnetic activity.

## 1.2 Space Weather and Importance of Prediction

The circumstances on the Sun that can affect the operation and reliability of space and ground-based systems, as well as human life on Earth, are referred to as space weather. The Sun's expelled magnetic fields, radiation, particles, and materials can interact with the Earth's upper atmosphere and surrounding magnetic field to cause a range of consequences. One of the earliest instances of Space Weather affecting humans was on December 21, 1806, when Alexander von Humboldt observed that his compass had become erratic during a bright auroral event [21]. Richard C. Carrington saw two rapidly brightening patches of light in the centre of a sunspot group he was viewing on September 1, 1859, during his regular observations of sunspots. Another British astronomer, R. Hodgson, had noticed the same thing. This is the first precise description of a solar flare, which is defined as a powerful radiation emission caused by the abrupt release of magnetic energy held inside twisted flux tubes in the solar atmosphere.

Many critical infrastructure and services that our modern society relies upon, such as telecommunications, global navigational networks, satellite broadcasts, polar air-traffic and high frequency radio communications are dependent on near-Earth space environmental conditions. This environment, termed as space weather, is neither constant, nor benign; it is a variable environment that is governed by changing radiation, magnetic, and high energy particle fluxes from the Sun. The most severe of space storms generated by the Sun have been known to cripple satellites, trip electric power grids and lead to large-scale communication blackouts (Figure 1.3).

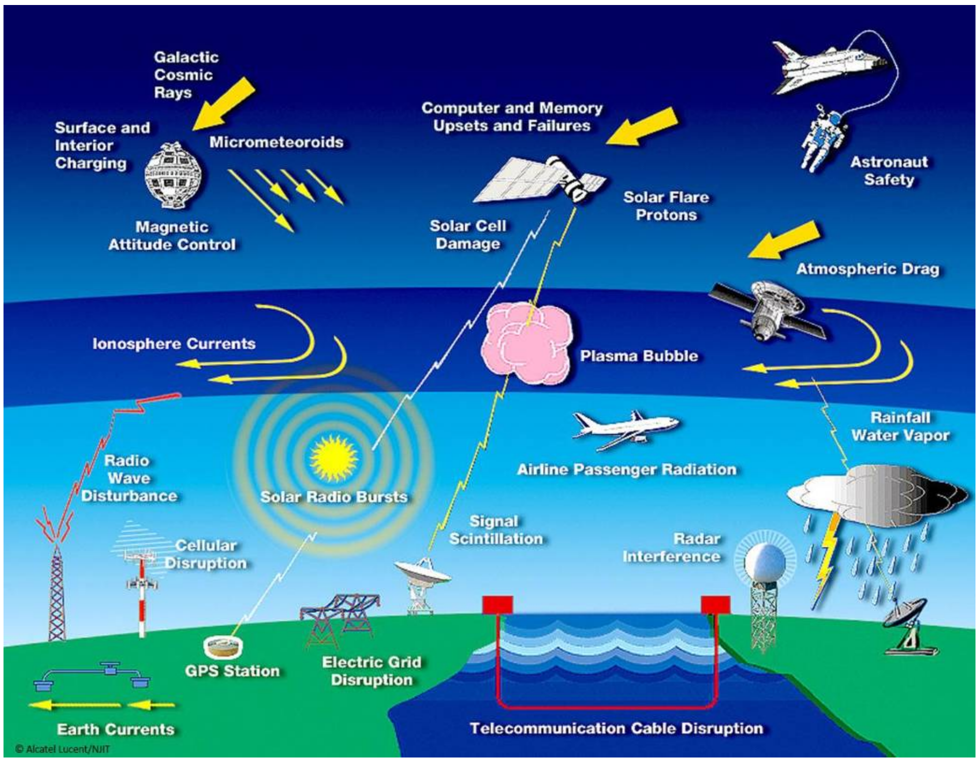


FIGURE 1.3: Space weather phenomena have an impact on technology and infrastructure. Image Credit/Copyright: Alcetel/NJIT

Sunspots are strongly magnetized regions. With magnetic fields 10,000 times stronger than the Earth's, they occasionally erupt on the Sun's surface and lead to Solar Flares [11]. These are believed to result from complex interactions of plasma flows and magnetic fields within the Sun (Brandenburg and Subramanian 2005 [7], Charbonneau *et al.* [8] 2010, Nandy *et al.* [18] 2011, Mitra *et al.* 2014 [16]). The magnetic structure and properties of sunspots (Hahn *et al.* 2005 [12], Hazra, Nandy and Ravindra 2015 [13], Pal *et al.* 2018 [19]) and rate of magnetic flux emergence (Cheung *et al.* 2018 [9]) determine whether they can produce energetic solar storms. Currently, physical model based prediction of their occurrence is not possible and one observational prescription (Ilinodis, Zhao and Kosovichev 2011) [14] has not proven to be robust.

In the recent times, the amount of solar data that is being collected by the observatories, both ground based and the ones launched into space, is increasing exponentially. Instruments improve in spatial, temporal, and/or wavelength resolution with each new mission. In any of these three areas, higher resolution equates to more data volume. It is very diffcult task to work through this data and analyse it manually. Automation is the need of the hour and machine learning as an automated data driven approach is an extremely valuable asset [3].

While Sunspot emergence prediction is an important problem, it is also a difficult problem. Solar flares and Sunspots result from complex interactions of plasma flows and magnetic fields within the Sun. The few attempts at machine learning based solar storm predictions has shown great potential, although the forecast success rate has not reached desirable levels (Ahmed et al. 2013 [1], Bobra and Couvidat 2015 [5], Bobra *et al*. 2016 [6], Nagem *et al.* 2018 [17], Poduval and Berger 2018 [20]). We have identified several deficiencies in these preliminary studies. First of all, these studies applied machine learning to few derived parameters from solar observations (with the assumption that these parameters are the only relevant ones). Second they did not have any dynamic information whereas, often information exists in the rate at which a parameters change in sequential images. Furthermore they did not incorporate any solar surface velocity information (dopplergram maps) which contains important information because acoustic wave propagation and turbulent flows are perturbed by strong near-surface magnetic field. Works using time series data have been extremely successful in predicting solar flares [26]. In this study we intend to progress in this field of using machine learning to improve our comprehension of solar activities by working on these deficiencies and producing better and improved models. While extensive work has been done in predicting the flaring potential of active regions, no tools are available to predict the emergence of the active regions, themselves. This study is meant to work on predicting the emergence of sunspots, which, being precursors of solar flares, would help in trimming lead times on flare detection and allow for better forecasts of space weather.

# Chapter 2

# Data Selection and Processing

## 2.1 Description of Data

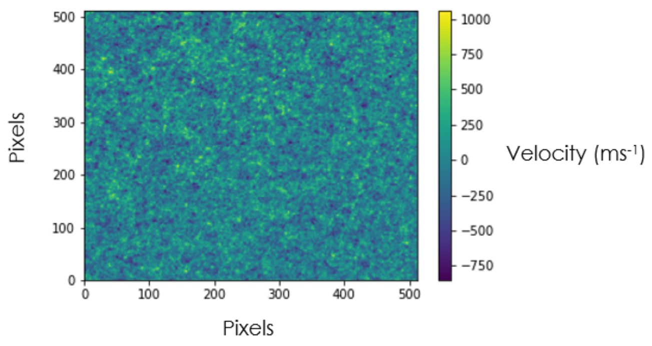
Our study considers data from December 2011 to November 2017. Our dataset consists of 382 positive and negative regions. A positive region is one where a sunspot emerged and a negative region is one with relatively low magnetic activity and/or no sunspot. Out of the 382 regions, 185 were from the northern hemisphere and 197 were from the southern hemisphere. Keeping the numbers similar from both hemispheres was ensured to avoid bias in the model based on the location of the region.

For this work we used Dopplergram patches of regions we were interested in. The selection of these regions will be expanded upon in the next section. They are a commonly used in the field of Helioseismology, a field that deals with detecting how sound waves interact with the Sun's interior structure, especially magnetic fields. We chose to use these for our study because the emerging sunspot region should boost the local speed of sound, thereby hastening the refracted return of sound waves passing through that part of the solar interior. This should lead to an identifiable signature which our machine learning models should learn to identify and use to distinguish dopplergrams of regions which are going to experience sunspot emergence from regions which will not.

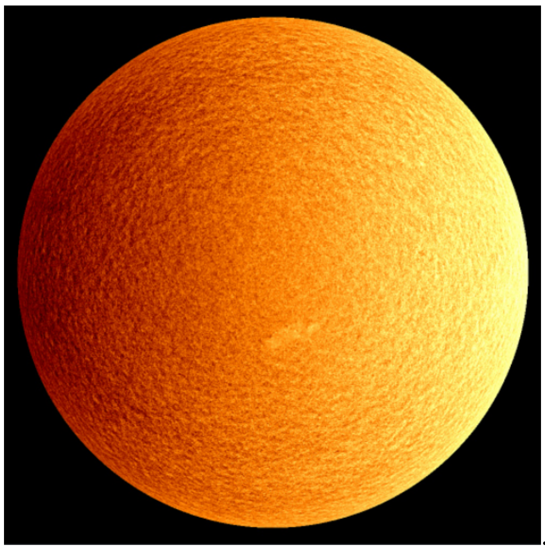
| | |
|---|---|
| series | = hmi.V_45' |
| duration | = 76 hour |
| cadence | = 1800 sec |
| columns | = 512 |
| rows | = 512 |
| scale | = 0.0301 |

TABLE 2.1: Data Export Parameters of the Dopplergrams.

The data we are using was provided by the Helioseismic and Magnetic Imager (HMI) aboard Nasa's Solar Dynamics Observatory (SDO) [23]. It was obtained from the the Joint Science Operations Center (JSOC) maintained by Stanford at http://jsoc.stanford.edu/. The JSOC database contains data products froma variety of missions and instruments, including the HMI. Parameters of the data used in this study are detailed in the table 2.1. For the purpose of our work, we used the *hmi.V_45s* dataseries. This series contains Dopplergrams with a cadence of 45 seconds.To create this dataseries, the Helioseismic Magnetic Imager (HMI) aboard the Solar Dynamics Observatory takes a series of images every 45 seconds in an extremely narrow range of wavelengths in visible light of the solar photosphere.



(A) Patch



(B) Full Disc

FIGURE 2.1: A typical LOS Dopplergram [1]

The wavelengths correspond to a region around the 6173 Angstroms spectral line of neutral iron. From the images it captures, the instrument constructs a set of images which extract other characteristics of the photosphere. For this Dopplergram dataset, it calculates the velocity of gas flows on the surface of the Sun using by analysing the

[1]These Dopplergrams have been taken from JSOC. Image Credit: NASA/SDO/AIA/HMI/Goddard Space Flight Center.

shifting of spectral lines. This spectral line shift is due to the Doppler effect, hence the name Dopplergrams.

## 2.2 Selection criteria and Data Acquisition

For the purpose of this study the first step was to create a dataset of active and inactive regions. To achieve this we used movies created via the Solar Monitor to look at the Sun from December 2011 to November 2017. Throughout this time period we categorised regions with high magnetic activity which experienced Sunspots in the active region category. To create the control group category we looked at geographically same regions during a different temporal period of relative inactivity. Through this process we created a dataset for training our models.

In the selection process, we implemented the criteria of only choosing regions where the Sunspot emerged in the Stonyhurst Coordinate bound of longitude and latitude $\pm 45^{\circ}$ This was done because after selection of the regions, we tracked them for 3+1 days. The tracking started 3 days before the Time of Emergence (ToE) and was done till 1 day after the emergence. The coordinate bound ensured that the regions being tracked will not go out of view in the relevant time duration. Moreover ensuring that the data is not leaning too much in the direction of the edges is better for data integrity and minimizes limb effects.

While the dataseries has images at 45 sec intervals, for our study we used dopplergrams of 30 min intervals. This was done to reduce the storage and processing power required for the model to work. As detailed in table 2.1, for each region, dopplergrams were acquired for a duration of 76 hours (3+1 days), with cadence of 1800 sec. These patches were of size 512 x 512. As detailed in the code in Appendix A.1, Postel projection was also done as part of the request queries submitted to JSOC for processing. The reason for choosing the postel projection is that it has the advantages of all points on the map being proportionally accurate distances from the centre point and all points on the map being in the correct direction.

As mentioned before, the data was acquired from JSOC. This was done using Python scripts for requesting queries and downloading them post processing by JSOC .The script for submitting the request to JSOC has been provided in Appendix A.1. This script can be referenced for the relevant parameters, also provided in 2.1 and the modules used (like drms). Further details regarding getting data are also available at http://jsoc.stanford.edu/How_toget_data.html. Once the query has been submitted and processed, we can download the data. I have also provided the script I used for downloading the data in Appendix A.2. The data we get from the site is in the form of Flexible Image Transport System (FITS) files. This format is regularly used in astronomy and was developed as a standard specifically for astronomical data. It is useful for concise storage of data and provides all the relevant metadata about the image in the form of headers. Once the data has been downloaded, we need to extracting the relavant matrix from this files so that they can be used for analysis by the models. The script for achieveing this has also been provided in Appendix A.3. This script also splits the data into a training set and a testing set where the size of both sets can be specified by the user. Once these three scripts have been used, your data is ready for being used in a machine learning model of your choice.

Once the data had been acquired and processed, we used machine learning models to attempt to use it for achieving our goal of predicting the emergence of sunspots. Neural Networks were our choice of model for doing this task. Machine learning is a desirable path for this study because of the large scope of the data. Before getting into the details of how we used the data with machine learning models, lets first discuss a little bit about the models themselves in the next section.

# Chapter 3

# Machine Learning Models

## 3.1 Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data[15]. The machine model uses sample data, termed as training data to *learn* and improve at doing desired tasks. These tasks might involve prediction, data generation, extrapolation among others. These algorithms are used in scenarios where using conventional models is difficult. There are three broad approaches in Machine Learning.

- Supervised learning: In this approach the model is given training data with labels. The models learns the rule to map the data to the labels. Then it can be tested on test data where the model is given data without labels and it has to figure out the label using the rule it has learned [22]. Supervised learning algorithms can be used in solving a variety of problems including but not limited to active learning, classification and regression [2].

- Unsupervised learning: In this approach, the model is given training data without any labels. The model attempts to identify the constitution of the data and find structure in it. There is no feedback involved here as opposed to Supervised Learning. The field of density estimation in statistics, such as calculating the probability density function, is a key application of unsupervised learning [25].

- Reinforcement learning: In this approach, the model has to interact with a dynamic environment. The model is given a goal it has to achieve, some metric it has to maximise or minimise, and it figures out by interacting with the data, the most efficient path to achieving that goal. This approach works on a rewards based system where the machine is given reward as feedback as it gets better at achieving the given task [4].

## 3.2 Neural Networks

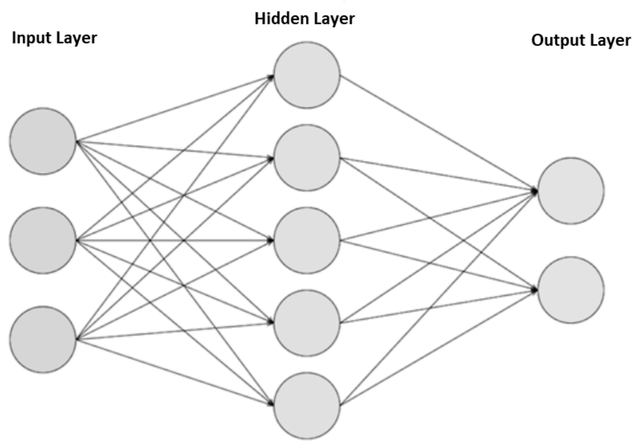


FIGURE 3.1: Neural Network Schematic

A widely used example of Machine learning models are neural networks. They are named based on the similarity of the architecture of a standard artificial neural network and the biological network of neurons in a brain, as seen in the figure 3.1. Here each circle denotes a node, which can be looked at as an individual neuron in a brain. Despite the visual resemblance, the working of an artificial neural network has little in common with a brain.

A neural network has three layers ,

- Input Layer: As is evident from the name, accepts input for the model.
- Hidden Layer: There can be one or more hidden layers in a neural network. They are the site of the calculations that allow the network to learn patterns.
- Output Layer: The input having been transformed by the hidden layers, reaches the output layer which gives out the output. This output can be of a number of forms.

To better understand the working of a neural network we shall further look at the building blocks of a neural network, the nodes. As mentioned above, neural networks are made up of nodes, which receive input, use an activation function to mix it with their internal state and an optional threshold, and then use an output function to produce output. External data, such as photographs and papers, are the first inputs. The final outputs, such as recognising an object in an image, complete the task. The activation function is notable for providing a smooth transition as input values change, i.e. a tiny change in input results in a small change in output. The network is also made up of connections, each of which serves as an input to another neuron by passing the output of one neuron. Here, every individual link is assigned a certain value called the weight which indicates its relative relevance. Multiple input and output connections are possible for a single node. The lines connecting the circles in figure 3.1 represent the connections. And lastly we have the propagation function, which computes a node's input as a weighted sum of its prior nodes' outputs and connections.

To accomplish its goal of learning, a neural network uses a number of parameters.

- Learning Rate: This determines how large the model's corrective steps are in adjusting for faults in the context of every single input. A large value of the learning rate reduces the amount of time it takes to train the model but reduces overall accuracy, whereas using a small value for the learning rate results in taking longer time to train but this route allows for the possibility of a much better accuracy as compared to a large learning rate.

- Cost Function : We strive to minimise a cost function to improve the network's performance at the relevant task. This is dependent on what the neural network is intended to perform. And lastly,

- Backpropagation : Backpropagation is a technique for adjusting connection weights to compensate for learning errors. The quantity of error is effectively distributed throughout the connections. Backprop estimates the gradient of the cost function associated with a particular state in relation to the weights in a technical sense. Many approaches, such as stochastic gradient descent, can be used to update the weights.

There are many kinds of neural networks, built and designed for being better at specific types of tasks. For example dense neural network is one where all the layers are fully connected. That is, ever node is connected to every other node in the adjacent layers. Because of their relevance to our work, we are going to elaborate on two types of neural networks in the coming sections, Convolutional neural networks and Long Short Term Memory neural networks. The first one is designed to be better at handling images, while the latter outperforms the competition when it comes to data in the form of time series. Since this study primarily focuses on image based data, lets understand Convilutional Neural Networks a little better.

## 3.3 Convolutional Neural Networks

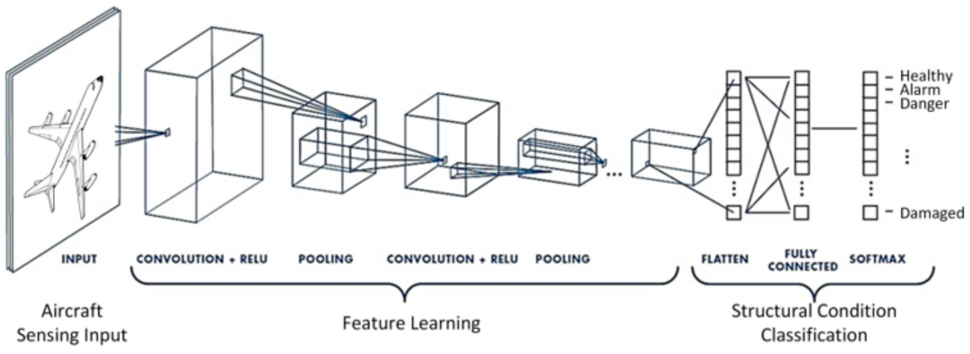


FIGURE 3.2: Convolutional Neural Network Schematic.
Credits : Tabian et al.[24]

Convolution is a method of producing a third array of numbers by multiplying two arrays of numbers, usually of different sizes. It is the process of adding each element of an image to its nearby neighbors, weighted by a matrix called the kernel, in the context of image processing. This can be used for reducing the size of an image. Traditionally, kernels have to be defined analytically to perform specific operations on the image for size reduction. In Machine Learning, however, a kernel of the desired size is initialized with randomized values, and through iteration, the values are optimized for the best possible classification. We essentially extract the critical features in the image through convolution and then feed them to the remaining layers which are vanilla dense neural networks.

A convolutional neural network is essentially a modified dense neural network designed for working better with images as input data. As seen in the figure 3.2, there is a classification section which is very similar to how the standard dense neural network works. The feature learning part is what sets CNNs apart. They have two main layers,

- Convolution layer : This layer takes a tensor as input. This tensor has the dimensions (number of inputs) x (input height) x (input width) x (input channels). The convolution layer implements convolution on the tensor as described above and outputs an abstract feature map. Here, much like the dense neural network, the nodes start with randomized weights and through the learning process using the Backpropagation algorithm, the model figures out the optimum kernel which provides the best results for the overall model. This has an advantage over the traditional convolution process where one has to figure out what kernel to use through previous experience with no systematic approach.

- Pooling layer : By merging the outputs that we get from a group of neurons from a single layer and feeding it into an individual neuron in the adjacent layer of the model, pooling layers minimise the dimensionality of data. Small clusters are combined via local pooling. The feature map's neurons are all affected by global pooling. There are two types of pooling that are commonly used: maximum and average. The maximum value derived from each locally present group of neurons in the feature map is used in max pooling, while the average value is used in average pooling as seen in the figure 3.3.

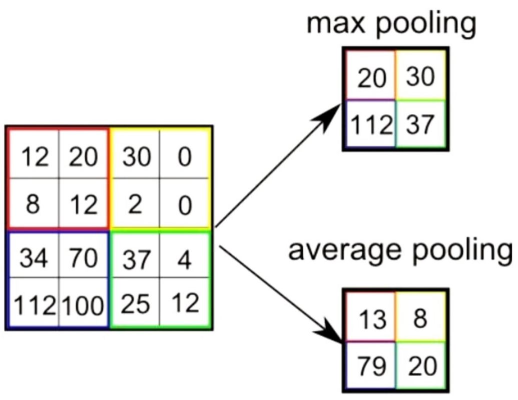


FIGURE 3.3: Working of the Pooling layer in a CNN

Once the images are processed through the convolution and pooling layers, and their features are abstracted, the result is then fed into the next part of the neural network which is just like a normal dense neural network. Through this architecture, CNNs boast of much better results over dense neural networks in dealing with images. They have an additional advantage of reducing the size of the images in the feature learning process leading to lesser commputational resources required for the model to work.

# Chapter 4

# Analysis and Results

## 4.1 Methodology

This project was primarily aimed at using machine learning methods to conduct data analysis of Dopplergrams to predict the emergence of Sunspots. After acquiring the data as detailed in in Chapter 2, we used the machine learning models we described in Chapter 3 to perform our study. The study can be divided into two approaches,

- Single Image
- Snapshot Extraction from Time Series

In the first approach, we used Dopplergrams of Time of Emergence of the Sunspot and fed them to the model. We used the code detailed in Appendix A.3 to convert the FITS files acquired from JSOC and fed them to a Convolutional Neural Network. The dataset was then split into a training set and a test set in the ratio 70:30. This is a common way of using CNNs for image classsification. The neural network creates a feature map of the images and based on this, attempts to segregate dopplergrams with sunspot emergence from ones with no sunspot. Similar work has been done by Dhuri et. al [10] using Magnetograms instead of Dopplergrams with promising results.

In the second approach, we used a time series. As detailed in section 2.2 we tracked the active regions for 3+1 days. So we had dopplergram time series with a cadence of 45 sec. For the purpose of this study we used a cadence of 30 minutes to reduce computational resource requirements and attempted to extract a meaningful snapshot from this time series to then feed to the convolutional neural network model. Our method of extracting this snapshot was taking the variance of each pixel value of a dopplergram across the time series as shown in the figure 4.1

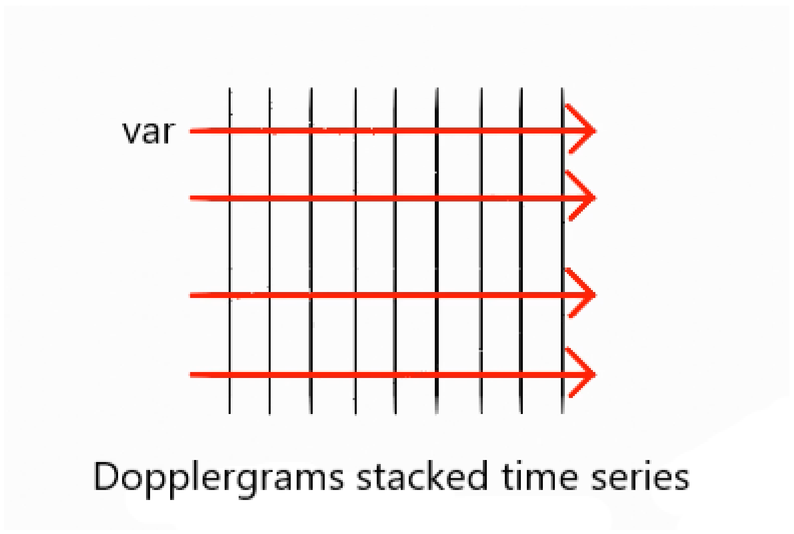


FIGURE 4.1: Representation of the snapshot exraction process

Once the extraction was done and we had a snapshot representing the time series of all the regions in the dataset, we followed the procedure of the previous approach and fed this snapshot to a CNN model. For help in understanding of the study, a sample CNN model used in the study has been included in Appendix A.4

## 4.2 Results

The approaches undertaken in this study did not yield promising results. The methodology detailed in the previous section was performed 5 times to maintain statistical integrity. The results that we got are detailed in the table 4.1. The mean accuracy was less than ideal and the model was not performing as it should to make its usage feasible in the real world.

| | Mean Accuracy | Standard Deviation |
|---|---|---|
| Single Image | 0.446 | 0.027 |
| Snapshot Extraction | 0.549 | 0.039 |

TABLE 4.1: Mean Accuracy of the Convolutional Neural Networks.

# Chapter 5

# Conclusion

The study did not lead to desired results. Using dopplergrams alone with neural networks does not seem to be adequate for the model to be able to differentiate between regions with sunspot emergence and regions without sunspot emergence. It is a difficult problem and requires more detailed investigation. One possible area that can be explored in the future is using Long Short Term Memory (LSTM) neural networks as they are more suitable for handling time series.

As magnetograms being used in a similar fashion with convolutional neural networks have given positive results [10], incorporating this data into the input for our models might also prove useful.This would help improve the robustness of the model. Furthermore, the results gained from including magnetic field data will hopefully allow us to get better insight into the solar processes that drive sunspot emergence.

For now, dopplergrams cannot give us a data driven model for accurately predicting the emergence of sunspots on the surface of the sun. But it remains an interesting problem worth exploring.

# Appendix A

# Codes

## A.1 Sending requests for Data

The following code was used for submitting requests for the Dopplergrams used in the study to JSOC.

```python
# -*- coding: utf-8 -*-

#importing relevant modules
from astropy.io import fits
from astropy.coordinates import SkyCoord
import astropy.units as u
from datetime import datetime as dt_obj
from datetime import timedelta
import drms
import json
import numpy as np
import pandas as pd
from pandas.plotting import register_matplotlib_converters
from pathlib import Path
import os.path
import os
from datetime import datetime
from scipy import stats
import sunpy.map
from sunpy.coordinates import frames
from sunpy.time import TimeRange, parse_time
from shutil import copyfile
import tarfile
import time
from tqdm import tqdm, trange
import urllib
import urllib.parse as parse

datadir = '/content/drive/My Drive/cessi_ms/requests_inactive/'
ardatafile = 'inactive.txt'          # dataset for download
```

```
col_names  =['sno', 'harp','toe','lon','lat'] # columns in the input file
ardata     = pd.read_csv(datadir + ardatafile,sep='\t',header=None, names=col_names) # read to dataframe
#--columns of input file---
noaa_list  = ardata['harp']
date_list = drms.to_datetime(ardata['toe'])


lon_list   = ardata['lon']
lat_list   = ardata['lat']
#----
print(date_list)
count      = len(noaa_list) # number of entries
ardata # printing the dataframe

# ======== Data export parameters ========

series    = 'hmi.V_45s'  # dataseries
ntime     = 56             # we split the download duration into 56 segments
dur       = '76h'         # each download segment covers 3h data . Changed this to 76h, so that in one  batch we have data for whole 76h at every 30min
cad       = '1800s'       # cadence for download  Changed this to 30 min
cols      = 512           # dimension
rows      = 512
scale     = 0.0301

op        = 'exp_request'
#---------------------------------------------To be changed
email     = 'astrophysics.vishal@gmail.com'           #  jsoc registered id
#---------------------------------------------
method    = 'url-tar'                  # url or url-tar
protocol = 'FITS,compress Rice'
ffmt      = '{seriesname}.{T_REC:A}.{segment}'
oformat   = 'txt'

JSOC      = 'http://jsoc.stanford.edu'
FETCH     = JSOC + '/cgi-bin/ajax/jsoc_fetch'
# field     = 'field'
# azim      = 'azimuth'
# incl      = 'inclination'
# disamb    = 'disambig'
```

```python


reqfile  = datadir  + 'inactive.txt'   #!!! this file will contain the list of jsoc request id's
errfile  = datadir + 'inactive_errors.txt'      # the cases of export errors will be saved here
#---------------------------------------------

Path(reqfile).touch()  # create empty request and error files
Path(errfile).touch()
timestep=3;

def jsoc_request(i, noaa,datetime,lon,lat):
    ds       = '{0}[{1}/{2}@{3}][?Quality>=0?]'.format(series,datetime,dur,cad)

    #--- update the process if required -----
    process  = 'n=0|Maproj,map=Postel,clon={0},clat={1},scale={2},cols={3},rows={4}'    .format(lon,lat,scale,cols,rows)
    #process='n=0|no_op'
    #-----------------------------------------

    RESP     = '/tmp/jsoc_export.'+str(os.getpid()) + '_' + str(i)
    if os.path.exists(RESP):
        os.remove(RESP)

    # url parsing all the inputs
    ds_pq        = parse.quote(ds)
    notify_pq    = parse.quote(email)
    protocol_pq = parse.quote(protocol)
    ffmt_pq      = parse.quote(ffmt)

    # setting up the wget download
    cmd = 'op={0}&ds={1}&process={2}&method={3}&format={4}&protocol={5}&filenamefmt={6}&notify={7}&requestor=none'    .format(op,ds_pq,process,method,oformat,protocol_pq,ffmt_pq,notify_pq)
    #print(cmd)

    toget = '{0} {1}?"{2}"'.format(RESP,FETCH,cmd)
    os.system('wget -S -nv -O {0}'.format(toget))


    #--- reading the request response ---
    f     = open(RESP, 'r')
    line  = f.readline()
    while line:
        print(line,end='')
        line = f.readline()
        if "JSOC_" in line:
            requestid = line.split("=")[1].strip()
        if "size" in line:
            size = line.split("=")[1].strip()
            #print('\nrequestid= ', requestid)
        if "wait" in line:
            wait=line.split("=")[1].strip()

        if "status" in line:
            status=line.split("=")[1].strip()
    f.close()


    # if request fails or there is a queue- wait and note it in the error file
    try:
        in_status = int(status)
    except:
        with open(errfile, 'a') as f:
                f.write('{0}\t{1}\t{2}\t{3}\t{4}\n'.format(i,noaa,datetime,lon,lat))
        return None

    #-- print the request file path
    print('resp= ', RESP)
    print("="*10)
    #----
    # if the run is successful note the request id's in file
    with open(reqfile, 'a') as f:
        try:
            f.write('{0}\t{1}\t{2}\t{3}\t{4}\t{5}\n'.format(i,noaa,requestid, RESP,size,wait))
        except:
            pass
    return None

# ====== Loop for the JSOC requests ===========

#--------------------------------------------To be changed
n1 = 0          # noaa start number from the list; set 0 to start from begining
n2 = count        # noaa end number; set as variable count for taking all the entries
#--------------------------------------------
```

```
m1 = 0          # start number for download segment; typically 0
m2 = ntime      # end number for download segment; typically ntime

# the n,m  values can be changed to run for specific cases when
   error occurs

for i in range(n1, n2):

    dtime_1 = date_list[i]
    noaa   = noaa_list[i]
    lon    = lon_list[i]
    lat    = lat_list[i]
    dtime = dtime_1 + timedelta(hours=1*timestep)
    for j in range(m1, m2):
        print('i, j = {0}, {1}'.format(i, j) )
        dt1 = dtime - timedelta(hours=j*timestep)
        dtf = dt1.strftime('%Y.%m.%d_%H:%M:%S_TAI')
        index = str(i)+'_' + str(j)
        now = datetime.now()
        print('now = ', now)
        jsoc_request(index,noaa,dtf,lon,lat)
        time.sleep(100)
```

## A.2 Downloading the Data

Once the requests had been processed, the following code was used for downloading the data.

```
# -*- coding: utf-8 -*-

#importing relevant modules

from datetime import timedelta
from datetime import datetime as dt_obj
import drms
from IPython.display import clear_output
import numpy as np
import pandas as pd
from pathlib import Path
import os.path
import os
from shutil import copyfile
import tarfile
import time
import urllib
import urllib.parse as parse


p1 = 0         # start number from the request list
p2 = 382       # end number from the request list

email    = 'astrodata.vishal@gmail.com'  # JSOC email list
dir = '/home/vsingh/Documents/requests_inactive/' # where request
    and error files are saved
reqfile= dir+ 'north_request.txt'
outdir      = dir+'data/inactive_north/'
method   = 'url-tar'

if not os.path.exists(outdir):
            os.mkdir(outdir)

req = pd.read_csv(reqfile,sep='\t',header=None, names=['index','
    date', 'requestid', 'resp','size','wait_time'])

req['index'] = req['index'].apply(str) #split the i_j index into i
    and j
req[['i', 'j']] = req['index'].str.split('_',expand=True)
req = req.drop(['index'], axis=1)
cols=['i','j','date', 'requestid', 'resp','size','wait_time']
reqf = req.reindex(cols,axis=1)
reqf

reqf[reqf.duplicated('date',keep=False)]


reqf2=reqf.drop_duplicates(subset ="date", keep = 'first').
    reset_index(drop=True)
reqf2

ilist = reqf2['i']
jlist = reqf2['j']
requestlist = reqf2['requestid']
resplist    = reqf2['resp']
datelist = reqf2['date']

ival  = ilist.drop_duplicates().reset_index(drop=True)
icount = len(ival)

def index_data(inum):
    idf = reqf[reqf['i'] == str(inum)]
    return idf
```

```
icount

c   = drms.Client(email=email, verbose=True) # setup the drms client
def jsoc_download(date,request_id,outdir, num):
    num = num
    # create the individual folders for different index
    out_dir    = outdir + 'AR_'+ str(date)
    if not os.path.exists(out_dir):
            os.mkdir(out_dir)
    # create empty files for each request id to avoid repeat downloads
    export_file = out_dir+'/jsoc_export_'+ request_id
    if os.path.exists(export_file):
        print("Download already exists! Skipping!!!")
        time.sleep(5)
        return None

    # create the request object from id
    r=c.export_from_id(request_id)

    # check the download status
    check = r.status
    if check > 0:
        print('Status = ', check)
        print('\nDownload not ready. Try again later')
        return None

    # create a file containing details of all the downloaded files
    Path(out_dir+'files.txt').touch()
    r.data.to_csv(out_dir+'files.txt',header=True, index=False, sep='\t', mode='a')

    # download the files
    r.download(out_dir)

    # untar them if needed then delete the tar file
    if method == 'url-tar':
        tarf = out_dir+'/'+request_id+'.tar'
        outdir_tar = out_dir+'/'+request_id
        if not os.path.exists(outdir_tar):
            os.mkdir(outdir_tar)
        tf   = tarfile.open(tarf)
        tf.extractall(outdir_tar)
        tf.close()
    Path(export_file).touch()
    return None

print('i\tj\tdate\t')
for i in range(icount):
    inum   = ival[i]
    idf = index_data(inum)
    idate = idf.date.values[0]
#     itime = idf.time.values[0]
    fcount = len(idf)
    print('{0}\t{1}\t{2}'.format(inum,fcount,idate))

n1 = 0          # start number for index; = 0 if starting from
    begining
n2 = icount     # end number; = icount if doing all

# for loop for the download
for i in range(n1, n2):
    inum   = ival[i]
    idf = index_data(inum)
    idate = idf.date.values[0]
    ireq  = idf.requestid.values
    fcount = len(idf)

    m1 = 0          # start number for the segment; typically 0
    m2 = fcount     # end number; typically filecount, fcount


    for j in range(m1, m2):
        print('{0}\t{1}\t{2}/{3}'.format(i,idate, j+1, m2))
        num = str(j)
        request_id = ireq[j]
        now = dt_obj.now()
        print('now = ', now)
        jsoc_download(idate,request_id,outdir, num)
    #clear_output()
```

## A.3 Data Processing

The data we got from JSOC was in the Flexible Image Transport System (FITS) file format. To use it for analysis we had to extract the image matrices and for efficient usage of the data by the machine learning models, were converted to datacubes.

```
#data_prep

from astropy.io import fits
from random import randrange
import glob
import numpy as np
import pandas as pd
import scipy.io as sio

def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list

#active
datadir = '/home/ML/Output_files/harp_active/AR_'
ardatafile = '/MS/vs16ms007/CESSI/ML/active_harp_list.txt'
col_names  =['1','harp','3','4', '5', '6'] # columns in the input
    file
ardata      = pd.read_csv(ardatafile,sep='\t',header=None,names=
    col_names)
harp_list   = ardata['harp']
harp_list=Remove(harp_list)
fname_active=[]
for ar in harp_list:
    fn=glob.glob(datadir+str(ar)+'/*.fits')
    if len(fn)<3:
        continue
    fn=fn[1]
    fname_active.append(fn)

#inactive_north
datadir = '/home/ML/Output_files/harp_inactive/north/AR_'
ardatafile = '/MS/vs16ms007/CESSI/ML/north_2.csv'
col_names  =['harp','2','3','4', '5'] # columns in the input file
ardata      = pd.read_csv(ardatafile,sep='\t',header=None,names=
    col_names)
harp_list   = ardata['harp']
fname_inactive_north=[]
for ar in harp_list:
    fn=glob.glob(datadir+str(ar)+'/*.fits')
    if len(fn)<3:
        continue
    fn=fn[1]
    fname_inactive_north.append(fn)

#inactive_south
```

```python
datadir = '/home/ML/Output_files/harp_inactive/south/AR_'
ardatafile = '/MS/vs16ms007/CESSI/ML/south_2.csv'
col_names  =['harp','2','3','4', '5'] # columns in the input file
ardata     = pd.read_csv(ardatafile,sep='\t',header=None,names=
    col_names)
harp_list  = ardata['harp']
fname_inactive_south=[]
for ar in harp_list:
    fn=glob.glob(datadir+str(ar)+'/*.fits')
    if len(fn)<3:
        continue
    fn=fn[1]
    fname_inactive_south.append(fn)

#creating target array
target_active_unit=[1,0]
target_inactive_unit=[0,1]
fname_inactive=np.concatenate((fname_inactive_north,
    fname_inactive_south))
target_inactive=np.tile(target_inactive_unit,(len(fname_inactive)
    ,1))
target_active=np.tile(target_active_unit,(len(fname_active),1))



#creating data cube
data_cube=[]

c=0;
for fname in fname_active:
    hdul = fits.open(fname)
    hdul.verify("silentfix+warn")
    dat=hdul[1].data
    if c==0:
        data_cube=dat;
        c=1
    elif c==1:
        data_cube=np.dstack((data_cube,dat))
for fname in fname_inactive:
    hdul = fits.open(fname)
    hdul.verify("silentfix+warn")
    dat=hdul[1].data
    data_cube=np.dstack((data_cube,dat))


target=np.concatenate((target_active,target_inactive))
```

```
data_cube=data_cube.T

#Splitting data_cube into test and train
percent=0.70 #percentage of data used for training
train_len=int(percent*len(target))
x_test=[]
x_train=[]
y_test=[]
y_train=[]
randind=[]
init_len=0;
while(len(randind)<train_len):
    for i in range(init_len,train_len):
        randind.append(randrange(len(target)))
    randind=Remove(randind)
    init_len=len(randind)


#putting values into train
for i in range(0,len(target)):
    if i in randind:
        x_train.append(data_cube[i])
        y_train.append(target[i])
    else:
        x_test.append(data_cube[i])
        y_test.append(target[i])
x_test=np.asarray(x_test)
x_train=np.asarray(x_train)
y_test=np.asarray(y_test)
y_train=np.asarray(y_train)

sio.savemat('/home/ML/Output_files/active_region.mat', {'x_train':
    x_train,'x_test':x_test,'y_train':y_train,'y_test':y_test  })
```

## A.4 CNN

Here is a sample CNN code used by us in the study.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from keras.layers.convolutional import Conv2D # to add
    convolutional layers
```

```python
from keras.layers.convolutional import MaxPooling2D # to add
    pooling layers
from keras.layers import Flatten ,Dropout# to flatten data for
    fully connected layers
from scipy.io import loadmat
import numpy as np

ar_set = loadmat("/home/ML/Output_files/active_region.mat")
x_train =  ar_set["x_train"].astype(np.float32)
x_test =  ar_set["x_test"].astype(np.float32)
y_train =  ar_set["y_train"].astype(np.float32)
y_test=  ar_set["y_test"].astype(np.float32)

# reshape to be [samples][pixels][width][height]
x_train = x_train.reshape(x_train.shape[0], 512, 512, 1).astype('
    float32')
x_test = x_test.reshape(x_test.shape[0], 512, 512, 1).astype('
    float32')

#normalize the data
scal=np.max((np.max(x_train),np.max(x_test)))
x_train = x_train / scal
x_test = x_test / scal

#y_train = to_categorical(y_train)
#y_test = to_categorical(y_test)
num_classes = y_test.shape[1] # number of categories

#works
def convolutional_model():

    # create model
    model = Sequential()
    model.add(Conv2D(16, (5, 5), activation='relu', input_shape
   =(512, 512, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(8, (2, 2), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Flatten())
    model.add(Dense(600, activation='relu'))
#     model.add(Dropout(0.5))
#     model.add(Dense(300, activation='relu'))
#     model.add(Dense(600, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    # Compile model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# build the model
model = convolutional_model()

# fit the model
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=1, batch_size=40, verbose=2)

# evaluate the model
scores = model.evaluate(x_test, y_test, verbose=0)
print("Accuracy: {} \n Error: {}".format(scores[1], 100-scores[1]*100))
```

# Bibliography

[1] Omar W. Ahmed et al. “Solar Flare Prediction Using Advanced Feature Extraction, Machine Learning, and Feature Selection”. In: *Solar Physics* 283.1 (2011), pp. 157–175. DOI: 10.1007/s11207-011-9896-1.

[2] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning series)*. second edition. The MIT Press, 2009.

[3] John A. Armstrong and Lyndsay Fletcher. “Fast Solar Image Classification Using Deep Learning and Its Importance for Automation in Solar Physics”. In: *Solar Physics* 294.6 (2019). DOI: 10.1007/s11207-019-1473-z.

[4] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2011.

[5] M. G. Bobra and S. Couvidat. “SOLAR FLARE PREDICTION USINGSDO/HMI VECTOR MAGNETIC FIELD DATA WITH A MACHINE-LEARNING ALGORITHM”. In: *The Astrophysical Journal* 798.2 (2015), p. 135. DOI: 10.1088/0004-637x/798/2/135.

[6] M. G. Bobra and S. Ilonidis. “PREDICTING CORONAL MASS EJECTIONS USING MACHINE LEARNING METHODS”. In: *The Astrophysical Journal* 821.2 (2016), p. 127. DOI: 10.3847/0004-637x/821/2/127.

[7] Axel Brandenburg and Kandaswamy Subramanian. “Astrophysical magnetic field and nonlinear dynamo theory”. In: *Phys. Rep.* 417 (June 2004). DOI: 10.1016/j.physrep.2005.06.005.

[8] Paul Charbonneau. “Dynamo Models of the Solar Cycle”. In: *Living Reviews in Solar Physics* 7 (2010). DOI: 10.12942/lrsp-2010-3.

[9] M. C. M. Cheung, M. Schüssler, and F. Moreno-Insertis. “Magnetic flux emergence in granular convection: radiative MHD simulations and observational signatures”. In: *Astronomy  Astrophysics* 467.2 (2007), pp. 703–719. DOI: 10.1051/0004-6361:20077048.

[10] Dattaraj B. Dhuri et al. “Application and Interpretation of Deep Learning for Identifying Pre-emergence Magnetic Field Patterns”. In: *The Astrophysical Journal* 903.1 (2020), p. 27. DOI: 10.3847/1538-4357/abb771.

[11] T. Gold and F. Hoyle. “On the Origin of Solar Flares”. In: *Monthly Notices of the Royal Astronomical Society* 120.2 (1960), pp. 89–105. DOI: 10.1093/mnras/120.2.89.

[12] Michael Hahn et al. “Spatial Relationship between Twist in Active Region Magnetic Fields and Solar Flares”. In: *The Astrophysical Journal* 629.2 (2005), pp. 1135–1140. DOI: 10.1086/431893.

[13] Soumitra Hazra, Dibyendu Nandy, and B. Ravindra. “The Relationship Between Solar Coronal X-Ray Brightness and Active Region Magnetic Fields: A Study Using High-Resolution Hinode Observations”. In: *Solar Physics* 290.3 (2015), pp. 771–785. DOI: 10.1007/s11207-015-0652-9.

[14] S. Ilonidis, J. Zhao, and A. Kosovichev. “Detection of Emerging Sunspot Regions in the Solar Interior”. In: *Science* 333.6045 (2011), pp. 993–996. DOI: 10.1126/science.1206253.

[15] Tom Mitchell. *Machine Learning*. 1st ed. McGraw-Hill Education, 1997.

[16] Dhrubaditya Mitra et al. “Intense bipolar structures from stratified helical dynamos”. In: *Monthly Notices of the Royal Astronomical Society* 445.1 (2014), pp. 761–769. DOI: 10.1093/mnras/stu1755.

[17] Tarek A.M. Nagem et al. “Deep learning technology for predicting solar flares from (Geostationary Operational Environmental Satellite) data”. In: (2018). URL: http://hdl.handle.net/10454/15683.

[18] Dibyendu Nandy, Andrés Muñoz-Jaramillo, and Petrus C. H. Martens. “The unusual minimum of sunspot cycle 23 caused by meridional plasma flow variations”. In: *Nature* 471.7336 (2011), pp. 80–82. DOI: 10.1038/nature09786.

[19] Sanchita Pal et al. “Dependence of Coronal Mass Ejection Properties on Their Solar Source Active Region Characteristics and Associated Flare Reconnection Flux”. In: *The Astrophysical Journal* 865.1 (2018), p. 4. DOI: 10.3847/1538-4357/aada10.

[20] Bala Poduval and Tom Berger. “Predicting Solar Eruptive Events Using Artificial Neural Networks”. In: *Solar Heliospheric and INterplanetary Environment (SHINE 2018)*. July 2018, p. 157.

[21] Russel. “Geomagnetic Storms”. In: *Windows to the Universe. National Earth Science Teachers Association.* (Mar. 2010).

[22] Stuart Russell et al. *Artificial Intelligence*. Upper Saddle River, NJ, United States: Prentice Hall, 2010.

[23] P. H. Scherrer et al. “The Helioseismic and Magnetic Imager (HMI) Investigation for the Solar Dynamics Observatory (SDO)”. In: *Solar Physics* 275.1-2 (2011), pp. 207–227. DOI: 10.1007/s11207-011-9834-2.

[24] Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei. “A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures”. In: *Sensors* 19 (Nov. 2019), p. 4933. DOI: 10.3390/s19224933.

[25] Allen Tucker. *Computer Science Handbook, Second Edition*. 2nd ed. Chapman and Hall/CRC, 2004.

[26] Xiantong Wang et al. “Predicting Solar Flares with Machine Learning: Investigating Solar Cycle Dependence”. In: *The Astrophysical Journal* 895.1 (2020), p. 3. DOI: 10.3847/1538-4357/ab89ac.