**POLITECNICO DI TORINO**

Master's Degree in Computer Engineering

Master Thesis

# Deep Neural Networks For Detection Of Solar Corona Mass Ejections


Candidate: Alberto Calò

Supervisor: Professor Enrico Magli

Academic year 2018/19

# Acknowledgements

First I would like to thank Professor Enrico Magli, for his expertise, guidance and patience throughout the process of writing this thesis and for the opportunity he gave me. I would like to thank Assistant Professor Diego Valsesia for his always present availability and his technical advices.
Thanks to ESA and NASA for the CME catalog I used in this work. This CME catalog is generated and maintained at the CDAW Data Center by NASA and The Catholic University of America in cooperation with the Naval Research Laboratory. SOHO is a project of international cooperation between ESA and NASA.
A special thanks to my family that has always supported me and to my friends that always encourage me.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Purpose of the Thesis

Space exploration and astronomical knowledge reached by humanity in the last century have laid the foundations for new discoveries, however as a species we do not have a complete knowledge of the space that surrounds us. Goals such as the creation of satellite networks or the man in space have suffered and still suffer today from problems caused by our star. One of these problem is caused by an activity called Solar Corona Mass Ejection which can cause problems due to the high energy it contains. Not only in space, but also on earth, we can feel the effects of these activities because the most powerful ones can create geomagnetic storms. The purpose of this thesis is to provide a tool, based on modern Artificial Intelligence technologies that is able to support researchers in the recognition and detection of these particular events. The work done tries to classify these activities based on the observable variations that happens in solar images sequences taken in the various years of research. To achieve this goal, techniques based on deep convolutional neural networks have been used. To solve the space-time problem, a first technique of separating spatial information from temporal information was tested. Later, in order to understand if the space-time approach was valid a network based on the 3-dimensional convolution was tested.

## 1.2 Coronal Mass Ejection

A Coronal Mass Ejection is a huge release of solar matter during solar eruption. This kind of event generally occur with solar flares and they are related with the most well-know Solar Wind. A CME release large quantities of plasma and electromegnetic radiation into space above the sun's surface. These ejections can both sorrund the space near the sun (solar prominence) or go through the inter-

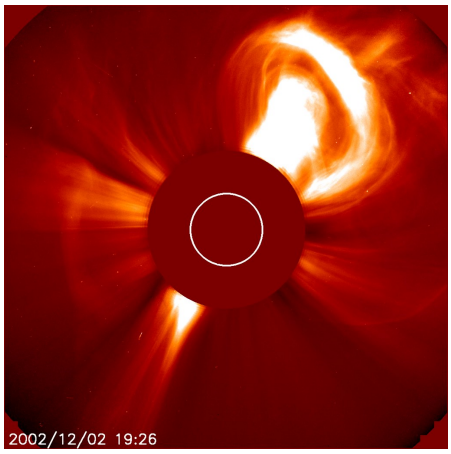


Figure 1.1: [1]A Coronal Mass Ejection

planetary space (interplanetary CME or ICME). These types of events are able to eject millions of coronal mass components, which is the mass relative to the outermost part of the sun, and carry a magnetic field that is more intense than the interplanetary magnetic field [2][3]. When a CME is directed towards the Earth interacts with the Solar Wind and the Interplanetary Magnetic Field. The propagation speed is therefore linked to this wind and this magnetic field. This means that after recognition, it is possible to evalute a solar event after 8 minutes from the begin, which is the time the light takes to reach the Earth from the Sun, while instead a CME spends one or more days before arrive on Earth [4].

### 1.2.1 Effects

The most well-know effect is Norther and Southern Lights (aurora borealis and aurora australis) that is the result of the meeting of solar energetic particles and Earth's magnetic field at the magnetic poles. CME can cause strong aurorae in large region, disrupt radio transmission and cause damage electrical transmission line facilities as well; the last one, potentially, can cause power outgate. The large energy product by a CME increase the number of free electrons in the ionosphere that is dangerous for people at high altitude. The same problem happens for instance at the astronouts in the space station [3]. Around the Earth, due the high energy particles moved by the Solar Wind demages to silicon-based components may happen; so this is a real big problem for the entire satellite system.

#### Future Risk

According to a report the chance of Earth being hit by a Carrington-class storm between 2012 and 2022 is up to12%[5]. The Carrington Event was a powerfull geomagnetic storm from 1855 to 1867. On the Earth it produced telegraph

disturbances[6] and aurorae lights seen from Roma[7], Cuba, Hawaii and Jamaica [3].

### 1.2.2 Detection

The following images represent two sequences of solar activity for two different days. Observing the flares moves you are able to detect if a CME happens or not.

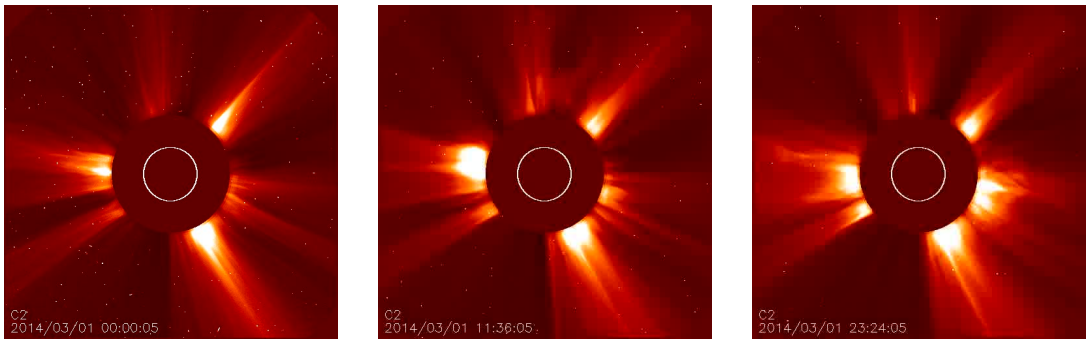


Figure 1.2: [1]Three images figure out daily sun activity. s can be seen with the naked eye in the three images there are few variations, with the exception of the intensity of the light of the last, which however is not sufficient to mark the activity as CME.

In the sequence above not relevant changes were recorded. Something seems to happen but it is not enough relevant to be marked as CME event.

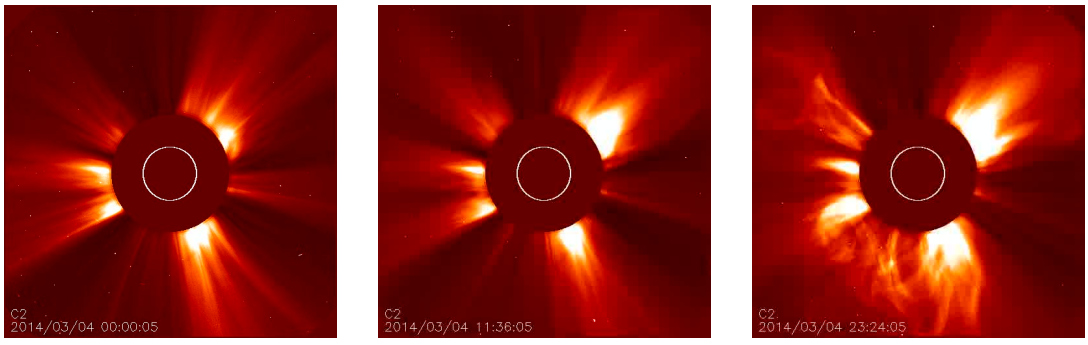


Figure 1.3: [1]Three images figure out daily sun activity. As can be easily seen in the last image, solar activity has increased considerably to the point of marking this activity as CME

The last sequence is more relevant; in fact small changes seems to happen between first and second images, but large differences are found moving between second and third images. As a result of this a new CME event is cataloged.

## 1.3 SOHO/LASCO

SOHO is the acronym of Solar & Heliospheric Observatory, it is a NASA and ESA project to study the Solar Corona and the Solar Wind. Among the numerous instruments inside the spacecraft there is LASCO (Large Angle and Spectrometric Coronagraph). The LASCO Catalog contains all CMEs manually identified since 1996. LASCO has three telescope C1, C2 and C3 but only C2 and C3 is used. The catalog is incomplete because of the absence of a strong automatic CME detector program [8]. This thesis worked on the images provided by the C2 telescope.

# Chapter 2

# Neural-Network-Based Classification of Images

In this chapter you see the main classification techniques and fundamental tools that make up a neural network, in particular a deep neural network, and a new approch that attempts a classification by extracting features from images and then treating them as time sequential informations.

## 2.1 Image Classification

Image Classification is a process inside the computer vision field that provide or try to provide classification of images working on visual context. An example is a Network able to distinguish within a dataset of animals which of these images represent a cat or a dog. In this context the informations extract from an image are called features.

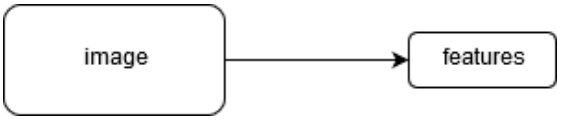


Figure 2.1: Features From An Input

The features generation algorithm give you information about the image; these informations are used by the last layer to classify the content of the image itself.

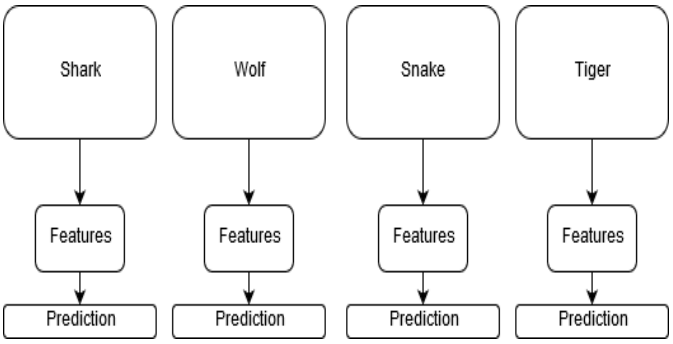


Figure 2.2: Classification Example

A Feature is an extremely important concept in Artificial Intelligence especially in Pattern Recognition. It is a property of an image, sound, or phenomenon in general that has been observed and is measurable. [9] Simply a feature is a property or variable on which our models will perform classification and prediction. In Computer Vision you are dealing with images and the achievement of a high accuray is given by how good you are at extracting features from images that feed the network. The neural networks that best serve this purpose are convolutional networks. CNNs are the most used neural network for classification purpose[10]. You speak of convolutional networks because the mainly technique used to extract features is convolution. In the next sections it will be explained when and how to exploit this mathematical tool by analyzing different problems, which can be solved by choosing, in rigth way, the most appropriate convolution. In particular we will see how the convolution in one dimension can give us useful information about the temporal trend; how two dimensional convolution can give information about the images; finally, how three dimensional convolution could give we information about sequential images. As said before currently the best algorithms for this task are based on Convolutional Neural Network. The binary classification allows to evaluate an image in a dataset and assign it a label that can be chosen between two values only, since the dataset contains only two types of content. For example, if we want to evaluate whether a set of mountains is covered with snow or not, we can choose for the single instance between the "snow-covered" label and the "not snow-covered" label. The multiclass classification allows to discover which class an image belongs to within a dataset going to evaluate various and different possibilities. For example, if you have a dataset that contains images of dogs, cats and elephants you want to assign to the single image only one of several different label values, obviously going to correctly predict a cat like a cat, a dog like a dog and so on.

## 2.2 Time Series Classification

An accurate temporal forecast is today very important in various sectors: from the prediction of faults in the automotive sector to the value of actions in the financial world, from the forecast of arrival of a transport system to the prediction of weather conditions, and so on. As the name suggests, the data must have a temporal or sequential logic.

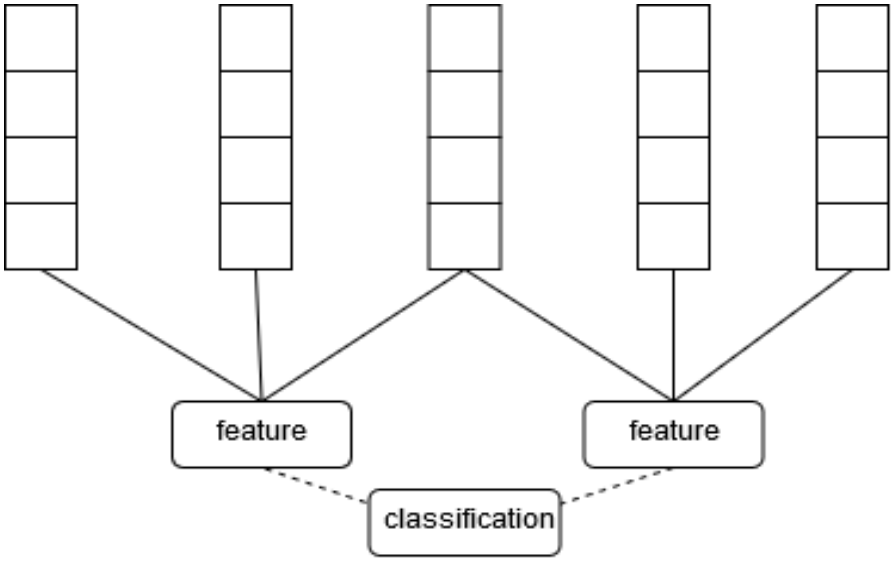


Figure 2.3: Features capture in time series. Following the time line, from left to right, we can observe the vectors that represent our data with their characteristics and properties. These vectors follow a sequential logic. Note that sometimes features are not extracted from sequential inputs, but you try directly to classify or predict (just imagine single input values instead of vectors; in this case it is useless to extract features because you could think of training the net simply by changing weights and biases).

As shown in figure 2.3 it is possible to extract features from events locally (in a temporal sense). Once again, through convolution it is possible to obtain relations between the various input properties. In particular you can try to find out what are the characteristics that over time change the final property of the input. In addition to images, other sequential data such as audio or texts can be finely processed with the Convolutional Neural Networks to reach state-of-the-art performance for document classification and speech recognition[11] and it is therefore possible to try to exploit these potentials in other areas such as videos or, as in this work, images in sequence. There are various areas in which this instrument is used: the classification of ECG signals (electrocardiogram), images classification, motion sensor data classification, and so on.

## 2.3 Neural Network

Trying to simplify the neural biological networks, the field of artificial intelligence has given birth to the so-called artificial neural networks (ANN). This type of networks are composed of "neurons" connected to each other,

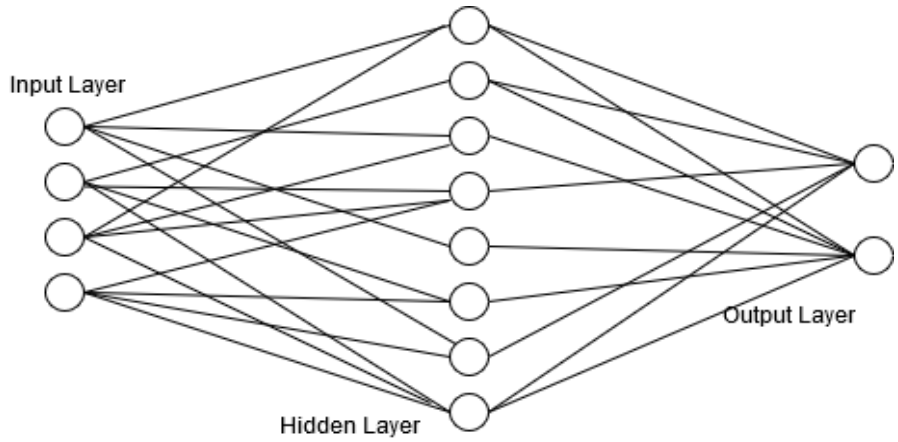


Figure 2.4: As you can see this network is composed by 3 layers: the first one is the input one, that is the data entry point (data is raw or pre-processed); the second is a hidden layer in which you can see relations, which are weighted; the third represents the final layer that through other weighted relationships will be the layer from which you will extract the information for our classification.

generally, by non-linear mathematical models that give them the peculiarity of being able to evolve over the training modifying biases and weights.

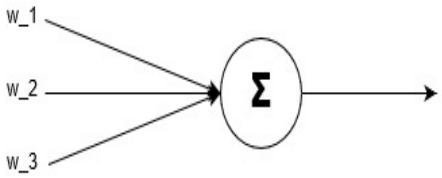


Figure 2.5: Figure 2.4 shows how the layers are composed by different neurons. Each neuron is generally represented as in this image: you have a neuron which is the sum of the weights, remembering that the weights represent the relations with other neurons.

Given an input (i.e. pixels of an image) a neurons multiplies it by the associated weight and adds the result to all the other inputs with their relative weights. At this point the neuron passes through an activation function that generates the final result. This type of neural networks are called *feedforward* because of flow is one directional: from the input layer to the final layer crossing the various hidden layers.

### 2.3.1 Convolutional Neural Network

One of the most important NN models is the Convolutional Neural Network, which is the most widely used model for resolving image-driven classification problems[12]. We can define a CNN as composed mainly of five key layers:

- INPUT layer: generally in the context of computer vision these are images. In theory, any data can be treated by a neural network, as long as it can be represented by an integer or a real. As seen in the previous paragraphs, there are many situations in which neural networks can be applied. An important note must be made about the quality and quantity of data: when possible you must try to use a large number of input data (whether they are used to train the network or to test the network); moreover it is very important to have quality data and to know which data properties are actually useful for our purpose (in this perspective you often talk about data pre-processing; even raw data can be used).

- CONVOLUTIONAL layer: this type of layer exploits the mathematical model of convolution to obtain an output starting from neurons that are linked together by a particular logical region (a more in-depth analysis of this part will follow shortly).

- SUBSAMPLING layer: it is used to reduce the size of the extracted features, it can be useful to resize the layers due to the computational effort trying not to affect the information obtained. The most common pooling layers are max and average pooling. Pooling is a process of discretization[13]. The objective is the down-sampling of the input representations (generally here the input is the output of another layer: for example, after extracting features by convolution from an image it is possible to resize what we have obtained) [15] reducing dimensionality and allow to formulate hypotheses on the features contained in the sub-regions binned [13].

- FULLY CONNECTED layer or GLOBAL AVERAGE POOLING layer: the fully connected layers are used to connect each neuron of a layer with the neurons of the next layer; in this way each neuron of the next layer is connected with all the neurons of the previous level. The number of neurons between one layer and another varies: we can find layers that have a lower number of neurons than the next layer. However, in convolutional neural networks this type of layer is used after the features extraction through the convolution mechanisms and therefore in CNN there are layers with a number of neurons higher than the subsequent layers. This is done in order to progressive reduce the size of the agtents that appear in the network. Using

of one or more of these layers allows you to generate the final classes of belonging of a specific input. Tipically you use stack fully connected layer and in general they perform well, but they introduce heavy computation due the huge number of parameters that could slow down the network and maybe overfitt. It is also possiible to use Global Average Pool on the feature maps [16]. Like the max-pooling the GAP performs a reduction in dimensionality, but unlike the max-pooling the GAP allows to perform an extreme reduction in the dimensionality, however not losing points about the quality of information extracted and also allowing a lower computational effort compared to one fully connected layers stack.

- OUTPUT layer: it represents the layer containing all the possible classes of belonging that our network is able to predict. It is often convenient to use two concepts: output and logit. The logit represents the exit from the last layer of the network before going through an activation function. The output instead represents the value produced by the activation value. While the output is used to verify the accuracy of the classification, the logit is used to manage an optimizer and then to train the network.

Although various types of layers are used in the creation of this network, as the name easily suggests, the main component is the Convolutional layer. *Convolution*, in functional analysis, is an operation of two functions that produces a third function which expresses how one of the starting function changes its shape due to the other one. The following represent the mathematical formula of convolution:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

### 2.3.2 1-D Convolution

As the name implies, convolution occurs in one dimension. Two different vectors called Input and Kernel participate in the convolution in the form *(Input * Kernel) ($\Delta T$)* producing an output vector. As can be seen in the image below, the kernel size is generally small: in fact, we want to have a receptive field able of capturing information regarding the inputs considered to be nearby. In this way it is possible to segment the input locally and obtain features on these individual segments.

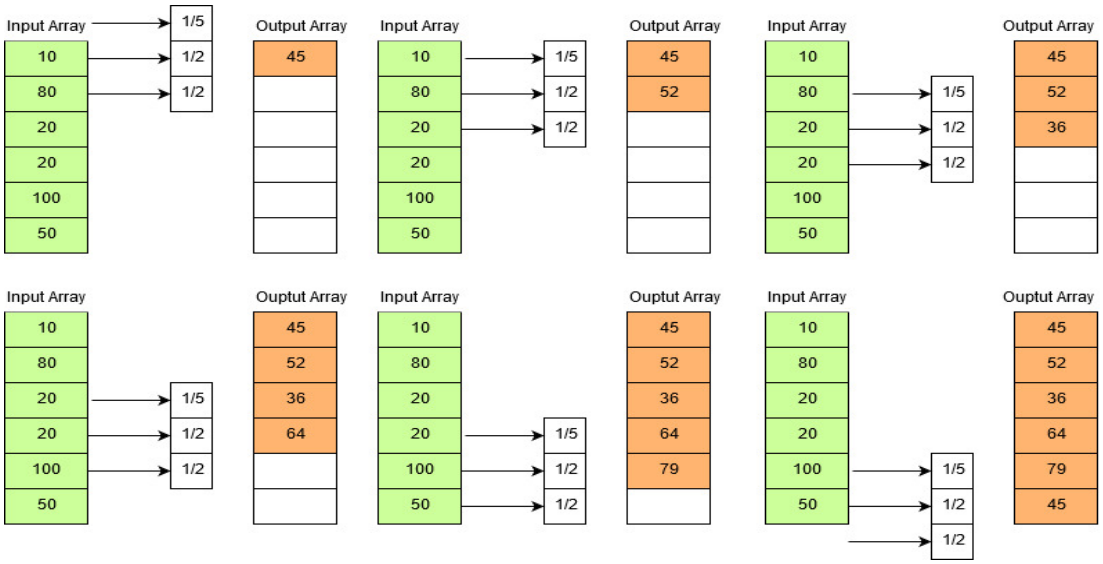


Figure 2.6: An example of how a vector is convolved with another generating a new vector. The kernel, represented by the white vector, moves across the entire input array. In particular, at each iteration (the step is defined by strides: strides equal to one means moving one step forward, strides equal to two moving two steps and so on. In this example the strides is equal to one) multiply the input values that metch the floating window, or rather the kernel. Once multiplied, the final value is added and obtained. In the second iteration for example the calculation is $(10 * 1/5) + (80 * 1/2) + (20 * 1/2) = 52$. As you can see in the first and in the last iteration the sliding windows does not completely match the input array: in this case the cell assumes a value of zero and therefore the calculation in the first iteration is $(0 * 1/5) + (10 * 1/2) + (80 * 1/2) = 45$. The choice to use 0 and start from a cell with an index lower than the starting one is dictated by padding: in this case the padding is *same*; in other cases, as in padding *reflect* instead of using zero, the opposite value is reflected, in this case the last of the vector.

As mentioned above the convolution takes place in the form *(Input * Kernel) ($\Delta T$)*. This means that we are going to evaluate Input and Kernel as a domain varies, this domain can be both spatial and temporal; in this specific case, the temporal character of the domain was highlighted in this context since this type of convolution is particularly suitable for the analysis of the series following a temporal logic.

### 2.3.3 2-D Convolution

The convolution seen previously works in one dimension, but in reality there is no upper limit to the dimensionality of the functions, vectors or matrices to which it can be applied. For example, if we consider an image, it can be seen as a pixels matrix and therefore be considered as a two-dimensional structure. The multidi-

mensional convolution works as the one-dimensional convolution, therefore starting from two functions $f$ and $g$ of n-dimensions, a *res* function of n-dimensions is obtained as well. By creating a 2-dimensional Kernel you can use this mathematical tool to extract information from an image. Inputs and kernels are therefore in the form *(Input * Kernel)* $(\Delta x, \Delta y)$.

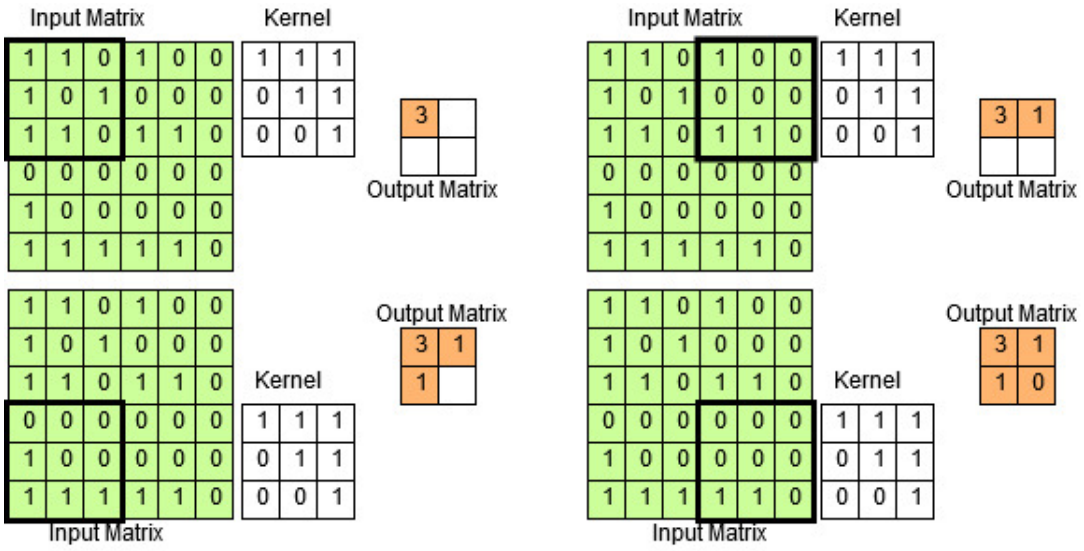


Figure 2.7: An Example of 2-D Convolution. In this example the kernel is a two-dimensional matrix as well as the input (if you are on an initial layer this input could be an image where each cell of the matrix represents a pixel). In this case the strides is three, in fact we can see how the first cell of the 3 * 3 kernel is in the first iteration in position (0,0) while in the second it is in position (0, 3). The calculation of the first iteration is (1*1) + (1*1) + (1*0) + (1*0) + (0*0) + (1*1) + (1*0) + (1*0) + (0*1) = 3. In this case you do not start from (-1, -1), but from (0, 0) because the padding used is *valid*.

### 2.3.4 3-D Convolution

A 3-dimensional neural convolution network is generally used to extract features from three-dimensional matrices or to establish a relationship between 3 dimensions. It generally used to extract patterns from three-dimensional images by studying fields such as height, width and depth. A specific use is in the medical field for brain tumor segmentation [17], or even the detection and segmentation of videos [18]. Over the years various attempts have been made, such as the recognition of human actions, however due to the large number of necessary data and the difficulties faced when training 3D convolution kernels only limited successes have been obtained [19].As video analysis uses frames as a temporal sequence of images in this work you evaluated the usefulness of using this type of convolution to extract information about the space-time transformations that occur in the

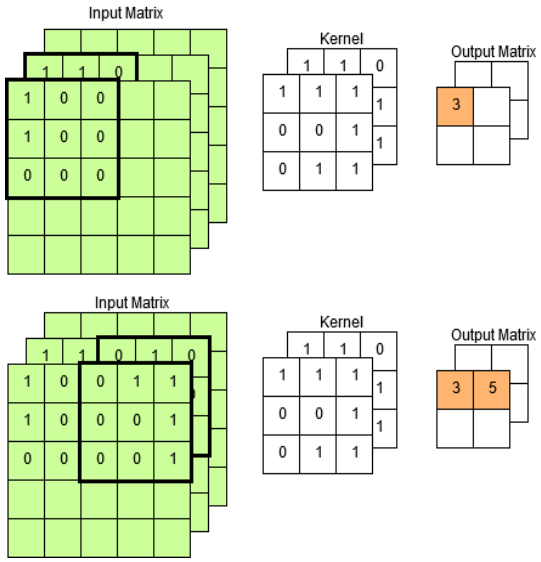


Figure 2.8: An example of 3 dimensional convolution. As you can see the kernel is a 3 dimensional matrix and the input matrix is threedimensional as well. In this case the padding is *valid*, it means you start from position (0,0) and every iteration the kernal does not spill out the matrix. The first calculation is [(1*1)+(0*1)+(0*1)+(1*0)+(0*0)+(0*1)+(0*0)+(0*0)+(0*1)] + [(1*1)+(1*1)+(0*0)+(0*0)+(0*0)+(0*1)+(0*0)+(0*0)+(0*1)] = 3. On this case the strides is two on te x and y axes, despite in z direction is 1.

time sequences of solar photographs. Inputs and kernels are in the form *(Input * Kernel)* $(\Delta x, \Delta y, \Delta z)$ or keeping in mind the purpose of this work you can think of transforming the last dimension from spatial to temporal obtaining *(Input * Kernel)* $(\Delta x, \Delta y, \Delta t)$.

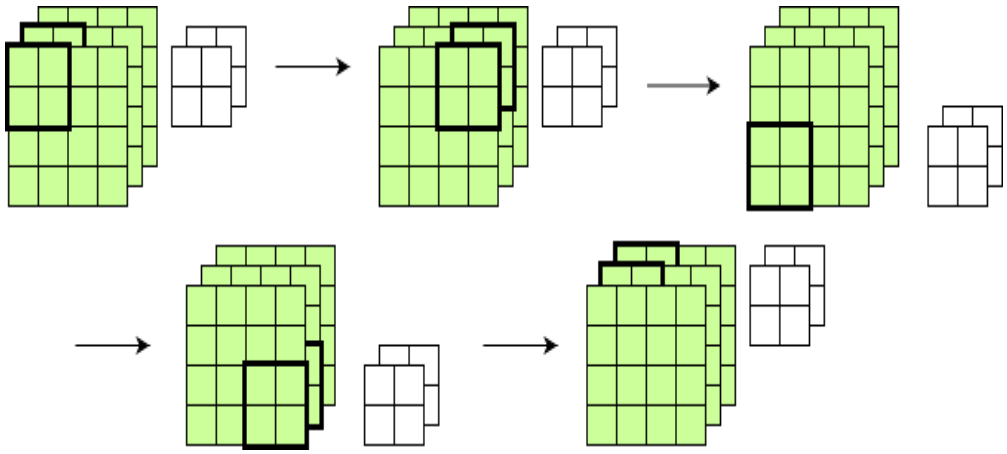

Figure 2.9: In this figure the trend of the convolution is shown and how it moves in the three dimensions: you have first a displacement on the x axis, successively on the y axis and finally on the z axis.

### Disjoint Space from Time

The most common technologies in the field of deep learning adopt an approach like the one seen above, that is to take advantage of the convolution in three dimensions when dealing with spatio-temporal data. The work done during this thesis has also tried a new type of approach: the separation of spatial dimensions from temporal dimensions. In particular through the use of the convolution in two dimensions a features extractor was built; features that were then the entry of a new network that through the one dimensional convolution tries to produce a time series classification.

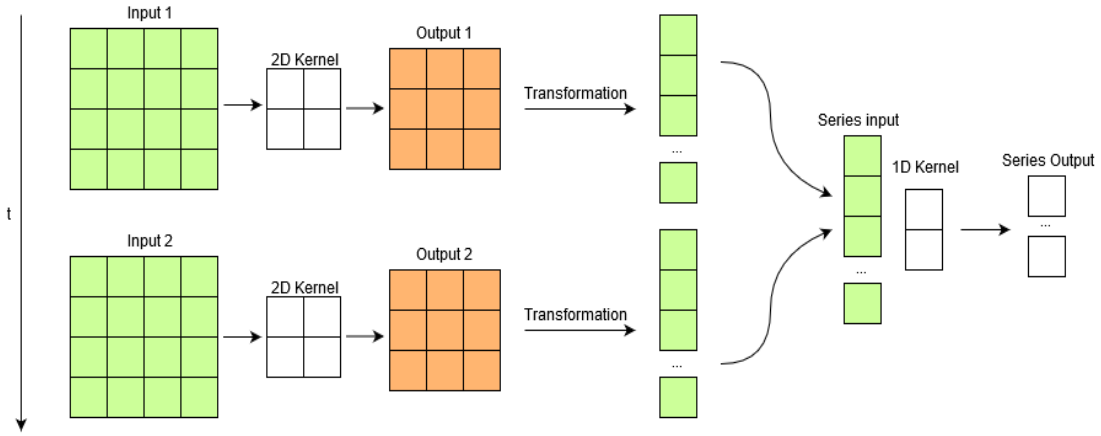


Figure 2.10: Composed 3D Convolution. The figure explains how it is possible to separate the spatial dimensions from the temporal dimensions. On the left side of the image you see how a pair of images, which have a sequential logic between them, are elaborated individually with the convolution techniques in 2 dimensions.Once the features maps of both images are obtained, through appropriate transformations, they are a layer input convoluted for the time dimension.

Some works have been carried out in this direction, but the approach involves two-dimensional extractions followed by temporal analysis in the single blocks [20].The one tested in this thesis is, as mentioned, an approach that aims first at the complete extraction of all the features from all the images in the sequences and then completes the classification through temporal analysis.

# Chapter 3

# Data Pre-Processing

## 3.1 The FITS file

FITS (Flexible Image Transport System) is the data format that is most used for scientific data in the astronomical field. FITS file is structured on matrices and multidimensional arrays and it allows the analysis and storage of these files (as in our case the photos from the coronograph) in a format that allows, besides the storage of the image, the creation of tables for the storage of information (therefore you have a header that allows you to have access to numerous information such as the date of creation of the file and the date on which the image was taken, or who treated it, or the coronograph that took the image, and so on) [14]

## 3.2 The Time filter

You want to create sequences of images that can feed a neural network able to extract space-time information from these sequences. Analyzing just the temporal concept it is clear that 2 images taken at **t_1** and **t_2** that represent the transaction from a state of "quiet" of the sun to a state that causes a CME, cannot be considered in the same way as two images that similarly produce this transaction, but which are taken in a range **t_1** $>>$ **t_2**. An analysis carried out throughout the data set showed that the most common number of seconds between two images is 720. In order to obtain the maximum number of pairs close to each other all the images at a distance between 710 seconds and 730 seconds are considered valid for our purposes, while the others are discarded. It is useful to note that the images at the beginning of the dataset have a higher average than the most recent images that have an average like the one computed. This trend show you how manual photos management has been improved over time.

## 3.3 Size Error And Photo Error

Once discarded images that do not follow the temporal logic previously descripted, the aim next is to analyze which of these images is actually usable for our purpose. The first check performed is on the FITS file headers: in this way anomalous values in the dimensions can be easily identified and images that do not respect the standard dimensions for height and width values can be discarded. The allowed value for both sizes is 1024: the images are therefore 1024 * 1024. Although the

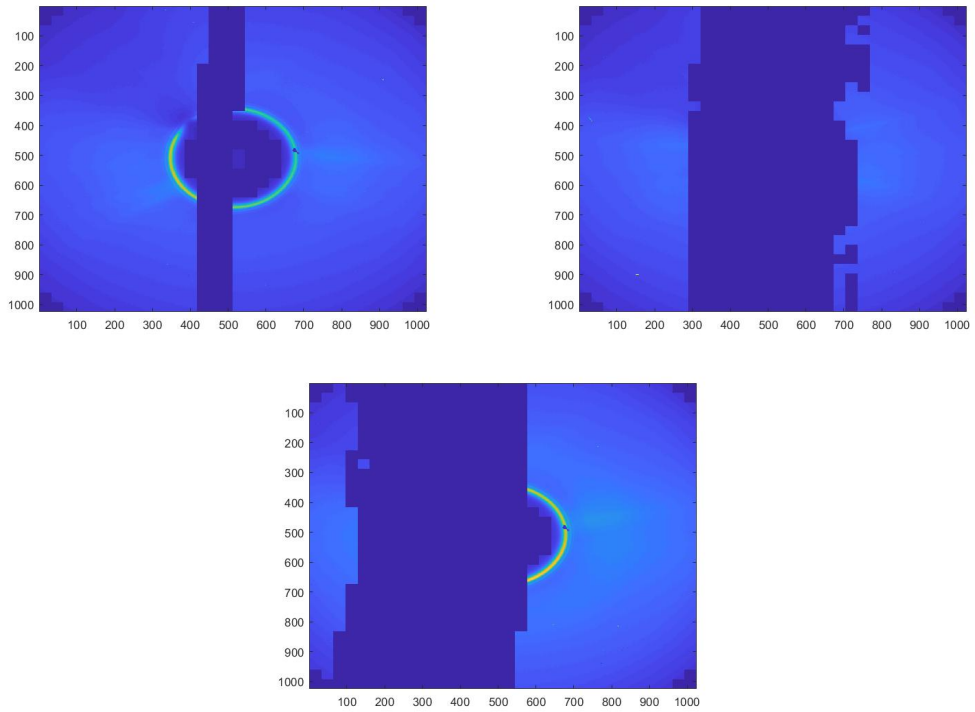


Figure 3.1: Discarded images. The three images in the figure represent corrupt images that have been discarded. It is evident to a visual analysis as well that these images can cause problems in the future training of the network. In the first image, although the solar corona is visible, coronal mass expulsion may have occurred in the portion of the image where we have no information. In the third image half of the solar corona and half of the space around the sun are inaccessible. In the second the solar corona is even completely inaccessible.

header gives you a first advice on the images to be discarded a more accurate analysis of the properties it is done: it was discovered that in some of these images one dimension, or both, does not have the expected values. The image is therefore again processed even if the header returned the correct values: when unexpected values of x or y are found, the image is discarded. After this filtering, using statistical analysis on samples, it is discovered that some images pass the dimensional filter, but have "switched-off" pixels. Figures 3.1 and 3.2 show the meaning of "switched-off". Normal "switched-off" pixels depend on the darkening

of the center of the sun in order to better capture the lights of the solar corona: the value associated with these pixels is therefore zero. It has been noted that an image that optimally darkens the solar corona has a value of zero pixels equal to 62463 or 62464. By studying all the filtered images, the photos that have a value much higher than the previous values of zero pixels are discarded (remember that the coronograph is subject to possible radiations coming from space and it is therefore possible that the images are not transferred to Earth in the correct way). In order to avoid an excessive downsizing of the dataset only the strongly corrupted images have been discarded: the images that had less than double pixels that are expected to be corrected have been kept as good images.

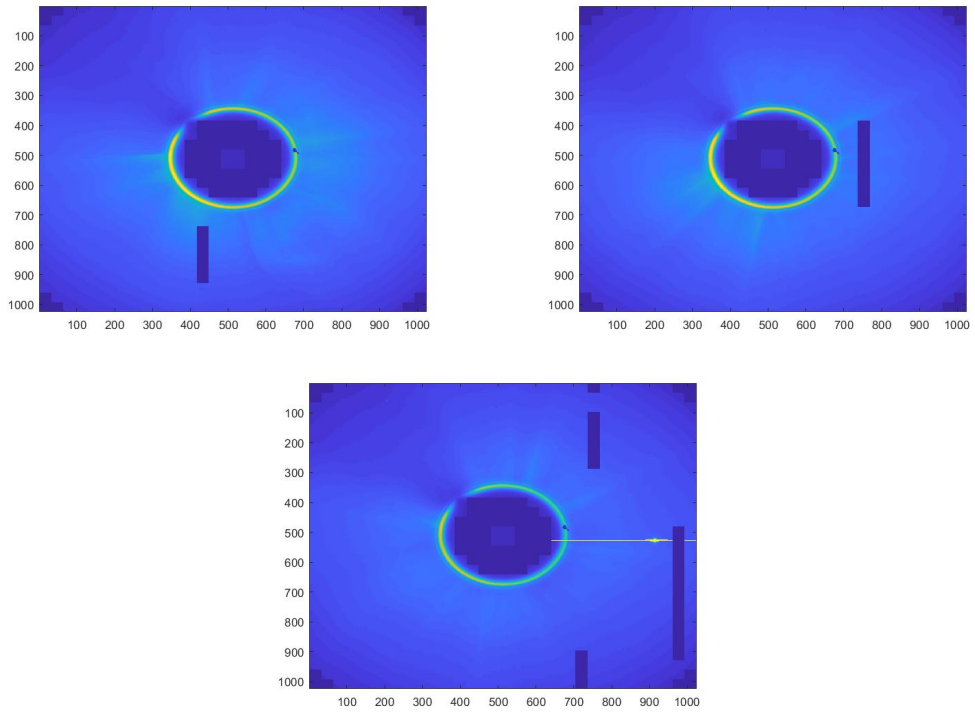


Figure 3.2: Images with small errors not discarded. Although some information has been lost in these images it is clear that the main information is still available to us and therefore it was decided to use these photos in the future training of the network.

Another important thing is the images that compose coronograph first months usage: in this part of the dataset, in fact, the center of the sun is darkened, but the value of the pixels is not zero such as those discussed above. In this situation the number of pixels equal to zero of an image must be exactly zero. Particular attention therefore has to be paid to avoid that correct images are treated as corrupted.

## 3.4 Polar Transformation

To have an effective representation of round images like the sun, remainig images have been transformed into polar coordinates going to reduce also the dimensionality of the y axis. This step is the last step in filtering and transforming the data. In the following paragraphs an analysis is made on the contents of the images themselves and a focus is done on the right choice of the best temporal sequences length. Figure 3.3 shows two sample of the tranformation ouput.

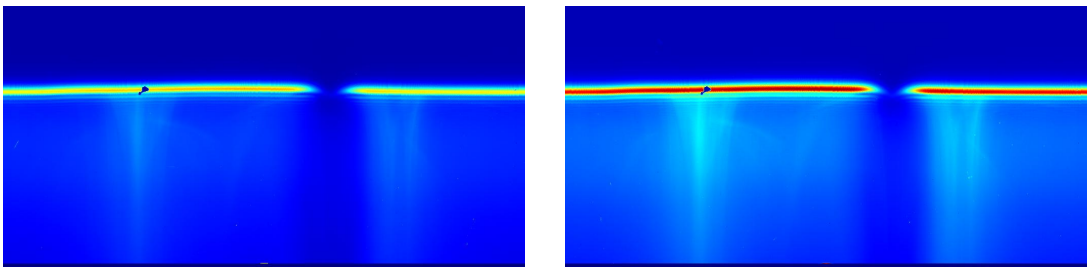

Figure 3.3: Two samples obtained after filtering and transforming.As can be easily observed, you have gone from polar coordinates as in the figure 3.2 to cartesian ones.

## 3.5 Sequences selection

Due to the large number of data and their quality, it was necessary to analyze these images and discarded the corrupted images before being able to feed the network with these data. In addition to a purely qualitative selection, we must remember that our intent is to classify time sequences. In order to get the best trade-off between the total sequences number and the length of the each sequence, the following table can be observed:

| Seq Length | No. of Sequences | No. of Events |
|---|---|---|
| 2 | 168550 | 139431 |
| 3 | 150656 | 187042 |
| 4 | 134200 | 221918 |
| 5 | 119268 | 245979 |
| 6 | 106057 | 261836 |
| 7 | 95393 | 274425 |
| 8 | 86909 | 285493 |
| 9 | 80007 | 295321 |
| 10 | 73401 | 301329 |
| 11 | 67017 | 300601 |
| 12 | 60833 | 297734 |
| 13 | 54864 | 290119 |
| 14 | 49167 | 278965 |
| 15 | 43657 | 264289 |
| 16 | 38332 | 246179 |
| 17 | 33315 | 225691 |
| 18 | 28985 | 206249 |
| 19 | 25651 | 191166 |
| 20 | 22439 | 174194 |

Table 3.1: The possible lengths of the sequences and the relative number of sequences obtained on the data held. Next to this information it is also possible to see the number of events that can be obtained starting from a certain length.

| No. of Events | No. of Sequences |
|---|---|
| 0 | 27456 |
| 1 | 2804 |
| 2 | 3064 |
| 3 | 3462 |
| 4 | 3823 |
| 5 | 4248 |
| 6 | 4285 |
| 7 | 4052 |
| 8 | 3839 |
| 9 | 3715 |
| 10 | 12653 |

Table 3.2: Number of sequences per number of events with length 10. It can easily be seen that the number of sequences made up entirely of non-events is much greater than in all the other types of sequences. In second position we find sequences entirely constituted by events. These two pieces of information allow us to understand that the general rule is that when the sun produces events these happen for a long time, as when the sun is in a quiet phase it does not produce event for a long time (generally longer than when it produces events).

The goal is to get the best balanced sequences. It means that the number of

*sequences containing Not-Events* should be as close as possible to the number of *sequences containing Events*. For instance, choosing 2 as length, 93538 sequences contain only Not-Events, 10593 sequences contain a Not-Event and an Event and 64419 contains only Events are generated; you cannot take this because the sequences are not balanced enough. The length that most closely fits this property is 10. Using 10 you obtain the most uniform distribution of sequences contain events and not events. The number of sequences which contain no events and sequences that only contain events still remains to be balanced. Furthermore, we need to balance the number of *sequences that produce an event* and *sequences that do not produce an event* (this is done in the following paragraph while creating the datasets).

## 3.6 Data Analysis

Before proceeding with the creation of data sets it is useful to take a look at how they are actually composed. Analyzing the images in a given period and computing the average for each image, we get obtain following graphic:

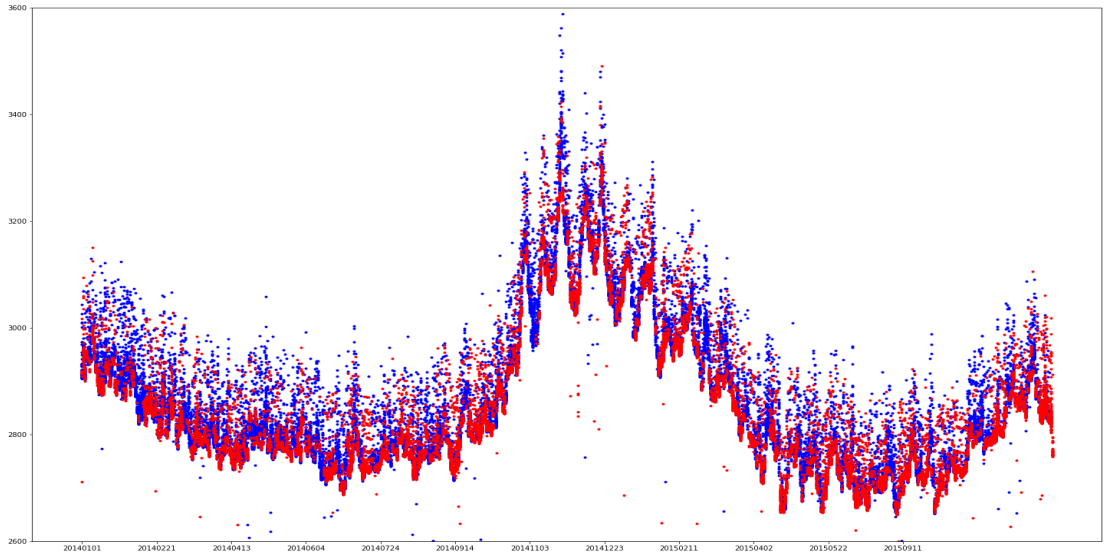


Figure 3.4: Image's mean per date in range 01/01/2014 - 31/12/2015

In the x-axis it is represented every day of which we hold data starting from January 1st of 2014 until December 31st of 2015 (a total of two years of images). For each of these days, the average of each image associated with that day is calculated (since each image is composed of pixels, the average value of all the pixels contained in a given image can be calculated). The y-axis therefore indicates the average value of each image.
As you can see, there is a difference, even if not particularly strong, between the

average of the not-cme (in red) and the cme (in blue) on a particular day; this makes us notice how the distinction and therefore the classification of CMEs may be possible for a purely visual analysis. It's important to note that this difference is visible only when CME and not-CME occur on the same day, or at most in neighboring days (if you take images at a great distance over time, like a CME on April 4th of 2014 and a not-CME on December 23 of 2014, in not-CME turns out to have a much higher average than CME, and this makes them incomparable, this could be due to the fact that the coronograph orbits with the Earth around the Sun, and therefore performs a periodic movement, which is easily observed in the image above, between Aphelion and Perihelion, changing the exposure of the taken photo); this can give value to the type of experiment dealt with in this thesis in that by analyzing how features change over time we try to find a change in the image from the sun at the local level. We recall as mentioned in the previous paragraph that to train the network, series of length 10 are used where on average the "temporal distance" between one image and the other is 720 seconds; with the exception of some particular cases where some images are at a distance of 1440 seconds, this last choice was made in order not to excessively reduce the size of the final datasets.

Despite these differences at the local level, you can see that often not-CME and CME of the same day have almost identical averages and this could lead to a difficulty in their correct classification. Furthermore, it can easily be seen how outliers exist for both classes, which could cause the classification of certain sequences to fail. The problem of exposure also occurs at the local level even if with less impact. To avoid this problem, each image of each sequence in the tests described in the following paragraphs is normalized by subtracting each image from its support. The average subtraction per channel is used to center the data around zero for each channel (in this case a pixel in the image is not RGB, but is represented by a float). This generally helps the network learn faster because the grades are uniformly for that channel.

# Chapter 4

# Models Definition and Evaluation

## 4.1 Temporal Convolutional Neural Network

The approach performed in this work consists of a neural network divided mainly into two parts:

- **Features Extractor**: the first is a deep convolutional network designed to extract spatial features from a sequence of images. Being focused on spatial concepts, this type of network extracts information from the single image in order to obtain features (exactly 64 for each image) which then will be used by the second part of the network. Once the features from each single image are obtained, through a Global Avarege Pooling and a reshaping of the last layer, the data show only temporal information.

- **Temporal Network**: information in input at this point must to be handle in order to learn how they evolve over time. For this purpose this Temporal Convolutional Network is used, taking advantage from one dimensional convolution potentiality. At the end, information obtained is used for classification of sequences and for computation of loss that allows us to train the network itself.

The following figure represents the main part of the features extractor:

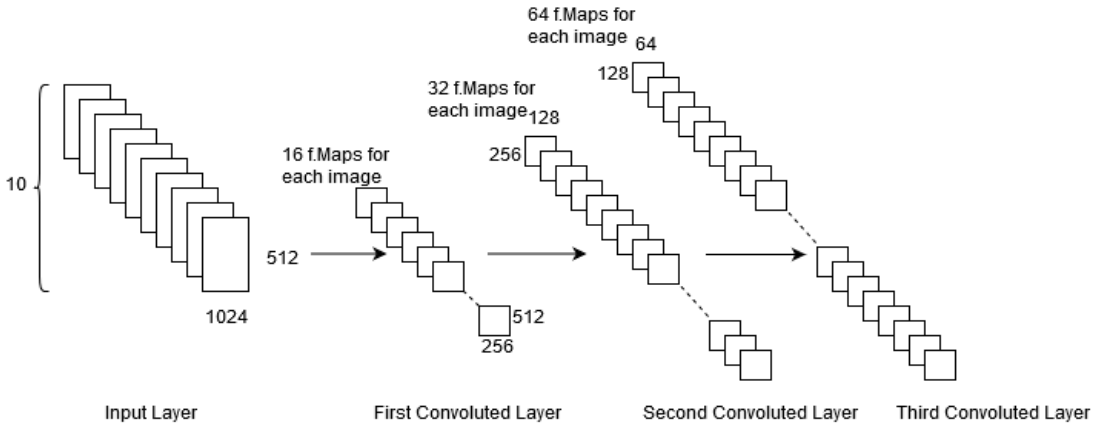


Figure 4.1: Features Extractor Starting From a Sequences of 10 images, for representative simplicity only the features maps of a single image are shown.

Starting from a sequence of 10 images, each polarized and therefore of size 1024 * 512, we pass through a convolution in 2 dimensions in which the associated weights and the relative kernel have dimension:

$$[3, 3, 1, 16]\ [1, 2, 2, 1]$$

The first two dimensions (3, 3) represent the receptive spatial kernel field, the third (1) the number being at the first layer represents the number of input channels of each image. Remember that the **input layer** is given in the format:

$$[Batch\_size\_per\_seq\_length, 1024, 512, 1]$$

Finally the last dimension (16) represents the number of features extracted at this level. In order to perform subsampling a *Kernel Dimensionality Reduction* is used, instead of the common average or max pooling strategies. Therefore the strides used have the same spatial dimensions as the kernel:

$$[1, 3, 3, 1]\ [1, 2, 2, 1]$$

Obviously first and last dimension are equal to 1. In this way we can reduce the dimension of x and y input image axis. This action produces **first hidden layer** with dimension:

$$[Batch\_size\_per\_seq\_length, 512, 256, 16]$$

Proceeding in this way, because the same type of convolution is used, the other hidden layers are obtained. Using the previous layer as input and using a kernel of dimensions:

$$[3, 3, 16, 32]\ \ , [1, 2, 2, 1]$$

we get the **second hidden layer** with dimension:

$$[Batch\_size\_per\_seq\_length, 256, 128, 32]$$

And with:

$$[3, 3, 32, 64]$$

as kernel, you get the **third hidden layer** with dimension:

$$[Batch\_size\_per\_seq\_length, 128, , 64]$$

We finally got the 64 features. Now we need to reduce the spatial dimensionality so that upon exiting this network we only have the above mentioned features. The GAP (Global Average Pool) is then used as solution.

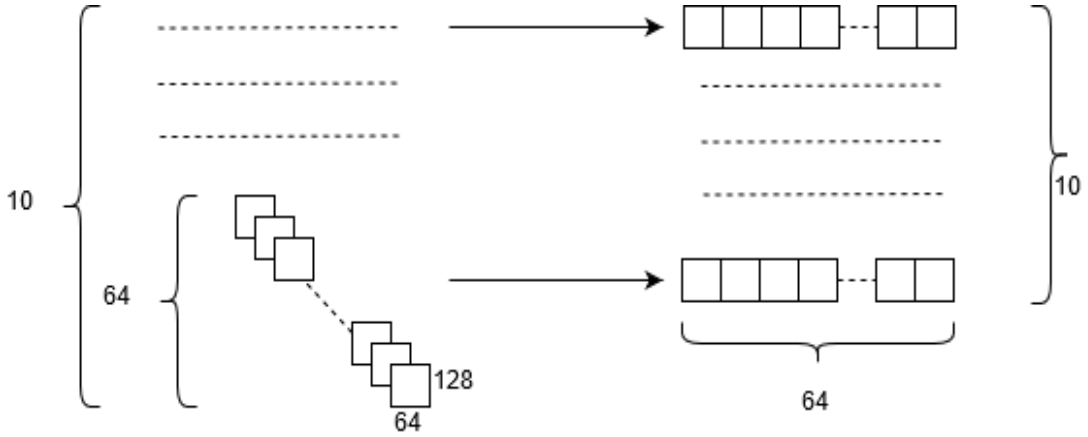


Figure 4.2: Global Average Pooling, for representative simplicity only the features maps extracted from a single image are shown. Note that each cell on the right is not a matrix, but a scalar number.

As the name suggests, the GAP is nothing more than an average pool layer used to reduce the global size of the input layer. In this solution it is used to bring the dimension of the third layer hidden to:

$$[Batch\_size\_per\_seq\_length, 1, 1, 64]$$

$$[Batch\_size\_per\_seq\_length, 64]$$

At this point, remembering that the length of the series is equal to 10, a reshaping occurs in order to obtain as **output of the feature extractor** a tensor with dimension:

$$[Batch\_size, 10, 64]$$

As you can see we are trying to move from two spatial dimensions to a temporal

one, at this point this type of information can be treated, and then passed, to the second part of the network, the temporal one.
The following image shows the first layer of this part of the network, **the input layer of the temporal network**:

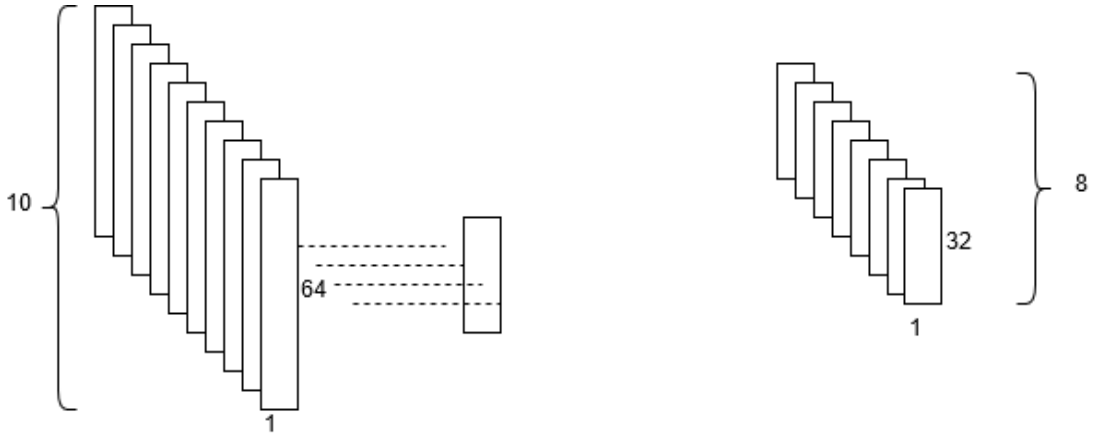


Figure 4.3: Temporal Network at input layer. On the left side the input of the temporal network is shown; using a kernel and convolution in 1 dimension the dimensionality is reduced. On the right side you can se the products of these convolution: a smaller number of features have been summerized.

This layer takes in input the 64 features of each of the 10 images that make up the sequence to reduce the dimensionality in order to understand the temporal progression of the sequence. Starting from a size input:

$$[Batch\_size, 10, 64]$$

Going through a one-dimensional convolution with a shape kernel:

$$[3, 64, 32]$$

The first dimension (3) represents the time window (and therefore the cardinality of extracted features that you want to observe together) that the kernel is able to associate with a specific location; second (64) and third (32) represent respectively the number of inputs and the number of outputs associated with the relative weights and biases (the latter only for the outputs). The first temporal hidden layer is obtained:

$$[Batch\_size, 8, 32]$$

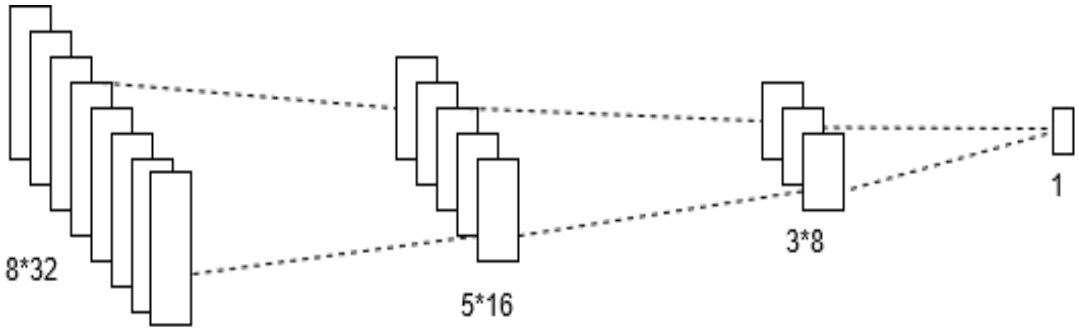


Figure 4.4: Temporal Network at input layer..

Figure 4.5 shows how the process described above can be repeated for the other layers in order to obtain a logits unit that can be used for prediction.
Proceeding in the same way with a kernel of dimensions:

$$[4, 32, 16]$$

the second hidden temporal layer of dimensions is obtained:

$$[Batch\_size, 5, 16]$$

With:

$$[3, 16, 8]$$

you get the third:

$$[Batch\_size, 3, 8]$$

and finally with:

$$[3, 8, 1]$$

you get the scalar value that will be used for the prediction:

$$[Batch\_size, 1]$$

Once this value is obtained, you can go through an activation function to get our prediction. Since our aim is to obtain a binary classification, the sigmoid is used as activation function. The sigmoid function is an increasing monotonic function that comes from -$\infty$ and arrives at +$\infty$, but at value 0 in the x-axis it is exactly 0.5. Using this function we can classify a sequence entering our network as *a sequence that generates a CME* if the output value is greater than 0.5, as *a sequence that does not generate a CME* otherwise. To do this the $y$ obtained from the sigmoid is rounded to 0 or to 1.

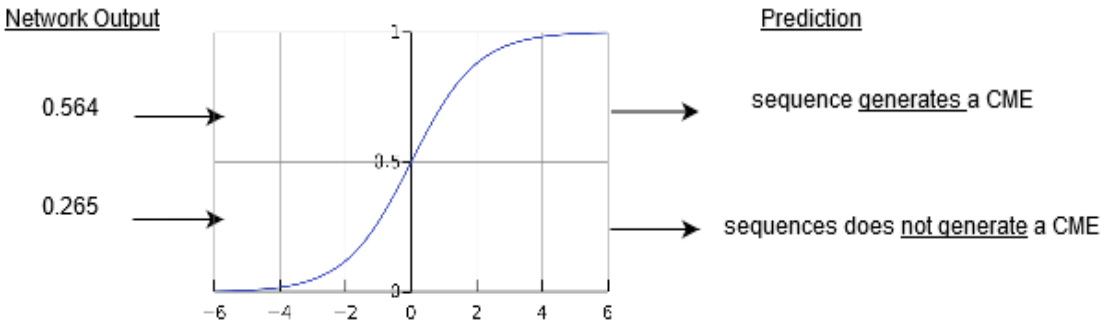


Figure 4.5: [21]Example of how Sigmoid Function Work

$$S(x) = \frac{1}{1 + exp(-x)}$$

$$S(x) = \frac{\exp(x)}{exp(x) - 1}$$

Using this function the *sigmoid_cross_entropy_with_logits* of *tensorflow* during training is used to give information to the optimizer, so that it can be able to perform the backpropagation. An Adam Optimizer is used to perform the learning.

**Summary**

$$Input : [Batch\_size\_per\_seq\_length, 1024, 512, 1]$$

$$Hidden\_Layer_1 : [Batch\_size\_per\_seq\_length, 512, 256, 16]$$

$$Hidden\_Layer_2 : [Batch\_size\_per\_seq\_length, 256, 128, 32]$$

$$Hidden\_Layer_3 : [Batch\_size\_per\_seq\_length, 128, 64, 64]$$

$$GAP : [Batch\_size\_per\_seq\_length, 1, 1, 64]$$

$$GAP_{\text{reshaped}}, Temporal\_Input : [Batch\_size, 10, 64]$$

$$Hidden\_Layer_{1\_\text{temporal}} : [Batch\_size, 8, 32]$$

$$Hidden\_Layer_{2\_\text{temporal}} : [Batch\_size, 5, 16]$$

$$Hidden\_Layer_{3\_\text{temporal}} : [Batch\_size, 3, 8]$$

$$Logits, Outputs : [Batch\_size, 1]$$

## 4.2 Testing the Datasets

To create the datasets a note must be made: the events in the catalog are cataloged with a value from q0 to q5, which represents the quality (intended as intensity) of the event.
From now, different datasets are defined:

- the first one, called D0, contains events of all qualities;

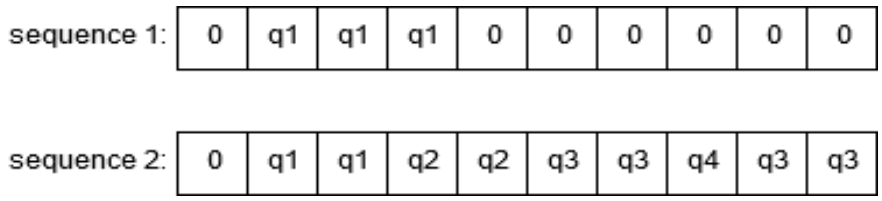


Figure 4.6: Example of sequences in D-0. Sequence 1 does not produce a CME: in fact the image in position 10 is labelled with 0. Sequence 2 produces a CME: in fact image in position 10 is labelled with 1. Remember that a *NOT CME* is labelled with 0 and a *CME* with quality q0, q1, q2, q3, q4, q5 is labelled with 1

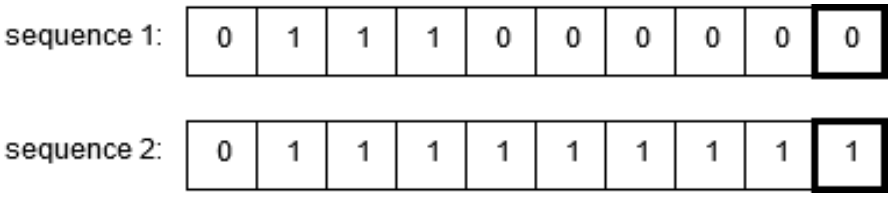


Figure 4.7: Example of sequences in D-0 with label. This figure reproduces the figure above replacing the quality with the label 1.The cells with the thick lines represent the label of the entire sequence.

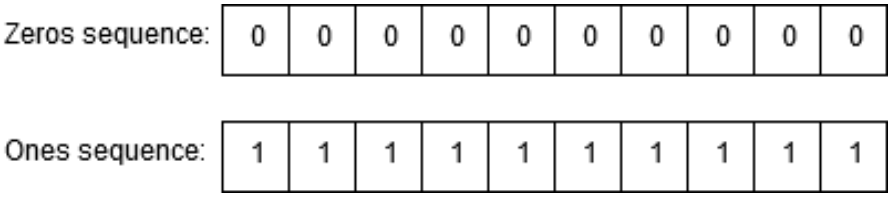


Figure 4.8: Example of sequences in D-0 with images marked similar. In the dataset D0 (in D1 and in D2 as well) it is common to find sequences like the two shown in figure: often the images in the sequence contain the same marking values.

- the second one, called D1, contains only events with quality 3, 4 and 5;

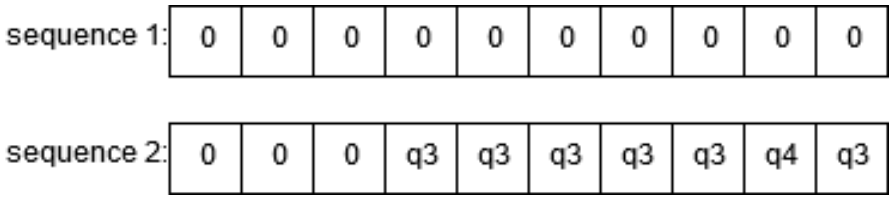


Figure 4.9: Example of sequences in D1. Sequence 1 does not produce a CME: in fact the image in position 10 is labelled with 0. Sequence 2 produces a CME: in fact image in position 10 is a q3 quality event marked as 1.

- the third and last one, called D2, contains all events qualities within the sequences, but the event that appears in position 10 has quality 3, 4 or 5.

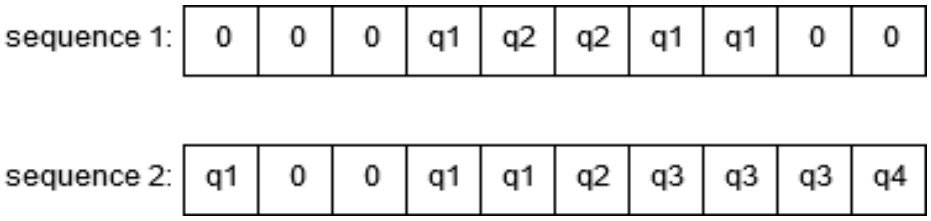


Figure 4.10: Example of sequences in D2. Sequence 1 does not produce a CME: in fact the image in position 10 is labelled with 0. Sequence 2 produces a CME: in fact image in position 10 is a q4 quality event marked as 1.

Of course the work done in figure 4.7 is done for all the datasets.
In order to allow the network to learn to recognize both types of sequences, it is necessary to create fairly balanced training sets. Furthermore, to avoid that variance and standard deviation can give us altered values during validation and testing, these last two datasets have also been balanced so that the number of labels of CME generating sequences is similar to the number of sequences that do not generate CME. This causes a further elimination of data, especially in training sets.

### 4.2.1 D-0

**Training Set**

Label 0: 27917, *after resizing*: 20000
Label 1: 19059, *after resizing*: 19059
Sequences 0 ones: 12907 (0 are labelled with 1)
Sequences 1 ones: 1488 (880 are labelled with 1)
Sequences 2 ones: 1696 (987 are labelled with 1)
Sequences 3 ones: 1889 (1071 are labelled with 1)
Sequences 4 ones: 2134 (1215 are labelled with 1)

Sequences 5 ones: 2308 (1340 are labelled with 1)
Sequences 6 ones: 2355 (1380 are labelled with 1)
Sequences 7 ones: 2154 (1347 are labelled with 1)
Sequences 8 ones: 2158 (1438 are labelled with 1)
Sequences 9 ones: 2081 (1512 are labelled with 1)
Sequences 10 ones: 7889 (7889 are labelled with 1)

**Validation Set**

Label 0: 6299, *after resizing*: 5500
Label 1: 5445, *after resizing*: 5454
Sequences 0 ones: 3027 (0 are labelled with 1)
Sequences 1 ones: 445 (214 are labelled with 1)
Sequences 2 ones: 466 (239 are labelled with 1)
Sequences 3 ones: 566 (286 are labelled with 1)
Sequences 4 ones: 584 (291 are labelled with 1)
Sequences 5 ones: 673 (339 are labelled with 1)
Sequences 6 ones: 669 (349 are labelled with 1)
Sequences 7 ones: 746 (418 are labelled with 1)
Sequences 8 ones: 610 (357 are labelled with 1)
Sequences 9 ones: 706 (499 are labelled with 1)
Sequences 10 ones: 2453 (2453 are labelled with 1)

**Testing Set**

Label 0: 8980, *after resizing*: 5700
Label 1: 5700, *after resizing*: 5700
Sequences 0 ones: 3712 (0 are labelled with 1)
Sequences 1 ones: 483 (298 are labelled with 1)
Sequences 2 ones: 495 (304 are labelled with 1)
Sequences 3 ones: 540 (330 are labelled with 1)
Sequences 4 ones: 615 (366 are labelled with 1)
Sequences 5 ones: 651 (382 are labelled with 1)
Sequences 6 ones: 751 (466 are labelled with 1)
Sequences 7 ones: 659 (425 are labelled with 1)
Sequences 8 ones: 639 (437 are labelled with 1)
Sequences 9 ones: 544 (381 are labelled with 1)
Sequences 10 ones: 2311 (2311 are labelled with 1)

**Evaluation**

The following tables represent the scoring from the traning of the network and the validation step. The parameters represent the *learning rate* (that is, how parameters change during the descent of the gradient; the smaller is this changing, the more specific characteristics could be noticed, but there is a risk of generalizing too little the network: it may not be able to recognize other sequences than those used during training. In this case we talk about overfitting. On the other hand if it is too high we can jump between different values too quickly, risking to skip local lows. In this case we talk about underfitting), the number of *epochs* of training (the number of times that the whole training set passes through the network), the *training loss* (or how much the predicted value is far from the real value), the *validation loss* (like the training loss, but the calculation is performed on the validation set) and *validation accuracy* (how much the batch validation passed to the network is accurate, in percentage terms).

Params: *learning_rate*: 1e-6, *epochs*: 2.
Each epochs run for 3h and 12m.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 1 | 31.1602 | 79.7908 | 51.67% |
| 2 | 26.3456 | 66.5545 | 51.93% |

Params: *learning_rate*: 1e-4, *epochs*: 3.
Each epochs run for 3h and 47m.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 1 | 30.9787 | 61.3146 | 47.79% |
| 2 | 10.3879 | 13.3667 | 44.35% |
| 3 | 10.1101 | 12.7287 | 46.92% |

Params: *learning_rate*: 1e-5, *epochs*: 5.
Each epochs run for 3h and 26m.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 1 | 0.6977 | 0.7139 | 49.29% |
| 2 | 0.6853 | 0.7069 | 49.61% |
| 3 | 0.6816 | 0.7065 | 47.98% |
| 4 | 0.6792 | 0.7086 | 50.09% |
| 5 | 0.6775 | 0.7100 | 50.15% |

Table 4.1: Experiments on D-0

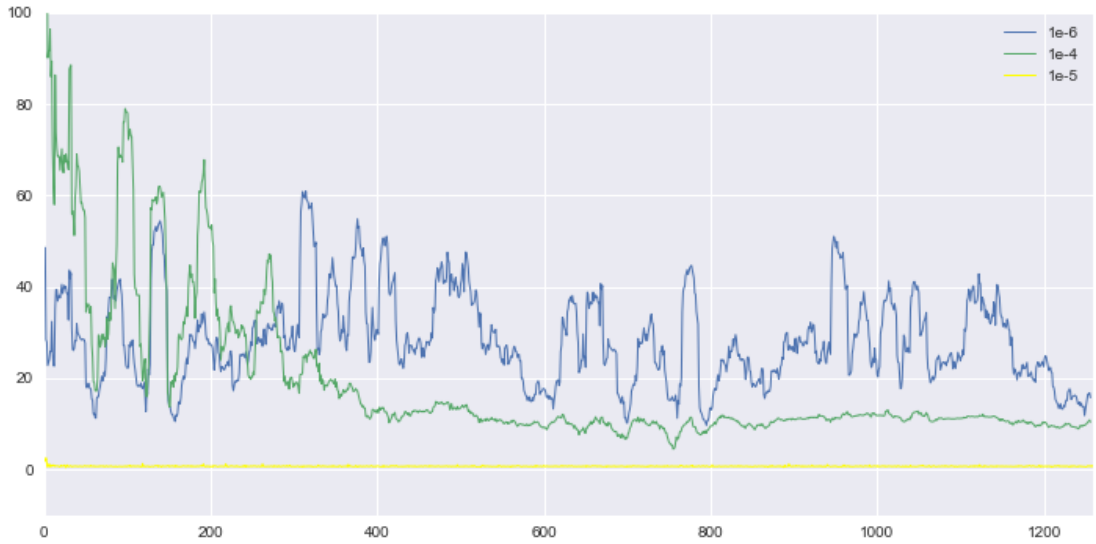


Figure 4.11: D-0 is trained over two epochs with different learing rate

As can be seen from the image and the data in the table, all three experiments seem to decrease towards limit values. It is evident that training with *learning_rate* = 1e-5 decreases to its minimum value faster than the others; this minimum value is also the smallest recorded. The training with *learing_rate* = 1e-6 and *learning_rate* = 1e-4 decreas slowly to a limit that is approximatly near to 10, that is higher than 0.6853 recorder for the training with *learning_rate* = 1e-5. For that reason the experiment is again performed with *learning_rate* = 1e-5 increase the number of epochs to 5 as reported in the previous table.

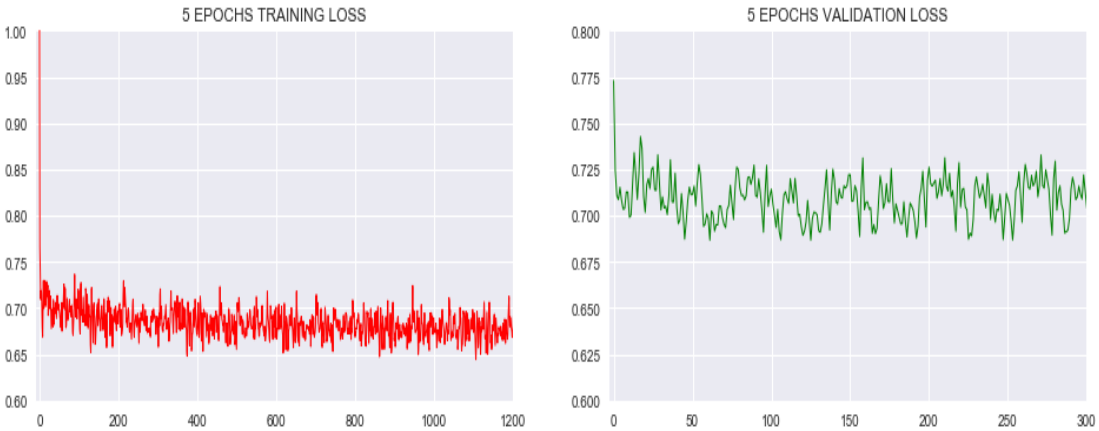


Figure 4.12: D-0 losses training and validation on experiment 3

As can be seen from Figure 4.8, the loss of the training set decreases rapidly until it converges to a value of 0.67. This value seems to diminish very little during all the ages, so it is useless to continue further in the training. A similar argument can be made by observing the loss of the validation-set, which converges rapidly

to a value of 0.70 and in the last epochs oscillate slightly.

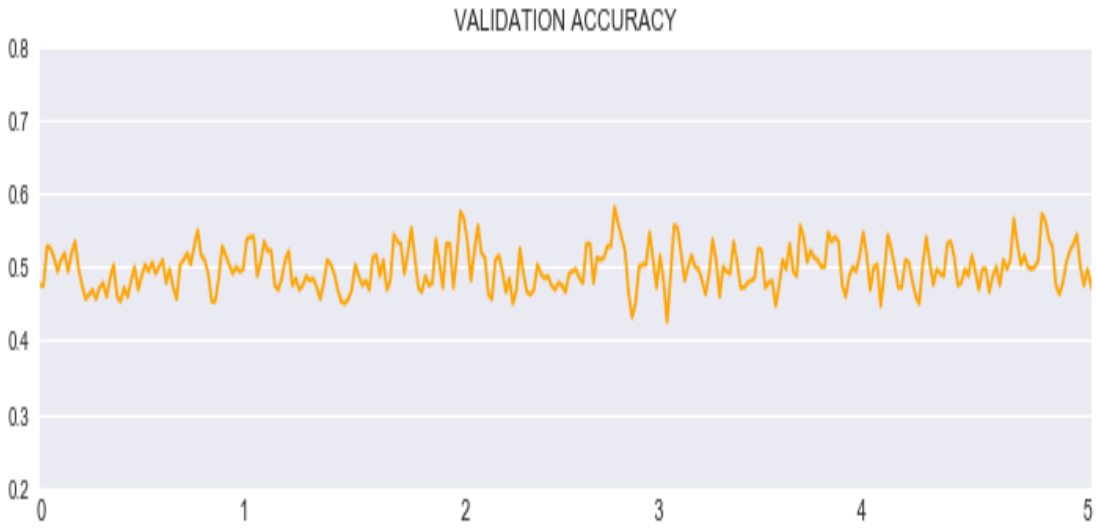


Figure 4.13: D-0 Validation Accuracy over 5 epochs

Figure 4.9 shows us how the accuracy fluctuates around the value of 50% for all training periods (observed in last section of table 4.3 as well). This information combined with the training loss, which states that the network has stopped learning, makes us understand that this type of network is not able to learn information for the D-0 dataset.

### 4.2.2 D-1

#### Training Set

Label 0: 19070, *after resizing*: 1700
Label 1: 1632, *after resizing*: 1632
Sequences 0 ones: 1549 (0 are labelled with 1)
Sequences 1 ones: 119 (105 are labelled with 1)
Sequences 2 ones: 126 (109 are labelled with 1)
Sequences 3 ones: 138 (116 are labelled with 1)
Sequences 4 ones: 155 (131 are labelled with 1)
Sequences 5 ones: 138 (117 are labelled with 1)
Sequences 6 ones: 118 (102 are labelled with 1)
Sequences 7 ones: 99 (84 are labelled with 1)
Sequences 8 ones: 99 (86 are labelled with 1)
Sequences 9 ones: 86 (77 are labelled with 1)
Sequences 10 ones: 705 (705 are labelled with 1)

**Validation Set**

Label 0: 4702, *after resizing*: 500
Label 1: 474, *after resizing*: 474
Sequences 0 ones: 443 (0 are labelled with 1)
Sequences 1 ones: 34 (30 are labelled with 1)
Sequences 2 ones: 43 (31 are labelled with 1)
Sequences 3 ones: 41 (33 are labelled with 1)
Sequences 4 ones: 41 (33 are labelled with 1)
Sequences 5 ones: 38 (34 are labelled with 1)
Sequences 6 ones: 33 (26 are labelled with 1)
Sequences 7 ones: 29 (27 are labelled with 1)
Sequences 8 ones: 31 (25 are labelled with 1)
Sequences 9 ones: 26 (20 are labelled with 1)
Sequences 10 ones: 215 (215 are labelled with 1)

**Testing Set**

Label 0: 6064, *after resizing*: 406
Label 1: 406, *after resizing*: 406
Sequences 0 ones: 392 (0 are labelled with 1)
Sequences 1 ones: 23 (23 are labelled with 1)
Sequences 2 ones: 24 (21 are labelled with 1)
Sequences 3 ones: 18 (16 are labelled with 1)
Sequences 4 ones: 21 (19 are labelled with 1)
Sequences 5 ones: 24 (23 are labelled with 1)
Sequences 6 ones: 23 (20 are labelled with 1)
Sequences 7 ones: 20 (20 are labelled with 1)
Sequences 8 ones: 21 (20 are labelled with 1)
Sequences 9 ones: 23 (21 are labelled with 1)
Sequences 10 ones: 223 (223 are labelled with 1)

**Evaluation**

Params: *learning_rate*: 1e-5, *epochs:* 10.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 1 | 0.7722 | 0.7998 | 49.53% |
| 2 | 0.7134 | 0.7427 | 47.56% |
| 3 | 0.7120 | 0.7413 | 47.10% |
| 4 | 0.7104 | 0.7392 | 47.33% |
| 5 | 0.7090 | 0.7392 | 47.10% |
| 6 | 0.7076 | 0.7386 | 46.57% |
| 7 | 0.7064 | 0.7377 | 46.41% |
| 8 | 0.7052 | 0.7396 | 46.18% |
| 9 | 0.7040 | 0.7401 | 46.18% |
| 10 | 0.7029 | 0.7415 | 46.18% |

Table 4.2: Experiment on D-1

In this experiment is evaluated the net just using a *learning_rate* = 1e-5, due the goal now is not to find the best fitting for the net, but if the net is able or not to predict our datas. As shown in D-0, a *learning_rate* of 1e-5 is a good one to perfom the training. Due the size of D-1, that is small than D-0, the number of epochs is doubled in order to be sure that the train is not stopped early.
As shown in table 4.4 and after in figures 4.10 and 4.11 the training loss seems to converge rapidly after a few eras to the limit value of 0.70, while the validation loss, similarly, converges around 0.74.


Figure 4.14: Net trained on D-1 dataset. Are shown the training and validation losses.

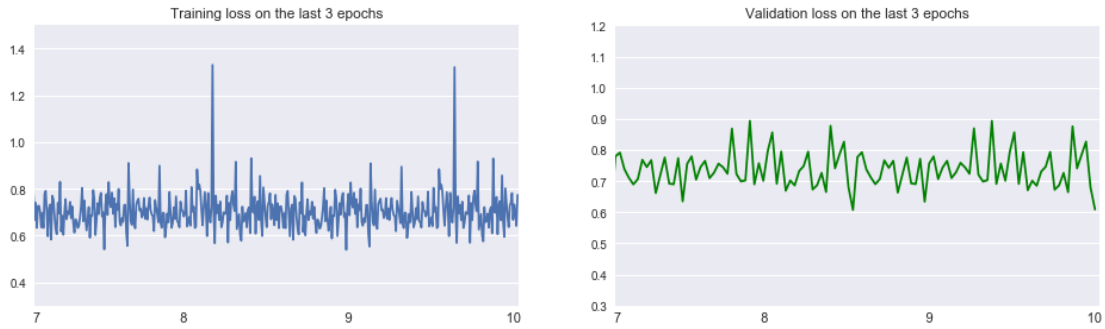


Figure 4.15: Zoom on last 3 epochs on Training the Net with D-1 dataset

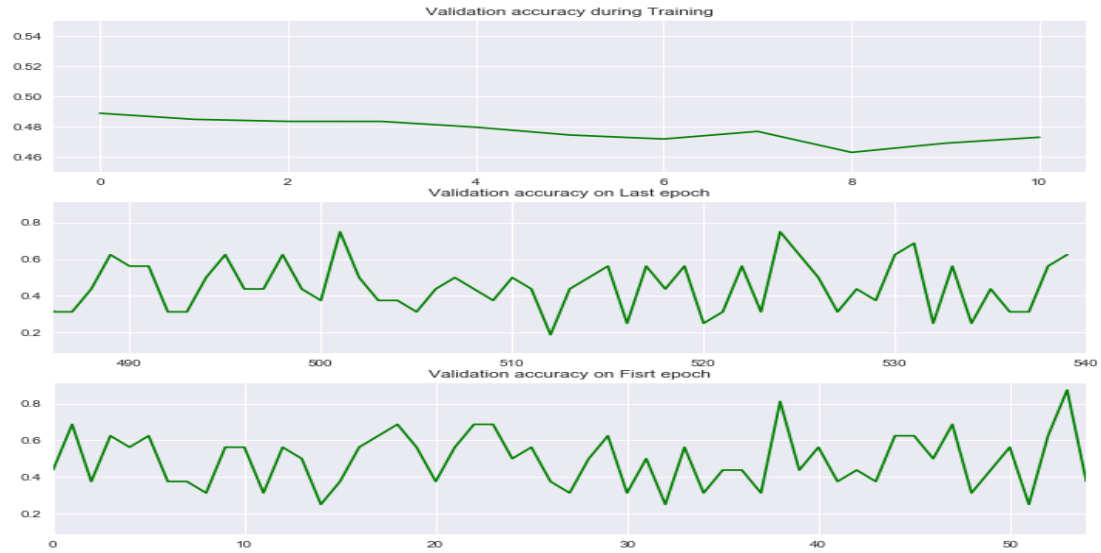


Figure 4.16: Validation Accuracy on D-1. The first figure shows the general trend. The second and third ones show the trand in a specifc epoch.

The figures below show the trend of Validation Accuracy. They are reported first and last epochs that represent best and worst performance. How you can see the accuracy fluctuate around the 50%. It can be said that the model fails to learn even on D-1.

### 4.2.3 D-2

**Training Set**

Label 0: 12759, *after resizing*: 1800
Label 1: 1743, *after resizing*: 1743
Sequences 0 ones: 1459 (0 are labelled with 1)
Sequences 1 ones: 97 (63 are labelled with 1)
Sequences 2 ones: 105 (74 are labelled with 1)
Sequences 3 ones: 108 (70 are labelled with 1)

Sequences 4 ones: 116 (71 are labelled with 1)
Sequences 5 ones: 119 (72 are labelled with 1)
Sequences 6 ones: 135 (90 are labelled with 1)
Sequences 7 ones: 111 (79 are labelled with 1)
Sequences 8 ones: 102 (65 are labelled with 1)
Sequences 9 ones: 116 (84 are labelled with 1)
Sequences 10 ones: 1075 (1075 are labelled with 1)

**Validation Set**

Label 0: 3072, *after resizing*: 560
Label 1: 554, *after resizing*: 554
Sequences 0 ones: 400 (0 are labelled with 1)
Sequences 1 ones: 27 (11 are labelled with 1)
Sequences 2 ones: 29 (13 are labelled with 1)
Sequences 3 ones: 33 (16 are labelled with 1)
Sequences 4 ones: 33 (16 are labelled with 1)
Sequences 5 ones: 43 (18 are labelled with 1)
Sequences 6 ones: 43 (28 are labelled with 1)
Sequences 7 ones: 43 (22 are labelled with 1)
Sequences 8 ones: 35 (19 are labelled with 1)
Sequences 9 ones: 43 (26 are labelled with 1)
Sequences 10 ones: 385 (385 are labelled with 1)

**Testing Set**

Label 0: 4131, *after resizing*: 400
Label 1: 401, *after resizing*: 401
Sequences 0 ones: 330 (0 are labelled with 1)
Sequences 1 ones: 19 (12 are labelled with 1)
Sequences 2 ones: 18 (11 are labelled with 1)
Sequences 3 ones: 15 (7 are labelled with 1)
Sequences 4 ones: 17 (11 are labelled with 1)
Sequences 5 ones: 29 (15 are labelled with 1)
Sequences 6 ones: 22 (13 are labelled with 1)
Sequences 7 ones: 27 (15 are labelled with 1)
Sequences 8 ones: 20 (15 are labelled with 1)
Sequences 9 ones: 20 (18 are labelled with 1)

Sequences 10 ones: 284 (284 are labelled with 1)

### Evaluation

Params: *learning_rate*: 1e-5, *epochs*: 5.

| EPOCH | TRAINING LOSS | VALIDATION LOSS | VALIDATION ACCURACY |
|---|---|---|---|
| 1 | 18.1357 | 19.4116 | 50.81% |
| 2 | 0.8102 | 0.7626 | 55.43% |
| 3 | 0.7267 | 0.7000 | 54.89% |
| 4 | 0.7187 | 0.6980 | 51.63% |
| 5 | 0.7117 | 0.6987 | 48.36% |

Table 4.3: Experiment on D-2

[Aggiungere i valori dei TP, TN, FP e FN per spiegare il perchè quei valori come 54, 55% non possono essere presi in considerazione: far vedere che per brevi alcuni periodi la rete predice sempre lo stesso risultato come ad esempio 0 e che quando arrivano in sequenza sequenze che non producono eventi spesso si hanno valori come 100% di accuratezza che spostano verso l'alto l'accuratezza generale dell'intervallo e di una determinata epoca, e che quindi non possono essere considerati come veritieri]

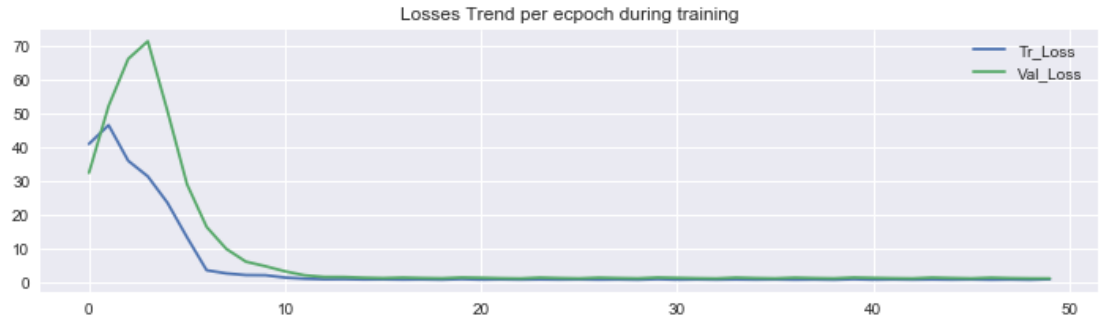


Figure 4.17: Net trained on D-2 dataset. Are shown the training and validation losses.

As shown in table 4.5 and in figure 4.13 both losses tend to fall quickly. Already in the middle of the second epoch we are almost close to the convergence value, which is around 0.72 for the training-set and around 0.69 for the validation-set.

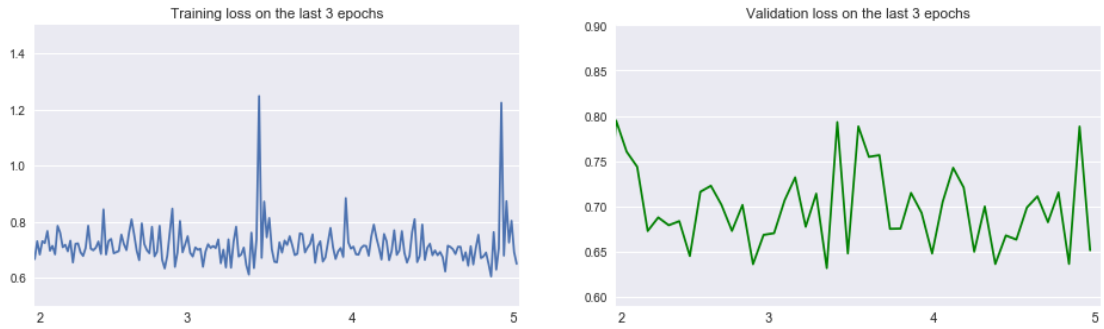


Figure 4.18: Zoom on last 3 epochs on Training the Net with D-2 dataset

In figure 4.14 we see how, especially for the validation loss, in the last two epochs the value stops around 0.698, moreover while this value is blocked, in training it goes down, even if very little. We therefore stop and observe the accuracy value shown in figure 4.15 and in table 4.5.


Figure 4.19: Validation Accuracy on D-2. The first figure shows the general trend. The second and the third ones the accuracy over first and last epoch

Figure 4.15 shows how the accuracy value still fluctuates around 50%. We can therefore state that the network randomly predicts the class to which the sequences belong.
With these experiments it was not possible to classify the classes of belonging of the sequences in a fairly correct way, independently of their quality. You might think that the space-time approach is not suitable for this type of data or simply that the network can be right, but not able to learn from this kind of data.

## 4.3 3-Dimensional Neural Network Model

In order to understand if a space-time approach can work on this type of data, a new network model has been defined to try to classify the sequences. This model is no longer structured on the division of tasks by the sub-networks within the network. The new structure tries to use the convolution using three dimensions, trying to mix the spatial approach with the temporal one.

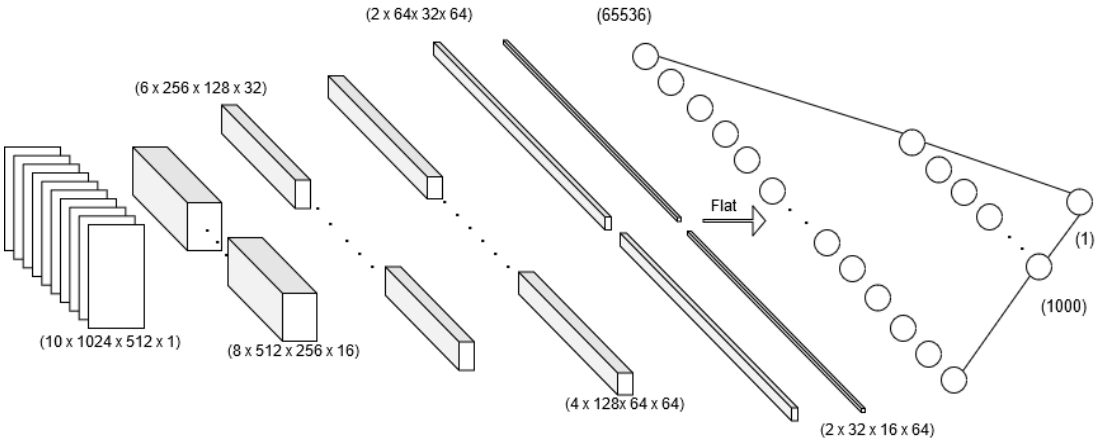


Figure 4.20: 3-D network

Staring from a sequences of [Batch_size, 10 , 1024, 512, 1] and pass throug a 3-D convolution with kernel dimensions: [3, 3, 3, 1, 16] and strides: [1, 1, 2, 2, 1] you get the **first hidden layer**. The **second hidden layer** is compute with [3, 3, 3, 16, 32] as kernel size and [1, 1, 2, 2, 1] as strides. The **third hidden layer** is compute with [3, 3, 3, 32, 64] as kernel size and [1, 1, 2, 2, 1] as strides. The **fourth hidden layer** is compute with [3, 3, 3, 64, 64] as kernel size and [1, 1, 2, 2, 1] as strides; this last one hidden layer is used to reduce time dimensionality over the same number of features. For subsampling this last layer is used a **max_pooling** strategy in order to reduce computationally effort for the fully connected node. After flat this layer all neurons are connected with the **first fully connected node** followed by a **second fully connected node** in order to get one neuros used for the classification.

**Summary**

$$Input : [Batch\_size, 10, 1024, 512, 1]$$

$$Hidden\_Layer_1 : [Batch\_size, 8, 512, 256, 16]$$

$$Hidden\_Layer_2 : [Batch\_size, 6, 256, 128, 32]$$

$$Hidden\_Layer_3 : [Batch\_size, 4, 128, 64, 64]$$

$$Hidden\_Layer_4 : [Batch\_size, 2, 64, 32, 64]$$

$$Max\_Pooling\_Layer : [Batch\_size, 2, 32, 16, 64]$$

$$Fully\_Connected\_Layer_1 : [Batch\_size, 65536]$$

$$Fully\_Connected\_Layer_2 : [Batch\_size, 1000]$$

$$Outputs, Logits : [Batch\_size, 1]$$

### 4.3.1 D-2

In order to perform this network, useful to show if a 3D approach can be done, the D2 dataset (described in the paragraph 4.2.3) has been used: in this way you can immediately understand if these types of experiments are valid or not.

#### Evaluation

To evaluate this new network you started from information obtained from the previous network. In fact, although the previous network was not able to give you adequate accuracy, it was useful in giving you some indications concerning some parameters such as the learning that you decided to set with a starting value of 1e-6 and the number of epochs that you have seen it necessary to make it grow in particular when the datasets are small (like the D2 that is used in this training).

Params: *learning_rate*: 1e-6, *epochs*: 30.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 1 | 16.7932 | 19.8135 | 51.74% |
| 2 | 4.2734 | 8.0294 | 52.50% |
| 3 | 1.5243 | 5.2119 | 51.50% |
| 5 | 0.7040 | 4.2173 | 54.75% |
| 10 | 0.2572 | 3.9651 | 56.71% |
| 15 | 0.2154 | 4.4007 | 56.71% |
| 20 | 0.0429 | 5.3073 | 59.61% |
| 21 | 0.0374 | 5.3575 | 60.30% |
| 22 | 0.0758 | 5.4979 | 60.42% |
| 28 | 0.0974 | 4.6807 | 62.04% |
| 29 | 0.0794 | 4.6807 | 60.4% |
| 30 | 0.3685 | 5.0527 | 54.15% |

Table 4.4: 3-D experiment on D-2. For ease of visualization only some periods are reported.

Table 4.4 shows the first experiment conducted on this network. This first training was launched for 30 epochs with a learning rate of 1e-6. As you can easily

see from the table the training has a positive trend concerns our purpose: although the validation loss has an oscillatory trend, the validation accuracy grows over time in parallel with the decrease of the training loss. With this first training it was observed that the network is able to learn something, for this reason by modifying the parameters it was decided to train the network for another 20 epochs in order to see if it can actually achieve better results.

Params: *learning_rate*: 1e-6, *epochs*: 20.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 41 | 0.00519198 | 4.148036 | 65.39% |
| 42 | 0.00487719 | 4.139599 | 65.16% |
| 43 | 0.00460525 | 4.151326 | 65.05% |
| 44 | 0.00434201 | 4.155872 | 64.58% |
| 45 | 0.00451232 | 4.280730 | 65.51% |
| 46 | 0.00367322 | 4.181511 | 65.62% |
| 47 | 0.00343623 | 4.186375 | 65.16% |
| 48 | 0.00401666 | 4.177502 | 66.20% |
| 49 | 0.00279858 | 4.198667 | 66.55% |
| 50 | 0.00258480 | 4.198675 | 66.32% |

Table 4.5: This table shows the last 10 periods of the second training. As you can see the results are positive with the passing of the epochs.

As you can see in table 4.5 the training loss continues to decrease, while the validation loss fluctuates between 4.1 and 4.2. In this situation also the accuracy has grown: in fact you have gone from 60% of the first training to a 66% in this last training. At this point it was decided to try yet another training by modifying the learning rate, decreasing it to 1e-7, to try and see if it is possible to find new local minimums. Since you could in the subsequent training incur in overfitting the last epoch of this part of training is saved (as well as the number 30 in the previous training).

Params: *learning_rate*: 1e-7, *epochs*: 20.

| EPOCH | TRAINING_LOSS | VALIDATION_LOSS | VALIDATION_ACCURACY |
|---|---|---|---|
| 66 | 0.00104314 | 4.720064 | 65.62% |
| 67 | 0.0010270 | 4.724249 | 65.62% |
| 68 | 0.0010010 | 4.726898 | 65.51% |
| 69 | 0.0009806 | 4.729340 | 65.51% |
| 70 | 0.0009605 | 4.732010 | 65.51% |

Table 4.6: This table figure out the last five epochs of the last training. How is easly visible the training loss slowly decreases while the validation loss increases. It is explained by overfitting. Due this the training is finally stopped.

In these following pages the graphs obtained from these trainings will be ana-

lyzed in order to understand the overall network trend and then move on to the testing set.


Figure 4.21: Losses in 3-D network over D2 dataset. his graph represents the overall trend of the network in the three trainings. We can observe, in green, the validation loss which tends to decrease and then slowly go up again in the last epochs and the training loss, in blue, which constantly decreases.

As shown in figure 4.21 both the validation loss and the training loss decrease rapidly, breaking the value of 5 in the very epochs. As is usually for neural networks, the value of training loss is lower than validation one.

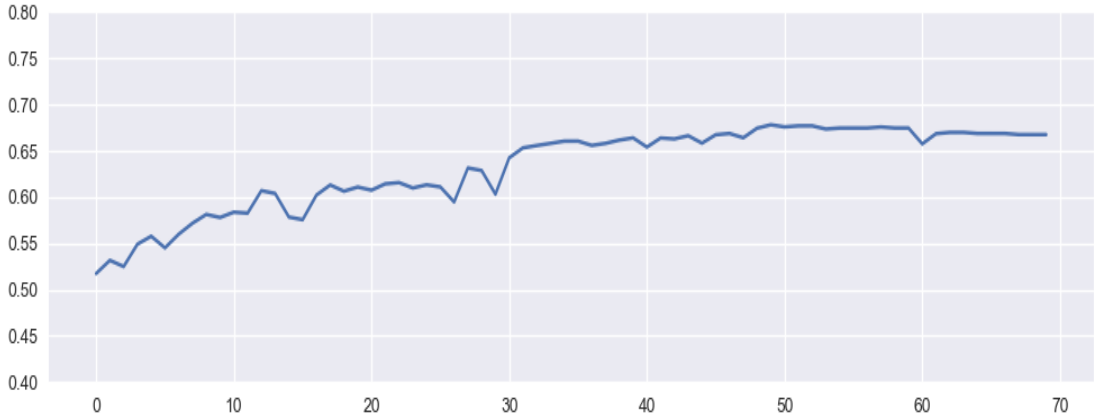


Figure 4.22: Validation Accuracy in 3-D network over D2 dataset. This image represents the overall trend of accuracy on the validation set during all 70 training periods. It can be seen how the score obtained during the epochs is increasing, with the exception of the last part of the training where, due to some overfitting, it tends to decrease.

The trend of training loss can be approximated to a descending function, while the trend of the validation loss is fluctuating: in fact between the twelfth and the twenty-third epoch it grows and then falls towards the end of the first training

(remember that the first training lasted 30 epochs ). After a downward peak around the thirtieth epoch the value of the validation loss seems to fade, and slightly rise, around a convergence value for the entire second training (between the epoch 30 and epoch 50). This value then grows constantly in the last training (between epoch 50 and epoch 70): this behavior is most likely due to an overfitting situation (analyzing the performance of the training loss it can be seen that it continues to fall, look at the image 4.23). In parallel to the study of the losses one you must analyze the trend of the accuracy, as shown in figure 4.21. As can be seen, the accuracy trend is increasing, except for the last part of the network and some downward peaks that can be observed in different epochs. In particular it is evident that the network quickly learns in the first epochs passing from about 50% accuracy of the first epoch to around 65% of the epoch 35 (in the middle of the total training). Around the time 50 the highest peak of correct prediction is reached and subsequently until the end of the training accuracy decreases constantly giving value to the idea of overfitting (in fact in figure 4.20 we noticed how the validation loss grew while the training loss continued to fall). Summary it can be said that this type of network in 3 dimensions is able to learn a fair amount of sequences by recognizing which ones produce an event and which ones not. In the next section you will try to rerun the validation set on the network loaded with the parameters of different epochs to try to refine the accuracy achieved.

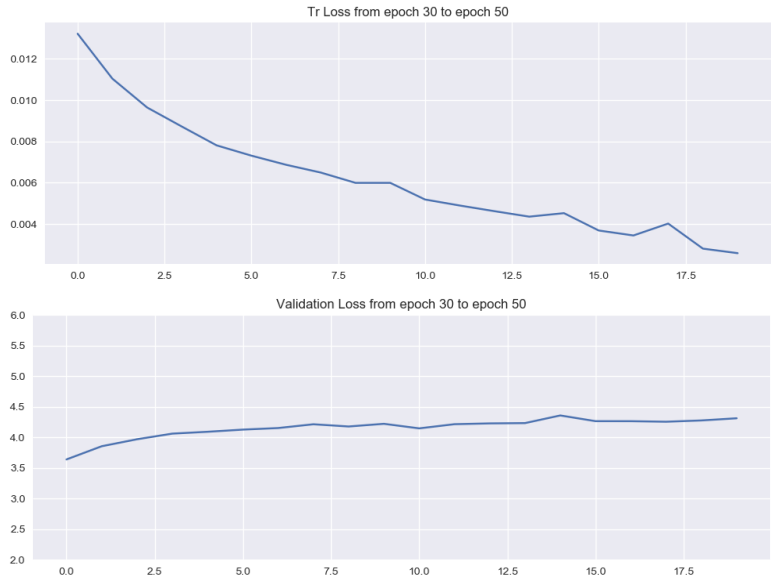


Figure 4.23: Training Loss and Validation Loss in 3-D network over D2 dataset from 30 to 50 epochs.In the first image you can see how the training loss constantly decreases, while in the second image, even if only slightly, the validation loss grows.

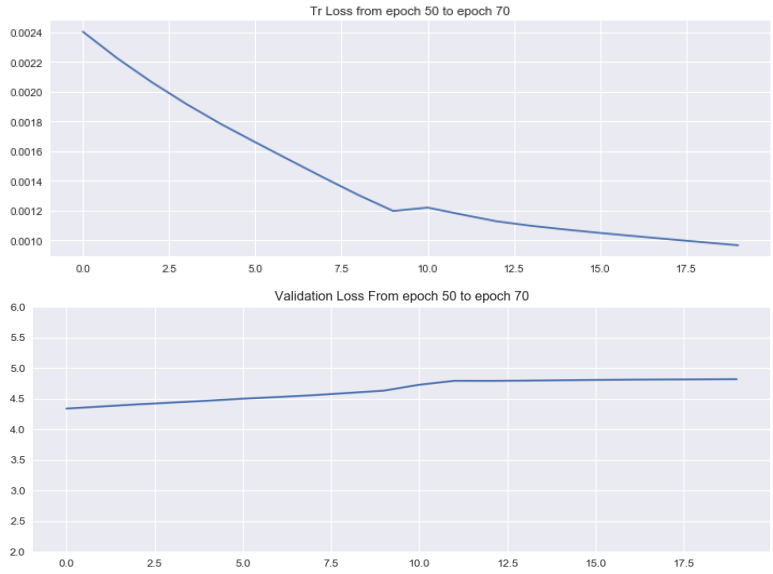


Figure 4.24: Training Loss and Validation Loss in 3-D network over D2 dataset from 50 to 70 epochs.In the first image you can see how the training loss constantly decreases, while in the second image, even if only slightly, the validation loss grows.

### Validation & Testing

Once the training has been completed analyzing the predictions made on the validation set, it was noticed that in most cases the threshold value for which a sequence was classified as "producing a CME" or as "not producing a CME" is variable and is not the value 0.5. It must be remembered that the value 0.5 is the threshold for which the activation function, used in the network to produce the output, that is the sigmoid, uses to give as output a 1 or a 0 (the labels for "producing a CME" and "not producing a CME"). To find the best threshold value that best fits the data in our possession, the validation set calculation function was relaunched using a mobile threshold: we started from 0.1 to 0.9 with step 0.1 in order to find what value produce the highest accuracy.

| EPOCH | BEST VAL FITTING | VALIDATION ACC |
|---|---|---|
| 30 | 0.6 | 65.71% |
| 50 | 0.8 | 68.58% |
| 70 | 0.8 | 67.06% |

Table 4.7: Accuracy validation for Mobile Threshold on 3D network.For epochs 30, 50 and 70 the accuracy is calculated again using a mobile threshold. For each of these epochs the value of the threshold for which the best result and the result itself is obtained is reproduced.

Table 4.7 shows the values for the validation accuracy calculated on the network parameterized with the variables and weights of epochs 30, 50 and 70. For each of

the afore mentioned epochs the threshold values that produce the highest accuracy are calculated. It can be noted, as similarly happened for the values during the training, that the parameters that give the highest result are those of the epoch 50, which using the value 0.8 produce an accuracy of 68.58%, that is more than two percentage points compared to the value obtained with the same set used during training. Once this information is obtained, you can choose the epoch 50 with a threshold value of 0.8 as epoch to test the testing set.

| EPOCH | TRESHOLD | TESTING ACCURACY |
|---|---|---|
| 30 | 0.6 | 68.04% |
| 50 | 0.8 | 72.17% |
| 70 | 0.8 | 71.07% |

Table 4.8: Test on the 3D network.

Table 4.8 shows the results obtained on all three eras. As previously stated, the value to be taken into consideration is epoch 50 which allows us to obtain a final result of 72.17% accuracy on the testing set.

### 4.3.2 D-3

After this result a new Dataset is defined:

- D3 contains, in addition to not cme events, all events quality within the sequences, but the event that appears in position 10 has quality q4 or q5:

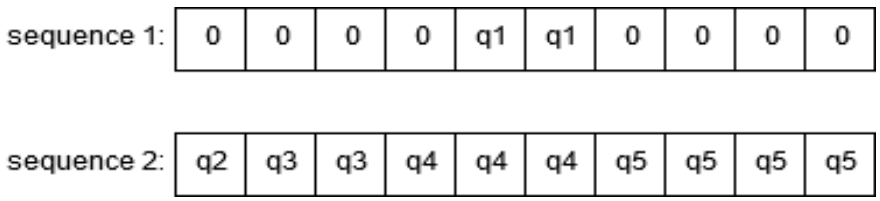


Figure 4.25: Example of sequences in D3. Sequence 1 does not produce a CME: in fact the image in position 10 is labelled with 0. Sequence 2 produces a CME: in fact image in position 10 is a q5 quality event marked as 1. Sequences containing both cme and not cme are extremely rare: in fact in this dataset the general behavior is all zero or all one within a sequence.

Training Set) Label 0: 700 sequences, Label 1: 689 sequences.
Validation Set) Label 0: 243 sequences, Label 1: 243 sequences.
Testing Set) Label 0: 140 sequences, Label 1: 140 sequences.

### Evaluation

The following images show the losses and the accuracy trend:

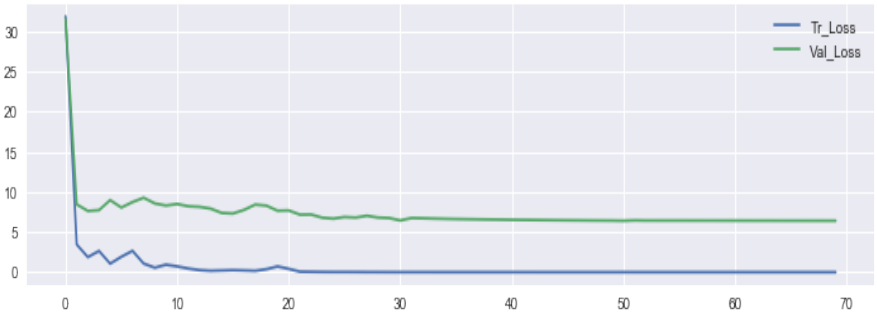


Figure 4.26: Losses in D3. The figure shows the trend of losses. As you can easily see, both losses rapidly decrease.

The training is performed for 50 epochs with a learning rate of 1e-6 and then for other 20 epochs with learning rate of 1e-7.

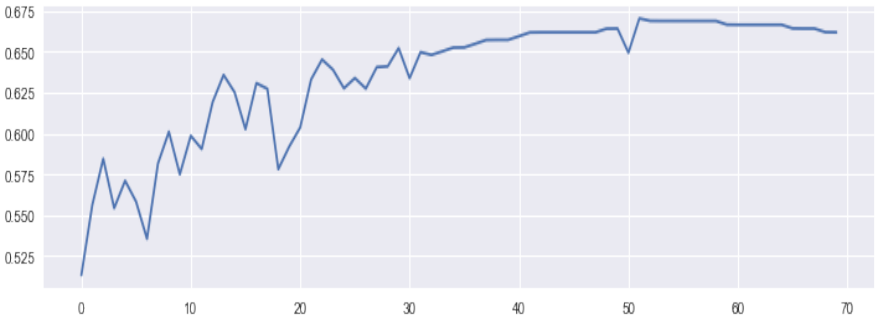


Figure 4.27: Accuracy in D3. As you can see, the accuracy has a growing trend up to around the epoch 50. After this time the accuracy decreases slightly, due to a small overfitting due to a too low larning rate.

### Validation & Testing

As done for the D2 dataset the best threshold was calculated.

| EPOCH | BEST VAL FITTING | VALIDATION ACC |
|---|---|---|
| 30 | 0.7 | 62.60% |
| 50 | 0.8 | 63.58% |
| 70 | 0.6 | 62.96% |

Table 4.9: D3 validation threshold. The same operation done for D2 experiment. For each epoch is computed the threshold that perform the best accuracy using validation set.

Also in this case, the epoch 50 allows us to achieve the best accuracy. Proceeding with the testing set and using the 50 age as a session and a threshold

equal to 0.8, an accuracy of 72.14% is reached. It is not an improvement from the previously dataset. This could be explained by two reasons: the high size reduction occurred in D3 sets and the similarity between q3 (present in D2) and q4/q5 quality events; in the last case the quality q3 is closer to q4 and q5 than to a not-event.

# Chapter 5

# Conclusion

## 5.1 Conclusion

As seen in the previous chapter, the first network, the one composed of a part of features extractor and one of convolution over time, did not produce significant results in any of the datasets created: we concluded that the network was not able to learn from these data and in fact the classification was random. This result does not preclude the possible goodness and functionality of this type of network; the results show only that the data in our possession are not suitable for this network. Instead a space-time approach with a network that uses 3 dimensional convolution is possible: in this case, although the experiments have been carried out with reduced datasets and of a quality higher than the average of the total data, it has been possible to reach an accuracy of 72.17%, as seen with the D2 dataset. With this experiment about three-quarters of the sequences tested at the end of the process were correctly classified.
Another important conclusion concerns the quality and quantity of data. As seen in the chapter about data pre-processing and in the chapter in which training, validation and testing sets were defined, most of the starting data were discarded due to time filters, intrinsic errors in the images and for the balance requirement of the various datasets. As can be seen from the various analyzes and results obtained from the 3-dimensional convolution network, the quality and quantity of images has increased over time, indicating a photos management and retrivial from the coronograph that has improved over time. This last indication can make us think that it is possible in the future to retest these networks in order to try to achieve higher accuracy. In conclusion it is therefore possible to use the realized network that performs the best accuracy to help those who currently manage the classification, in this way they can have a technological tool able to give feedback to their work. It is also possible in the future, when data will be better, retrain

the network and try to predict whether a sequence will produce a CME or not trying to anticipate solar events becoming able to take precautions before a CME event occurs.

# Bibliography


[1] Images taken from https://cdaw.gsfc.nasa.gov/CME_list/

[2] https://www.swpc.noaa.gov/phenomena/coronal-mass-ejections
"Coronal Mass Ejections (CMEs) are large expulsions of plasma and magnetic field from the Sun's corona. They can eject billions of tons of coronal material and carry an embedded magnetic field (frozen in flux) that is stronger than the background solar wind interplanetary magnetic field (IMF) strength."

[3] https://en.wikipedia.org/wiki/Coronal_mass_ejection

[4] https://www.spaceweatherlive.com/en/help/
what-is-a-coronal-mass-ejection-cme

[5] https://aviationweek.com/awin/major-solar-
event-could-devastate-power-grid "Recent calculations suggest there is a 6-12% chance of another storm at that level in any given year."

[6] https://aviationweek.com/awin/
major-solar-event-could-devastate-power-grid
"That storm took down parts of the growing U.S. telegraph network, starting fires in the process and subjecting some telegraph operators to electric shock."

[7] https://books.google.it/books?id=73IRAAAAYAAJ&pg=PA732&dq=1859+
aurora+boreale&lr=#v=onepage&q=1859%20aurora%20boreale&f=false
Aurora boreale in Roma, in La Civiltà Cattolica, Anno decimo, vol. III della serie quarta, Roma, 1859, pp. pp. 732-733.

[8] https://sohowww.nascom.nasa.gov/about/instruments.html

[9] Bishop, Christopher (2006). Pattern recognition and machine learning. Berlin: Springer. ISBN 0-387-31073-8.

[10] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller (2018). Deep learning for time series classification: a review. arXiv:1809.04356v4

[11] https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d
"Convolutional Neural Networks (CNNs) is the most popular neural network model being used for image classification problem."

[12] https://medium.com/datadriveninvestor/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8
"CNN's are really effective for image classification as the concept of dimensionality reduction suits the huge number of parameters in an image."

[13] https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks
"Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned."

[14] https://fits.gsfc.nasa.gov/ See documentation.

[15] https://computersciencewiki.org/index.php/Max-pooling_/_Pooling (2018).

[16] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, Serge Belongie; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2921-2930

[17] Bora Erden, Noah Gamboa, Sam Wood; Stanford University; 3D Convolutional Neural Networkfor Brain Tumor Segmentation, 2017

[18] Rui Hou,Chen Chen and Mubarak Shah; An End-to-end 3D Convolutional Neural Networkfor Action Detection and Segmentation in Videos, 2015

[19] Lin Sun, Kui Jia, Dit-Yan Yeung, Bertram E. Shi; Human Action Recognition Using Factorized Spatio-Temporal Convolutional Networks; The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4597-4605

[20] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, Manohar Paluri; Dartmouth College; A Closer Look at Spatiotemporal Convolutions for Action Recognition, (2018)

[21] The function at the center of the figure is in By Qef (talk) - Created from scratch with gnuplot, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4310325