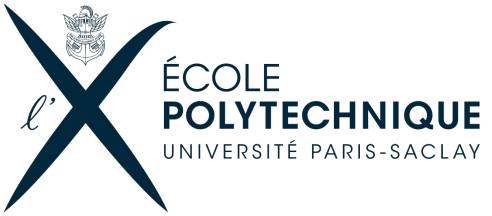
Ecole polytechnique

Research internship report

Deep Solar Imaging for Geomagnetic Storms Prediction



Student : Mhamed Hajaiej X2014
Supervisor : Cyril Furtlehner
Referent : Yanlei Diao
Organism : Inria Saclay
Date : April - July 2017

Abstract

The solar activity has a cycle of around 11 years during which the geomagnetic storms increases to a peak. The geomagnetic storms affect the Earth's magnetic field and can damage the electrical systems on Earth. However, they are still considered to be unpredictable events and the current physical models are not sufficient to explain the complexity of the Sun's magnetic field. In order to have a better understanding of the solar activities, space mission were lunched to survey the Sun. They provide solar images from different detectors as well as space weather indicators measured in the environment surrounding the Earth. We exploit the data provided by those missions. We explore the deep autoencoder approach to provide compact representation of the solar magnetogram images and we extract information about the solar coronal mass ejection. Those representations are correlated later with the solar activity indicators.

Contents

1 Introduction

This report summarize the work done during the 4 months internship at **Inria** Saclay with the **TAU** team : a research team that focuses on machine learning. This internship takes place in the context of an international collaboration with the **CWI** (National Research Institute for Mathematics and Computer Science) in Amsterdam. The project is called Machine Learning for Space Weather. It aims to couple physics-based simulations with Artificial Intelligence in order to enhance the current state-of-the-art simulation for Space Weather and to create a 'grey-box' model that can decide the relative importance between physical and empirical estimations depending on real-time conditions observed by the satellites.

In addition to its scientific interest from the point of view of fundamental physics, space weather may have a big influence in our technology dependent society as it has the ability to damage satellites and electronic circuits. This damage can affect the space based technology like the GPS, but also ground based technology and even the electrical grid. One long-term goal of the project is to find an efficient method to predict when the solar state is likely to affect the Earth magnetic field in order to be able to prevent damages to satellites and electronic networks. Traditional physics methods are based on magneto-hydrodynamic equations but are not capable of fully solving the problem. Considering the big database of solar images, we think that a data driven approach may help to make progress on this problem.

Our contribution to the project consists in applying deep learning techniques to tackle the geomagnetic storms' prediction problem. Thanks to the progress in the computational power and an increase in the quantity of available data, the deep learning techniques have recently achieved spectacular results in many domains and especially in image recognition. In this internship, we take advantage of the various solar images made available by the spacecrafts **SOHO** (Solar and Heliospheric Observatory) launched in 1995 and more recently **SDO** (Solar Dynamics Observatory) launched in 2010. We explore these images, and their relation with the spatial weather and the geomagnetic storms.

In the two first sections of this report, we introduce the reader to the subject. We start by presenting general information about the spatial weather and the available images. We then present the deep learning basics and the main layers that we are using in our work. Afterwards, we focus on unsupervised learning for finding compact representations of a particular type of solar images. Finally, we present the current progress in the spatial weather prediction.

2 Spatial weather knowledge and the solar images

2.1 Space weather

Space weather can be defined as the study of the Sun and the space environment surrounding the Earth and of their evolution in time.

2.1.1 Origin

Constantly, a stream of charged particles like electrons and protons is released by the Sun. This stream is called the solar wind. In addition to the charged particles, the solar wind also contains the **HMF** (heliospheric magnetic field) : a component of the solar magnetic field dragged by the solar wind from the Sun. This solar wind interacts with the Earth's magnetic field causing temporary disturbances that we call geomagnetic storms. In fact, the Earth's magnetic field protects the Earth from the charged particles of the solar wind and the cosmic rays that can destroy the atmosphere including the ozone layer. During geomagnetic storms the Earth's magnetic field is weakened.

2.1.2 Solar cycle

The solar sunspot cycle is a cycle of around 11 years. During this cycle, the solar activity goes from a minimum to a maximum during which the number of the sunspots, the solar flares and the geomagnetic storms increases, and then goes down again. Early in the cycle, sunspots appear in the higher latitudes and then move towards the equator as the cycle approaches its maximum. We can see this periodical structure in the sunspot butterfly diagram of **Fig 1**

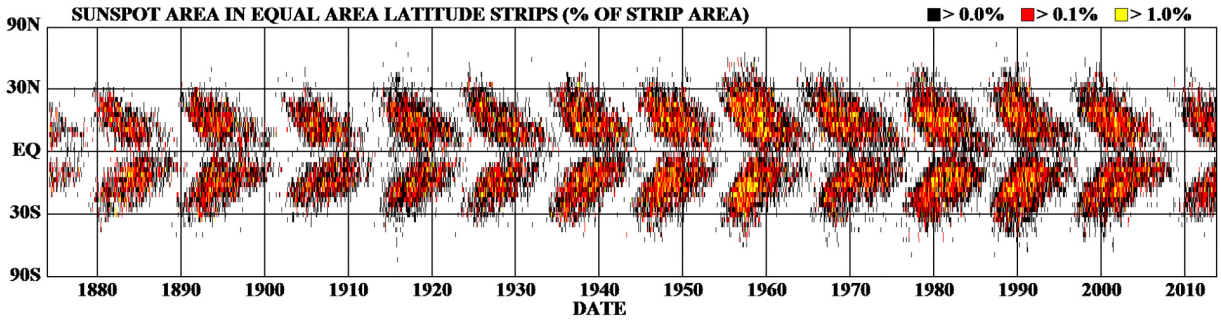


Figure 1: Sunspot butterfly diagram. The percentage of the sunspot area as a function of time (x axis) and of the latitude (y axis)

2.1.3 Solar wind :

The solar wind is a stream of charged particles released from the sun. It can be divided into two categories depending on its speed:

- Slow solar wind : around 300-500 Km/s, mainly created by active regions in the Sun called "streamer belt"
- Fast solar wind: around 750 Km/s, mainly created by active regions in the Sun called "coronal holes"

2.1.4 Solar activity

Several phenomena are responsible for the solar activities and can affect the space magnetic field. For example we have :

- Coronal holes : Areas of the Sun's corona allowing a great number of particles to escape from the Sun. They contribute to the solar wind with particles traveling with the fast solar wind speed.
- Coronal mass ejection : A **CME** (coronal mass ejection) is an unusually large release of plasma and magnetic field from the solar corona. We can observe around 3/day at the solar maximum, and 1 every 5 days at the solar minimum. They contribute to solar wind and may cause geomagnetic storms.

2.2 The geomagnetic storms' danger

The geomagnetic storms have negatives effects on electrical systems. For example, it affects the radio communications and the navigation systems like GPS. It also creates an induced current that causes damage to the electricity grid (like the Quebec incident in 1989). Finally, it damages satellites and spacecrafts by causing damage to their electrical systems and also by slowing down satellites and making them change orbit slightly.
Some dangerous solar storms happened in the past like the solar storm of 1859 also known as the Carrington event and the one of 2012 which fortunately missed the earth. It' estimated that the economic cost of a geomagnetic storm of this scale nowadays can rise to billions, or even trillions US dollars.

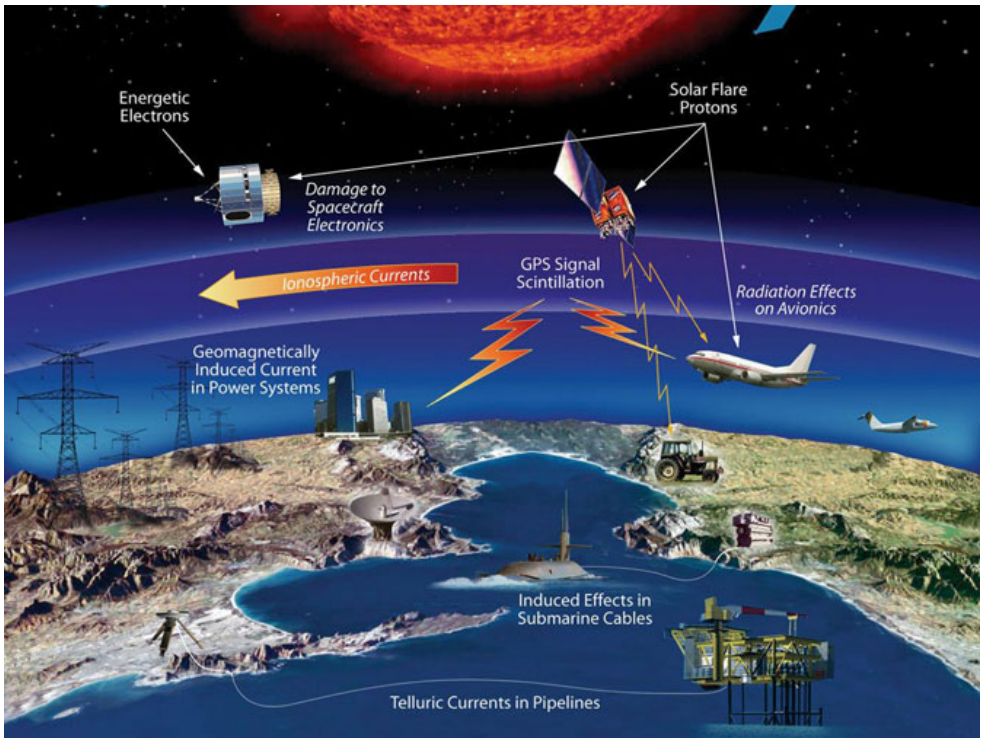


Figure 2: The potential effects of the geomagnetic storm on Earth

2.3 The data

The data is a set of images of the Sun taken regularly by different instruments. Some instruments take an image every few minutes, other every few hours. The three main instruments are :

- **LASCO** (The Large Angle and Spectrometric Coronagraph) which monitor the solar corona by using an optical system to create an artificial solar eclipse : in these images we can see the particles' ejection from the Sun. These images are taken every 30 minutes.**Fig 3**
- **EIT** (Extreme ultraviolet Imaging Telescope) which produces images of the Sun at wavelength like at 284Å and 304Å. Different wavelength correspond to different depths near the solar surface : it allows us to see the active regions which may release significant amounts of energy and particles during the solar events. These images are taken every 6 hours.**Fig 4**
- **MDI** (Michelson Doppler Imager) which produces a magnetogram of the Sun showing the polarity and the intensity of the Sun magnetic fields : it allows us to see the sunspots and the active regions. These images are taken every 90 minutes.**Fig 5**

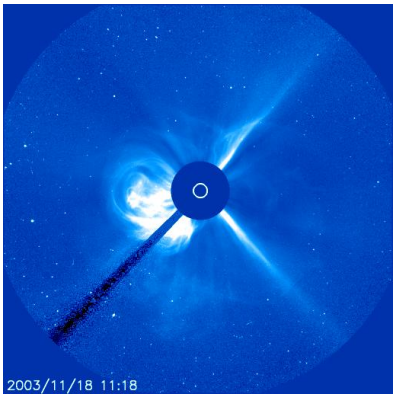


Figure 3: LASCO C3 image

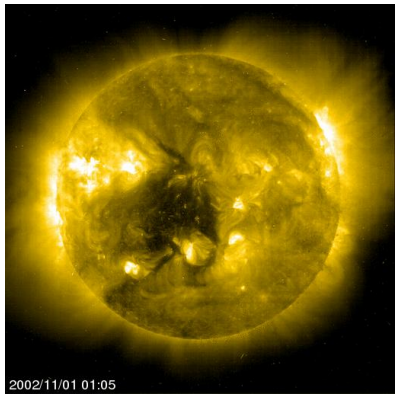


Figure 4: EIT 284 image

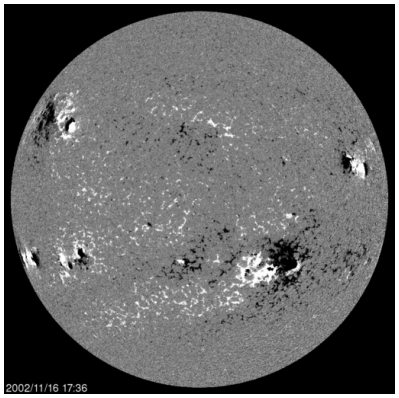


Figure 5: HMI Magnetogram

The images are available in 512x512 and 1024x1024 resolution. The 512x512 are chosen for computational efficiency.

In addition to these images, we can use the data of the solar wind and the space weather taken regularly at the point L1: Lagrange point between Earth and the Sun, located at approximately 1/100 of the Earth-Sun distance. This data is available as a per hour average. Finally, we have a small set of solar activity events that was manually labeled by domain experts. We chose to start working with the data of the solar cycle from 1996 to 2007 as it is completely covered by SOHO.

3 Deep learning

3.1 Definition

Deep learning is a branch of machine learning that is based on artificial neural networks. Neural nets are composed of a sequence of layers. Every layer takes as inputs the previous layer and uses it to compute its value. The general equation is : $X_i = f_i(W_i X_{i-1} + h_i)$ where X_i and X_{i-1} are the value of the current layer and the previous layer, W_i represents the weights, h_i the bias and f a non linear function called the activation function. In fact, researches have proved that using non linear functions between the layers, the neural net can approximate any continuous function. Two of the most famous activation functions are the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and the rectified linear unit **ReLU** $f(x) = max(0, x)$. W_i and h_i are the parameters of the model, they are randomly initialized before being learned during the training of the neural net.

3.2 Training the network

Training a neural net is an optimization problem where we use stochastic gradient descent techniques in order to find the best parameters. During the training, the data is passed to the neural net in batches, and a pass of all the training data set is called an epoch. The training is done using the backpropagation algorithm : first the data is fed forward. We start by the input layer and every layer computes its values using the previous one until we reach the output. Once we reach the output, we compare the computed output to the desired one and we obtain an error using a loss function. Afterwards, we use this error and the chain rule in order to compute the gradient and apply the gradient descent to update the weights and to minimize this error.

3.3 Useful layers

3.3.1 Dense layer

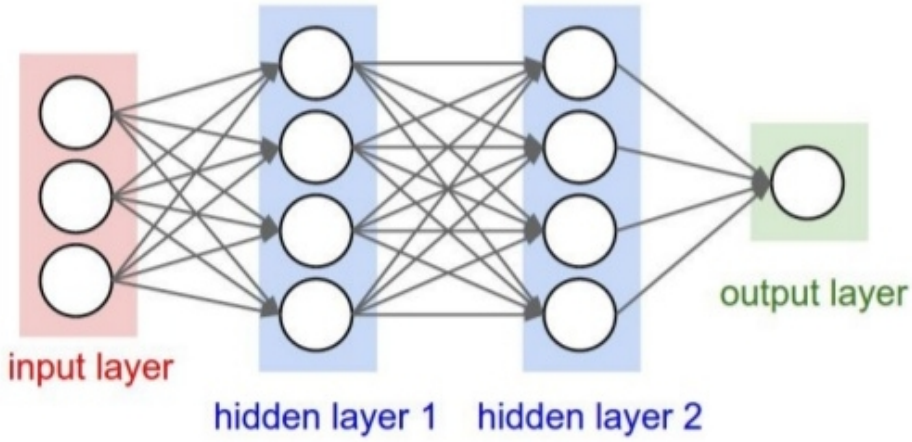


Figure 6: Neural net with dense layers

The most standard layer is the dense layer. In this kind of layers, every neuron is linked to all the neurons of the previous layer. This leads to a very big number of parameters : the number of parameters for this layer is equal to one plus the number of neurons, multiplied by the number of neurons in the previous layer.

3.3.2 Convolutional layer

When applying neural nets to images, the number of model parameters is very high and the computation becomes very complicated and time consuming because of the big number of pixels in an image. The idea to solve this problem is to exploit local connectivity and parameter sharing : every neuron is connected to only a limited number of neurons from the previous layer and the neurons have the same parameters. We call this operation convolution. Using the same small set of parameters for different regions in the images makes the computation faster and allows the model to learn the presence of some structures in the whole image. The learned structures can get more and more complicated as we stack the convolutional layers. The use of convolutional neural nets enhanced the performances of the neural nets, especially when applied to images.

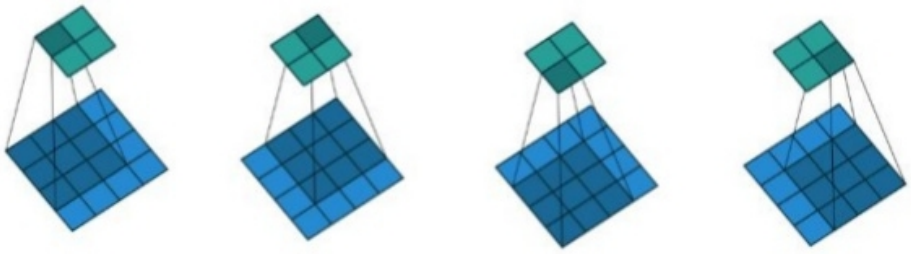


Figure 7: Convolutional layer

Usually, the convolutional layers are followed by a pooling layer : a non linear down-sampling layer, the most common one being the max pooling where we apply the max operator. For example, If we have a matrix of $2n$ by $2n$, we can divide it to an n by n non overlapping squares of size 2x2. The result is an n by n matrix where the values are the result of the max operator applied to the 2x2 squares.

Finally, we define the deconvolution as the opposite of a convolution followed by a pooling layer. A deconvolution is simply a convolution followed by an up-sampling operation which produces a an approximation of what would be the representation of the image in a higher resolution.

3.3.3 Recurrent layer

When dealing with data that is presented in a sequential way, for example images in a video or words in a sentence, recurrent neural networks are a common technique to be used. The idea is that recurrent layer receives the elements of the sequence one by one while keeping an internal state about the past elements. Thus, the prediction is not based only on the last element, but on all the preceding elements and their order. Once more, the number of the parameters is very small comparing to a classic dense layer.

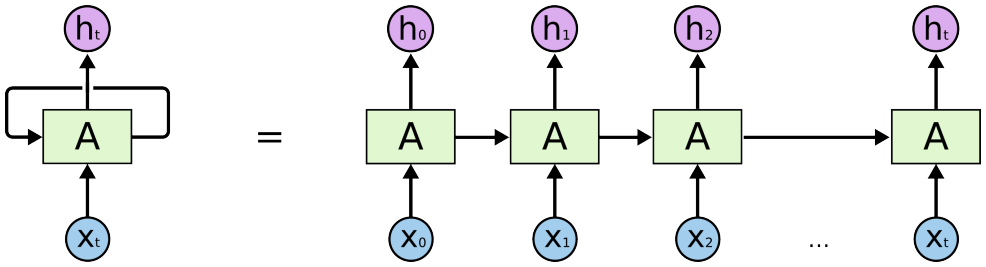


Figure 8: Recurrent layer

A problem that recurrent layers faces is the gradient propagation problem. Computing the gradient through many layers or sequence steps, is done by multiplying the gradient. Multiplying by values superior to 1 or between 1 and 0 multiple times leads to two problems, respectively: the exploding gradient and the vanishing gradient. Those problems prevent the neural net from learning the proper weights. One solution is to use the LSTM (Long Short term memory) layers. They are an improved version of the recurrent layer. They have a better control over the gradient flow using the input gate and the forget gate.

4 Unsupervised representation for the HMI magnetogram data

The charged particles escaping from the Sun don't affect the Earth magnetic field immediately but they have to travel for a period of one to a few days. This causes a time delay between the observed solar images and the space weather. Moreover, as we have seen in the previous section, the solar particles speed is not fixed which causes the time delay to be variable and makes the problem more complicated. Furthermore, due to the rotation of the Sun on itself, the particles ejected from the Sun corona do not travel in a straight lines. Finally, although we have a huge number of solar images, they are not directly labeled or linked to the observed space weather variables.

The idea here is to use the MDI magnetogram images in order to create a compact representation of the solar state. Using this representation, we try to classify the solar states depending on the events occurring in the Sun. Once we have the new representation, we tackle the time-lag problem and try to predict others indicators measured at L1 like the solar wind speed and the Magnetic field z component which are indicators for the geomagnetic storms. The need for the compact representation comes from the fact that, when we have an unknown time lag , it's easier to correlate few representative features with the result than finding the correlation using the whole images. In order to solve this unsupervised learning task, we use autoencoders and we try different architectures.

4.1 Image preprocessing

The magnetogram images are mostly composed of grey colour, with some white and black spots. This is clearly visible in the colours histogram **Fig 9** where 0 corresponds to the black colour and 255 to the white colour. We can identify a large peak in the grey colour and two small peaks in the white and black colour. We would like this distribution to be preserved later by the autoencoder. Especially the black and white pixels as they are the pixels that contains the information about the magnetic field intensity and polarity.

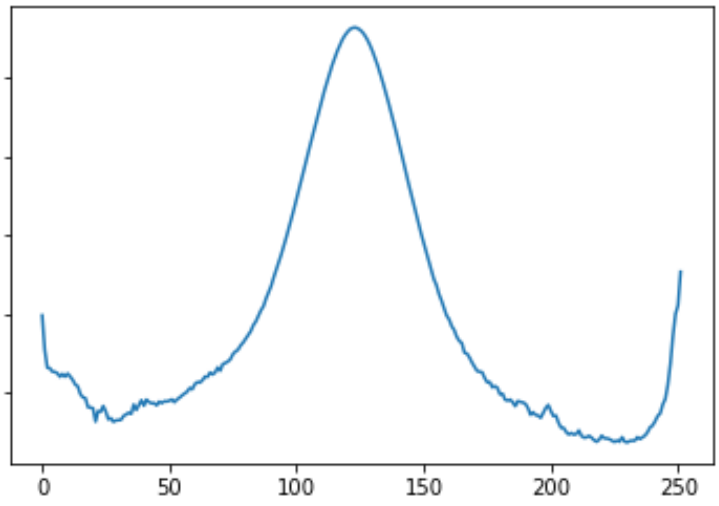


Figure 9: Magnetograms images colour distribution

One of the problems that we have faced is that some of the magnetogram images taken by SOHO are corrupted. This is caused by some material problems and also by a problem in 1998 that caused a near loss of the spacecraft. Most of the corrupted images have horizontal black bands. Those bands have a visible

mark on the 2D Fourier transform : they appear as a vertical centred line. In **Fig 10,11,12 and 13** we can see the original images and their Fourier transform. We have made an automatic classifier based on the Fourier transform to classify the images into good images and corrupt ones. Evaluating manually the performance of this classifier on a subset of images gives a good result. Almost all the images classified as corrupt are corrupted, and we don't see a lot of corrupted images classified as good ones. Classifying all the data gives 48 797 good images and 13 001 corrupted ones. We evaluate the result using the first 1000 images of the solar cycle : we find 12 corrupted images in the 422 images that were classified as good ones and 2 good images in the 578 images that were classified as corrupted.

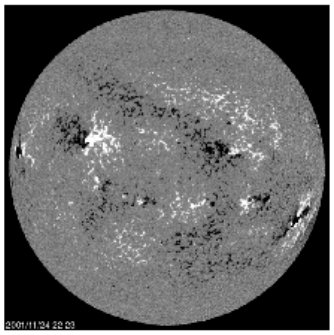


Figure 10: Normal magnetogram

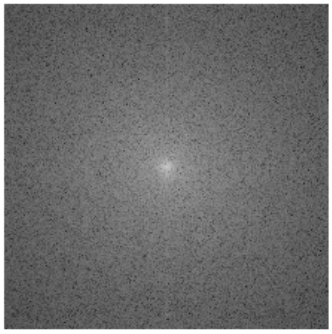


Figure 11: Fourier transform

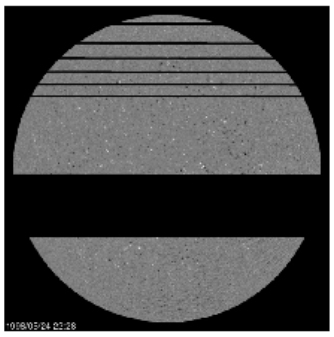


Figure 12: Skewed magnetogram

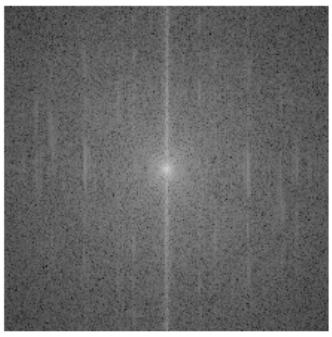


Figure 13: Fourier transform

4.2 Autoencoders for dimensionality reduction

We can make a first attempt to reduce the dimensionality of the images by using a low pass filter or the wavelet decomposition. As we can see, the results are not satisfying when we set the dimensionality reduction to the targeted value : around 500 variable to describe one magnetogram image. The result are shown in **Fig 14**

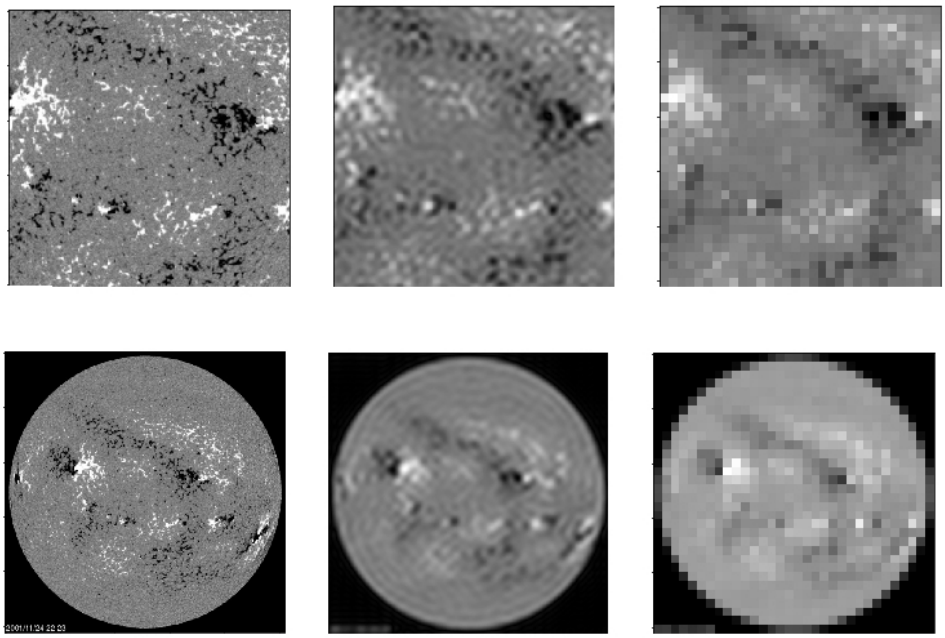


Figure 14: The first row is a crop of 256x256 of the image.The second row is the full 512x512 image. From left to right, we have the original image, a low pass filter and a wavelet filter

4.2.1 Autoencoders

Autoencoders are neural networks that are usually used for unsupervised learning. They can be used for denoising, as generative models or like in our case for dimensionality reduction. In the case of the autoencoder, the first layer and the last layer of the neural net should have the same shape and the autoencoder tries to optimise the reconstruction of the input : minimize an error that evaluates the difference between the input and the output.

We force the autoencoder to pass by a bottleneck : a layer that is smaller than the inputs layer. If the autoencoder can reconstruct its input passing by that layer, we can use the results of that layer as a compact representation of the inputs as it's sufficient to reconstruct the output which is very similar to the input. We control the dimensionality reduction rate by controlling the size of the smallest layer.

We have tried different types of autoencoders :
a) A pure convolutional/deconvolutional architecture

b) A convolutional/deconvolutional net with dense layers in the middle

c) A convolutional/deconvolutional net with PCA applied on the middle layer for more dimensionality reduction

d) A convolutional variational autoencoder

4.2.2 Loss function

The autoencoder tries to reconstruct the input image. In order to do that, we need a metric measuring the similarity between two images. The magnetogram images are greyscale and standard metrics are

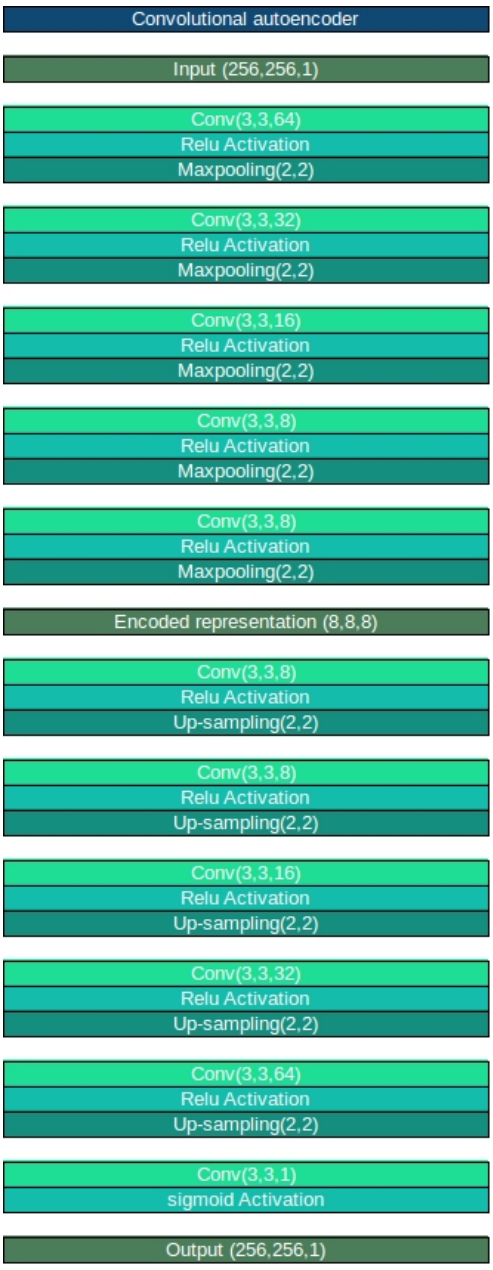


Figure 15: a) Convolutional autoencoder

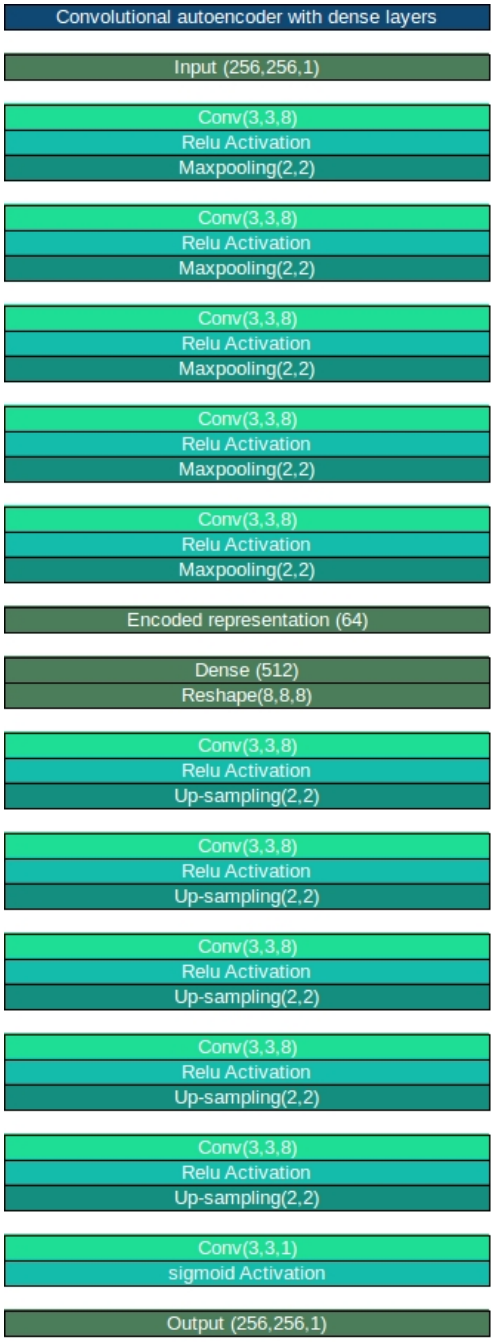


Figure 16: b) Convolutional autoencoder with dense layers

the mean squared error and the binary crossentropy : when normalizing the grey intensity to the range [0,1], the grey intensity can be interpreted as the probability of the pixel to be white. As the images are mainly composed of a middle grey colour, a lot of autoencoders architectures get stuck in a local minimum

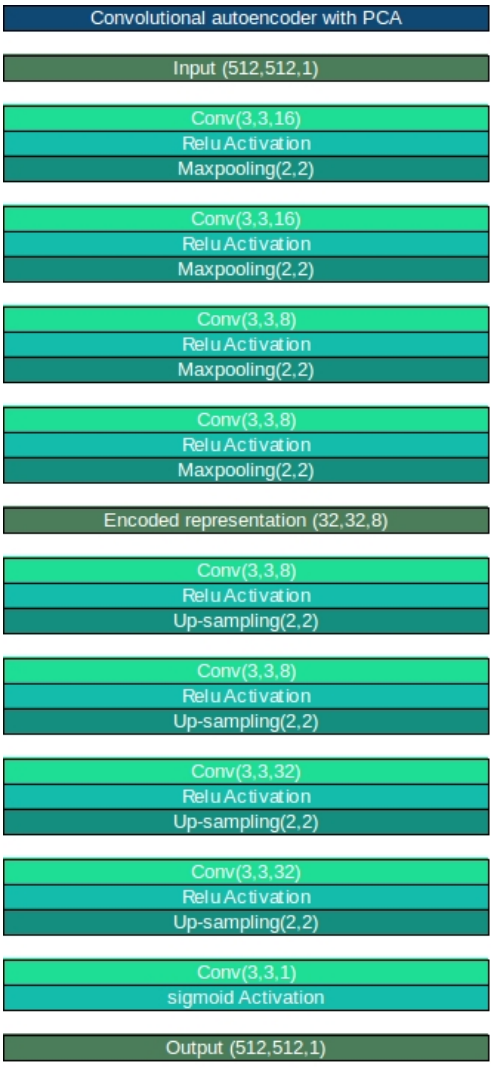


Figure 17: c) Convolutional autoencoder with PCA

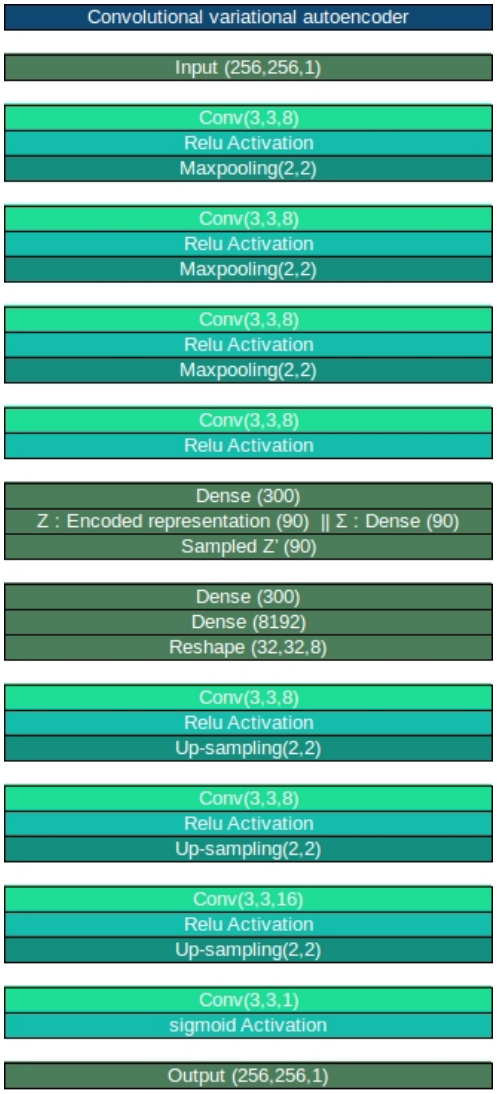


Figure 18: d) Convolutional variational auto-encoder

corresponding to the mean image where all the sun is grey. This problem was more recurrent when using the mean squared error than when using the binary crossentropy. We have decided to use a custom loss that avoid this problem by focusing on the most important parts of the images which are the black and the white spots on the Sun.

$$loss = \frac{(y_{true} - y_{pred})^2}{(y_{true} + \epsilon)^\alpha} + \frac{(y_{true} - y_{pred})^2}{(1 - y_{true} + \epsilon)^\alpha} \tag{1}$$

This loss is convex because only y_{pred} depends on the model parameters : This is important as we are using gradient descent algorithms to minimize the loss function. With this loss function, we give more importance to the pixels that are white or black in the original image, corresponding to $y_{true} \simeq 1$ or

$y_{true} \simeq 0$. α is a parameter, the higher it's the more the autoencoder will focus on the black and white spots and ignore grey spots. Typical values range between 0.25 and 0.4 . Increasing alpha too much will result in a bad reconstructed image replacing all the grey space by black or white spots as we can see in **Fig 19,20,21 and 22**. We add a small value ϵ in order to avoid division by zero.

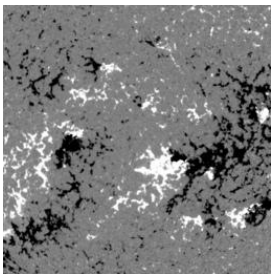


Figure 19: Original image

Figure 20: Autoencoder output with $\alpha = 0.2$

Figure 21: Autoencoder output with $\alpha = 0.25$

Figure 22: Autoencoder output with $\alpha = 0.45$

Using this loss function, the autoencoders were able to avoid the grey-image local minimum most of the time, but tended to give images where we overestimate the black and white spots. Every autoencoder architecture was tried with the different losses.

Finally, we always have to look at the reconstructed images and to manually compare them to the original ones in order to determine whether the results of the autoencoder are satisfying or not. We need the autoencoder to keep the large sunspots and the big clusters of small sunspots. The reconstructed images were presented to the domain expert at CWI team to confirm the results.

4.2.3 Image cropping

The original images resolution is 512x512, but it's easier for the autoencoder to learn the representation of smaller images. That's why we have tried to train the autoencoder on image patches of different sizes. For every patch size, we searched for the smallest possible size of the encoded layer as we can see in **Fig 23**. We observe that the number of features per pixel needed to represent the image decreases when we increase the patch size, but at the cost of a higher training time because we have to increase the complexity of the autoencoder. When we approach patches that included all the image, the number of features per pixel increases again. The problem seems to be caused by presence of the Sun's borders. Thus, we have three options for the patch size. A patch of 256x256 taken at the center of the image. A patch of around 300x400 taken at the center of the image. The reason is that according to domain experts, most of the important phenomena happens near the Sun equator and far from the poles. For practical reasons, we work with a size of 288x416 instead of 300x400 : When passing through max pooling and through deconvolution, the size is devided by 2 at every step and we need a size that we can divide by 2 multiple times. Finally, we can consider the whole image in order to keep all the magnetogram information.

4.3 Autoencoder architectures

4.3.1 Pure convolutional/deconvolutional autoencoder

Our autoencoder network is composed by 5 pairs of convolution and maxpooling, followed by 5 pairs of deconvolutional layers. We use the rectified linear unit **ReLU** as an activation function for all the layers except for the last one where we use the sigmoid. Applying this architecture to the 256x256 patches allows us to compress it to 512 features.

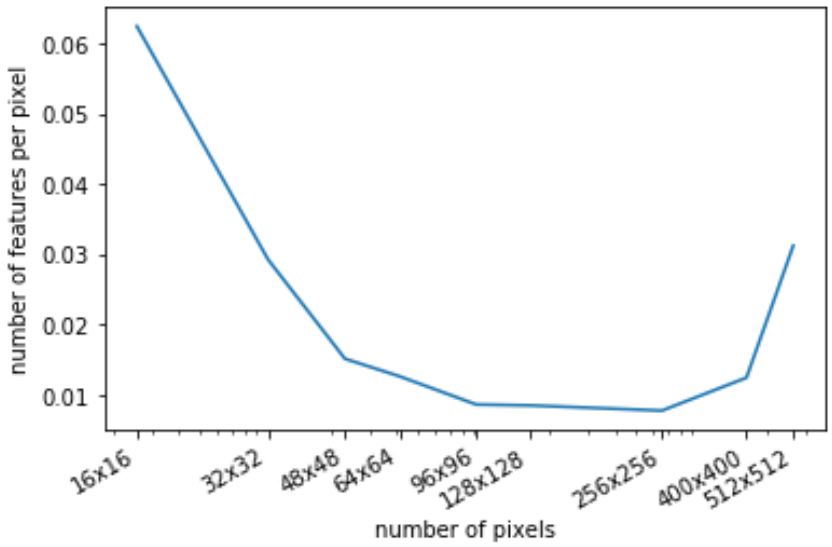


Figure 23: Number of feature per pixel in the decoded layer as a function of the size of the image patch

4.3.2 convolutional autoencoder with PCA

This is basically the same autoencoder but with a PCA applied to the encoded representation after training the autoencoder. The principal component analysis **PCA** is a procedure based on linear algebra and orthogonal transformations that project the data into new axes. Every axis is chosen by maximizing its variance while being orthogonal to the previous axes.

As we can see in **Fig 24** , Most of the variance in the data can be explained by the first few features and we can proceed to another dimensionality reduction. We apply this method to the full image : we start by creating a compact representation of 8192 features using the convolutional autoencoder. Then, we use the PCA to downsize the representation from 8192 features to a smaller dimension of 500.

4.3.3 convolutional autoencoder with dense layers

We keep the same architecture of the convolutional autoencoder and we add 3 dense layers between the convolution phase and the deconvolution phase. Using this technique, we can reduce the dimensionality even more and represents the 256x256 patches with only 64 features instead of 512

4.3.4 convolutional Variational autoencoder

Variational autoencoders[Doe16][KW13] are autoencoders that makes an assumption on the latent variables distribution, usually taken as a Gaussian distribution. In this case, we add a new term to the loss that enforces the latent distribution to be as close as possible to the wanted distribution : a spherical multivariate Gaussian distribution in our case.

More precisely, let us denote the autoencoder inputs by x, the latent representation by z and the output by $\widetilde{x}$. ϕ and θ are the weights and bias parameters of the encoder and the decoder phase of the autoencoder. The encoder part is $q_\phi(z|x)$, it learns to approximate the true posterior distribution $p(z|x)$. The decoder part is $p_\theta(z|x)$.

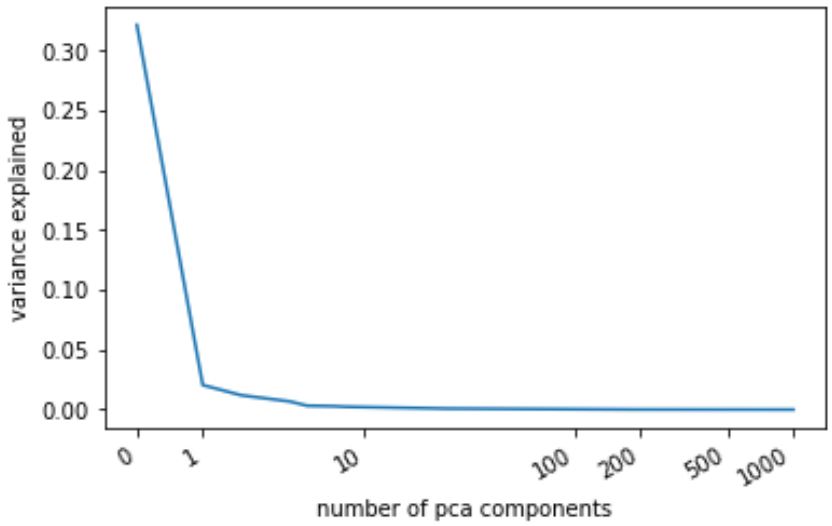


Figure 24: Variance explained by PCA components

First, the encoder transform the inputs x to the latent representation z and yields two parameters : the mean $\mu(x)$ and the variance $\Sigma(x)$. Then, we randomly sample similar points z' using the normal distribution $\mathcal{N}(\mu(x), \Sigma(x))$. Finally, the decoder maps z' to the original space resulting in $\widetilde{x}$.
The parameters ϕ and θ are learned while optimizing the loss function :

$$L(\phi,\theta,\text{x}) = \text{-E}_{q_\phi(z|x)}[log p_\theta(z|x)] + KL(q_\phi(z|x)||p_\theta(z))$$

Where KL stands for Kullback–Leibler divergence. The first term of the loss is the reconstruction loss. It enforces the autoencoder to reconstruct the input. The more the input and the output are different, the higher is the value of this loss. The second term measures how close is the distribution of $q_\phi(z|x)$ to $p_\theta(z)$ that we assume to be a normal distribution in our model.
With this structure, we can reduce the 256x256 patch to a latent dimension of 90.

One advantage is that the variational autoencoder is a generative model. We can easily generate new images by taking random samples from the latent representation. This can be helpful in order to balance classes when tackling the prediction problem. We can see some new generated data in **Fig 25**.

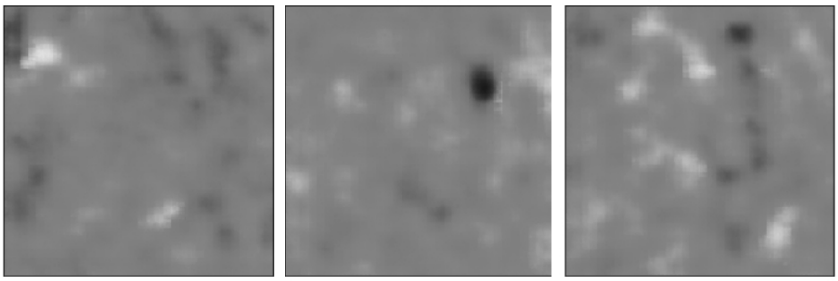


Figure 25: New generated images using the variational autoencoder

4.3.5 Results

As the solar state evolution is slow over time, the evolution of the compact representation features should be smooth over time. We can verify this smoothness in **Fig 26,27,28 and 29**. We also verify the stability of these representations, meaning that similar images should have similar representations. We find some unusual values of the features for some images. We verify that they correspond to corrupted images that were not filtered by the Fourier filter applied to the magnetograms during the preprocessing step.

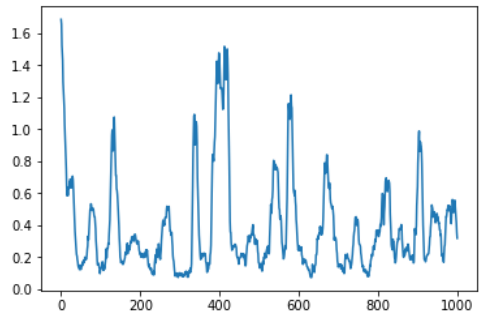


Figure 26: Evolution of features of the convolutional autoencoder over time

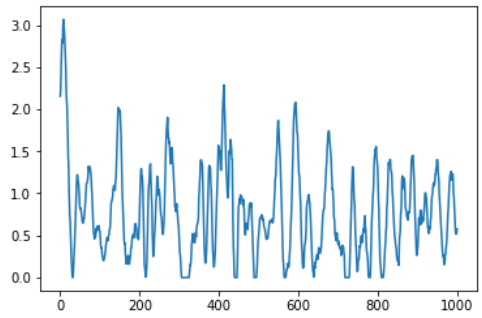


Figure 27: Evolution of features of the convolutional autoencoder with dense layers over time

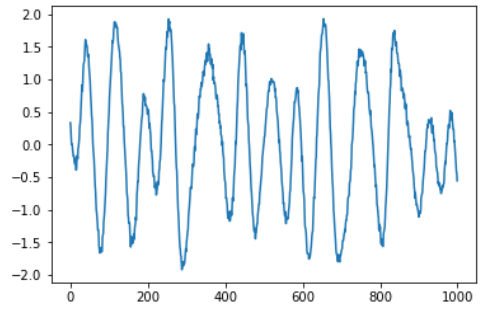


Figure 28: Evolution of features of the convolutional autoencoder with PCA over time

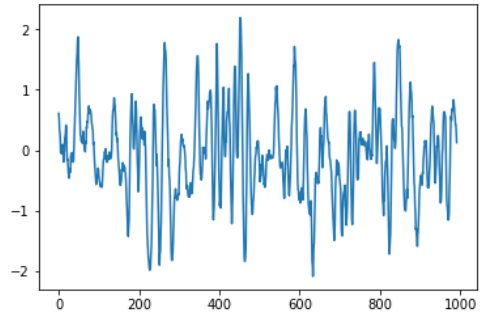


Figure 29: Evolution of features of the convolutional variational autencoder over time

The features learned by the autoencoders differ from one to another. The features obtained by the pure convolutional architecture represents local area of the image as a consequence of the network architecture. The other autoencoders give global features but different ones. We also observe that the PCA features representations are more and more noisy as the index of the corresponding principal component increases. We can observe some of these features in **Fig 30,31,32 and 33**

We have created compact representations using the four techniques. We can visually compare the results of the reconstructed output in **Fig 34**. Later, we will try the different representation in the prediction problems.

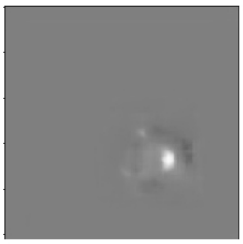


Figure 30: Example of features learned by the convolutional autoencoder

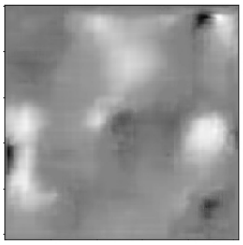


Figure 31: Example of features learned by the convolutional autoencoder with dense layers

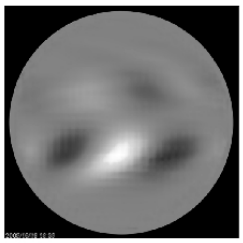


Figure 32: Example of features learned by the convolutional autoencoder with PCA

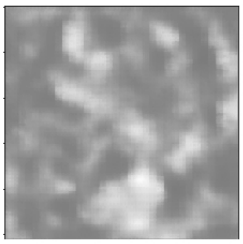


Figure 33: Example of features learned by the convolutional variational autencoder

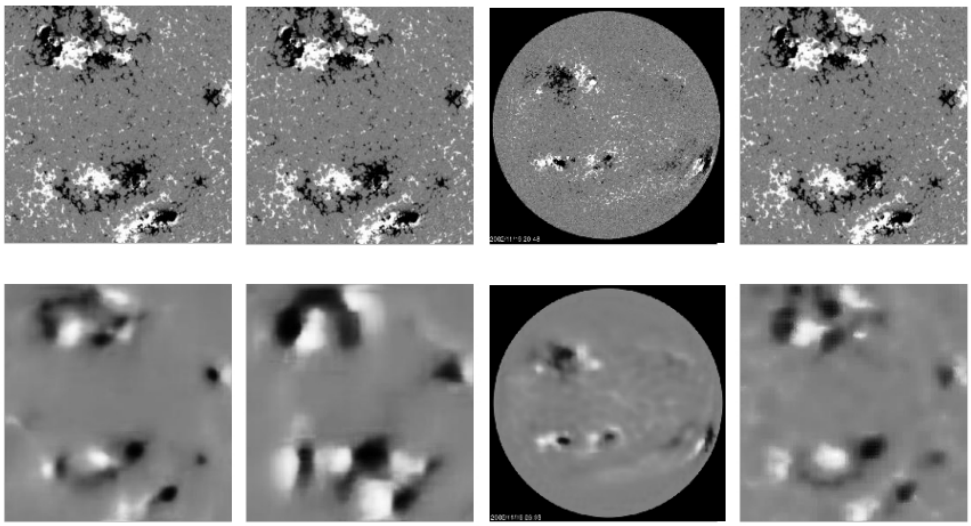


Figure 34: Examples of autoencoders outputs and inputs. The top row are the original inputs, the bottom line are the reconstructed images. From left to right : Convolutional autoencoder, Convolutional autoencoder with dense layers, convolutional autoencoder with PCA, convolutional variational autoencoder

5 Space weather prediction

5.1 The dst index

The first thing that we have tried to do was to find a correlation between the extracted features and the dst index. The disturbance storm time index(**dst index**) measured in μT on the earth surface measures the strength of the ring current created by particles coming from the Sun around the Earth. This current creates a magnetic fields that is opposite to the Earth's magnetic field making it weaker. A negative dst value means that the Earth's magnetic field is being weakened. This happens during solar storms. We have tried plotting the time series of the dst index and the images features and testing the Pearson correlation coefficient assuming different time-lags. However, we couldn't observe any significant correlation.

The prediction of the dst-index directly seems to be a hard task, that's why we decided to first use the created image representation to classify events occurring directly at the Sun : in this problem, we don't have to handle the time-lag which is an extra difficulty, and we can test the efficiency of the representations that we have obtained. Afterwards, we will try to predict other variable that are affected by the Sun state and that affects the Earth's magnetic fields like the solar wind speed and the interplanetary magnetic field Z components.

5.2 Prediction of solar events

We use a dataset indicating some solar events like: Coronal holes, lepping, pseudo streamer and strahl. These are events that happens in the Sun and they can affect the Earth's magnetic field. This dataset was manually compiled by domain expert. As we can see on the MDI magnetogram images **Fig 35,36,37 and 38**, it is not easy to make the distinction between the four categories. Of the 48000 MDI images, we have around 8000 images that were labeled with at least one of the four categories. For every category we have around 2000 images. Finally, an image may have multiple labels. For the remaining images, not having a labels doesn't mean that there is no event happening, but most probably that the image was not looked at, thus we can't use them in the model training or evaluation. Finally, splitting the dataset into train and test dataset was a bit tricky. Random splitting into train and test leads to a very accurate classification (99%) but this is a misleading result because non i.i.d samples results in cross-validation problems. In our case, the solar events last in time, meaning that one event may correspond to various successive images that are similar. Random splitting can affect one of those image to train and another one to the test set and the model will be memorizing the already seen images instead of generalizing. The perfect solution for us would be to split the data chronologically and to use the past for training and the future for testing. However, we couldn't do this because of the distribution of the labeled events. As we can see in **Fig 39 and 40**, the events are not evenly distributed over time: For example, the Lepping event is concentrated at the beginning of the considered time period while the Strahl event is present only at the end of the time period.

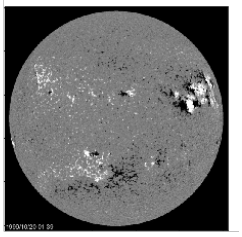


Figure 35: Coronal hole

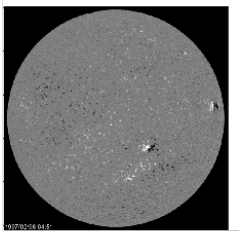


Figure 36: Lepping

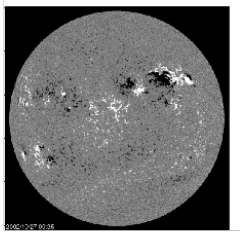


Figure 37: Pseudo streamer

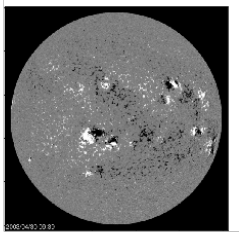


Figure 38: Strahl

We do a random sampling over time slices of data instead of single instances. As every slice will include several events from their start to their end, plus at most two parts of two other events, this will reduce the effect of having images of the same events appearing in the train set and in the test set. The average results are presented in the following table.

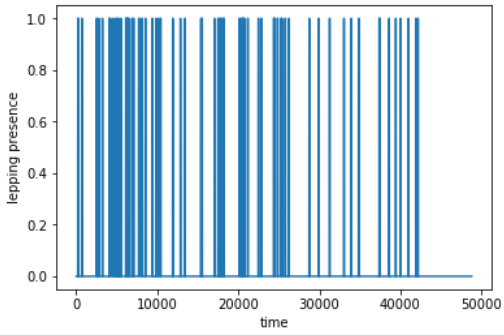


Figure 39: lepping distribution over time

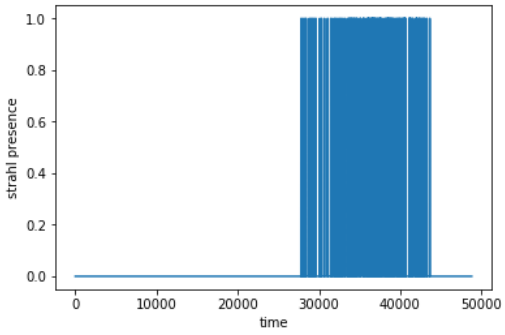


Figure 40: Strahl distribution over time

Event	precision	recall	accuracy	F1-score
Coronal hole	0.74	0.36	0.62	0.48
Lepping	0.90	0.51	0.77	0.65
Pseudo streamer	0.66	0.93	0.78	0.77
Strahl	0.55	0.98	0.73	0.70

Denoting the number of the true positives by TP, the false positives by FP and the false negatives by FN we have : $Recall = \frac{TP}{TP+FN}$ representing the fraction of the relevant events that are successfully retrieved. $Precision = \frac{TP}{TP+FP}$ representing the fraction of retrieved events that are relevant. $F1score = \frac{2}{\frac{1}{recall}+\frac{1}{precision}}$: a weighted measure of recall and precision. The three parameters are in [0,1] and a higher value indicates a better performance. The classification of the four events shows that the difficulty of the classification depends of the event itself. However, as we have a small data set of labeled events :8000 images, 6000 for training and 2000 for testing , we can't say for sure that the result are significant. Moreover, theses results depend on the partition of the data set into a training set and a testing set. The variance of the results is high especially for the coronal hole prediction where we can see the F1 score dropping to 0.21 .

Finally, when comparing the different representation performances, the convolutional autoencoder with PCA applied to the whole image seems to be the one giving the best results. This may mean that the lateral parts of the Sun that are removed when applying the central patches contain some relevant information.

5.3 Prediction of the solar wind and the interplanetary magnetic field

We decided to use the autoencoder features in order to predict the solar wind speed and the interplanetary magnetic field z components B_z. The solar wind speed and B_z are measured by the satellite in the point L1. The particle traveling from the Sun can take from one day to five days to reach L1. Also, particles don't keep a constant speed after being released from the Sun : they are accelerated. We can use a 1D propagation model where the dimension is r: the distance from the center of the Sun to the particle, to model their speed, but we don't obtain an expression of the velocity v_r as a function of r but as a function of v_r and r [KG99] [ZH08].

$$\frac{v_r}{c_s} = \sqrt{4(\frac{r_s}{r}) - 3 + 2ln[(\frac{r}{r_s})^2((\frac{v_r}{c_s})]}$$

With c_s a constant and r_s the critical position depending on the ejection speed. We can solve this

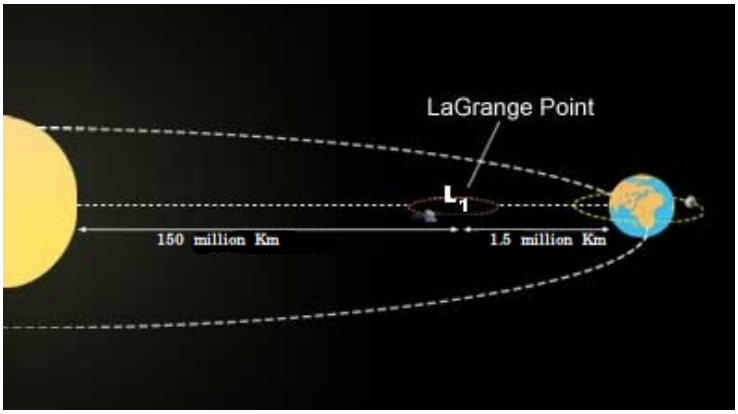


Figure 41: Lagrange point and distance to the Sun and Earth

equation numerically for a given value of r. Finally, there is the fact that the particles are not moving in straight lines but in loops starting and ending at the Sun, those loops can reach even as far as Jupiter.

In order to handle the time lag problem when predicting the solar wind based on the representation obtained from the autoencoder, we have tried several approaches :

- A constant time lag
- Assuming two different time lags : One lag correspond to the fast solar wind, arriving in L1 with a speed superior to 600 Km/s . Assuming an average speed of 750 Km/s, the time lag should be around 55hours. Another lag for the slow solar wind arriving in L1 with a speed inferior to 600 Km/s. Assuming an average speed of 400 Km/s, the time lag should be around 91 hours.
- Finally We have also tried a continuous mapping between the speed and the time Lag. We assume that the speed is constant from the Sun to L1, which is actually wrong, and we compute the time lag based on the Sun-L1 distance.

In the two last approaches, we end up by having intersecting time lines : some events are happening later on the Sun, but detected sooner on L1, which is not very realistic as we will probably have shock phenomenon. We also end up by having different values of solar wind speed in different moments t_i, corresponding to the same image t_0.

In order to enhance our model, we can consider two ideas :

- Use the autoencoder as a time series data and let the model learn the time lag on its own.
- Use the complete 1D propagation model and discretization of space between the Sun and L1 in order to get a more realistic solution to the time lag problem.

5.4 LASCO detector

5.4.1 The detector

Visualizing the LASCO images over some time periods while monitoring the dst index or the solar wind speed shows that there might be a direct relation between the CME observed by LASCO and those variables. In many cases, disturbances in those variables can be related to CME events happening a few days earlier.

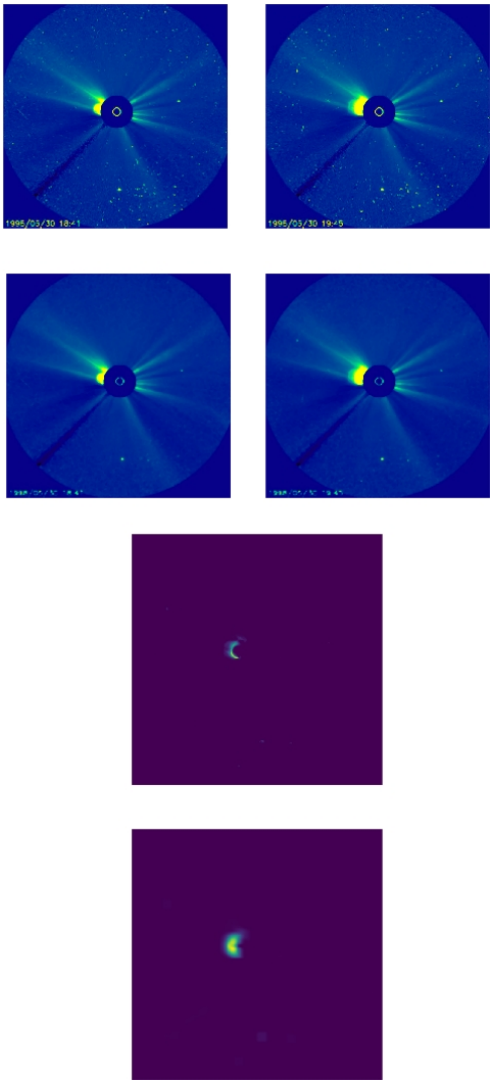


Figure 42: LASCO detector steps first example: From top to bottom we have the two successive original images, the denoised images, the pixel-wise difference and the convolution

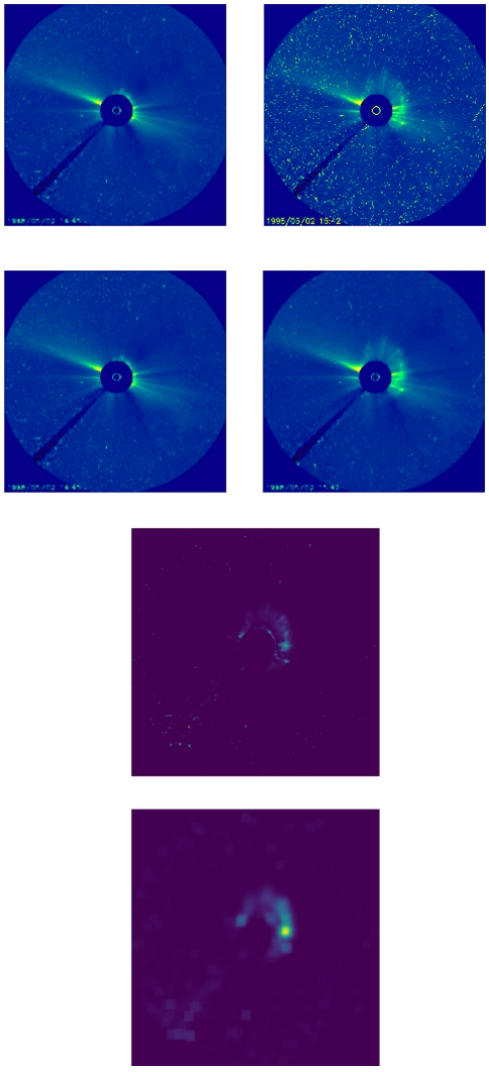


Figure 43: LASCO detector steps second example: From top to bottom we have the two successive original images, the denoised images, the pixel-wise difference and the convolution

We have decided to make an automatic module that uses the LASCO images in order to detect the CME. We start with a simple method : First we start by taking two successive images and by denoising the images using a median filter. We also remove the pixels corresponding to the date at the bottom left of the image. Next, we compute the square of the difference between the corresponding pixels in the two successive images : this step allow us to visualize the movement between the two images. Afterwards we apply a convolution with a mean filter. Finally, we take the max intensity of the resulting image in order to obtain an indicator of the CME activity. This value holds information about the maximum intensity of "movement" between the two images in a local area due to convolution and the max operator. The

convolution step grants also more robustness to noise. In fact, the denoising step don't remove all the noise from the image. However, the remaining noise is usually present as isolated pixels. Applying the local mean filter using convolution will mitigate the effect of the isolated pixel in regards to the upcoming max operation while keeping the effect of the areas where the pixels are grouped. We are interested in those areas as they are the one indicating the CME. Those steps are shown in **Fig 42 and 43**. Finally, we can remark that the LASCO images can be very noisy as we can see in the original images of the second example.

In order to check the output of our detector. We have made a manual labeling by looking at the LASCO images of one month : Mai 1998. We gave a value between 0 and 1 corresponding to the CME activity observed for every date. Even thought this is not a very accurate measure, but a subjective visual one, it should be enough to verify the performances of our detector. We apply the automatic detector to the same images and compare the detector output with the manual labelling in **Fig 44**.

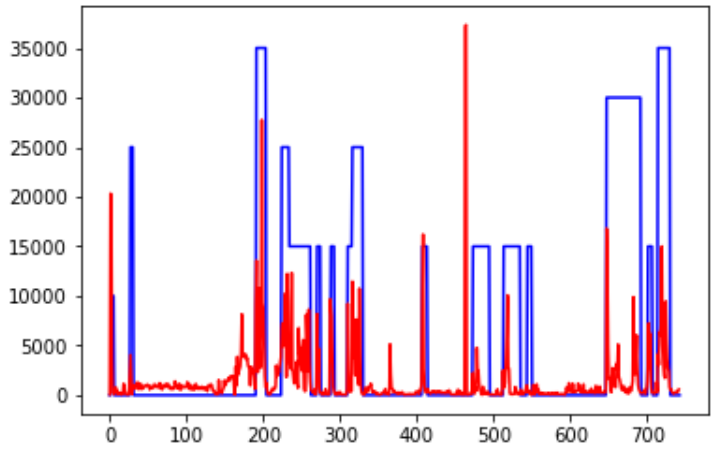


Figure 44: Comparing the automatic LASCO detector to manual labelling : the two curves show the evolution of the CME activity as a function of the time during the month of Mai 1998. The blue curve corresponds to the manual labelling while the red one corresponds to the detector

When we compare the two curves we notice a first difference for the peak Superior to 35000 on the automatic detector. We notice that this peak corresponds to the presence of a skewed image that our detector can't handle. For the time period where x is between 170 and 190, we have another event detected by the automatic detector but that was not manually labeled. A more carefull look to the images allows us to see a small CME that we didn't consider during the manual labeling. For the images where x is between 640 and 700, we seem to overestimate the duration of the CME during the manual labelling. For the rest of the month, the automatic detection is close to the manual labelling. Considering these results, our detector seems to be working, but is very sensible to skewed images.

Finally, we still can enhance the model in the future if needed. We can work on :

- Making an automatic detection to remove the bad images like we did for the magnetogram images.
- Working more on removing noise, especially that some images are very noisy. Also, sometimes celestial objects like stars, comets or Venus are visible in the image and are not completely removed by our denoising method.

- Finally, we can enhance the detector output by adding the location, the size and the speed of the CME for example.

5.4.2 Correlation between the LASCO detector and the solar wind

First, we can visualize the distribution of the solar wind in **Fig 45**. We can see a clear peak at 400 Km/s corresponding to the slow solar wind, and another smaller one around 600 Km/s for fast wind. This distribution can be easily approximated using a Gaussian mixture model with two components as we can see in **Fig 46**.

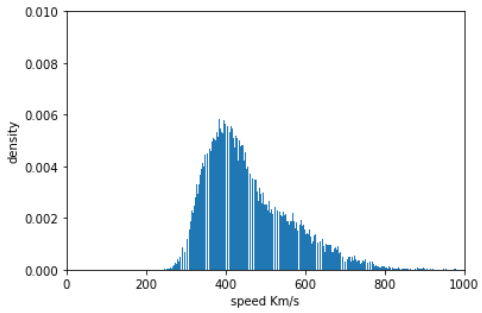


Figure 45: The solar wind distribution during the years 1995-2007

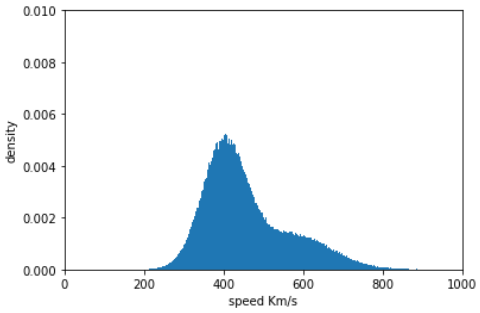


Figure 46: The generated wind distributions using a two components Gaussian mixture model

We set a threshold for the values detected by the LASCO detector and we consider that we have detected a CME if we are above that threshold. We assume different time lags and we investigate the solar wind distribution conditionally to the fact that we have detected a CME or not. Let us denote by Y the solar wind speed, by X the LASCO detector value and by Z the binary variable indicating the presence of a CME activity. This means that Z=1 if and only if X $>$ threshold. We compare the distributions of $p = Y|Z$ and $q = Y|\bar{Z}$. We don't obtain a perfect separation between the two categories of wind speed but we can observe the difference between the two distribution as a function of the chosen time lag. We use two different ways to evaluate the difference between the two distribution : First, we measure the Kullback–Leibler divergence $D_{KL}(p||q) = \sum_y p(y)log(\frac{p(y)}{q(y)})$ which measures the similarity of two probability distributions. For the second measure, we estimate the mean of every conditional distribution as well as its confidence interval using the central limit theorem. As q is supposed to represent the slow solar wind while p the fast solar wind, the two confidence intervals should be separated. Thus, we measure the difference between the two confidence interval extremities. We observe the evolution of those measures as a function of the time lag in **Fig 47 and 48**. We observe That we have a peak around 55 hours. As expected, the time lag is close to the time spent by the fast solar wind particles to arrive from the Sun. However, this separation is not good enough. For instance, the separation between the two means is around 20 while it's optimally around 200, and the KLD values prove that the two distributions are quite similar.

We have tested the Pearson correlation coefficient between the solar wind speed and the values given by the LASCO detector for various time lags. Once again, we observe a peak in the Pearson correlation coefficient for a time lag close to 55 hours. However, the values of the correlations coefficients are low and don't exceed 0.09 for the whole period of the Solar cycle. Further investigations of this coefficient

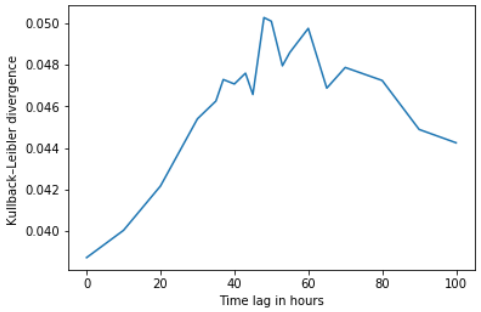


Figure 47: The Kullback–Leibler divergence between $p = Y|Z$ and $q = Y|\bar{Z}$ as a function of the time lag

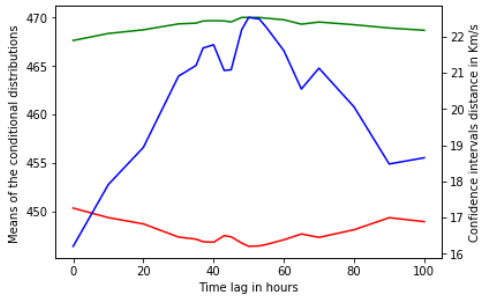


Figure 48: The green and the red curve represents respectively the means of the distribution of p and q as a function of the time lag. The blue curve is the difference between the confidence interval of those means as a function of the time lag

proves that the correlation can be very high for some months, but that for other time periods we have no correlation. For example, using a time-lag of 55 hours, we reach a correlation of 0.35 for the month of 11-2000 while the correlation coefficient value is only 0.08 for the month of 02-2006. The pearson coefficient have even small negative values for some months. The curves for the two specified months are shown in **Fig 49 and 50**

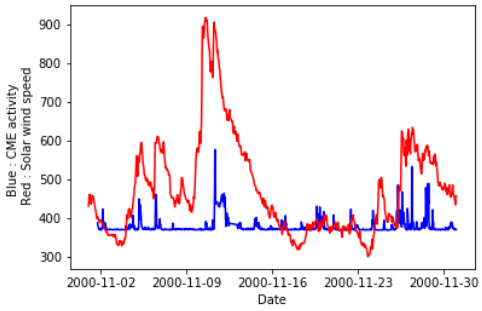


Figure 49: The solar wind speed in red and the LASCO detector CME activity prediction in blue for the month of 11-2000

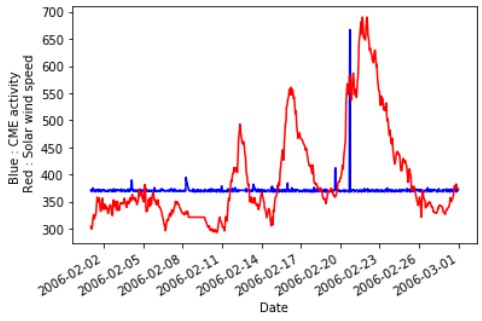


Figure 50: The solar wind speed in red and the LASCO detector CME activity prediction in blue for the month of 02-2006

We take the ensemble of the CME detected and we consider the values of the solar wind measured at L1 in the following few hours : in the period of time during which we expect the particles released during the CME to reach L1. We can observe that most of the observation of fast solar wind at L1 are preceded by a CME activity. However, only around one out of ten CME is followed by a fast solar wind. The reason for this is that the detected CME are not necessarily directed towards the Earth. In that case, they won't affect the Earth's space weather. The next step is to find a way to determine if the CME is directed towards the Earth.

5.4.3 The Computer Aided CME Tracking

The Computer Aided CME Tracking or **CACTUS** [RB04][RBV09] is an already existing CME tracker that has made its data available to public. It's estimated to being able to retrieve about 75% of the humanly cataloged CME as well as other true CME events that were small and that were not cataloged. The CACTUS detector uses the same LASCO images as our detector but is more sophisticated. It includes a polar transformation of the Sun image and a time - distance representation where we can see the CME as lines in the obtained images. Those lines can be easily recognized using the HOUGH transform. In addition to the dates of the CME, the module provides other information about the CME speed, angle and number of ejections. Those informations will be useful to determine if the CME is headed towards the Earth or not.

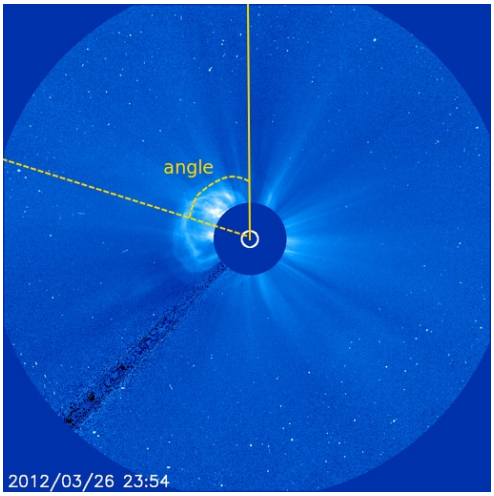


Figure 51: The CME angle

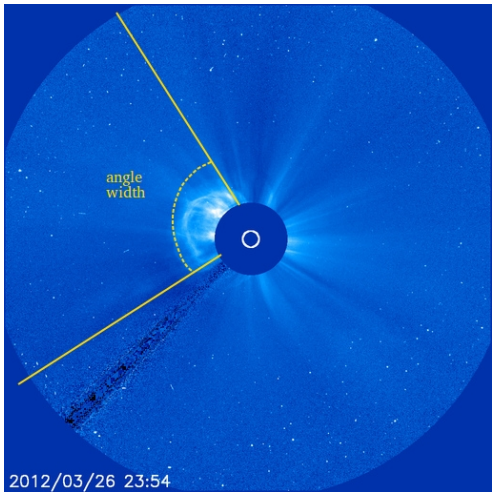


Figure 52: The CME angle width

We start by comparing the events detected by our detector and the events detected by the CACTUS tracker. If we consider the events detected by cactus the ground truth, and the ones detected by our detector as the predictions, we obtain a recall of 0.36, a precision of 0.48 and a similarity in the number of the detected events. If we allow a time tolerance of 10 hours between the two detectors, we obtain a recall of 0.87 and a precision of 0.83 which prove a similarity in the events' detection between our LASCO detector and the CACTUS tracker.

After verifying the similarity of the two detectors, we study the relation between the properties of the CME given by the CACTUS tracker like its speed or angle and between the effect of the CME on the space weather, used as a proxy to determine if the CME is headed towards the Earth or not. The method that we have used is the following : We consider every CME detected by cactus. We estimate its arrival time to the point L1 and its duration. We measure the maximum values of the solar wind speed, the proton density and the Kp index as well as their variances during the expected time period. Finally, we separate the CME into two categories depending on some of their properties and we compare the maximum values and the variances between the two categories. A higher maximum of solar wind speed, Kp index or proton density, as well as a high variance of those values is an indicator of solar activity. We find out that the most important property of the CME is its angle width. In fact, the events with a high angle width are called HALO CME. The reason is that the corresponding LASCO images shows the ejection of the particles in almost all the directions forming a HALO around the Sun.

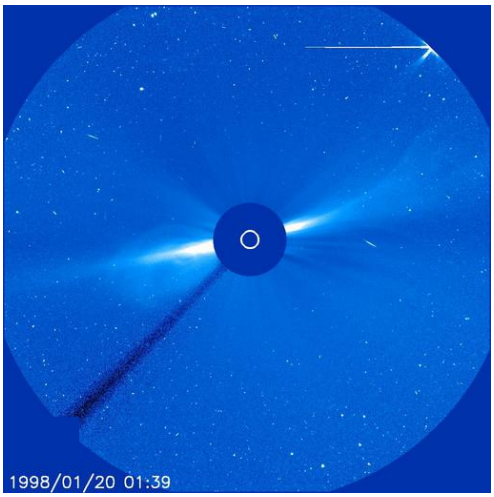


Figure 53: Small angle width CME

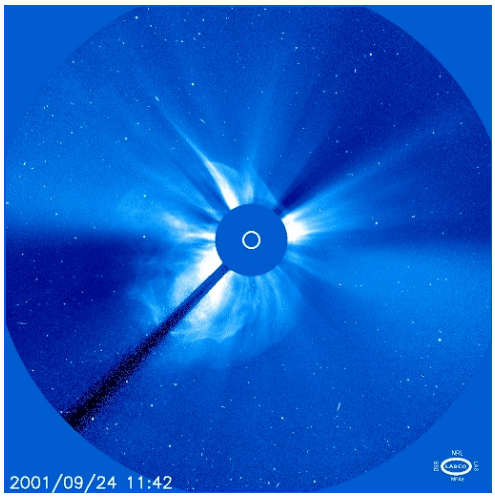


Figure 54: HALO CME

After filtering using CME width, around 2/3 of the selected CME are followed by fast solar wind in the expected period. Another way to see the influence of the HALO CME is to compare the distributions of the K_p index during the period following the HALO CME, the period following the normal CME and during the whole solar cycle. The K_p index quantifies the disturbances in the horizontal components of the Earth's magnetic field. Values equal to 5 or higher in a scale of 0 to 9 and equivalently values equal to 50 or higher in a scale of 0 to 90 indicates the presence of the geomagnetic storms. We can see these distributions in **Fig 55** and we observe that the HALO CME is related to the higher values of the K_p index. In fact, 45% of the CME with an angle width superior to 150 are followed by a K_p index superior to 50. Those events cover 48% of the cases where the K_p index rises above 50. We can use the detection of the HALO CME and their estimated effect period to predict the solar wind speed. However, it's difficult to predict the exact period and the duration of the time period during which the HALO CME will cause a fast solar wind. We decided to use the Dynamic Time Warping technique to measure the similarity between the estimated periods and the real time periods where we measure a fast solar wind. The dynamic time warping is an algorithm used to find a non linear mapping between two time series and to measure their similarity . It finds the optimal mapping between the two series and it's able to adapt to the deformations of time, speed and duration. Applying this algorithm, with a 40 hours time window, to the prediction made by the HALO CME gives a better similarity measure comparing to the hole set of CME. This result looks promising and it's going to be studied more deeply during the remaining period of the internship.

As the next step of the CME prediction, we can adapt our current LASCO detector to provide more information about the CME. We can also consider making a new LASCO detector, similar to the CACTUS tracker, that can provide us with CME proprieties. Finally, we can consider using autoencoders to find a compact representation of the LASCO images that contains the needed proprieties of the CME.

5.5 Time sequences representations for solar wind prediction

In addition to detecting whether we have a fast solar wind in the following hours, we attempt to predict the solar wind speed at every hour. We have decided to use the features that we have extracted with the

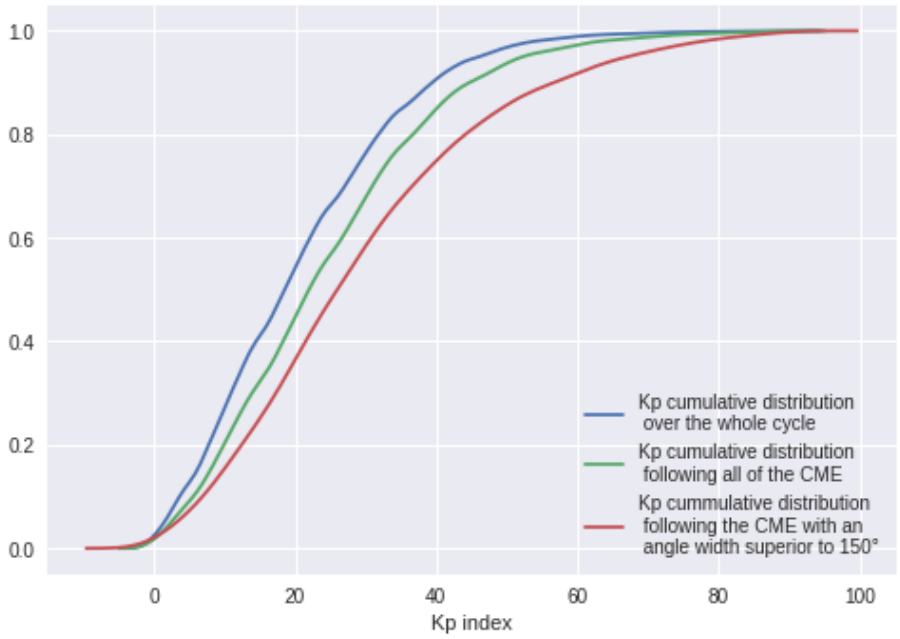


Figure 55: Kp index cumulative distributions

autoencoder and the data of the LASCO detector from the period [t0 -5days ; t0- 1day] in order to predict the speed value at t0. The prediction is done using the data from a time period of four days and we leave the time lag problem to be taken care of by the learned model. The different images are not taken at a regular time interval. For example, some instances may have 20 magnetogram images available during the considered four days period while other have more than 60 images. We have tried three options in order to solve this problem :

- For every feature, we consider its evolution on time during the 4 days and we represent the feature by the parameters of a curve that we fit to this evolution. For example we can use a linear, polynomial or spline interpolation. For every feature, we can also use the mean squared error between the real curve and the fitted curve as an input to the prediction.
- We divide the 4 days into time steps : 20 for the autoencoder features and 200 for LASCO. For every time step, if we have available images at that date, we use one of the images features: the latest one chronologically for example. We fill empty time steps with a special value that is going to be ignored when applying the Recurrent neural net.
- We do the same thing as the previous method, but we fill in the empty times steps using linear interpolation instead of tagging them with a special value.

Finally, for the three options, we set a minimum number of images that needs to be available during the 4 days windows in order to use it in the prediction. Otherwise, we consider that we don't have enough data to represent the instance.

The prediction task is not an easy one. The data set is unbalanced : more than 90% of the instances have a slow wind speed. We also have to split the training set and the testing set chronologically, which

produces a difference in the solar wind speed distribution between the training and the testing sets. We try to solve this problem by balancing the weights of the different instances depending on their solar wind speed : The higher is the solar wind speed of a training instance, the higher is the attributed weight. We perform the prediction task using neural nets. We compare our results to the mean-prediction and to a random predictor based on the solar wind distribution in the training set. We summarize the results and compare the different data representations in the following table. The MSE is measured for the solar wind speed ranging from 0 to 1 while the RMSE is rescaled to the original values of the solar wind in Km/s. We observe the increase in the Pearson correlation when we extend the representation of the solar state. However, the improvement of the RMSE is not substantial comparing to the mean predictor.

Data	Pearson correlation	MSE	Rescaled RMSE
Random predictor	0	0.039	165
Mean predictor	-	0.028	139
Magnetogram compact representation	0.05	0.0541	193
HALO CME	0.14	-	-
Sequential	0.236	0.0211	121
Sequential + HALO CME	**0.335**	**0.0208**	**120**

We also plot a comparison between the evolution in time of our predictions, and the true values. The predicted values tend to be close to the mean value and we don't have a strong predictor, but we are able to explain a part of the variance in the data. We compare the predicted solar wind speed to its true values in **Fig 56**. We can identify three different behaviours. First, around the hour 600, the prediction is very close to the true value. The second behaviour is when we can predict the variations of the solar wind, but we don't have the right amplitude. This behaviour can be seen around the hour 800. Finally, for some time periods, we can't detect the variations at all. this is the behaviour that we see around the hours 200 - 400.

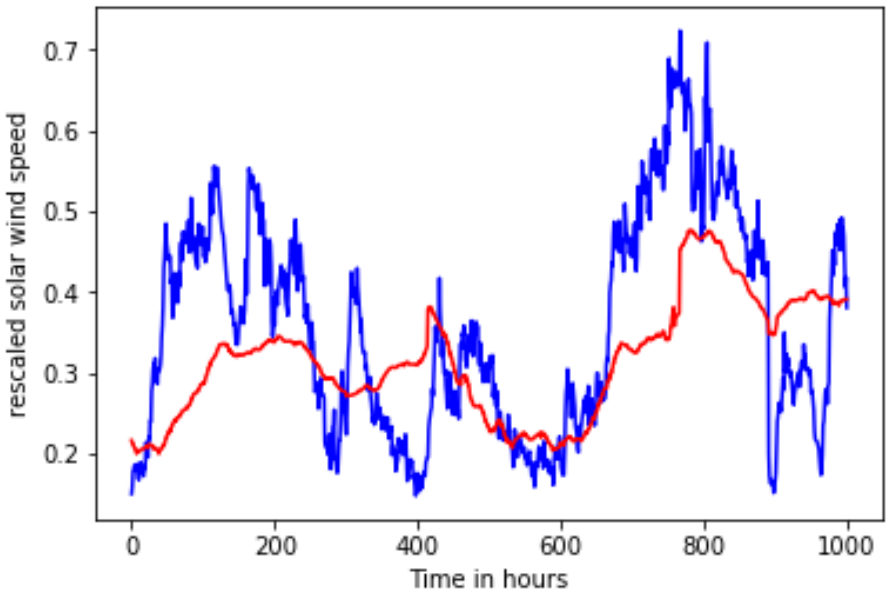


Figure 56: Prediction of the solar wind speed in red compared to the true value in blue

6 Conclusion and future work

In this report, we described the geomagnetic storms' prediction problem and the main techniques that we have developed to tackle the problem. We first presented basic knowledge about the space weather and the available data. Then, we briefly explained the most important deep learning techniques that we have used in our work. Afterwards, we detailed the work done on the magnetogram images and the autoencoders approach for dimensionality reduction. We tested different architectures and we showed that they give satisfying results in term of the compactness and the smoothness of the encoded representation. We then presented the detection of the CME using the LASCO images, we compared our detector to an existing one and we showed that we need to extract more information about the CME. Finally, we discussed the correlations between the extracted features and the space weather perturbations alongside with the solar wind speed prediction. We found a relation between the HALO CME and the magnetic disturbances affecting the Earth. We could also explain a part of the variance in the solar wind speed values even thought the prediction is not strong enough.

For the moment, we have a satisfying representation of the magnetogram images, but we need to improve the CME information extraction in order to improve the prediction of the perturbations: for instance, we still have difficulties in determining their exact arrival time and their duration. Moreover, the images of the EIT instruments were not used yet and we have to test the information that they contain about the space weather. Finally, we can extend the current predictions to the rest of the satellite indicators.

References

[KG99] R. Keppens and J. P. Goedbloed. "Numerical simulations of stellar winds: polytropic models". In: 343 (Mar. 1999), pp. 251–260. eprint: astro-ph/9901380.

[RB04] E. Robbrecht and D. Berghmans. "Automated recognition of coronal mass ejections (CMEs) in near-real-time data". In: 425 (Oct. 2004), pp. 1097–1106. DOI: 10.1051/0004-6361:20041302.

[HS06] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: http://science.sciencemag.org/content/313/5786/504.full.pdf. URL: http://science.sciencemag.org/content/313/5786/504.

[ZH08] Bertalan Zieger and Kenneth C. Hansen. "Statistical validation of a solar wind propagation model from 1 to 10 AU". In: *Journal of Geophysical Research: Space Physics* 113.A8 (2008). A08107, n/a–n/a. ISSN: 2156-2202. DOI: 10.1029/2008JA013046. URL: http://dx.doi.org/10.1029/2008JA013046.

[RBV09] E. Robbrecht, D. Berghmans, and R. A. M. Van der Linden. "Automated LASCO CME Catalog for Solar Cycle 23: Are CMEs Scale Invariant?" In: 691 (Feb. 2009), pp. 1222–1234. DOI: 10.1088/0004-637X/691/2/1222. arXiv: 0810.1252.

[KW13] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 [stat.ML].

[Doe16] C. Doersch. "Tutorial on Variational Autoencoders". In: *ArXiv e-prints* (June 2016). arXiv: 1606.05908 [stat.ML].

[SAM16] Schuh, Michael A., Angryk, Rafal A., and Martens, Petrus C. "A large-scale dataset of solar event reports from automated feature recognition modules". In: *J. Space Weather Space Clim.* 6 (2016), A22. DOI: 10.1051/swsc/2016015. URL: https://doi.org/10.1051/swsc/2016015.