Auroral detection in coloured all-sky images

Not confidential

Tachet Alexia

Student at ENSTA Bretagne

FISE 2022

Specialised in Observation systems and artificial intelligence (SOIA)

Project for

the University Centre in Svalbard (UNIS)

Norway

ENSTA Bretagne supervisor : M. Bonnafont

UNIS supervisor : Ms. Partamies

March 14, 2022 - August 19, 2022

Acknowledgements

Firstly, I want to thank Mr.Toumi who helped me to find some ideas in the beginning of the project.

I want to thank Mr.Bonnafont for his supervision and the answers he brought me.

I also want to thank Mr.Syrjäsuo for his help when I had to make some technical choices. I thank Ms.Partamies for her support and her supervision.

Eventually, I thank Ms.Partamies and Mr.Syrjäsuo for the visit of the Svalbard observatory.

Abstract

English

During this project, I had to automate the detection of auroras on all-sky coloured images. I had images captured with a camera Sony a7s and coming from the Kjell Henriksen Observatory located near Longyearbyen, in Svalbard (Norway). It is more important not to miss an aurora than to avoid detecting false auroras. Therefore, I tried to improve the accuracy and the recall rather than the precision. I prepared the training, validation and testing data sets. Then I used classification methods without Convolutional Neural Networks (CNN). I selected an efficient dimension reduction algorithm and a classifier. For the dimension reduction, the Uniform Manifold Approximation and Projection (UMAP) does not require to optimize lots of parameters. The chosen classifier is a Random Forest. It also gives satisfying results without optimizing parameters. The best accuracy obtained on the validation data set was around 80%, and the best recall was around 75%. These results have been reached thanks to the Orthogonal Local Binary Pattern Combination (OLBPC) applied on local areas of the images with the RGB, HSV and Lab color systems. The values distribution of each channel on RGB and Lab channels also enabled to reach similar results. However, the method based on Scale Invariant Feature Transform (SIFT) was not efficient at all. The methods based on fine tuning of pretrained models offer especially good results even on the testing data set. By taking into account the accuracy and the recall, the best models using the "ResNet50" are those associated with classifiers composed of two layers of 150 neurons, a dropout rate of 0.3 and a regularization "L2" (with a regularization rate equal to 0.1 or 0.5). It enables to achieve a recall of 86%, a precision of 94% and an accuracy of 89% on the testing data. The best model using "InceptionV3" is the one associated with a classifier composed of two layers of 150 neurons, a dropout rate of 0.7 and a regularization "L2" with a rate equal to 0.5. It enabled to reach an accuracy of 88%, a recall of 87% and a precision of 92% on the testing data. Unfortunately, the recall was always worse than the precision. At the end, we notice that the misclassified images contain clouds, moonlight or faint auroras. Eventually, I tried to split all the classes into homogeneous classes but it was problematic. It did not enable to reach better results. Indeed, the same label has been attributed to all images.

Français

Durant ce projet, j'ai automatisé la détection des aurores sur des images plein ciel et en couleurs. Il était plus important de ne manquer aucune aurore plutôt que d'éviter les fausses alarmes. J'ai donc cherché à améliorer l'exactitude (Accuracy) et le rappel (Recall) plutôt que la précision (Precision). Je disposais pour cela de photos capturées à l'aide d'un appareil photo Sony a7s à l'observatoire Kjell Henriksen, à Longyearbyen, au Svalbard (Norvège). Dans un premier temps, j'ai préparé les données d'entrainement, de validation et de test. Ensuite, j'ai utilisé des méthodes non basées sur des réseaux de neurones convolutifs (CNN). J'ai choisi un algorithme de réduction de dimension théoriquement efficace et ne nécessitant pas d'optimiser de nombreux paramètres, le "Uniform Manifold Approximation and Projection (UMAP)". La classification, elle, se fait par une forêt aléatoire (Random Forest). Cet algorithme ne nécessite pas non plus d'optimiser des paramètres. La meilleure exactitude sur les données de validation était autour de 80%, et le meilleur rappel était autour de 75%. Ces résultats ont été atteints grâce à l'algorithme d'extraction de caractéristiques « Orthogonal Local Binary Pattern Combination (OLBPC) » appliqués sur les différentes régions de l'image avec les canaux de couleurs RGB, HSV et Lab. Un autre extracteur de caractéristiques, basé sur les distributions de valeurs de chaque canal du système de couleur RGB et HSV, a permis d'atteindre des résultats similaires. Néanmoins, les méthodes basées sur la méthode d'extraction « Scale Invariant Feature Transform (SIFT) » n'étaient pas efficaces du tout. Le réentrainement (fine tuning) de modèles de CNN pré-entrainés permet d'atteindre de bien meilleurs résultats y compris sur les données de test. Si on prend en compte l'exactitude et le rappel, les meilleurs modèles utilisant "ResNet50" sont ceux associés à un classifieur constitué de deux couches de 150 neurones, un taux de decrochage (Dropout rate) de 0.3 et une régularisation "L2" (avec un taux de régularisation égal à 0.1 ou 0.5). Ces modèles permettent d'atteindre un rappel de 86%, une exactitude de 94% et une précision de 89% sur les données de test. Le meilleur modèle utilisant "InceptionV3" est celui associé à un classifieur constitué de deux couches de 150 neurones, un taux de decrochage de 0.7 et une régularisation "L2" avec un taux égal à 0.5. Ce modèle permet d'atteindre un rappel de 87%, une exactitude de 88% et une précision de 92% sur les données de test. A la fin, on note que les images mal classifiées contiennent des nuages, de la lumière de la lune ou des aurores à peine visibles. Finalement, j'ai essayé de subdiviser chaque classe en sous classes homogènes pour améliorer la classification. Malheureusement, cette étape amenant à de nombreuses problématiques, les tentatives de classification qui s'en sont suivies a donné de très mauvais résultats. En effet, la même étiquette a été attribuée à toutes les images.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This internship is entitled "Auroral detection in coloured all-sky images". It takes place in the University Centre in Svalbard (UNIS), Norway, and it is not confidential. My supervisor at UNIS is Ms. Partamies, professor in middle atmospheric physics.

This project is a continuation of experiments to automate the classification of all-sky images in two classes: "Aurora" or "No aurora". This experiment was especially developed on grey scale images with image processing and classification methods. The main idea is to improve the existing results thanks to coloured images and Deep-Learning methods.

This project could help to make an alarm when there is an aurora and also help to better understand the auroral activity. Indeed, the models of auroral activity tend to be improved throughout the years. The aim is to miss the least number of auroras. Therefore, we will prefer a high rate of false alarms than a high rate of false negative.

The aim is to improve the automatic detection of auroras. The idea is to classify all images in two different families: "Presence of aurora" and "Absence of aurora".

A camera Sony a7s installed in the Kjell Henriksen observatory, close to Longyearbyen ($78.15°N$ $16.04°E$.), allows to take colour all-sky images at regular time intervals. The technical details of this camera will be further described.

In total, more than one million images have been taken for three winters (from 2019 to 2022). Among the images that have been taken during the winter 2019 – 2020 and in January and February 2019, 37 384 images have been labelled in two classes by Ms. Partamies: "Aurora" and "no aurora". Thus, the detection of auroras appears as a supervised learning for a binary classification. There is a time resolution of six minutes between each image.

To work on this project, I have access to a computer at UNIS containing the described dataset. I work with Pycharm and python libraries (like "OpenCV", "ScickitLearn", "Keras", "Pandas" and "Tensorflow").

I will first present the university, the Kjell Henriksen Observatory and the camera Sony a7s. Next, I will deal with the explanation on the auroral activity. Then, I will develop a state of the art of the different existing method to achieve the classification. Then, on the basis of the state of the art, I will develop a strategy to implement and test the methods. Eventually, I will conclude about the obtained results.

Chapter 2

The University Centre in Svalbard (UNIS) and the Kjell Henriksen Observatory

2.1 The University Centre in Svalbard (UNIS)

The University Centre in Svalbard (UNIS) has been created in 1993 in Longyearbyen on the Spitzberg island in the Svalbard archipelago, in Norway. It is located at 78 degrees of latitude north and is the northernmost university. Longyearbyen is plunged in the polar night for four months each year. The director is Jøran Moen. The university depends on the universities of Tromsø, Bergen, Oslo and Trondheim. The disciplines taught are arctic geology, arctic geophysics, arctic biology and arctic technologies.

2.2 The Kjell Henriksen Observatory

The Kjell Henriksen Observatory (Figure 2.1) is located near Longyearbyen. It has been established in 1978 by the professor Kjell Henriksen [7]. There are around thirty radio and optical instruments belonging to universities or research centres worldwide like UNIS, the University of Oslo, the College of London, the National Institute of Polar Research in Japan, the Kyoto university, the Italian Space Agency, the Polar Institute of China, and lots of other ones. All these instruments are used for the research on middle and upper atmosphere.

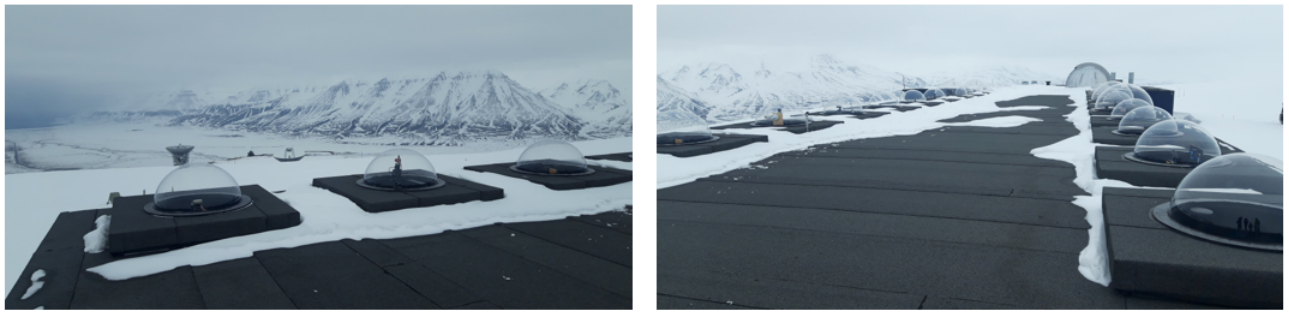

Figure 2.1: The roof of the Kjell Henriksen Observatory : Optical instruments are installed under the transparent domes

Chapter 3
Camera and data description

The camera Sony a7s allows to take colour all-sky images at regular time intervals. The following information are instrument specifications :

Time resolution: $10 - 30$ seconds

Exposure time: 4 seconds at night

Camera: Sony a7s

Lens: Sigma $8mm$ $f/3.5$ EX DG Circular Fisheye, full 180° FOV

Spatial resolution: $12M$ pixels (8.4 um $\times$ 8.4 um pixels)

Spectral resolution: RGB

The obtained images are as follows (Figure 3.1) :

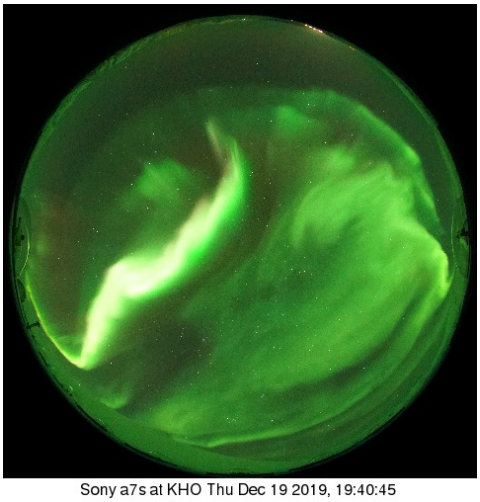

Figure 3.1: All-sky image of an aurora

In the beginning of this project, I have access to 37 384 images from the winter 2019-2020 and January and February 2019 manually labeled by Ms. Partamies in two classes: "Aurora" and "no aurora". These images have also been labeled in two other classes: "Cloudy" and "Cloudless". There is a time resolution of six minutes between each image.

At the end, I have access to 14 775 additional labeled images from January 2022.

Chapter 4
Auroral activity

I will describe here the auroral activity [28].

The sun emits particles like protons, electrons and ions. There are three types of solar emissions : the solar wind, the Corona Mass Ejection (CME) and the solar flares.

The solar wind consists in continuous charged particle stream with varying speed and low energy. These streams come from coronal holes which are areas with low density and low magnetic field. The particles are ejected along opened magnetic fields lines. The origin of the streams are in the polar regions when there is a low-level of solar activity. When the solar activity is intensifying, the streams can come from lower latitudes. Then, the emitted streams are heading towards lower density areas and form a cone. The slow solar stream has a speed of 300 kilometers per second. The high speed solar stream has a speed between 600 and 900 kilometers per second. The streams are not continuous in the space and form thin strands. Because the sun is rotating, the streams of the solar wind describe a spiral. The high speed wind happens often when the solar activity is minimal.

The Corona Mass Ejections (CME) are caused by the solar magnetic fields variations. Fragments of the sun's surface are ejected and take the form of solar plasma clouds. Numerous particles are ejected with a maximum speed of 1600 kilometers per second. The CME with the highest speed causes a shock wave in the solar wind, which is accelerated by this phenomenon. Then, these shock waves are responsible for the impact of particles on the earth. The CMEs happen often when the solar activity is maximal.

The solar flares are due to localized areas which are magnetically unstable and induce charged particles accelerations by modifying the magnetic fields. Moreover, a part of the magnetic energy is released by short emissions of X rays and extreme UV. Electrons and ions are also ejected. The most dangerous particles are the Solar Energetic Protons (SEP). They combine a high speed and great mass. They can reaches the half of the speed of light.

When the solar wind is approaching the earth, the particles of the solar wind arrive with a supersonic speed compared to the sound velocity in the plasma. The magnetosphere of the earth acts like a shield (Figure 4.1). It induces a bow shock upstream of the magnetopause, the outermost layer of the magnetosphere. This bow shock is located about 80 000 kilometers from Earth and is 100 kilometers thick. The bow shock slows down the solar wind. It also heats and disturbs it. Eventually, the solar wind reaches the magnetopause at an altitude of 64 000 kilometers. The solar wind compresses the magnetopause towards the earth and creates a magnetosphere distortion. The majority of the solar wind streams are deviated around the earth.

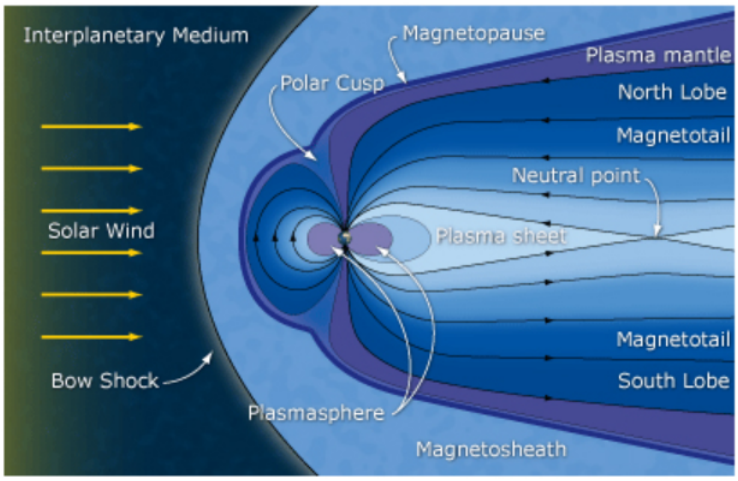

Figure 4.1: Schematic representation of the magnetic field of the Earth [28]

The solar magnetic field changes the polarity every eleven years, but the solar wind and the electromagnetic emissions can variate at all time scales. According to the relative direction of the earth's and the sun's magnetic field, the magnetic fields of the Earth and Sun can recombine (Figure 4.2). If the magnetic field of the sun is parallel and opposed to the terrestrial magnetic field, these magnetic fields combine. In this case, the magnetopause opens and the solar particles can enter the atmosphere. The particles can also enter the external magnetosphere through the magnetotail, where the magnetic field is weak. In this case, the particles are guided to the earth and enter the internal magnetosphere. Even without reconnection, particles can come in the atmosphere at the polar cusps, where the magnetosphere is weak.

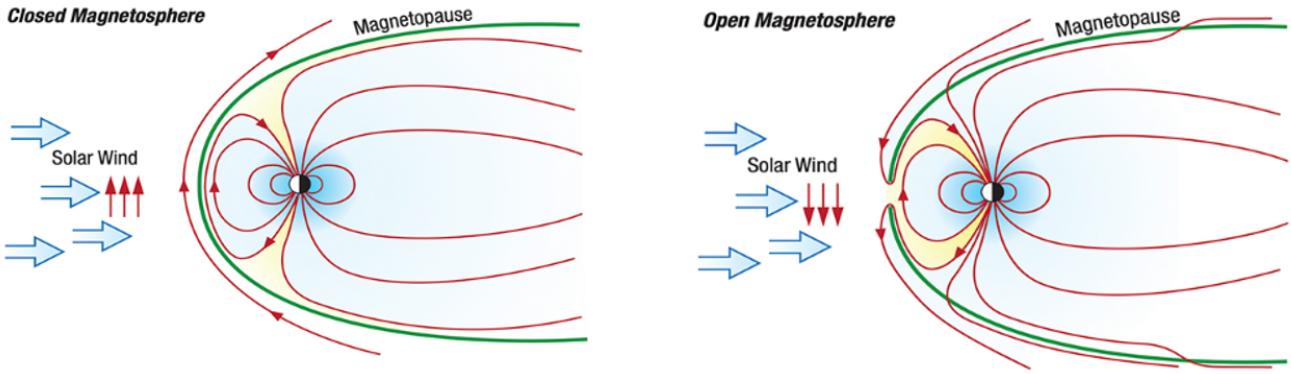

Figure 4.2: Schematic representation of the combinations of the Earth's and Sun's magnetic fields [28]

Once a particle has entered the magnetosphere, it is trapped. Then, the particles go back and forth bouncing from a pole to the other one and following the magnetic fields lines. This region along the magnetic fields lines is named the "Van Allen Radiation Belts".

Then, if the particles have enough energy, they will enter the atmosphere. The particles can also be stored in the magnetotail, in the plasma sheet.

The magnetic reconnection creates a magnetic storm which disturbs the terrestrial magnetic field. During this storm, the magnetotail is enlarged. It reinforces the electric fields in the magnetosphere. Therefore, the charged particles will gain energy and enter the atmosphere.

The substorms appear when the magnetotail is unstable and there are strong electric currents. In this case, a part of the electric current of the magnetotail is short-circuited through the ionosphere. Then, the electrons stored in the magnetosphere are propelled to the thermospere, the ionosphere and the high mesosphere (between 80 and 350 kilometers above sea level). Eventually, once stored in the atmosphere, the charged particles collide with oxygen and nitrogen atoms. These collisions release photons with different colors. The oxygen atoms produce green and red lights. The nitrogen atoms release blue light. The auroral substorms can appear frequently during these previously described magnetic storms but also during small disturbances caused by the solar wind variations.

The auroras occur mainly in the auroral ovals centered around the magnetic poles. These ovals change in depending on the solar activity. An auroral oval is four times wider on the night side. There still are auroras on the day side which are weaker because there are created by particles which have been previously caught in the magnetosphere.

There are also polar cap auroras which correspond to weak auroras that appear through the auroral ovals when the electrons comes through the magnetotail. Auroras happen in the thermosphere and exosphere, between 100 and 500 km altitude, while cloudy phenomena happen under 12 km altitude. It corresponds mostly to the troposphere (until 10 km altitude). Therefore, auroras are easily masked by different types of clouds. Clouds can also significantly change the aspect of an image containing a visible aurora. A cloudy sky can appear white, yellow or even dark. In the first case, the light reflected from the moon whitens the clouds. The colour yellow is caused by the light pollution. A dark sky is linked to high clouds. It can be distinguished from a sky that is not cloudy by the absence of stars.

Moreover, the light pollution, the moonlight and the sunlight (during the sunset and the sunrise), can create some confusions.

All auroral observers, whether they are tourists or scientists, are interested in knowing when there is aurora in the sky. Because we take millions of images every winter season it is a challenge to find the interesting times by visual inspection, and therefore, an automated routine is needed.

Chapter 5
State of the art

The image classification can be done in two ways. A first solution is to proceed with a preprocessing step, a features extraction, a dimension reduction, and finally, a classification step. Another solution is to replace these steps by a Convolutional Neural Network (CNN). I will further develop on the previously mentionned phases. Some classifiers and all the CNN require to choose hyperparameters. Some methods have been used in very similar contexts while others have been used in different contexts. Indeed, some methods were applied to proceed to a classification into several classes containing various types of auroras. Sometimes, the methods have been applied on gray scale images. Moreover, some articles give results obtained on data sets which do not contain images with clouds, moonlight, sunlight or artificial light. In our case, we deal with a binary classification on colored images containing clouds, moonlight, sunlight and artificial light.

First, I will give an explanation about the methods mentioned in some articles in the context of the auroral detection and classification. Some other associated methods which have not especially been used in this context will be described in annexes (B, C and D). Then, I will synthesise the various contexts of classification and their linked results given in the previous studies.

5.1 Preprocessing

The preprocessing step can contain one or several image processing's phases. Some of the preprocessing phases can be systematically used while other preprocessing phases depend on the features extraction.

All images must have the same orientation. In our case, all images have the same orientation and north corresponds to the top of the picture and east to the left of the picture.

One of the main steps is to choose the color system. The most famous color systems are the HSV (Hue Saturation Value) system and the RGB (Red Blue Green) system. Another possibility is to use the Lab color system. "L" corresponds to the perceived brightness, "a" is the coordinate on the "green-red" axis and "b" is the coordinate on the "blue-yellow" axis. On the contrary to the RGB and HSV systems, the advantage of the Lab system is that a variation of the values in the space corresponds to a proportional change of the perceived color. Indeed, in the RGB color system, if R, G and B have a similar value, the pixel is gray. If these values are close to 0, a small variation of one of the pixel's value will change the color. However, if these values are high, the same variation of one of the pixel's value will not change the gray appearance of the pixel. This choice highly depends on the chosen features extraction.

The available images have black corners which do not contain interesting information. However, these black corners induce a bias, especially if we want to compute the global statistical descriptors. To remove the black corners, we can crop the image. It also removes the parts of the image which correspond to low elevation angles. Therefore, if auroras are contained in these parts of the image, it can reduce the performance of the models.

Another possibility to remove the black angles is to apply the log polar or polar transformation. These transformations allow to transform each radius into an image row. Therefore, the black pixels (coming from the black corners) are confined on the right of the obtained image. Then, we can crop this obtained image.

We can also choose to reduce the noise on each image by computing the mean value of the pixels contained in a small window on the black corners and subtracting this value from all pixels [22].

To remove the noise, we can also use low-pass filters, like the median filter.

A possible preprocessing step is the normalisation of the pixels values. First, the values that are lower than a given percentiles (0.5 for example) are replaced by the value corresponding to this percentile. The values that are higher than another defined percentile (99.5 for example) are replaced by the value corresponding to this percentile. Then, the pixels values can be normalised.

We can reduce the size of the images to allow a faster training of the models.

Eventually, we could try to increase contrast by stretching the histograms or by reducing the number of bits, and see if it can improve the results. However, this method has not been used in the articles.

In practice, the preprocessing phases can be included in the features extraction methods. According to the chosen features extraction, we can apply some of the preprocessing steps beforehand.

5.2 Features extraction

The main idea is to transform a two or three dimensional image into a one dimensional vector by extracting the most relevant information. Then, the one dimensional vector can be directly input in a classifier to obtain a predicted label. We can also reduce dimensions of this vector before the classification step.

Many features extraction methods have been used to analyse images containing auroras. I will further develop explanations of some of them. However, these methods have often been used to classify different type of auroras or to detect auroras on gray scale images. In some articles, images containing hardly recognizable auroras, sunlight, moonlight or clouds have been removed. I will also describe in annexes some methods which are similar to the other described methods and which have also been used for image classification but not especially for aurora detection or classification (Annex B).

5.2.1 Edges and regions based methods

To describe our images, we could use edges descriptors. For example, an article [21] proposes to detect edges and to describe them thanks to the Fourier descriptor. This solution was used to

classify images which contain auroras into different categories. However, it would be difficult to extend the method to the binary classification between images with and without auroras. Indeed, this method implies that there are relevant edges to detect in each image.

Another solution is to use the hierarchical trees to analyse images [19]. The first step consists in partitioning the images into previously defined gray levels. Then, the regions are defined on the basis of these different gray levels. The regions contours can be described thanks to different methods. Eventually, we can compute the tree in which each node represent a contour with its corresponding descriptor. The child nodes correspond to contours of area contained in the current region. Therefore, each image can be represented by a tree and these trees can be compared to each other. In our case, we have colored images and the idea is to use this advantage to improve the classification. Moreover, as for the previously described method, this idea is based on the idea that the images contain regions with an important signification. Indeed, it is used to classify the different types of auroras.

5.2.2 Scale, rotation and translation invariant features

To extract features, we can decide to detect salient points and extract features in the regions containing these points. In our case, the extracted features have to be scale, rotation and translation invariant. A famous method meeting these criteria is the Scale Invariant Feature Transform (SIFT). This method has other advantages than the scale or rotation invariance. It is robust to the partial occlusions. However, even if it is robust to brightness changes, it is not the best brightness invariant descriptor [14].

Scale Invariant Feature Transform (SIFT) for gray scale images

The Scale Invariant Feature Transform begins with the salient points detection.

The first step consists of a scale space construction. To do that, we apply Gaussian filters with different sizes to the images. Then, we sub-sample one of the filtered images and we apply the same filters. This operation can be done a chosen number of times.

We need to define the number of octaves m which corresponds to the number of times an image is sub-sampled. Therefore, it also matches the number of sizes of the resulting images. Given σ the standard deviation of the smallest Gaussian filter, we need to decide the number of scales n which corresponds to the number of applied filters before applying a Gaussian filter with a standard deviation of 2σ. The value $k = 2^{1/n}$ is the value by which the standard deviation of a filter is multiplied to obtain the standard deviation of the following filter. The number of images for each octave is given by the number $l = n + 3 = 5$.

We can, for example, choose $m = 3$, $n = 2$, $k = \sqrt{2}$ and $l = 5$. The second step consists in detecting the extrema in the Difference of Gaussian (DoG).

Given a selected octave, we can compute the difference between two consecutive filtered images. The result is called the Difference of Gaussian (DoG). Then, we stack the DoG for a given octave. Eventually, we can compare each pixel to its eight neighbours of the same scale and the nine neighbours of each neighbour's scale. If the considered pixel is the smallest or the greatest among all its neighbours, it is considered as a salient point.

A salient point is composed of the coordinates x and y and a corresponding scale (scale where the extremum has been found).

The final step of the salient point detection consists of the removal of the instable points. It can be points with a low contrast or localised on edges with a small curvature.

Then, we can compute the orientation for each salient point.

To do that, we first consider a neighbourhood for each salient point with a dimension of 3σ where σ corresponds to 1.5 times the scale of the considered salient point.

Then, we can compute the amplitude a and the orientation θ of the gradient of each pixel (x_i, y_i) in the neighbourhood.

Given an image V filtered at the scale of the salient point. we know that :

$$a(x_i, y_i) = \sqrt{(V(x_i + 1, y_i) - V(x_i - 1, y_i))^2 + (V(x_i, y_i + 1) - V(x_i, y_i - 1))^2}$$

$$\theta(x_i, y_i) = arctan(\frac{V(x_i, y_i + 1) - V(x_i, y_i - 1)}{V(x_i + 1, y_i) - V(x_i - 1, y_i)})$$

Then, each orientation is weighted by the amplitude. Eventually, all the obtained orientations are weighted by a Gaussian window (corresponding to the neighbourhood). It allows to give less importance to the pixels which have low gradient amplitude or which are localized at the image edges.

The final phase is to create histograms of the orientations. An orientation is considered when its corresponding value exceeds eighty percent of the most represented direction.

The last phase of the SIFT algorithm is the computation of the descriptor vectors of each salient point.

We consider a neighbourhood of 16×16 pixels around each salient point. We divide this area into 16 blocks with a size of 4×4. Then, we can compute an histogram of the (eight) directions in each block. In the end, we obtain a vector with a value for each direction of each block.

Scale Invariant Feature Transform (SIFT) for coloured images

We have coloured images and we can apply the SIFT method on each channel. To do that, we can use different color systems. Nevertheless, we want to achieve a partial illumination invariance. If we use the HSV color system, we obtain a scale and shift invariant descriptor, with respect to light intensity. However, it would be partially invariant to light color changes [14].

Another solution is to use the Opponent SIFT. The Opponent color system is linked to the RGB system as follows :

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

O_3 only includes intensity information. O_1 and O_2 contain colors information but also take relative

brightness changes into account. If SIFT is applied to the channels O_1 and O_2, it will be insensitive to absolute intensity but it will take relative brightness into account. Therefore, this system would be suitable for our images.

Another system which could be appropriate is the Transformed color system. The Transformed color system is linked to the RGB system as follows :

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix}$$

With $\mu_R,\ \mu_G,\ \mu_B$ *the mean values*

and $\sigma_R\ \sigma_G\ \sigma_B$ *the standard deviations*

If SIFT is applied to those channels, it will be scale invariant and invariant to changes in light, color and arbitrary effects.

Another solution is to give different weights for each channel or to create a linear combination between the three channels of the RGB color system. For example, we can apply SIFT to the channels G and to the linear combination 2G-R-B. It allows to give more importance to the green channel. Indeed, aurora images often contains green color.

An article [14] relates results for those methods. The best results were obtained for the Opponent SIFT. Good results were also obtained when SIFT was applied to the channel G and to the linear combination 2G-R-B. Among the three best methods, we can also consider the Transform SIFT. On the contrary, SIFT only applied to the linear combination 2G-R-B does not offer a good accuracy.

5.2.3 Textures descriptors

There are lots of different texture analysis methods. We will see the benefits and drawbacks of a few of them.

First of all, we can compute statistical analysis of the different values of each channel. The first order statistics are descriptors that take into account one pixel at a time. Therefore, it does not consider the spatial distribution of the different gray levels.

To consider the global texture, a possibility is to divide the image into different patches before computing descriptors. We can consider the brightness distribution in north-south and east-west direction and the Ordinal Spatial Intensity Distribution (OSID). Contrary to the SIFT method, the OSID method is robust against complex brightness changes [20] [26].

We can also compute second order statistics like the Gray Level Cooccurence Matrices (GLCM) and the associated Haralick parameters [31] (described in annex B) [18] or the Gray level Aura Matrices (GLAM) [13] [21] [29] [31].

The Local Binary Pattern (LBP) is considered as a geometrical and structural approach [14] [20].

Eventually, a common approach in aurora images detection is to use sets of filters. Among the numerous possible methods, there are the Laplacian Pyramid, the Steerable Pyramid [6] [12]

(described in annex B) and the wavelets decomposition with orthogonal (Daubechies) and non-orthogonal (Gabor) wavelets basis [10] [23].

The Laplacian Pyramid does not take into account the orientations. Therefore, it will not allow to keep all information about oriented and elongated structures.

The orthogonal wavelets form a basis which does not respect the Nyquist criterion. It induces aliasing.

This noise in the sub-bands can be avoided by using non-orthogonal wavelets basis. However this type of wavelets basis induces redundant information. Therefore, we have to choose carefully the parameters to reduce these redundancies.

Another solution is to use the Steerable Pyramid [6] [12]. It allows to take into account the oriented and elongated structures. Moreover, it provides a non-orthogonal decomposition which allows to avoid aliasing.

First order statistics

The mostly used statistical descriptors are the minimum, the maximum, the mean and the standard deviation. However the first three descriptors are sensitive to the absolute brightness changes, which depends on the moonlight, the sunlight or the camera for example. The standard deviation is impacted by the relative brightness changes. Nevertheless, it is untouched by the absolute brightness changes [22] [23].

Moreover, the previously mentioned descriptors are global descriptors. Therefore, they do not bring any information about the images global structure.

Brightness distribution

An example of local intensity descriptors is the brightness distribution in (magnetic) north-south and east-west directions. This means first to place have the north-south axis along the vertical axis, and the west-east axis along the horizontal axis, which is already the case for our images. If we use the Cartesian coordinate system, the images are circular. We need to adapt by adding a normalisation [23].

Given an image I and $I(x, y)$ the pixel corresponding to the column x and the row y, the average brightness in each row corresponds to :

$$P(y) = \frac{1}{C_E(y) - C_W(y)} \times \sum_{x=C_E(y)}^{C_W(y)} I(x, y)$$

$$C_E(y) \ : \ column's\ number\ of\ the\ eastern\ most\ pixel,\ for\ the\ row\ y$$

$$C_W(y) \ : \ column's\ number\ of\ the\ western\ most\ pixel,\ for\ the\ row\ y$$

Therefore, the brightness distribution in north-south direction is defined by the following vector :

$$f_1 = [P(0)\ P(1)\ ...\ P(y_{max})]$$

The average brightness in each column corresponds to :

$$Q(x) = \frac{1}{L_N(x) - L_S(x)} \times \sum_{x=L_N(x)}^{L_S(x)} I(x, y)$$

$$L_N(x) \ : \ row's \ number \ of \ the \ northern \ most \ pixel, \ for \ the \ column \ x$$

$$L_S(x) \ : \ row's \ number \ of \ the \ southern \ most \ pixel, \ for \ the \ column \ x$$

Therefore, the brightness distribution in north-south direction is defined by the following vector :

$$f_2 = [Q(0) \ Q(1) \ ... \ Q(x_{max})]$$

Then, we can compute a distance between two vectors :

$$S = \min_k ||f^{(i)} - rot(f^{(j)}, k)||$$

The rotational operator $rot(f, k)$ corresponds to a circular shift of vector elements by k positions. It is justified by the invariance in the north-south direction. It allows to obtain a short distance between an image containing a bright arc in the north and another one containing a bright arc in the south.

Therefore, to find the minimum distance, we have to compute for each pair of vectors a lot of distances, corresponding to the number of rows for the north-south direction (and corresponding to the number of columns for the east-west direction). It induces a large number of calculations and a long computing time.

Moreover, it induces an impossibility to reduce dimensions. Indeed, a dimension reduction corresponds to a linear conbination between the vector elements. Then, if the we use rotated vectors, the dimension reduction can change the meaning of these vectors.

To solve the previously mentioned problems about the computing time and the impossibility to reduce dimension, we could prefer solutions which do not involve these issues. For example, we can apply a polar or log-polar transformation before computing the average brightness in each column. It does not correspond anymore to a brightness distribution in the north-south direction, but to the brightness of the many concentric circles of different sizes forming the images. We can consider that the small circles in the middle of the images do not have the same meaning as the wide circle on the outlines of the images. In this case, the circular shift of the vectors are not justified anymore.

However, computing the average brightness in each line after applying a polar or log-polar transformation corresponds to the computing of the brightness average of each circle radius. In this case, a circular shift of the vectors is justified because all radii have the same meaning.

In our project, we have colored images. Therefore, we can extend these types of calculation to other parameters than the brightness of the gray scale images, like the hue, the saturation and the value for example.

Ordinal Spatial Intensity Distribution (OSID)

A previously detailed method, SIFT, is robust to different variations or distortions. However, a better robustness against complex brightness changes can be achieved with the Ordinal Spatial Intensity Distribution (OSID) [20] [26].

The OSID method corresponds to a two-dimensional histogram where the pixel intensities are grouped in the spatial space as well as the ordinal space.

To proceed to the OSID we first have to pre-process the images. These images are smoothed with a Gaussian filter to remove the noise.

Then, we can detect intensity extrema and create local patches around these extrema to apply the rest of the algorithm to each patch. However, in our case, an article [20] proposes to use the whole image.

The first step consists in binning the pixels in the ordinal space. The pixels are binned in a certain number of bins where each bin contains pixels with similar intensities. If there are four hundred intensity levels and five bins, we will have eighty intensity levels for each bin. It will allow to obtain complex brightness changes invariant features.

The second step consists in binning the pixels in the spatial space. The image is divided in a certain number of angular sectors. This subdivision is appropriate for circular images. The pixels are labelled according to their area. It will allow to capture the structure information.

Eventually, we can create a two-dimensional histogram. The X-axis indicates the pixel intensity distribution. The Y-axis indicates the spatial area. Then, the rasterization consists in transforming the two-dimensional histogram into a one-dimensional vector. At the end, this vector is normalised.

Gray Level Aura Matrix (GLAM)

I will describe here the Gray Level Aura Matrices (GLAM) [13] [21] [29] [31].

We consider an image X as a $m \times n$ matrix. We set :

$$S = \{s = (i, j) | 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$$

We define the neighbours system $N = \{N_s, s \in S\}$ where N_s corresponds to the neighbourhood of the pixel s.

For $s, t \in S$, $s \notin N_s$ and $s \in N_t$ if and only if $t \in N_s$.

The pixel s is not included in its own neighbourhood and the neighbourhood is symmetrical.

The neighbourhood of the pixel s is described as follows :

$$N_{s=(i,j)} = \{r = (k, l) \in S | 0 < |(k-i)^2 + (l-j)^2| \leq d\}$$

$$\textit{with } d \ : \ \textit{a chosen distance}$$

Then, we define the Aura concept. Let $A, B \subseteq S$, we define the aura of A with respect to B :

$$V_b(A, N) = \cup_{s \in A}(B \cap N_s)$$

It corresponds to pixels included in B and which are the neighbours of all the pixels contained in A.

Let the aura measure of A with respect to B :

$$m(A, B) = \sum_{s \in A} |B \cap N_s|$$

where $|A|$ is the total number of pixels contained in A

Because the neighbourhood system is symmetrical, we can write $m(A, B) = m(B, A)$.

Let $J = \{S_i | 0 \leq i \leq G - 1\}$ a subdivision of the S, where G is the number of gray levels.

The aura matrix of J is :

$$A(J) = [m(S_i, S_j)]$$

where $m(S_i, S_j)$ corresponds to the aura measure between S_i and S_j, $0 \leq i, j \leq G$. We know that each pixel $s \in S$ has a gray level x_s corresponding to an integer between 0 and $G - 1$.

We define subsets of S corresponding to the different gray levels $S_g = \{s \in S | x_s = g\}$ where $g = 0, 1, ..., G - 1$.

If $m(A, B)$ is low, it indicates that the subsets A and B are clearly separated.

Then, we normalise the Aura matrix to have

$$\sum_{i,j=0}^{G-1} a(S_i, S_j) = 1$$

To compute the aura matrices, we first need to compute the gray levels subsets.

Then, we compute $m(S_g, S_{g'})$ where g, g' are gray levels of the previously defined subsets.

We set each element of the A matrix ($A = [m(S_i, S_j)]$) to 0. Next, for each pixel s, we define its gray level g, and we increment $m(S_i, S_j)$ for each pixel contained in the neighbourhood of s and with a gray scale g'.

We use the symmetry to reduce the computing time.

Eventually, we compute the similarity measure between two images. Let X_1 and X_2 two gray scale images and their associated Aura matrices : $A_1 = [a_1(S_i, S_j)]$ and $A_2 = [a_2(S_i, S_j)]$.

The similarity between them is computed as follows :

$$d(A_1, A_2) = \sum_{i,j=0}^{G-1} [a_1(S_i, S_j) - a_2(S_i, S_j)]^2$$

Because the neighbourhood is considered as symmetrical, we cannot catch the anisotropic textures. To solve this problem, we can apply a shiftable multi-scale transformation which allows to decompose the image into different spatial frequency bandwidths and to divide each of these frequency bandwidth into different orientations [17]. Therefore, we can construct a steerable pyramid (described in annex B), a shiftable multi-scale transformation detailed later in this report, and apply the GLAM algorithm to each level of the pyramid.

Let $B_0, B_1, ..., B_{k-1}$ the normalised gray level aura matrices steerable pyramid, $A = [B_0, ..., B_{k-1}]$.

A Basic Gray Scale Level Aura Matrix (BGLAM) corresponds to GLAM by considering only one pixel in the neighbourhood of each pixel. It is preferred to other types of GLAM because two images are identical if and only if their BGLAM are identical.

Structural and geometrical approach : Local Binary Pattern (LBP)

The Local Binary Pattern (LBP) is a gray-scale invariant texture measure. The main idea is to create a binary code to describe the local texture [14].

Given a pixel (x_c, y_c) with a gray level g_c and P neighbours, we can define the LBP code :

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} S(g_p - g_c) \times 2^p$$

$$with\ g_p\ :\ the\ gray\ level\ of\ the\ neighbour\ p$$
$$and\ S(x) = 0\ if\ x < 0,\ else\ S(x) = 1$$

Then, we can obtain 2^P possible values for a given pixel. Then, we can compute a histogram counting the obtained values for all pixels.

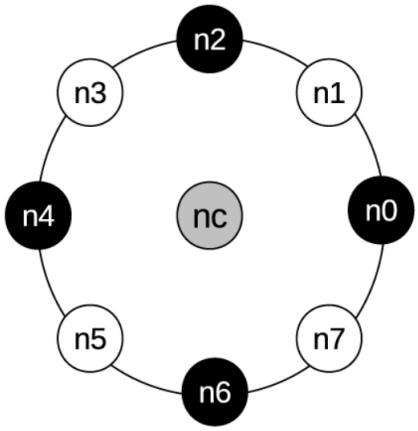

Figure 5.1: Schematic representation of a circular neighbourhood of a pixel n_c with a radius of one pixel [14]

If the defined neighbourhood is a circle with a radius of one pixel (Figure 5.1), we obtain the following formula:

$$LBP = S(n_0 - n_c) \times 2^0 + S(n_1 - n_c) \times 2^1 + S(n_2 - n_c) \times 2^2 + S(n_3 - n_c) \times 2^3$$
$$+S(n_4 - n_c) \times 2^4 + S(n_5 - n_c) \times 2^5 + S(n_6 - n_c) \times 2^6 + S(n_7 - n_c) \times 2^7$$

This classic LBP algorithm (with $P = 8$ neighbours) gives a result for each pixel of the image. There are $2^P = 2^8 = 256$ possible values. Then, we can create a histogram with 256 values, counting the number of pixels corresponding to each value. If we apply this algorithm to the three

channels of the colored images, we obtain a final feature with of 768 elements. A possibility to reduce the size of the final vectors is to use the Orthogonal Local Binary Pattern Combination (OLBPC). It can be computed as follows :

$$OLBPC = [OLBP1 \;\; OLBP2]$$
$$OLBP1 = S(n_0 - n_c) \times 2^0 + S(n_2 - n_c) \times 2^1 + S(n_4 - n_c) \times 2^2 + S(n_6 - n_c) \times 2^3$$
$$OLBP2 = S(n_1 - n_c) \times 2^0 + S(n_3 - n_c) \times 2^1 + S(n_5 - n_c) \times 2^2 + S(n_7 - n_c) \times 2^3$$

It allows to create two histograms with $2^4 = 16$ components. The final features contains 32 elements. If we apply it to the three channels, we obtain a final feature vector with 96 elements.

Eventually, we can apply the LBP algorithm directly to the image or divide the image into areas and apply the LBP algorithm to each area. In two articles [14][20], they divided the image into $3 \times 6 = 18$ areas. It allows to obtain local and global information. It gives information about the rotation invariant patterns. This descriptor is also robust against brightness changes.

Gabor wavelet decomposition

The wavelets decomposition allows to obtain a multiresolution representation. On the basis of this decomposition, we can compute features vectors [10] [23].

The mother Gabor wavelet can be written as follows :

$$g(x,y) = (\frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})+2\pi jWx})$$

$$\sigma_x \; and \; \sigma_y \; : \; the \; standard \; deviations$$
$$W \; : \; the \; modulation \; frequency$$

The Fourier transform of the previous wavelet is :

$$G(u,v) = e^{-\frac{1}{2}[\frac{(u-W)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}]}$$

$$\sigma_u = \frac{1}{2\pi\sigma_x} \; and \; \sigma_v = \frac{1}{2\pi\sigma_y}$$

A class of similar wavelets can be created by dilatation and rotation of this mother wavelet in the frequency domain :

$$g_{mn}(x,y) = a^{-m}G(x',y')$$
$$x' = a^{-m}(xcos(\theta) + ysin(\theta))$$
$$y' = a^{-m}(-xsin(\theta) + ycos(\theta))$$
$$\theta = \frac{n\pi}{k} \; and \; k : the \; number \; of \; orientations$$
$$a^{-m} : \; the \; scaling \; factor$$

These wavelets are filters used as points and lines detectors with adaptive scales and orientations.

The non-orthogonality of the wavelets basis induces redundant information in the filtered images. Therefore, we have to choose the parameters to reduce the redundancy.

The parameters are chosen on the basis of the following values :

$$U_i \text{ and } U_h \ : \ \textit{the center frequencies of the lower and upper frequencies wavelets}$$
$$k \ : \ \textit{the number of orientations}$$
$$S \ : \ \textit{the number of scales}$$

The parameters are chosen as follows :

$$a = (\frac{U_h}{U_i})^{-\frac{1}{S-1}}$$
$$\sigma_u = \frac{(a-1)U_n}{(a+1)\sqrt{2ln(2)}}$$
$$\sigma_v = tan(\frac{\pi}{2k})(U_h - 2ln(\frac{\sigma_u^2}{U_h}))(2ln(2) - \frac{(2ln(2))^2\sigma_u^2}{U_h^2})^{-\frac{1}{2}}$$
$$W = U_h \textit{ and } m = 0, 1, ..., S-1$$

With this parameters, the Gabor filters overlap until the half peaks levels. We still have to choose the parameters S, K, U_h and U_i.

To avoid sensitivity to absolute intensity changes, we have to choose the parameters so that we obtain $G(0,0) = 0$.

Given an image $I(x,y)$, the Gabor wavelets transform for a specific orientation and a specific scale is:

$$W_{mn}(x,y) = \int I(x_1, y_1) g^*_{mn}(x - x_1, y - y_1) dx_1 dy_1$$

The mean μ_{mn} and the standard deviation σ_{mn} of the magnitude of the transform coefficients are computed as follows:

$$\mu_{mn} = \iint |W_{mn}(x,y)| dxdy$$
$$\sigma_{mn} = \sqrt{\iint (|W_{mn}(x,y)| - \mu_{mn})^2 dxdy}$$

These previously mentioned parameters allow to create a features vector:

$$f = [\mu_{00} \ \sigma_{00} \ \mu_{01} \ ... \ \mu_{35} \ \sigma_{35}]$$

The distance between two features can be written as follows:

$$d(i,j) = \sum_m \sum_n d_{mn}(i,j)$$

$$d_{mn}(i,j) = |\frac{\mu_{mn}^{(i)} - \mu_{mn}^{(j)}}{\alpha(\mu_{mn})}| + |\frac{\sigma_{mn}^{(i)} - \sigma_{mn}^{(j)}}{\alpha(\sigma_{mn})}|$$

$$\alpha(\mu_{mn}) \; and \; \alpha(\sigma_{mn}) \; : \; the \; features \; standard \; deviation$$

Eventually, feature vectors are computed on the basis of wavelets decomposition. The Gabor wavelets are non-orthogonal. It allows to avoid aliasing because the Nyquist criterion is met. However, it induces information redundancy and the parameters have to be adapted to reduce it.

5.3 Dimensionality reduction

The dimensionality reduction is used to improve the efficiency. It allows to achieve better results, reduce computing time and storage need. It can also allow to obtain a three-dimensional visualisation. It removes the noise in the data by keeping only the most important features and removing the redundant ones. Moreover, it avoids overfitting. To reduce dimensions, we can select the most interesting variables. Another solution is to extract characteristics by creating relevant variables. There are linear and non linear dimensionality reduction. The Principal Components Analysis (PCA) is a well known linear dimensionality reduction. The aim is to find the dimensions which have the greatest variance in the data. Then, we can represent each point along these axes. The main problem of the linear dimensionality reduction is to only consider the Euclidean distances. However, two points can be close to each other according to the Euclidean distance but distant from each other if we consider the distance by following the surface defined by the points. The swiss roll illustrates this problem. Some algorithms based on neighbour graphs allow to consider the geodesic distances, like the isometric mapping (ISOMAP) [27] (described in annex C), the t-distributed stochastic neighbor embedding (t-SNE) [9] and the Uniform Manifold Approximation and Projection (UMAP) [11].

5.3.1 t-distributed stochastic neighbor embedding (t-SNE)

The main idea of the t-SNE algorithm is to consider the geodesic distance. First, we compute the similarities among points in the initial space [9]. For each point X_i, we center the Gaussian distribution around this point. Then, we measure for each point X_j $(i \neq j)$ the corresponding density under the Gaussian distribution previously defined. Then, we can normalise this computed density. In fact, we compute the normalised density as follows :

$$p_{ij} = \frac{e^{\frac{-||X_i - X_j||^2}{2\sigma^2}}}{\sum_{k \neq l} e^{\frac{-||X_k - X_l||^2}{2\sigma^2}}}$$

$$with \; \sigma \; : \; the \; standard \; deviation$$

The standard deviation is defined on the basis of a value called "the perplexity", which corresponds to the number of neighbours we want to consider for each point. This value is chosen by the user and allows to estimate the standard deviation of the Gaussian distributions defined for each point X_i. The second step consists in creating a space of lower dimension in which we will represent our data. This new representation is named Y and $Y = \{Y_1, ..., Y_n\}$. We first randomly distribute the

data in this new space. Then, we compute the similarities among the points by using a t-student distribution and not a Gaussian one.

$$q_{ij} = \frac{(1 + ||Y_i - Y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||Y_k - Y_l||^2)^{-1}}$$

Eventually, we want that the similarity measures q_{ij} coincide with the similarity measures p_{ij}. We can compare the similarity measures thanks to Kullback Leibler measures for example. This measure can be minimised by using the gradient descent.

5.3.2 Uniform Manifold Approximation and Projection (UMAP)

The UMAP algorithm is very similar to the t-SNE algorithm [11]. However, the UMAP algorithm is faster and captures the global structure. In the t-SNE algorithm, the standard deviation is the same for all points. In the UMAP algorithm, the standard deviation is chosen for each point according the distances of the K nearest neighbours. This K is the first parameter to choose to use the UMAP algorithm. We also need to choose the smallest distance between two points in the new space.

5.4 Classification

We can use a binary or a multi class classification. There are lots of algorithms of supervised classification : Support Vector Machine (SVM) [4], K-Nearest Neighbours (KNN), Bayesian classifier (described in annex D), decision tree and random forest [2]. Each method has advantages and disadvantages. Each model can induce "underfitting" or "overfitting". "Underfitting" happens if the model is not adapted enough to the data. "Overfitting" happens if the model is perfectly adapted to the training data and cannot be generalized to other data.

At the end of the classification, we obtain a confusion matrix (Table 5.1).

		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 5.1: Confusion matrix

On the basis of this confusion matrix, we can compute the precision P, the recall R and the accuracy A.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$A = \frac{TP + TN}{TP + TN + FN + FP}$$

In our case, we want to minimize the False Negative (FN) rate. Therefore, we prefer to improve the Recall rather than the precision. Obviously, we also want to improve the accuracy.

5.4.1 Linear regression and ridge classification

I will describe here the linear regression and the ridge classification [3] [16].

Let N inputs vectors $f_i \in \mathbb{R}^F$ corresponding to features vectors and N associated labels y_i. The linear regression enables to find the weights $\omega \in \mathbb{R}^F$ which minimize the cost function C_{LS}. This function depends on the differences between the the real labels y_i and the predictions $\omega^T f_i$:

$$C_{LS}(\omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \omega^T f_i)^2$$

If the number of features F is close to the number of inputs N, the weights can be perfectly adapted to the training data and give bad results on the validation and testing data. This is called "overfitting".

To solve this problem, the ridge regression consists in adding a regularization term. The high values of ω are penalized. The loss function becomes :

$$C_{RR} = (\omega) = \lambda\omega^T\omega + \frac{1}{N}\sum_{i=1}^{N}(y_i - \omega^T f_i)^2$$

$$with\ \lambda \in \mathbb{R}\ :\ the\ regularization\ coefficient$$

In the case of the binary classification, the real labels and the predictions can take the values 0 or 1.

5.4.2 Support Vector Machine (SVM)

We first place each training data sample in a space with F dimensions where F corresponds to the number of features. Each feature's value will correspond to the value of a specific coordinate [4] [21].

Then, the main goal of the SVM is to find the hyperplane which best separates two groups of samples. The chosen hyperplane has to allow to correctly classify as many data samples as possible. However, if some data has outliers, they have to be ignored. The chosen hyperplane also has to maximise the distances between the hyperplane itself and the data samples.

Sometimes, it is impossible to separate the data with a linear hyperplane. If there are very few misclassified samples, we can assume that it is due to outliers. If there are lots of misclassified samples, we can use non-linear hyperplane. That is the same as adding a coordinate which combines other coordinates and defining a hyperplane in this new space. This method is named "kernel SVM" [14].

The SVM classifier does not require to choose lots of parameters. We only have to decide if we want to find a non-linear hyperplane.

5.4.3 K-Nearest Neighbours (KNN)

As for the SVM classification, the first step consists in placing each data in the space with F dimension where F corresponds to the number of features [20][22][23][29].

The principle is to assign the label to a testing sample according to the neighbours of this sample. The chosen label corresponds to the most numerous class among the K nearest neighbours (training sample) of the currently assessed testing sample. We can also apply strict KNN algorithm. It means that we assign the label only if all the K neighbours have the same label.

To decide if samples are neighbours or not, we have to define a distance metric. We can use Manhattan, Minkowski, Euclidean or Hamming distances.

We also have to define K. It can induce "Underfitting" or "Overfitting". A low value of K can lead to obtain high variances and low bias in the results. A high value of K can induce a low variance and a high bias. The choice of K has to be done according to the input data. Indeed, if the data contain lots of outliers or noise, we would prefer a high value of K. It is also important to choose an odd number to avoid decision problems.

This method adapts well to new data. Moreover, we only have to choose a distance metric and a K value. However, this method needs to store lots of data and takes time to compute. It does not work well if the data have lots of features. Eventually, the choice of the K value strongly impacts the results.

5.4.4 Decision tree and random forest

A decision tree asks a series of questions which leads to the prediction. Each test corresponds to a node and each possible choice corresponds to a branch [2].

A random forest is a group of decision trees. A single tree can create rules to make decisions. A random forest randomly selects features and creates different uncorrelated trees on the basis of the selected features. Then, the recorded result corresponds to the mean of the results obtained thanks to all decision trees.

Lots of uncorrelated trees allow to make more precise predictions. It protects against individual errors and overfitting. It also allows to classify data with missing values or outliers. However, it requires more time to process than a single decision tree. To solve this issue, we can sample the training data.

5.4.5 Neural Network

A neural network can be used for a supervised classification. To use neural networks we have to implement the model and to proceed to a training phase. Vectors are placed at the entrance of the network. The output corresponds to the probability of belonging to the different classes. During the learning phase, we can adapt the neural network on the basis of the differences between the real labels and the predictions, by back propagation. I will now describe the architecture of a neuron and a neural network and the changes made during the learning phase.

A neuron is made of a vector of weights, a combination function and an activation function (Figure 5.2).

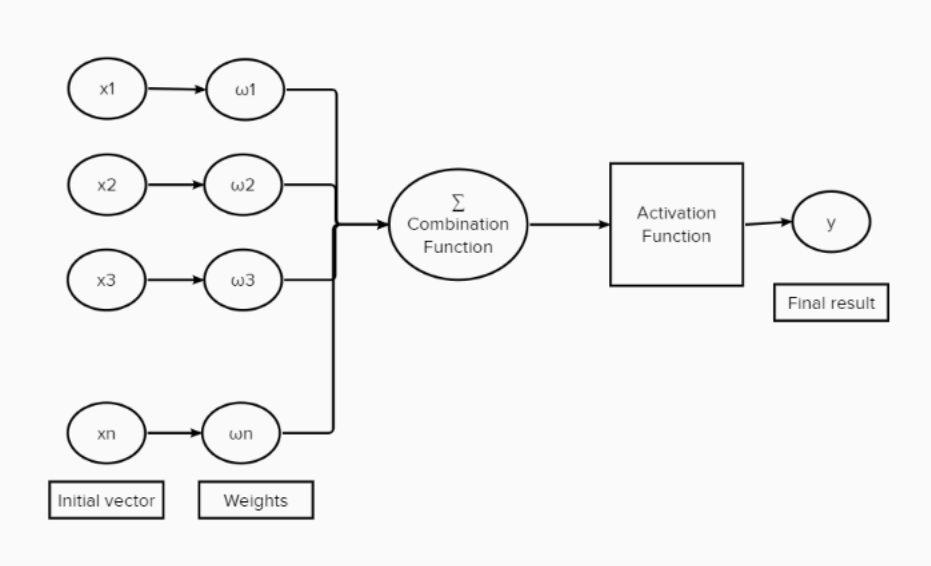

Figure 5.2: Diagram of an artificial neuron

Some of the most famous activation functions are the softmax function, the sigmoïd function and the Rectified Linear Units (ReLU) function.

The softmax function is :

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

$$With\ K\ :\ the\ size\ of\ the\ input\ vector$$

The sigmoïd function is :

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

The Rectified Linear Units (ReLU) replaces negative values by zeros :

$$h(x) = max\,\{0, x\}$$

During the learning phase, the weights are optimized by back propagation of the gradient of the difference between the real labels and the predictions (which corresponds to the loss function). This gradient is function of the input weights. The weights changes are proportional to the chosen learning rate. There are various optimization functions. One of the most popular and efficient functions is called "Adam". Each loss function computed for an image also depends on the number of images contained in the same class. If there are lots of images in the class, the associated weight of the loss function is low.

The batch size corresponds to the number of data samples for which we compute the loss function before simultaneously updating the weights.

During an epoch, the neural network updates its weights once on the basis of all data samples contained in the training data set. It reexamines all the data and updates the neural network at each epoch.

Between each epoch, the neural network is tested on a validation data set, which often comes from the same original data set as the training data set.

Eventually, the model is tested on a testing data set, which at best comes from a different source from the training and validation data set.

A neural network can be a feed forward network (Figure 5.3). It means that the information never goes back in the model. On the contrary, the recurrent models proceed information in cycles. These cycles allow to process the same piece of information many times by reinstating it in the network.

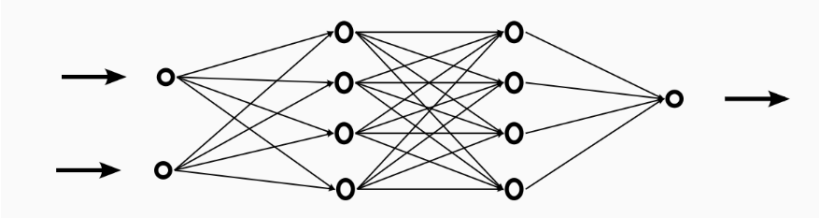

Figure 5.3: Feed forward neural network

If there is not enough vectors in the learning data set, the model can perfectly adapt its weight to the available data. In this case, the model is not appropriate anymore for other data. Then, the network provides bad results when it processes the testing data set. This problem is called "overfitting" and can be solved by some methods.

To avoid "overfitting", we can add a regularisation function to each layer. This is a modification of the loss function which allows to update the network during the learning phase. By adding a penalizing factor proportional to the weights of the layer, the regularisation function can avoid the over adaptation to the training data set. There are different types of regularisation functions.

The function L1 is :

$$\alpha \sum_{j=1}^{p} |\omega_j|$$

$$with\ \alpha\ :\ a\ scalar\ coefficient$$

$$and\ \omega_j\ :\ the\ weights\ of\ the\ layer$$

The function L2 is :

$$\alpha \sum_{j=1}^{p} \omega_j^2$$

The dropout is another method to reduce "overfitting". It corresponds to a temporary removal of some neurons during the training phase. Their values are replaced by zeros during this phase and reactivated to test the model. It requires to choose the dropout rate for each layer.

To implement a neural network, we have to choose the number of layers and the number of neurons contained in each layer, the dropout rate, the regularization function and the learning rate.

5.5 Convolutional Neural Network (CNN)

5.5.1 Principle

A Convolutional Neural Network (CNN) is an acyclic neural network. In the case of supervised deep learning, there is a learning step and a test step.

The CNN consists of two main parts. The first one is made of convolution layers. The second one corresponds to a neural network.

An image is given in the shape of a three dimensional matrix at the entrance of the CNN (Figure 5.4). The convolution part extracts features specific to each image and reduces the number of parameters to find in the neural network. The image goes through a succession of filters creating new images called convolution maps. Eventually, the obtained convolution maps are concatenated in a features vector and input to the neural network. Once a filter has been applied, each pixel goes through an activation function. This function is often a Rectified Linear Units (ReLU) which replaces the negative values by zeros.

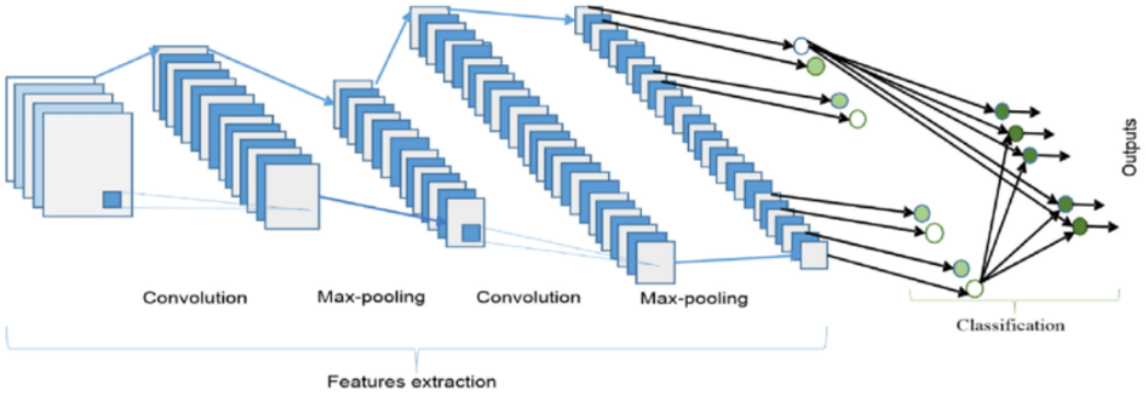

Figure 5.4: CNN diagram [1]

The convolution is a simple mathematical operation usually used for the image processing and recognition. The window containing the filter moves on the image (from left to right and from top to bottom) with a chosen step. It requires to define the filter, the size of the window and the step. Each filter takes into account all the layers of the image and produce a single layer. Therefore, the obtained convolution map obtained at the output of each step (composed of a group of filters) is a convolution map with a number of layers corresponding to the number of applied filters. For each part of the image, the convolution (by applying the chosen filter) allows to obtain a feature map which indicates where the features are localized in the image. The convolution filters are used for the detection of boundaries and shapes. The mean and Gaussian filters allow to reduce noise.

Between two convolution steps, we can down sample the filtered images by using a Max or Average Pooling. The Max Pooling (or Average Pooling) keeps the maximum value (or mean value) for each region swept by the filter to reduce the size of the image. It reduces the number of parameters to learn and it makes the features extraction invariant under translation. It also keeps the essential features contained in the image.

The neural network is the classification part. It is made of fully connected layers that combine the features to analyse and classify the image. The neural network successively apply linear combinations and activation function to classify the image. The output corresponds to a vector containing the probability of belonging to a each group.

During the learning step, the filters of the convolution part and the neurons of the neural network are updated by an algorithm of back propagation based on the difference between the real label and the prediction.

5.5.2 Transfer learning

The transfer learning consists in adapting a pretrained CNN model. The knowledge contained in such a network can be used in two ways. It can be used as a simple image features extractor. Then, we only have to replace the last layer with a layer adapted to the number of classes. The pretrained model can also be considered as a model initialisation. Then, the model can be more finely retrained and adapted to our data set. This method is called "fine tuning". The last layer has to be adapted to the number of classes. We also can add layers between the pretrained model and the last layer. To take advantage of the features extraction capacity of the original model, we have to choose a low learning rate (around 10^{-3}). It is a good way to slowly adjust the model to the new classification problem [3] [16] [8].

We can also use pretrained models as features extractors without adapting them to the new data set. Then, we need to add a classifier.

There are lots of pretrained models available in the "tensorflow" library : "AlexNet", "ZFNet", "VGG 16", "VGG 19", "GoogleNet", "ResNet", "Inception", etc. Some of these CNN models have particularities.

In the "ResNet" models, some steps are composed of two parallel paths. One of the paths is made of some filters, while the other one consists in keeping the original input. Then, the two paths are combined by adding together the two obtained convolution maps (Figure 5.5) [5].

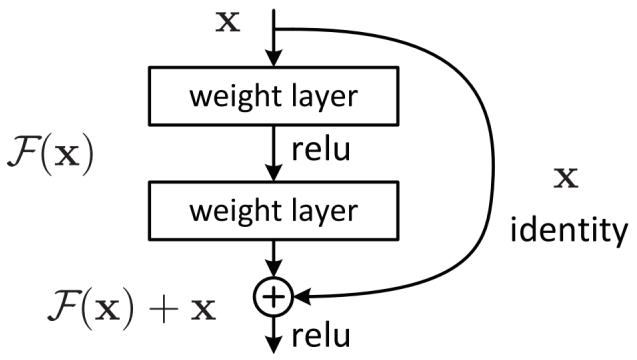

Figure 5.5: Building block of a "ResNet" model [5]

The "Inception" models also rely on the idea of using parallel paths before combining the obtained filtered images. In this case, each path is composed of a set of filters. At the end of each step, the

filtered images are concatenated. The following image (Figure 5.6) corresponds to the building block of the first version of the "Inception" model [24].

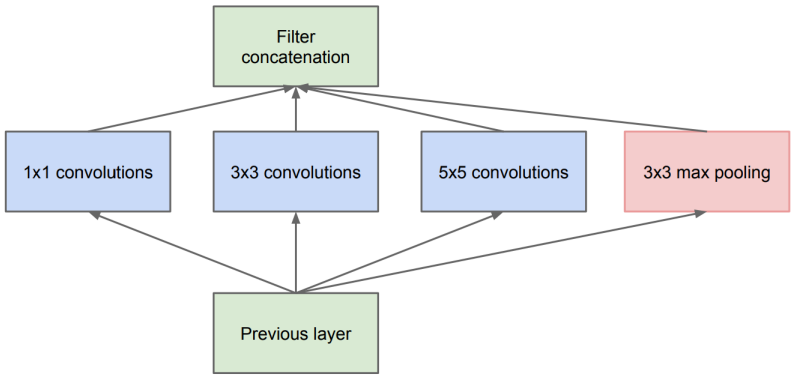

Figure 5.6: Building block of the first version of the "Inception" model [24]

Because one filter combines all layers into a single one, using a set of filters (with a size of 1×1) can reduce the number of layers if the number of filters is smaller than the number of original layers. It allows to reduce the computing time of the following steps. This method has been used in the next versions of the "Inception" model. Moreover, the computing time has been improved by replacing filters with an important size (5×5 for example) by more filters with smaller sizes (3×3 for example). Eventually, the filters with a size $N \times N$ have been replaced by two filters with sizes $1 \times N$ and $N \times 1$. The following image (Figure 5.7) corresponds to the building blocks used in the third version of the "Inception" model [25].

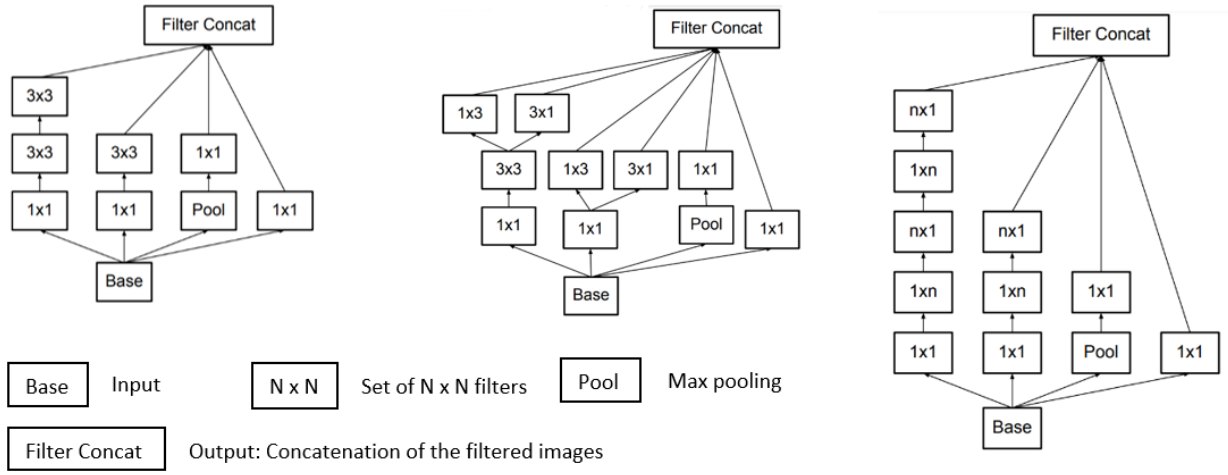

Figure 5.7: Building blocks used in the third version of the "Inception" model [25]

5.6 Results obtained in different articles dealing with various contexts of classification

I will first develop on an article [14] which deals with a binary classification of colored images in two classes, one containing images with auroras and another one containing images without aurora. Different features extraction methods were used and compared. At the end, the classification was made thanks to a kernel SVM. The OLBPC (with a radius of one pixel) method offered poor results with a mean cross validation error around $35,59\%$. The SIFT method was tried with different color systems. A first solution consists in applying the SIFT algorithm to a channel which is a combination of the three RGB channels. They hypothesised that the auroras often contains green. Therefore, they decided to give a weight of 2 for the green channel and to subtract the red and blue channels (2G-R-B). This solution offered a better result than the OLBPC method with a mean cross validation error of $13,7\%$. Then, the SIFT algorithm was used on the normalized RGB channels. It allowed to reduce the mean error to $8,67\%$. The SIFT algorithm was also applied to the channels G and the combination previously described (2G-R-B). The mean error was $8,63\%$. The SIFT method was also applied to the Opponent colors system. This solution gave the best result with a mean error of $8,51\%$. When this last method was tested on another data set, the obtained error was around 20%.

The second article [20] deals with a binary classification between images containing auroras and images which do not contain aurora. The available images are gray scale images centered at 557,7 nm with a bandwidth of 2,0 nm. Different features extractors have been compared with each other. Then, a KNN classifier has been used with different values of K. Regardless of the features extractor, the classification is improved until $K = 5$. Therefore, we will focus on the results obtained with $K = 5$. The first extractor consists in computing global statistical brightness descriptors : the minimum, the maximum, the mean value, and the range. Computing the mean, the minimum and the maximum values allows to obtain an error of 12%. The brightness descriptors have also been computed for the different local area of the images (with $3 \times 6 = 18$ area). It allows to improve the robustness against global brightness variations. Computing the minimum, the maximum and the mean values allows to obtain an error of 6%. Computing only the local mean value or the range allows to obtain similar results. The global LBP features extractor gives an error of 14% while the local LBP (with $3 \times 6 = 18$ area) gives an error of 7%. Eventually, the OSID method has been used by dividing the images into three and seven angular sections. These two methods give an error of 8%. We notice that the local extractors give better results than global extractors.

The fourth article [23] deals with a classification into classes which contain three different types of auroras (auroral arcs, patchy aurora and omega-bands). The available images are gray scale images centered around 558 nm. The features extracted are the brightness (mean and maximum value), the north-south and east-west brightness distribution and the vector linked to the Gabor wavelet decomposition as previously described. Next, the distances related to each feature are computed between all points. Then, the final distance between two points corresponds to the Euclidean distance. Eventually, the label is assigned thanks to the KNN algorithm. Images are randomly selected in the full data set. Then, if we apply a strict KNN, some images are not classified. 99% of the "auroral arcs" and 89% of the "patchy auroras" are correctly classified. However, only 12% of the "omega-bands" are correctly classified.

We focus now on an article [22] which deals with auroras detection on gray scale images centered

around 557,7 nm. It does not take into account images where the moon is above the horizon. First, we compute the mean value of brightness in the dark corners. It corresponds to the noise. Then, this value is subtracted from each pixel. Next, the mean and maximum brightness values are computed for each image. Eventually, the KNN classifier is applied to the testing data. Different values of K are tested. The higher the K value, the better the accuracy, until K achieves 17. If $K = 17$, the obtained error is around 8%.

The next article [21] deals with a classification of all-sky and gray scale images into six different classes corresponding to different types of auroras. The BGLAM is computed for each image. Then, a SVM enables the classification. It offers an accuracy between 34% and 42% if each image corresponds to a given type of aurora. If an unknown class is added before the labelling step, the obtained accuracy is between 50% and 53%.

An article [29] deals with a classification into three classes corresponding to different types of aurora on the basis of gray scale images. It is shown that the features extraction with the algorithm LBP clearly outperform the extraction based on the BGLAM extractor. The exact results depends on the K of the K-NN classification.

The next article [3] deals with a classification of colored images in six classes. Three classes contain images with three different types of auroras : "arc auroras", "diffuse auroras" and "discrete auroras". Another class contains images which show cloudy sky. The fourth class contains images which mainly contain moonlight. The last class contains images with a clear sky with visible stars but without auroras. Therefore, in this classification, there is no differentiation between cloudy skies without aurora and cloudy skies with auroras. The preprocessing step consists in cropping the image to remove the pixels at low elevation angles. For each image, we compute the first percentile of the brightness values and we subtract this value to each pixel. Then, the ninety-ninth percentile is computed and all pixels are divided by this value. Eventually, all values above 1 are reduced to 1. Then, the pretrained model "Inception V4" is used as a features extractor. At the end, the ridge classifier is used to allow the classification in the six previously described labels. The results show that the model separates well the classes containing moonlight, clouds and auroras. However, the classes containing auroras are confused. If we consider six classes, we obtain a mean error of $18,3\%$. If we consider three classes by combining the classes containing auroras, we obtain a mean error of $4,4\%$.

The next article [8] deals with colored images without clouds, moonlight or sunlight. The images are classified into seven different classes, corresponding to different types of auroras. A median filter is first applied on all images to remove the noise. Then, some well known CNN models are tested. However, these models are not pretrained. The "ResNet50" and "ResNet18" models allow to obtain an error around 8%. With the "ResNet50" model, the obtained recall is around 89%. With the "ResNet18" model, the obtained recall is around 87%. The "AlexNet" model gives an error around 12%. The "VGG-16" gives an error of 16% and the "VGG-19" gives an error of 18%. Therefore, the best result is obtained with the "ResNet50" model.

The last article [16] deals with a classification into six classes (Three classes containing three different types of auroras and three classes without aurora). This classification enables to obtain more information than if we only apply a binary classification (Images with or without aurora). Moreover, it improves the accuracy of this binary classification. The available images are gray scale images around 557,7 nm. First the images are re-scaled between the 0,5 and the 99,5 percentiles.

Then all pixel values are normalised. Eventually, we use transfer learning. We apply well known models pretrained on the "ImageNet" data set. However, they did not use "fine tuning" because they only had 6000 images. Therefore, it could induce "overfitting". Then, the classification is made thanks to a ridge classifier. Lots of models ("ResNet50", "InceptionV3", "InceptionV4", "ShuffleNetV2", "VGG-19", "PolyNet", etc.) have been tried and give similar results with an error between 6% and 12%. No model clearly outperforms another one, because the mean accuracy of a model is often contained in the confidence interval of the other ones. These confidence intervals have been found thanks to cross validation.

Chapter 6
Strategy

We would like to detect all auroras even if we also detect images without aurora. Therefore, we will prefer to improve the recall rather than the precision.

The project will be divided into four main parts. The first one consists in preparing the data. The second one consists of the selection of the most appropriate classification model. The third step consists in testing the best models thanks to a data set from another season to see which images are misclassified. Eventually, I will try to apply the best models to homogeneous classes.

I will first explain the training and testing data sets preparation. We use the data set from the winter 2019-2020.

To improve the accuracy, the recall and the precision, we can classify into more than two classes. Then, the accuracy, the recall and the precision can be computed for two classes.

The available images which are labeled ("Aurora" or "No Aurora") also have labels which determine if there are clouds or not. Therefore, we can split the images into four classes : "Cloudy with auroras", "Cloudless with auroras", "Cloudy without aurora", "Cloudless without aurora".

In a same class, two images can be really different. Moreover, two images from two different classes can be similar. Indeed, the global appearance of an image depends on lots of parameters like the sunlight, the moonlight, the opacity of the clouds or the light pollution. The images vary during the season because of these previously mentioned parameters. In the same file (containing images with the same label), the images are in chronological order. The dilemma is to know if we have to mix the data before splitting them into training data and validation data. If the data samples are randomly mixed before being separated, the risk is to have very similar images in the training data set and in the testing data set. However, if the images are not mixed before the splitting step, the induced problem depends on the percentage of validation data. If we do not have enough training data, the risk is to train the model with a small variety of images. If we do not have enough images in the validation data set, the model will be validated on a small variety of images which is not representative. It can induce bad or good results, which do not have significance.

There are six minutes between the capture of each available image. Overall, it avoids having the same image in the training data set and in the validation data set. However, we still have some images which are very similar in these two data sets. This is due to the fact that some sky conditions can last for hours.

A possibility is to remove a part of images which are neighbours and have a high similarity index. Then, we can mix the data before splitting them. We will have to define an appropriate similarity

index and a threshold above which two images are considered as similar images. A compromise has to be found to delete similar images and keep enough images.

The previously described step will allow to remove lots of images from the data set. Then, we will check the labelling. For lots of images, the presence or absence of auroras is obvious. However, for other images, the auroras are barely perceptible. These images are labeled "Aurora" or "No aurora". According to me, some images which have been classified as "No aurora" images clearly contain auroras. However, some images which have been classified as "Aurora" images contain auroras which are very difficult to see.

I can choose to place all images containing auroras which are hardly perceptible in the file "Aurora" or in the file "No aurora". I can also create a third label containing the images which are difficult to classify. However, if we consider that it is difficult to decide if an image contains an aurora or not, it is more difficult to define a threshold between a clearly visible and a barely perceptible aurora. Therefore, I won't consider this option. I prefer to place all images containing auroras in the same file.

To prepare the testing data set, I just need to split the data from January 2022 into four classes and to check the labels. This data set contains images which can have different aspects from the images in the training and validation data set. However, even if we tried to have a representative data set, there are probably a lack of representation of certain types of images.

During the second step, I will try to find the most appropriate classification model. Because the data set from January 2022 was available at the end of the project, I used the validation data set to look for the best models. I will first focus on the methods that do not include CNN. Then, I will try to use CNN models.

I will select a dimension reduction and a classification method and compare different combinations of preprocessing and features extraction methods. The following phase consists in selecting a Convolutional Neural Network (CNN). We will use the fine tuning and try different parameters combinations.

Once the most promising models are found, we can test their robustness thanks to the testing data set and see which types of images are misclassified.

Despite the subdivision of each class into sub classes "Cloudy" or "Cloudless", we see that the images within a same class can have very different aspects while two images from two different classes can look very similar (Figure 6.1). I want to avoid that a feature often but not systematically contained in the images of a given class becomes the only criterion of classification. It could improve the global accuracy on the training data set but ignore important features and give poor results on a different data set. As previously explained, the weather, the moonlight, the sunlight and the light pollution have a huge impact on the images. Therefore, I will try to divide the four available folders into homogeneous sub folders. To do that, I rely on the histograms of each image.

Once we obtain the new classification, we can apply the classification methods and see if it offers better results. Even if it is not the case, it will help see what images are difficult to classify.

The models are loaded and the parameters and the performances are written in a text file. The recording time is contained in the model's name and in the text file. It allows to link the model

with its associated parameters and accuracy. To change and test different methods, I choose to organise the program into classes.

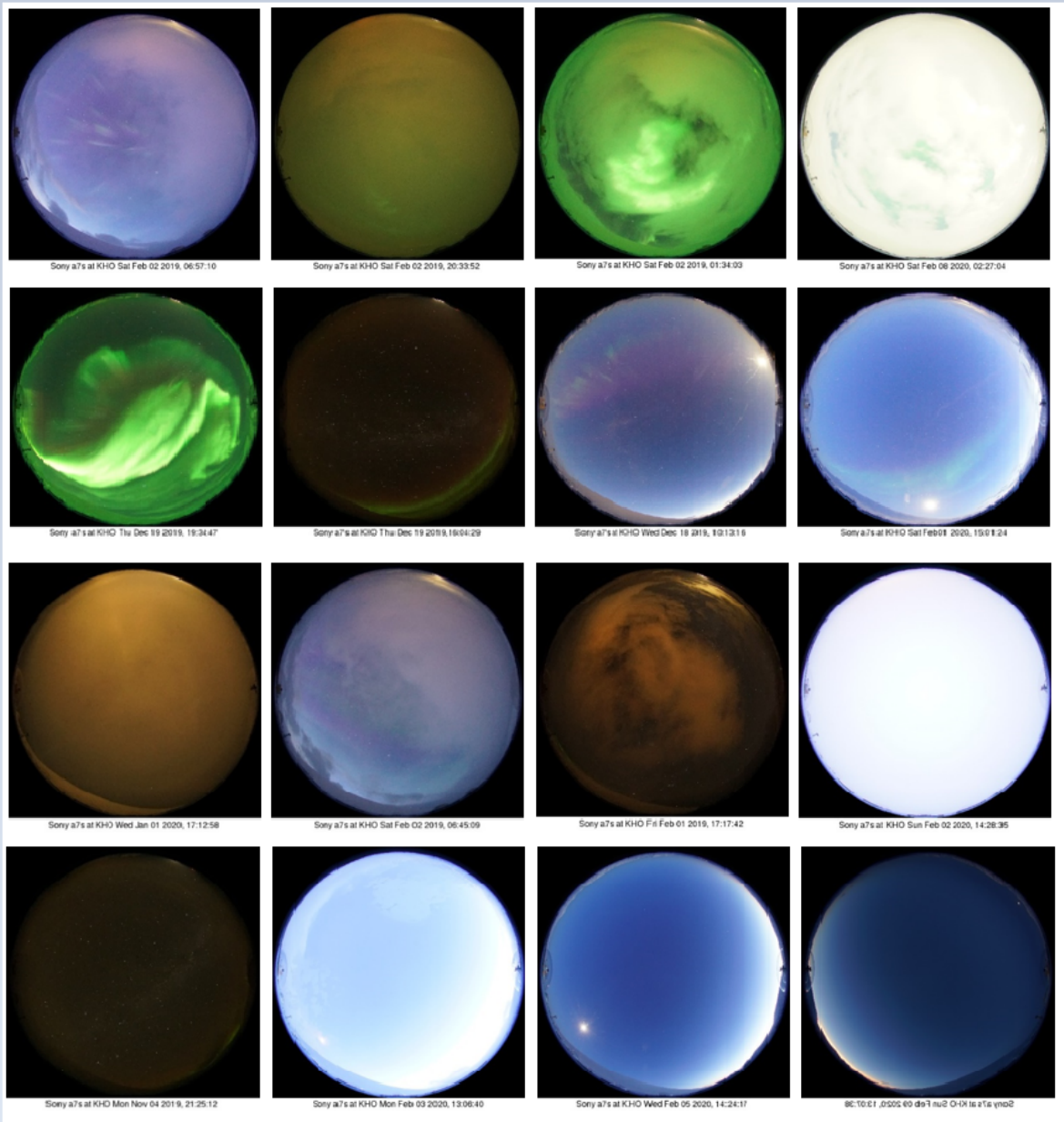

Figure 6.1: All-sky images: Images containing clouds and auroras on the first row; Images containing auroras but no clouds on the second row; Images containing clouds but no aurora on the third row; Images containing no aurora and no cloud on the last row

Chapter 7
Implementation

7.1 Data preparation

To create the training and the validation data sets, I have access to folders containing more than 41800 images with the labels "Aurora" or "No aurora". These images have been taken during the winter 2019-2020. I have also access to more than 37300 of these images in another folders where they are labeled "With clouds" or "Without clouds". Therefore, I first create four new folders containing the images of the same category : "Cloudy with auroras" (containing 1095 images), "Cloudless with auroras" (containing 5319 images), "Cloudy without aurora" (containing 18519 images), "Cloudless without aurora" (containing 7893 images).

Then, I remove images which are neighbours and have a high similarity rate. To assess the similarity, I compute the Structural Similarity Index (SSIM) [30] between two images which are neighbours. I first compute the SSIM of the two first images. If this index is higher than a previously chosen threshold, we consider that the images are similar. Then, we can remove the second image and compare the first and the third image. If the two compared images are not similar, the third image is not removed and becomes the reference image. Then, we can compare the third image to the fourth one. A compromise has to be found to delete similar images and keep enough images. If we apply a low pass filter before computing the SSIM, we will obtain a higher value. If we compare two dark images with stars located at different places, we will obtain a low value and consider these images as different images. However, if we apply a low pass filter, we cannot keep enough data to train the models. Choosing the threshold for which two images are considered as similar images also requires to make a compromise between keeping enough images and removing similar images. I choose not to apply a low pass filter and to set the threshold at $0,9$. After this step, there are only 13671 images. There are 751 images with the label "Cloudy with auroras". 4903 images have the label "Cloudless with auroras". There are 4080 images contained in the folder "Cloudy without aurora". 3937 images have the label "Cloudless without aurora".

Then, I check the labels of each image. As previously explained, I put all images containing hardly noticeable auroras in the classes "Cloudy with auroras" or "Cloudless with auroras". Eventually, I obtain 896 images in the class "Cloudy with auroras", 5297 images in the class "Cloudless with auroras". There are 3741 images with the label "Cloudy without aurora" and 3542 images with the label "Cloudless without aurora".

To create the testing data set, I use the 14775 images from January and February 2022. I split the images into four classes : 992 images are labeled "Cloudy with auroras", 7286 have the label "Cloudless with auroras", 4622 images are "Cloudy without aurora", and 1874 "Cloudless without

aurora". I do not need to remove the similar images. The testing data set will enable to assess the trained model on a completely new data set and to see if some images are difficult to classify.

7.1.1 Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) is a way to compare two images [30]. The SSIM allows to extract and compare luminance, structure and contrast. It produces a value between 0 and 1. A value close to 0 indicates that two images are very different. A value close to 1 indicates that two images are similar to each other.

Let x be an image :

The luminance corresponds to the average over all pixels.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$
$$with\ x_i\ :\ the\ pixel's\ value$$

The contrast is computed on the basis of the standard deviation of the pixels values.

$$\sigma_x = ((\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}})$$

The structure is computed as follows :

$$st_x = \frac{x - \mu_x}{\sigma_x}$$

The function of luminance comparison is computed thanks to the luminance measure :

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$
$$with\ C_1 = (K_1 L)^2$$
$$L = 255\ :\ The\ dynamic\ range\ for\ the\ pixel\ values$$
$$K_1\ and\ K_2\ are\ constants$$

The function of contrast comparison is computed thanks to the contrast measure :

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$
$$with\ C_2 = (K_2 L)^2$$

The function of structure comparison is computed on the basis of the structure measure :

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

$$with\ C_3 = \frac{C_2}{2}$$

$$and\ \sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y)$$

The SSIM is computed on the basis of the three previously described functions :

$$SSIM(x,y) = [l(x,y)]^{\alpha} [c(x,y)]^{\beta} [s(x,y)]^{\gamma}$$

$$with\ \alpha > 0\ ,\ \beta > 0\ ,\ \gamma > 0$$

These values correspond to the relative importance of each metric. To simplify, we set $\alpha = \beta = \gamma = 1$.

Eventually, the SSIM corresponds to the following expression :

$$SSIM(x,y) = \frac{(2\mu_x \mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

The previously described metric is global. We can compute the Mean Structural Similarity Index (MSSIM). We define a gaussian window with a size of 11 that we can shift pixel by pixel across the image.

The local statistics are computed as follows :

$$\mu_x = \sum_{i=1}^{N} w_i x_i$$

$$\sigma_x = (\sum_{i=1}^{N} w_i (x_i - \mu_x)^2)^{\frac{1}{2}}$$

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y)$$

$$with\ w_i\ :\ the\ gaussian\ function$$

The MSSIM corresponds to the following expression :

$$MSSIM(x,y) = \frac{1}{M} \sum_{j=1}^{M} SSIM(x_j, y_j)$$

7.2 Selection of the models

To select the classification models, I implement a program organised in different classes. In this diagram (Figure 7.1), we do not consider the fist step consisting in preparing the data.

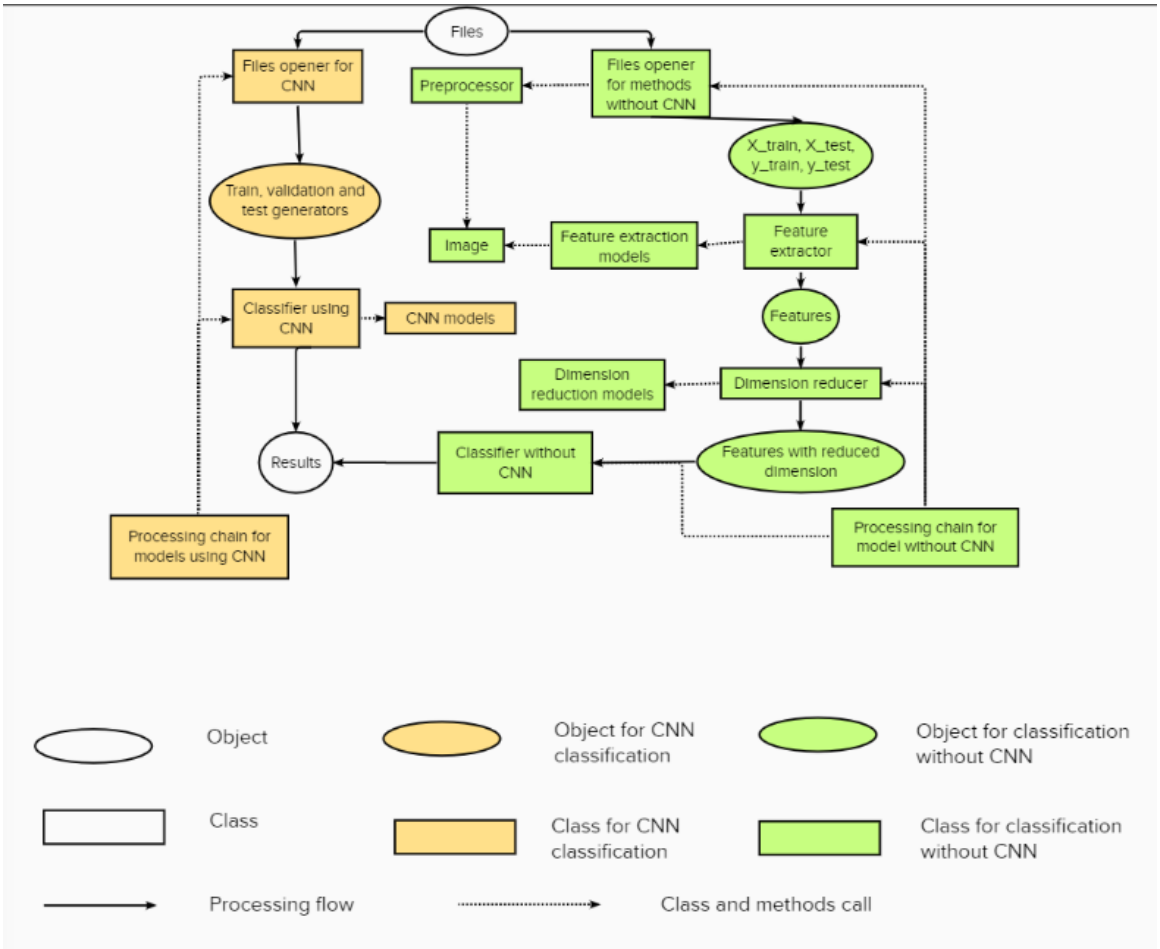

Figure 7.1: Diagram of the Python code

A part of the classes is used to find a combination of preprocessing, features extraction, dimension reduction and classification methods. The images are extracted from the files thanks to the class "Files opener for CNN", which uses the class "Preprocessor" to choose the preprocessing step. The class "Preprocessor" allows to choose a combination of preprocessing steps. Each single preprocessing step is a method written in the class "Image". The output of the class "Files opener for methods without CNN" consists of the variables "X train, X test, Y train and Y test". It corresponds to the images and the associated labels of the training data set and the testing data set. Then, the feature extraction is allowed thanks to the class "Feature extractor". This class calls methods from the class "Feature extraction models" which, in turn, call methods from the class "Image". Then, we can apply the dimension reduction to the extracted features thanks to the class "Dimension reducer". Eventually, the obtained features can be classified by methods contained in the class "Classifier without CNN". The file "Processing chain for models without CNN" allows to apply the previously described processing chain.

The second part of the classes allows to apply CNN. We can adapt the features extraction to put more data in the training, validation and testing data sets thanks to the "tensorflow" library. The variables "Train, validation and test generators" are extracted by the class "Files opener for CNN". Then, the class "Classifier using CNN" allows to train and test the chosen models. The processing chain is applied in the file "Processing chain for models using CNN".

7.2.1 Selection of methods without using CNN

Because the context is often not the same as those in the previously described articles, we cannot directly compare the obtained results. Indeed, we deal with a binary classification (With or without aurora) on colored images containing clouds, moonlight, sunlight or artificial light. In some articles, the available images were gray scale images.

I will compare different preprocessing steps and features extraction methods. I choose to remove noise by subtracting the mean value of the black pixels contained in the corners. I also choose not to apply an algorithm which improves the contrast as preprocessing step to avoid revealing irrelevant features. To compare the methods, I will choose a dimension reduction algorithm and a classifier.

The chosen dimension reduction is the UMAP algorithm because this method is theoretically efficient and does not require to choose lots of parameters. The minimum distance between two points in the new space is fixed to 0 and the number of points considered as neighbours of each pixel is fixed to 100.

We also choose the random forest as a classifier for its efficiency. Moreover, unlike the neural networks, it does not require optimizing parameters.

There are various features extractors that we could apply on different color channels. For each features extraction, it is possible to use different preprocessing steps. Therefore, it is impossible to try to apply all the potential combinations. That is why we will first focus on some of the features extractors that have been already tried. We will keep the methods which offered the best results. However, we cannot compare our results to the results obtained in the previously described articles. Indeed, we are not in the same context and we do not apply the same preprocessing methods, dimension reduction algorithms and classification methods.

As previously mentioned, the following features extraction methods have been tested :

- global statistical descriptors on gray scale images,

- local statistical descriptors ($3 \times 6 = 18$ area) on gray scale images,

- the north-south and east-west brightness distributions on gray scale images,

- the SIFT algorithm on a combination of the three RGB channels (2G-R-B),

- the SIFT algorithm on the normalized RGB system,

- the SIFT algorithm applied to the green channel and combination (2G-R-B),

- the SIFT algorithm applied to the Opponent colors system,

- the OSID algorithm by dividing the gray scale image into three angular sections,

- the OSID algorithm by dividing the gray scale image into seven angular sections,

- the BGLAM applied to the gray scale images,

- the global OLBPC applied to the RGB channels,

- the global LBP applied on gray scale images,

- the local LBP applied on gray scale images ($3 \times 6 = 18$ area),

- the Gabor wavelet decomposition on gray scale images.

The local statistical descriptors gave better results than the global ones. Therefore, we can try to compute local statistical descriptors on RGB channels, HSV channels or Lab channels [20].

I will first compute local statistical descriptors on the RGB, HSV and Lab channels. I will use the polar coordinate system to remove the black corners and to keep all other pixels of the image. Then, I will divide the images into $10 \times 10 = 100$ areas, and compute the standard deviation, the minimum, maximum, and mean values for each channel and each area.

Before computing local statistical descriptors, we can remove the noise by applying a low pass filter.

The East-West and North-South brightness distributions have not be compared to other features extractors. We can also try to compute other type of distributions. Because we have colored images, we can try to compute the distributions for different channels (RGB, HSV or LaB) [23]. We choose to use a features extractor which consists in computing the brightness distribution along the rows and the columns of the image in polar coordinates. It allows to give the corresponding distributions along the concentric circles and the radius of the images.

Before computing the value or hue distribution, we can also apply a low pass filter to the image to remove the noise.

The SIFT algorithm applied to the normalized RGB channels and to the Opponent color system enabled to obtain especially good accuracy [14].

I will apply the SIFT algorithm to the normalized RGB channels.

If we use this extractor, it is better to use the cartesian coordinate system (and not the polar one). Indeed, the SIFT extractor considers the direction of the features. Moreover, we cannot apply a median or Gaussian filter before the features extraction. Indeed, the SIFT algorithm is based on the features extraction at different scales and different orientations. Therefore, if we first apply a low pass filter, we cannot keep the information contained in the high frequencies.

If we apply the SIFT extractor to two different images, we will not detect the same number of features in each image. Thus, we will not obtain the same number of vectors for all images. To solve this problem, we can use the Bag of Words (BoW). The main idea is to keep all the vectors from all images and to create clusters of these images. Then, for each image, we can create a histogram counting the number of features (vectors) contained in each cluster.

To compute the SIFT extraction, we can use the "sift" method from the library "cv2".

The OSID on gray scale images enables to obtain similar results if the images are divided into three or seven images [20].

To extract features, we can apply the OSID algorithm to different channels of the color system RGB, to the channels H and V (Hue and Value) in the HSV system, or to the channels of the color system Lab.

The first step consists in applying a Gaussian filter. The second step consists in binning the image, which improves the contrast. Therefore, these two steps are not considered for the preprocessing phase.

The local LBP allows to obtain better results than the global LBP. Then, because we have colored images, we choose to apply local OLBPC algorithms on the RGB channels [20].

The Gabor wavelet decomposition has not been compared to the other features extractors [23].

The recall, the accuracy and the precision are equal to 1 when these methods are applied to the training data set. Therefore, we are sure that the learning phase works well. I applied the methods to the validation data set. The obtained results are mentioned in the following table (Table 7.1):

	RGB	HSV	Lab
Local statistical descriptor (10 x 10)	P = 78 %, R = 70 % A = 77 %	P = 76 %, R = 68 % A = 75 %	P = 77 %, R = 71 % A = 77 %
Values distribution of each channel	P = 78 %, R = 74% A = 79 %	P = 72 %, R = 68 % A = 73 %	P = 79 %, R = 74 % A = 79 %
Method based on OSID	P = 71 %, R = 73 % A = 74 %	P = 71 %, R = 72 % A = 74 %	P = 72 %, R = 74 % A = 75 %
Local OLBPC	P = 76 %, R = 74 % A = 77 %	P = 79 %, R = 74 % A = 79 %	P = 75 %, R = 73 % A = 76 %
BoW based on SIFT descriptor	Precision, recall and accuracy inferior or equal to 50 %	Ø	Ø

P : precision, R : recall, A : accuracy

Table 7.1: Results obtained by applying features extraction algorithm on the validation data set

When I tried to compute local statistical descriptors on different channels. The chosen channels did not impact the results and the obtained accuracy was around 75% while the recall was around 70%. Then, I computed the values distributions of the different channels along the row and columns after polar transformation. I obtained better results by applying this feature extractor to the RGB and to the Lab channels. The obtained accuracies were around 80% and the recalls around 75%. Applying this extractor to the HSV channels gave an accuracy of 73% and a recall inferior to 70%. The method based on OSID gave an accuracy of 75% whatever the chosen channels, and a recall between 70 *and* 75%. The local OLBPC provided an accuracy between 75 *and* 80% and a recall around 75%. The colour channels did not have any impact. However, the method based on SIFT was not efficient at all. Indeed, even if this method could be efficient, it requires to use

BoW, and therefore, we need to optimize lots of parameters. Therefore, we could conclude that the most appropriate feature extractors were the OLBPC applied on local areas of the images, and the values distributions of each channel of the RGB and Lab colour systems.

After choosing a feature extraction, if we had more time, we could also optimize the reduction of dimension and the classification algorithm.

7.2.2 Selection of the CNN models

Because I get the testing data set late in the project, I optimized the CNN model on the basis of the validation data set.

The implementation and the training of a model from the beginning requires to choose lots of parameters (Number of filters, size of the window, number of layers, number of neurons in each layer, etc.) or to work from a well-known model architecture. The pretrained version of such models is often available in "Keras" library. Moreover, we know that the use of pretrained CNN models offered better results for detection and classification of auroras and for lots of other image classification tasks. Therefore, we suppose that the "transfer learning" can allow to achieve better results than if we create and train a model from the beginning. It has been used and detailed in a previously mentioned article [16]. The features extraction was performed by pretrained CNN models and the classification was allowed by a ridge classifier. However, the "fine tuning" was not used. All the tested models offered similar results. Indeed, the cross validation allowed to find the mean accuracy and an uncertainty range. The data set is split into training and validation data set before training the model and saving the accuracy. This step is reproduced many times, while using different splits of the data each time. Then, the different accuracy values are saved as a mean accuracy and an uncertainty range. We can notice that even if a model gives a better mean accuracy than another one, each model's mean accuracy is contained in the uncertainty range of many other model. With the "fine tuning", this result could be investigated.

First, we can choose one of the models and compare various neural networks combinations to the output of the CNN model to classify the images. We use "fine tuning" with a low learning rate to avoid removing all information contained in the pretrained model. Then, we reproduce the experiment with another pretrained model. This approach allows to optimise the possible classification accuracy for a given pretrained model and to see if different models have the same behavior according to the final neural networks combination.

Nevertheless, because the "fine tuning" and the optimisation of neural networks require a long time (around ten hours to train a model), it will be impossible to test more than two pretrained CNN models. It will also be impossible to use cross validation for the chosen models and therefore, to know the uncertainty ranges of the accuracy. However, we will still see if some models give significantly better results than other models. We will also be especially careful not to select models which do not assign one of the labels. It means that, even if the model gives a good accuracy, we cannot rely on it. Indeed, if the testing data set includes a very small number of data with this label, the result can be improved by simply ignoring this label. Nevertheless, it annihilates all possibilities to classify this type of data.

We choose the batch size equal to 32 for the next steps. The number of training epochs is fixed to 10 to allow to test enough models. Indeed, each training epoch requires around one hour to

process. Moreover, we often see that the model does not improve after the fifth epoch. We will systematically keep the state of the model at the epoch with the best accuracy on the validation data set. It enables to avoid "overfitting".

For now, the learning rate is fixed to 0.0001. It should enable a slow but precise learning. It avoids removing all information contained in the pretrained model. We choose the "ResNet50" and "InceptionV3" models pretrained on the data set "ImageNet" and we add various combinations of neural network to the output. The chosen activation function between each layer is "ReLU" and the final one is "Softmax". Indeed, "Softmax" is adapted to the multiclass classification. I choose to use the optimizer "Adam". Moreover, because we do not have the same number of images in all classes, we adapt the weight of the difference between the real label and the predicted label according to the number of images contained in the class. If there are lots of images in a class, the associated weights of the errors will be low.

The classification is made into four classes and we record the results obtained on the validation data set by combining the classes with auroras for their part, and classes without aurora in the second class (Table 7.2).

For the majority of neural network combinations, we do not see big differences between the "ResNet50" and the "InceptionV3" models. However, the results can also be really different for a same neural network to the output according to the chosen pretrained model. Therefore, if we apply "fine tuning", the obtained results depend on the chosen pretrained model, and if we had more time, we could continue trying other models.

We notice that if the recall is exceptionally high, the precision is low, and that if the precision is high, the recall is exceptionally low. There is an anti-correlation through the extreme values. It happens if the model chooses to put almost all difficult-to-classify images in the same class. If these images are put in the class of the images containing auroras, the recall is high. If these images are put in the class of images containing no aurora, the precision is high.

We clearly see that the most appropriate regularisation function is "L2". When the regularisation rate is fixed to 0.03 and the dropout rate is fixed to 0.3, we notice that the best accuracies and recalls are obtained with two layers containing between 100 and 200 neurons. The models containing three layers with the same or different sizes often offer poor results. If I choose to keep two layers of 150 neurons, I cannot see a clear trend in the results when I change the regularization rate. The dropout rate does not change the results much either. The obtained accuracies and recalls are close to each other for the majority of the combinations.

To optimise parameters, we could use a Bayesian optimiser. Nevertheless, it would take lots of time and would not improve the results. Indeed, in our case, some parameters are not real values (type of regularization function for instance) and have to be selected in the beginning. Then, there are still lots of parameters to optimise, and each parameters can correspond to an infinite number of possible values. Therefore, a Bayesian optimisation would require an extremely long time to give interesting results. Moreover, it seems that we reach an accuracy which is difficult to exceed and that the differences between most of obtained accuracies are not significant. Even if a model gives a better accuracy than the other ones, its mean accuracy would probably be contained in the uncertainty range of some other models if we had time to proceed to the cross validation. Therefore, the optimiser could over-analyze accuracy differences between two models and try to

Regularisation	Dropout rate	Layers description	ResNet50	Inception V3
L1 ($\alpha = 0.03$)	0.3	2 layers of 150 neurons	Only one label is given «Cloudy without aurora »	Only one label is given «Cloudy without aurora »
L2 ($\alpha = 0.03$)	0.3	1 layer of 200 neurons	P = 79, R = 90, A = 86	P = 85, R = 92, A = 90
		2 layers of 100 neurons	P = 84, R = 94, A = 90	P = 89, R = 93, A = 92
		3 layers of 50 neurons	Never attributes the label « Aurora and clouds »	P = 91, R = 88, A = 91
		2 layers of 150 neurons	P = 93, R = 87, A = 92	P = 91, R = 92, A = 93
		1 layer of 200 neurons and one layer of 100 neurons	P = 89, R = 74, A = 86	P=86, R = 94, A=91
		1 layer of 150 neurons, 1 layer of 100 neurons and a layer of 50 neurons	P = 95, R = 60, A = 82	P = 89, R = 88, A = 90
		1 layer of 50 neurons, 1 layer of 100 neurons and 1 layer of 150 neurons	P = 94, R = 81, A = 90	P = 83, R = 87, A = 87
L2 ($\alpha = 0.1$)	0.3	2 layers of 150 neurons	P = 88, R = 92, A = 91	P = 87, R = 91, A = 91
L2 ($\alpha = 0.5$)			P = 88, R = 90, A = 91	P = 91, R = 91, A = 93
L2 ($\alpha = 1$)			P = 88, R = 92, A = 92	P = 81, R = 96, A = 89
L2 ($\alpha = 0.5$)	0.5	2 layers of 150 neurons	P = 91, R = 92, A = 93	P = 95, R = 86, A = 92
	0.7	2 layers of 150 neurons	P = 88, R = 95, A = 92	P = 82, R = 96, A = 90
		2 layers of 200 neurons	P = 89, R = 93, A = 93	P = 89, R = 92, A = 92
	0.8	2 layers of 300 neurons	P = 89, R = 90, A = 92	P = 95, R = 82, A = 91
L1 and L2 ($\alpha = 0.03$)	0.3	2 layers of 150 neurons	Only one label is given «Cloudy without aurora »	Only one label is given «Cloudy without aurora »

P : precision, R : recall, A : accuracy, α : regularization rate

Table 7.2: Results obtained by applying different CNN on the validation data set (Results given in percent)

find a maximum value of a "n-dimensional function" where the noise has a higher range than the actual variations of the "n-dimensional function" itself. Moreover, it does not consider if a label is ignored.

Because the methods with CNN give better results, I will focus on the testing phase of the CNN models.

7.3 Testing the CNN models

I will apply the previously described models on a new testing data set which contains images from another season (Table 7.3).

On the testing data set, we notice that the results are not as good as the results obtained with the validation data set. This can be solved by using a bigger training data set. The best models reach a recall and an accuracy between 85% and 90%. Unfortunately, the precision is always better than the recall. Therefore, we will more probably miss auroras.

Regularisation	Dropout rate	Layers description	ResNet50	Inception V3
L1 ($\alpha = 0.03$)	0.3	2 layers of 150 neurons	Only one label is given «Cloudy without aurora »	Only one label is given «Cloudy without aurora »
L2 ($\alpha = 0.03$)	0.3	1 layer of 200 neurons	P = 84, R = 82, A = 81	P = 99, R = 76, A = 86
		2 layers of 100 neurons	P = 90, R = 85, A = 86	P = 93, R = 82, A = 86
		3 layers of 50 neurons	P = 94, R = 66, A = 79	P = 92, R = 79, A = 84
		2 layers of 150 neurons	P = 97, R = 75, A = 85	P = 99, R = 78, A = 87
		1 layer of 200 neurons and one layer of 100 neurons	P = 95, R = 61, A = 76	P = 95, R = 81, A = 87
		1 layer of 150 neurons, 1 layer of 100 neurons and a layer of 50 neurons	P = 100, R = 40, A = 66	P = 91, R = 79, A = 84
		1 layer of 50 neurons, 1 layer of 100 neurons and 1 layer of 150 neurons	P = 99, R = 73, A = 84	P = 95, R = 85, A = 89
L2 ($\alpha = 0.1$)	0.3	2 layers of 150 neurons	P = 94, R = 86, A = 89	P = 95, R = 84, A = 89
L2 ($\alpha = 0.5$)			P = 94, R = 86, A = 89	P = 98, R = 78, A = 86
L2 ($\alpha = 1$)			P = 95, R = 84, A = 88	P = 91, R = 86, A = 88
L2 ($\alpha = 0.5$)	0.5	2 layers of 150 neurons	P = 96, R = 81, A = 88	P = 97, R = 73, A = 84
	0.7	2 layers of 150 neurons	P = 94, R = 82, A = 87	P = 92, R = 87, A = 88
		2 layers of 200 neurons	P = 96, R = 84, A = 89	P = 96, R = 80, A = 87
	0.8	2 layers of 300 neurons	P = 93, R = 81, A = 86	P = 98, R = 77, A = 86
L1 and L2 ($\alpha = 0.03$)	0.3	2 layers of 150 neurons	Only one label is given «Cloudy without aurora »	Only one label is given «Cloudy without aurora »

P : precision, R : recall, A : accuracy, α : regularization rate

Table 7.3: Results obtained by applying different CNN on the testing data set (Results given in percents)

The regularization function "L2" also gives exploitable results with the testing data set. If I choose a regularization rate at 0.03 and a dropout rate at 0.3, the recall does not exceed 85%. The best accuracies and recalls are obtained for two layers of 100 neurons and with the "InceptionV3" model associated to three layers, one with 50 neurons, the second one containing 100 neurons, and the last one containing 150 neurons. Like for the validation data set, we do not clearly see a trend in the results when we change the dropout rate and the regularization rate. However, if we consider the recall and the accuracy, the "ResNet50" model associated to a dropout rate at 0.3 regularization rate at between 0.1 and 0.5 with three layers enable to achieve a very good precision (between 90% and 100%) but a poor recall (between 40% and 85%). The "InceptionV3" model associated to two layers of 150 neurons, a dropout rate at 0.7 and a regularization rate at 0.5 enables to achieve the best recall (87%) and a good accuracy (88%).

Even if the differences are more clearly visible on the testing data set, we also seem to reach an accuracy and a recall difficult to exceed. The best accuracies and recalls are also close to each other.

If I consider the confusion matrix of the best models (Tables 7.4 and 7.5), I clearly see which classes are confused.

	True class				
Predicted class		« Auroras and clouds »	« Auroras without cloud »	« Clouds and no aurora »	« No aurora and no cloud »
	« Auroras and clouds »	551	2490	164	266
	« Auroras without cloud »	0	3533	0	43
	« Clouds and no aurora »	441	648	4454	907
	« No aurora and no cloud »	0	615	4	658

Table 7.4: Confusion matrix of the pretrained model "ResNet50" associated to a classifier composed of two layers of 15O neurons, a dropout rate of 0.3 and a regularization “L2” with a rate equal to 0.5

	True class				
Predicted class		« Auroras and clouds »	« Auroras without cloud »	« Clouds and no aurora »	« No aurora and no cloud »
	« Auroras and clouds »	653	735	205	132
	« Auroras without cloud »	33	5776	0	249
	« Clouds and no aurora »	302	305	4412	630
	« No aurora and no cloud »	4	470	5	863

Table 7.5: Confusion matrix of the pretrained model “InceptionV3” associated to a neural network of two layers of 150 neurons, a dropout rate of 0.7 and a regularization “L2” with a rate equal to 0.5.

The images containing auroras and clouds are often considered as images containing clouds but no aurora. The clouds prevent the detection of auroras. The images containing auroras but no clouds are mainly considered as images containing auroras and clouds. I think that it is due to the fuzzy threshold between images containing clouds and clear images. Hopefully, it does not impact the detection of auroras. The images containing clouds and no aurora are mainly well classified. The images containing no aurora and no cloud are considered as images containing no aurora but clouds. Like for the images with auroras, the threshold between images containing clouds and images containing no cloud is difficult to define. A sky is considered as cloudy if it is mainly cloudy in the centre of the image or cloudy enough to obscure the detection of auroral structures. It does not disturb the detection of auroras.

Next, I will visualize which images are misclassified by the best models. As we can see in Table 7.3 the best “ResNet50” models were those with two layers of 150 neurons, a dropout rate of 0.3 and

a regularization “L2” (with a regularization rate equal to 0.1 or 0.5). The “InceptionV3” model which offers the best accuracy and the best recall is the one with a neural network of two layers of 150 neurons, a dropout rate of 0.7 and a regularization “L2” with a rate equal to 0.5. We can see the images which contain auroras but are classified as images which do not contain aurora. These images are considered as false negatives (Figure 7.2). We can also see the images which do not contain any aurora but are classified as images with auroras. These images are false positives (Figure 7.3).

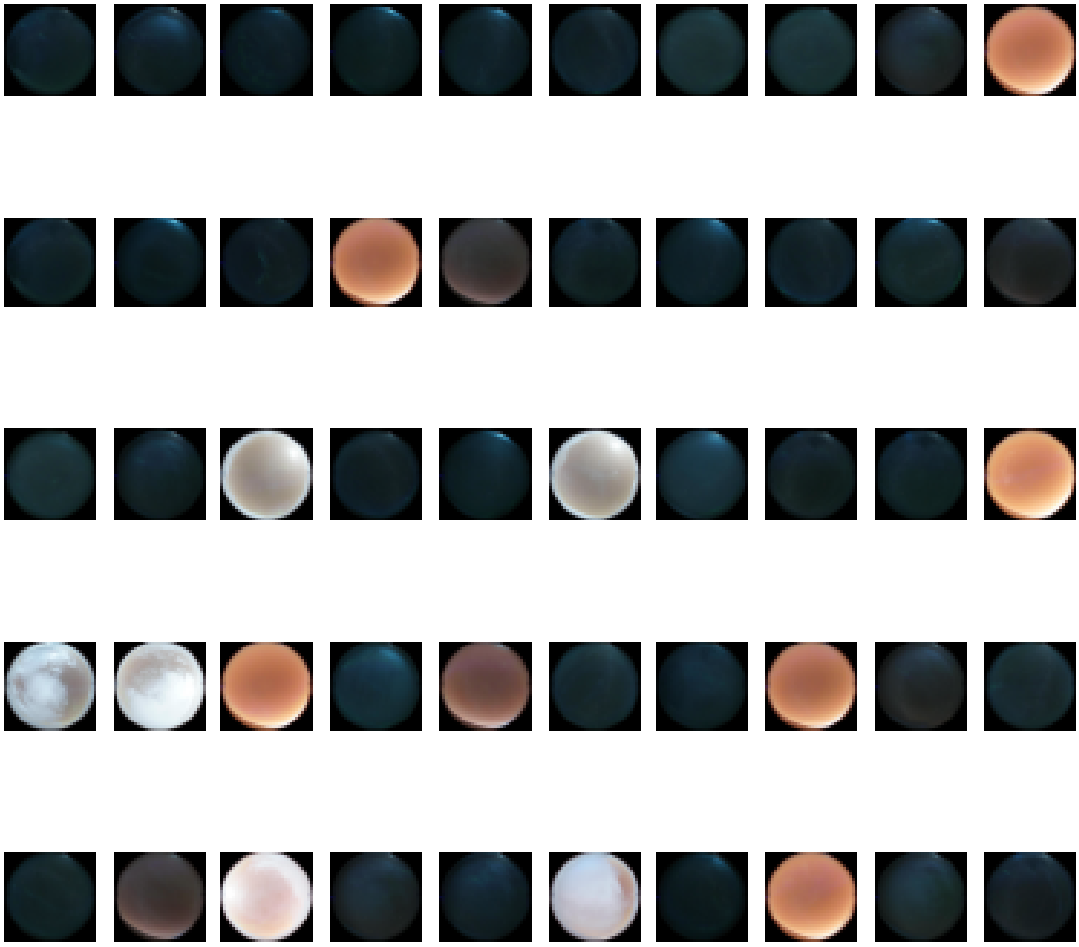

Figure 7.2: False negatives

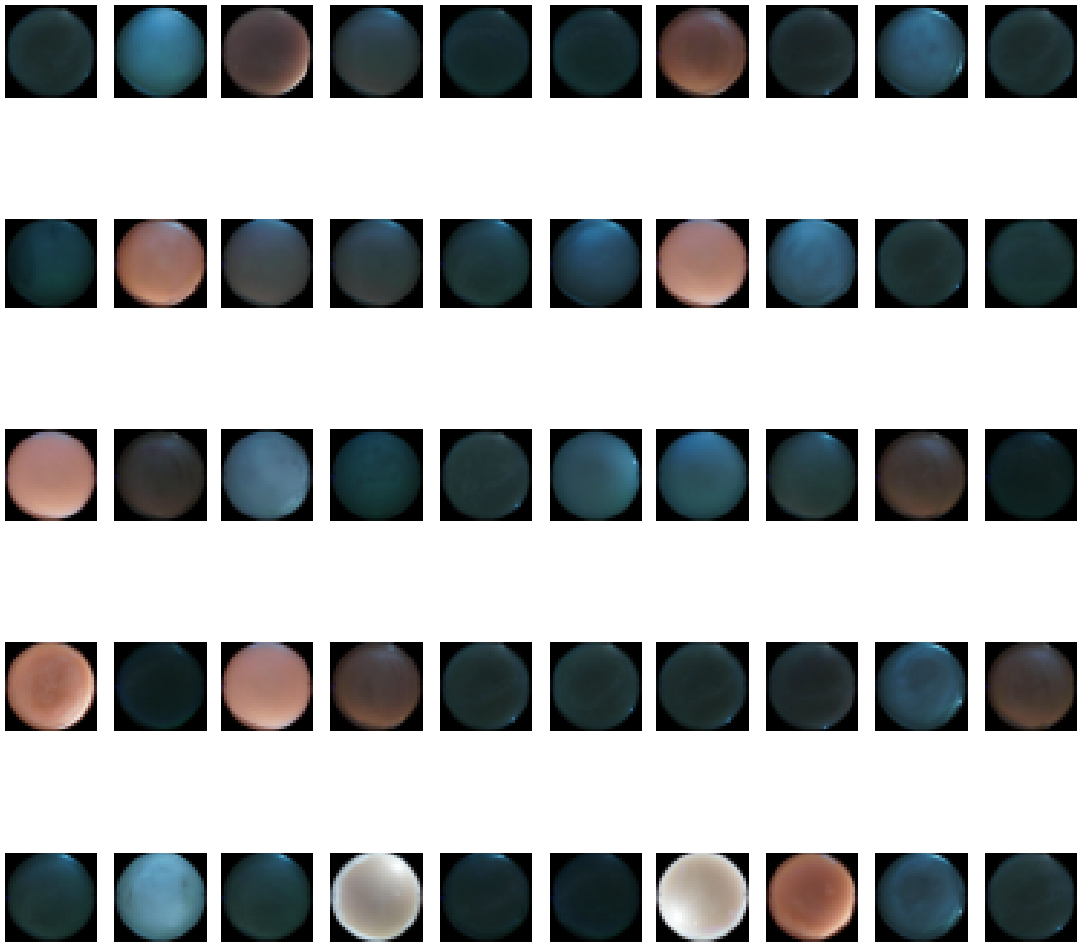

Figure 7.3: False positives

We can notice that the misclassified images are dark blue or gray images. They contain clouds, moonlight or faint auroras. The clouds and moonlight can create light arcs on the border of an image which can be confused with the features created by an aurora. The false negatives contain only faint auroras which are difficult to detect even for a human. We previously decided to label all images containing faint auroras as "Images containing auroras". It could explain the difficulty to obtain a better recall. Eventually, we can conclude that many of the images containing auroras and images which do not contain aurora have similar aspects.

Eventually, we can try to apply one of the best CNN models to the homogeneous sub classes obtained on the basis of the HSV histograms. Indeed, the background sky illumination covers the whole field of view and affects the colour of the observed auroras. Because a majority of images in a given class can contain the same colors, the models could base the classification only on the colors. For example, there are lots of green pixels on black backgrounds in the majority of images containing auroras. Therefore, the models could misclassify the few images containing red auroras or auroras in cloudy skies, and achieve good results. This new classification could improve the results and will enable to see which images can be easily misclassified.

7.4 Applying CNN models to homogeneous sub classes

To create sub classes where images have similar appearances, I rely on the HSV histograms. I compute the mean of each channel (H,S and V), the standard deviation and the most represented value of each channel. Each image is associated with the vector containing these previously described descriptors linked to the histograms. Then, for each class ("Cloudy with auroras", "Cloudless with auroras", "Cloudy without aurora" and "Cloudless without aurora"), I apply a dimension reduction method (UMAP) and a clustering algorithm (K-means) to create sub classes. If I choose to divide each file into less than ten sub classes, the images contained in a same sub class can have very different appearances. To avoid that some groups mix images with too different appearances, I choose to divide each class into thirty groups. Then, we can aggregate the obtained groups to obtain three, four or five sub classes.

Unfortunately, the classes (Annex E) based on the histograms have major disadvantages. Indeed, some classes contain much more images than other ones. Moreover, the distribution of images among the different classes is complicated. Indeed, if we choose to put in the same class all images of blue skies without aurora and cloud, dissimilar images will be contained in the same class. However, if we decide to divide these images into two groups (for light blue and dark blue images), the threshold between the two groups is difficult to define. Therefore, I try to adapt the aggregation to reduce the inequalities among the number of images contained in the classes. However, despite these precautions, one of the sub class contains less than two hundred images and another one contains almost two thousand images. Once this new labelling has been done, we obtain twenty-one sub classes spread across the four original classes. Even if it fails to improve the results, this new classification can allow to understand which images are difficult to classify.

In the class "Cloudy with aurora", there are only 896 images. Therefore, it is impossible to split this folder into lots of different ones. Therefore, we cannot separate the 172 light blue and dark blue images which are placed in the same sub folder. We have the same problem for the 147 light gray and dark gray images which are contained in another sub folder. 577 dark and brown images are contained in the third sub folder. It is impossible to define a threshold between the dark and the brown images.

In the class "Cloudless with auroras", there are 5297 images. Therefore, we need to subdivide this class into many groups. There are 1875 dark images with imperceptible auroras and 1124 images with dark sky and clearly visible green auroras. It was especially difficult to define a threshold between these two types of images. However, a single class for all these images would have too much elements compared to the other sub classes. One of the sub folders contains 903 light and dark gray images. There are not enough light gray images to create two sub folders. The last class contains 710 light and dark blue images. As for the previous class, there are not enough light gray images to create two folders.

The class "Cloudy without aurora" contains contains 3741. I decided to create seven different sub classes. There are 968 black and dark brown images. I created another class for the light brown images even if the threshold between them and the dark brown images was difficult to define. 713 light and dark gray images are contained in the same folder because there are not enough light gray images to create two different classes. Then, there are three groups with blue images. The first one contains 218 dark blue images. The second one contains 574 light blue images. Even if

the threshold was difficult to define, there is a great difference between the lightest and the darkest images. The third group contains 144 light blue images which can also be considered as white images. Eventually, 656 images have a color between the gray and the blue.

The class "Cloudless without aurora" contains 3937 images which I divided into six different classes. There are 777 gray images and 378 dark gray images. There are 1539 black or almost black images. There are also 264 dark blue images, 151 light blue images and 433 other blue images (between light and dark blue).

Eventually, we notice that lots of thresholds are difficult to define and some groups contain much more images than other ones. The obtained sub classes often tend to overlap. Therefore, the training step could tend to try to optimize the classification between the overlapped classes instead of focusing on the binary classification between "Aurora" and "No aurora" images. Moreover, it could ignore the classes containing too few images.

I select the CNN models which offer the best recalls and accuracies on four classes with the testing data set. The best models using the “ResNet50” were those with two layers of 150 neurons, a dropout rate of 0.3 and a regularization “L2” (with a regularization rate equal to 0.1 or 0.5). The best model using “InceptionV3” was the one with two layers of 150 neurons, a dropout rate of 0.7 and a regularization “L2” with a rate equal to 0.5. Unfortunately, the obtained results are bad. All images are classified as “images containing auroras and clouds”. I think that the models try to improve the results by finding the threshold between dark or light blue images for example. Because these thresholds are also difficult to see for a human, it disrupts the learning phase. Eventually, the obtained model is not efficient at all.

Chapter 8
Conclusion

During this project, I had to automate the auroral detection on all-sky colour images. The recall was more important to improve than the precision because we did not want to miss auroras. I first detailed most of the methods used in the previously conducted studies. Then I summarized the obtained results to decide which methods I could try by myself. A part of the methods did not rely on deep learning. In this case, the features were extracted thanks to different image processing algorithms such as SIFT, statistical descriptors, Gray Level Aura Matrices, Local Binary Pattern or Gabor wavelet decomposition. Then, the dimension of the feature vectors could be reduced before applying a classification algorithm. Otherwise, a CNN model could directly be applied to the images.

I first prepared the data sets. I split the available images into four classes: "Images containing auroras and clouds", "Images containing auroras but no cloud", "Images containing clouds but no aurora" and "Images containing no aurora and no cloud". The accuracy, the recall and the precision were computed for two classes: "Images containing auroras" and "Images without aurora". I prepared the training and validation data set by splitting a same original data set. I had to be careful not to have the same images in the training and the validation data sets. However, I had to have lots of different types of images in each of the data sets. Therefore, I had to remove the similar images before shuffling and splitting the data set. I also checked the labels and chose to attribute the label "Image containing auroras" to all images showing faint auroras. I also checked the labels of the testing data set coming from a season which is different from the training and testing sets. After preparing the data sets, I tried to choose the best classification model. I first used methods which do not require to use CNN. Then, I focused on the methods using CNN.

I describe here the selection of the model which did not require CNN models. I chose to select a dimension reducer and a classifier to compare different feature extraction algorithms. The chosen dimension reducer UMAP only required choosing K, the number of neighbours to consider, and d the smallest distance between two points in the new space. Moreover, it often provides satisfying results. I chose K = 100 and d = 0. The chosen classifier was a random forest which does not require to optimize parameters and is known as an efficient method too.

The best feature extractors were the OLBPC applied on local areas of the images, and the values distributions of each channel of the RGB and Lab colour systems. These methods enabled to reach an accuracy of 80% and a recall of 75%.

Then, I would have wanted to try some other feature extraction algorithms like the Gabor wavelet decomposition. Then, if I had more time, I would have selected one of the best feature extractors to try various dimension reduction algorithms and classifiers. The next step was using pretrained

CNN models while applying "fine tuning" and adding neural networks to the output to perform the classification. I selected two pretrained CNN models, "ResNet50" and "InceptionV3". I fixed some parameters like the learning rate (0.0001), the batch size (32), the number of epochs during the training phase (10), the activation functions (ReLU and Softmax), and the optimizer (Adam). Then, I added different neural network combinations by changing the number of layers and the number of neurons per layer, the dropout rate and the regularisation function.

The first aim was to compare the two different pretrained models with the same neural network combinations to see if the model has an impact. I describe here the results obtained by applying the classification models to the validation data set. Whatever the neural network combination, the "InceptionV3" model gave more stable results. Nevertheless, for most of the neural network combinations, the results were similar for the "ResNet50" model and the "InceptionV3" model. The second aim was to see if some neural network combinations can enable better classifications. Among the best models, the accuracies and recalls were close to each other. However, the regularization "L2" was the single one which offered satisfying results. Moreover, some combinations, like those with three layers, mainly gave poor results. The obtained recalls and accuracies were superior to 90%. However, it seemed impossible to exceed 93%. The CNN models gave much better results than the methods without CNN.

Because the obtained results were a little bit different according to the chosen CNN models, I would have wanted to test other models. I also wanted to apply cross validation to establish mean accuracies and uncertainty ranges, but I did not have enough time.

Then I selected the most promising methods to apply them to the testing data set. Indeed, because of a lack of time, I chose to proceed to the testing phase only for the CNN models. The obtained results were a little bit lower than those obtained on the validation data set. It allowed to select three models which offered better results than the other ones. If we consider the recall and the accuracy, the best models using the "ResNet50" were those with two layers of 150 neurons, a dropout rate of 0.3 and a regularization "L2" (with a regularization rate equal to 0.1 or 0.5). It enabled to reach a recall of 86%, a precision of 94% and an accuracy of 89% on the testing data. The best model using "InceptionV3" was the one with two layers of 150 neurons, a dropout rate of 0.7 and a regularization "L2" with a rate equal to 0.5. It enabled to reach an accuracy of 88%, a recall of 87% and a precision of 92% on the testing data. Unfortunately, the recall was always worse than the precision. It was probably due to the fact that all faint auroras had been classified as auroras. When I displayed the misclassified images, we could notice that the "false negatives" and the "false positives" had the same aspect. It was dark blue or grey images, with faint auroras or clouds. The classification would also be difficult for a human.

The last step consisted in creating homogeneous classes to proceed to a classification thanks to the CNN models. Indeed, a same class contained lots of different types of images while two different classes could contain similar images. Therefore, splitting the original classes into homogeneous sub classes could keep the model from relying on features which were often but not systematically associated with a given class. Unfortunately, the classes were difficult to divide. Some thresholds were fuzzy, and the obtained classes had very different sizes. I selected the CNN models which offered the best recalls and accuracies on four classes with the testing data set, but the obtained results were bad. All images were classified as "images containing auroras and clouds". Indeed, the models probably tried to improve the results by finding the threshold between dark or light

blue images for example, which was almost impossible because this threshold is even unclear for humans.

This automation of the detection will help scientists to establish the periods of time during which the auroras appear. It also will help to create a real time detector of auroras.

Bibliography

[1] M. Z. Alom, T. Taha, M. Nasrin *et al.*, 'A state-of-the-art survey on deep learning theory and architectures,' *Electronics*, vol. 8, p. 292, 5th Mar. 2019. DOI: 10.3390/electronics8030292 (cit. on p. 25).

[2] L. Breiman, 'Random forests,' *Machine Learning*, vol. 45, no. 1, pp. 5–32, 1st Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324 (visited on 11th Aug. 2022) (cit. on pp. 20, 22).

[3] L. B. N. Clausen and H. Nickisch, 'Automatic classification of auroral images from the oslo auroral THEMIS (OATH) data set using machine learning,' *Journal of Geophysical Research: Space Physics*, vol. 123, no. 7, pp. 5640–5647, 2018, ISSN: 2169-9402. DOI: 10.1029/2018JA025274. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2018JA025274 (visited on 11th Apr. 2022) (cit. on pp. 21, 26, 29).

[4] T. Evgeniou and M. Pontil, 'Support vector machines: Theory and applications,' vol. 2049, 1st Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12 (cit. on pp. 20, 21).

[5] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 26).

[6] D. Heeger and J. Bergen, 'Pyramid-based texture analysis/synthesis,' in *Proceedings., International Conference on Image Processing*, vol. 3, Oct. 1995, 648–651 vol.3. DOI: 10.1109/ICIP.1995.537718 (cit. on pp. 11, 12, 59).

[7] 'Kjell henriksen observatory (KHO).' (), [Online]. Available: http://kho.unis.no/ (visited on 27th Jul. 2022) (cit. on p. 2).

[8] A. Kvammen, K. Wickstrøm, D. McKay and N. Partamies, 'Auroral image classification with deep neural networks,' *Journal of Geophysical Research: Space Physics*, vol. 125, no. 10, e2020JA027808, 2020, ISSN: 2169-9402. DOI: 10.1029/2020JA027808. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2020JA027808 (visited on 14th Aug. 2022) (cit. on pp. 26, 29).

[9] L. V. D. Maaten and G. E. Hinton, 'Visualizing data using t-SNE,' *undefined*, 2008. [Online]. Available: https://www.semanticscholar.org/paper/Visualizing-Data-using-t-SNE-Maaten-Hinton/1c46943103bd7b7a2c7be86859995a4144d1938b (visited on 7th Aug. 2022) (cit. on p. 19).

[10] B. Manjunath and W.-Y. Ma, 'Texture features for browsing and retrieval of image data,' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, pp. 837–842, 1st Aug. 1996. DOI: 10.1109/34.531803 (cit. on pp. 12, 17).

[11] L. McInnes and J. Healy, 'UMAP: Uniform manifold approximation and projection for dimension reduction,' 9th Feb. 2018 (cit. on pp. 19, 20).

[12] G. Pandiselvi, V. Umamaheswari, S. P. Jothi and C. Balasubramanian, 'Steerable pyramid decomposition – rotation & scale invariant texture image retrieval,' p. 5, 2016 (cit. on pp. 11, 12, 59).

[13] X. Qin and Y.-H. Yang, 'Similarity measure and learning with gray level aura matrices (GLAM) for texture image retrieval,' presented at the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 1st Jan. 2004, pp. I–326, ISBN: 978-0-7695-2158-9. DOI: 10.1109/CVPR.2004.1315050 (cit. on pp. 11, 14).

[14] J. Rao, N. Partamies, O. Amariutei, M. Syrjäsuo and K. E. A. van de Sande, 'Automatic auroral detection in color all-sky camera images,' *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 12, pp. 4717–4725, Dec. 2014, Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2014.2321433 (cit. on pp. 9–11, 16, 17, 21, 28, 39).

[15] S. T. Roweis and L. K. Saul, 'Nonlinear dimensionality reduction by locally linear embedding,' *Science*, vol. 290, no. 5500, pp. 2323–2326, 22nd Dec. 2000, Publisher: American Association for the Advancement of Science. DOI: 10.1126/science.290.5500.2323. [Online]. Available: https://www.science.org/doi/10.1126/science.290.5500.2323 (visited on 7th Aug. 2022) (cit. on p. 61).

[16] P. Sado, L. B. N. Clausen, W. J. Miloch and H. Nickisch, 'Transfer learning aurora image classification and magnetic disturbance evaluation,' *Journal of Geophysical Research: Space Physics*, vol. 127, no. 1, e2021JA029683, 2022, ISSN: 2169-9402. DOI: 10.1029/2021JA029683. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2021JA029683 (visited on 11th Apr. 2022) (cit. on pp. 21, 26, 29, 41).

[17] E. Simoncelli, W. Freeman, E. Adelson and D. Heeger, 'Shiftable multiscale transforms,' *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 587–607, Mar. 1992, Conference Name: IEEE Transactions on Information Theory, ISSN: 1557-9654. DOI: 10.1109/18.119725 (cit. on pp. 15, 59).

[18] S. Singh, D. Srivastava and S. Agarwal, 'GLCM and its application in pattern recognition,' in *2017 5th International Symposium on Computational and Business Intelligence (ISCBI)*, Aug. 2017, pp. 20–25. DOI: 10.1109/ISCBI.2017.8053537 (cit. on pp. 11, 58).

[19] M. Syrjasuo, E. Donovan and M. Peura, 'Using attribute trees to analyse auroral appearance over canada,' in *Sixth IEEE Workshop on Applications of Computer Vision, 2002. (WACV 2002). Proceedings.*, Dec. 2002, pp. 289–295. DOI: 10.1109/ACV.2002.1182196 (cit. on p. 9).

[20] M. Syrjasuo and N. Partamies, 'Numeric image features for detection of aurora,' *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 2, pp. 176–179, Mar. 2012, Conference Name: IEEE Geoscience and Remote Sensing Letters, ISSN: 1558-0571. DOI: 10.1109/LGRS.2011.2163616 (cit. on pp. 11, 14, 17, 21, 28, 39, 40).

[21] M. Syrjäsuo, 'Automatic classification of auroral images in substorm studies,' *Proc. 8th ICS*, pp. 309–313, 1st Jan. 2007 (cit. on pp. 8, 11, 14, 21, 29).

[22] M. Syrjäsuo and E. Donovan, 'Analysis of auroral images: Detection and tracking,' vol. 38, 1st Jan. 2002 (cit. on pp. 8, 12, 21, 28).

[23] M. T. Syrjäsuo and E. F. Donovan, 'Diurnal auroral occurrence statistics obtained via machine vision,' *Annales Geophysicae*, vol. 22, no. 4, pp. 1103–1113, Apr. 2004, Publisher: European Geosciences Union. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00317287 (visited on 7th Jul. 2022) (cit. on pp. 12, 17, 21, 28, 39, 40).

[24] C. Szegedy, W. Liu, Y. Jia *et al.*, 'Going deeper with convolutions,' in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jun. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594 (cit. on p. 27).

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, 'Rethinking the inception architecture for computer vision,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2818–2826, ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.308. [Online]. Available: http://ieeexplore.ieee.org/document/7780677/ (visited on 28th Jul. 2022) (cit. on p. 27).

[26] F. Tang, S. H. Lim, N. L. Chang and H. Tao, 'A novel feature descriptor invariant to complex brightness changes,' in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 2631–2638. DOI: 10.1109/CVPR.2009.5206550 (cit. on pp. 11, 14).

[27] J. B. Tenenbaum, V. d. Silva and J. C. Langford, 'A global geometric framework for nonlinear dimensionality reduction,' *Science*, vol. 290, no. 5500, pp. 2319–2323, 22nd Dec. 2000, Publisher: American Association for the Advancement of Science. DOI: 10.1126/science.290.5500.2319. [Online]. Available: https://www.science.org/doi/10.1126/science.290.5500.2319 (visited on 7th Aug. 2022) (cit. on pp. 19, 60).

[28] UCAR/COMET. 'The sun, the earth, and near-earth space, 2nd edition.' (), [Online]. Available: https://www.meted.ucar.edu/spaceweather/sun_earth_space/ (visited on 27th Jul. 2022) (cit. on pp. 4, 5).

[29] Q. Wang, J. Liang, Z.-J. Hu *et al.*, 'Spatial texture based automatic classification of dayside aurora in all-sky images,' *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 72, no. 5, pp. 498–508, 1st Apr. 2010, ISSN: 1364-6826. DOI: 10.1016/j.jastp.2010.01.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364682610000441 (visited on 20th Jul. 2022) (cit. on pp. 11, 14, 21, 29).

[30] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli, 'Image quality assessment: From error visibility to structural similarity,' *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861 (cit. on pp. 34, 35).

[31] Xuejie Qin and Yee-Hong Yang, 'Basic gray level aura matrices: Theory and its application to texture synthesis,' in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Beijing, China: IEEE, 2005, 128–135 Vol. 1, ISBN: 978-0-7695-2334-7. DOI: 10.1109/ICCV.2005.43. [Online]. Available: http://ieeexplore.ieee.org/document/1541248/ (visited on 18th Jul. 2022) (cit. on pp. 11, 14, 58).

Chapter 9
Annexes

9.1 Annex A : Gantt diagrams

	Technical part of the project					Writing the report
	Monday	Tuesday	Wednesday	Thursday	Friday	Everyday
1	Meeting with Ms. Partamies	Collecting and checking the dataset + Understanding the different types of images				Introduction, UNIS presentation, Description of the observatory, Presentation of the dataset, Writing about methods described in the articles
2		Understanding the objectives				
3		Reading about the already explored methods in the context of aurora's detection: Pre-processing + Features extraction + Classification				
4						
5		Reading about Deep-Learning methods				
6		Reading about the results obtained and described in articles depending on the aim, the type of the data set and the used methods				
7		Applying and comparing different Pre-processing, features extraction and classification methods				Continuing the description of the methods, Presentation of the strategy, Explanations about the implementation
8		Applying and comparing different Deep-Learning methods (Neural network and CNN) on raw data to anticipate the behaviour (First accuracy results and computing time)				
9		Comparing the different possible Pre-processing methods and features extraction by using a same dimension reduction (UMAP for example) and classification method (A kernel SVM for example)				
10						
11		On the basis of a selection of Pre-processing methods and extraction features, comparing different neural network and their parameters (Manually and by using a Bayesian optimisation)		On the basis of a selection of Pre-processing methods, comparing different CNN architectures and their parameters (Manually and with Bayesian optimisation)		
12						
13						
14						
15		Using Post-processing methods to improve results				Presentation of the results, Conclusion, Abstract, Annexes
16						
17						
18						

Figure 9.1: Gantt diagram created in the beginning of the project

<table>
<tr><td></td><td colspan="4">Technical part of the project</td><td>Writing the report</td></tr>
<tr><td></td><td>Monday</td><td>Tuesday</td><td>Wednesday</td><td>Thursday</td><td>Friday</td><td>Everyday</td></tr>
<tr><td>1</td><td rowspan="22">Meeting with Ms. Partamies</td><td colspan="4">Collecting and checking the dataset + Understanding the different types of images</td><td rowspan="9">Introduction,
UNIS presentation,
Description of the observatory,
Presentation of the dataset,
Writing about methods described in the articles</td></tr>
<tr><td>2</td><td colspan="4">Understanding the objectives</td></tr>
<tr><td>3</td><td colspan="4" rowspan="2">Reading about the already explored methods in the context of aurora's detection:
Pre-processing + Features extraction + Classification</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td><td colspan="4" rowspan="2">Implementation of some classes and methods to anticipate the behaviour by using the previously mentioned methods</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td><td colspan="4">Reading about Deep-Learning methods</td></tr>
<tr><td>8</td><td colspan="4">Implementation of some classes and methods to anticipate the behaviour by using the CNN models</td></tr>
<tr><td>9</td><td colspan="4">Reading about the results obtained and described in articles depending on the aim, the type of the data set and the used methods to select the most pertinent methods</td></tr>
<tr><td>10</td><td colspan="4">Thinking about a detailed strategy based on the articles</td><td rowspan="9">Continuing the description of the methods,
Presentation of the strategy,
Explanations about the implementation</td></tr>
<tr><td>11</td><td colspan="4" rowspan="3">Preparing the data set: creating one folder for each class, removing similar images, correcting the labelling, creating subclasses to try to obtain homogeneous classes</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td></tr>
<tr><td>14</td><td colspan="2" rowspan="4">Implementation of different classes to test CNN models</td><td colspan="2" rowspan="4">Implementation of classes to test methods without CNN</td></tr>
<tr><td>15</td></tr>
<tr><td>16</td></tr>
<tr><td>17</td></tr>
<tr><td>18</td><td colspan="4" rowspan="3">Using "Fine Tuning" and testing various neural networks combinations at the output of two different pretrained CNN models + Testing different features extraction algorithms and classification methods without using CNN</td></tr>
<tr><td>19</td></tr>
<tr><td>20</td></tr>
<tr><td>21</td><td colspan="4" rowspan="2">Testing the best model's robustness with a new data set</td><td rowspan="2">Presentation of the results,
Conclusion,
Abstract,
Annexes</td></tr>
<tr><td>22</td></tr>
</table>

Figure 9.2: Gantt diagram corrected at the end of the project

9.2 Annex B : Features extraction

9.2.1 Gray Level Cooccurence Matrix (GLCM)

The Gray Level Cooccurence Matrix (GLCM) allows to count the pairs of pixels (with gray levels i and j) having a given relative position [31] [18]:

$$P_d(i,j) = card((x,y)/I(x,y) = i \; and \; I(x+dx, y+dy) = j)$$
$$with \; d = (r, \theta)$$
$$r \; : \; the \; chosen \; distance$$
$$\theta \; : \; the \; chosen \; direction$$

We obtain a matrix for each direction and each distance. These matrices contain lots of zeros. Therefore, it is difficult to exploit.

Moreover, it is not invariant under contrast changes or geometrical distortions.

To define the texture, we can use the seven Haralick parameters, which are defined as follows :

The homogeneity :

$$f_h = \frac{1}{N_c^2} \sum_i \sum_j P_d^2(i,j)$$

The contrast :

$$f_c = \frac{1}{N_c(L-1)^2} \sum_{n=0}^{L-1} [n^2 . \sum_{|i-j|=n} P_d(i,j)]$$

The correlation :

$$f_0 = \frac{1}{N_c . \sigma_x \sigma_y} | \sum_i \sum j[ij . P_d(i,j) - \mu_x \mu_y]|$$

The local homogeneity :

$$f_1 = \frac{1}{N_c} \sum_i \sum_j \frac{1}{1+(i-j)^2} P_d(i,j)$$

The entropy :

$$f_e = 1 - \frac{1}{N_c . log(N_c)} \sum_i \sum_j P_d(i,j) . log[P_d(i,j)]$$

The uniformity of the energy :

$$f_{unif} = \frac{1}{N_c^2} \sum_n P_d^2(n,n)$$

The directivity :

$$f_{dir} = \frac{1}{N_c} \sum_n P_d(n,n)$$

9.2.2 Steerable pyramid

We apply a low pass filter and a high pass filter to the same image [17]. The low pass filters can be median or Gaussian for example. To apply a high pass filter, we can for example use the Fourier transform and keep the high frequencies. We obtain two filtered images [6] [12].

Then, on the image filtered with the low pass filter, we apply a chosen number of oriented filters. It allows a features extraction along different directions.

The following image (Figure 9.3) details each step of the steerable pyramid.

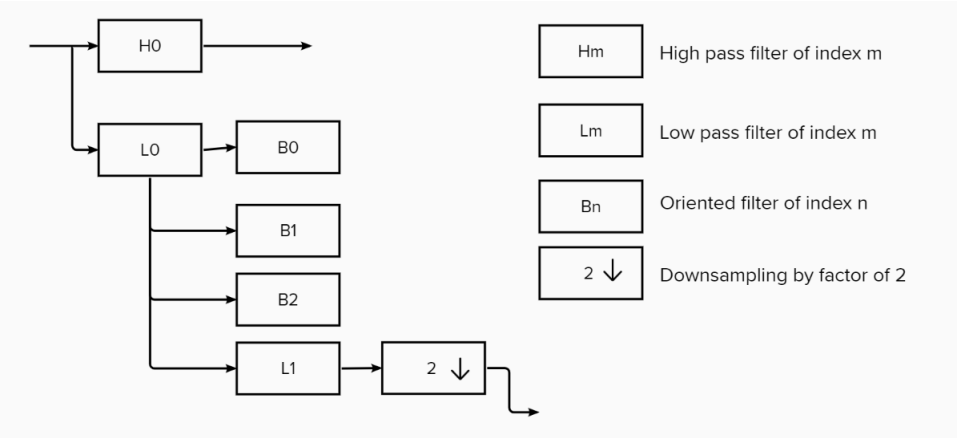

Figure 9.3: Schematic representation of the steerable pyramid

The orientations decomposition has to be steerable. It means that the resulted image after applying an oriented filtered can be broken down into a sum of responses of basic filters.

For N filters, the oriented filter of index n Bn is defined in the frequency domain as follows :

$$Bn(\omega) = B(\omega)(-jcos(\theta - \theta_n)^{k-1})$$

$$B \ : \ the\ basic\ filter$$

$$\omega \ : \ the\ frequency$$

$$\theta \ : \ the\ angle\ associated\ to\ the\ frequency\ \omega$$

$$\theta_n \ : \ the\ filter's\ orientation$$

The steerable pyramid corresponds to a non orthogonal basis. It minimizes the aliasing by meeting the Nyquist criterion. However, it induces information redundancy. Indeed, the number of pixels in the pyramid is higher than the number of pixels in the image.

The distance between two images can be computed in a similar manner as with the Gabor Wavelet decomposition. As previously mentioned, the steerable pyramid can be computed before applying the BGLAM to each level of the pyramid.

9.3 Annex C : Dimensionality reduction

9.3.1 Isometric mapping (ISOMAP)

The main idea of the ISOMAP algorithm is to consider the geodesic distance [27]. The first step is to determine which points are neighbours. In the original space of the dataset, we use the Euclidean distance to find, for each point X_i, all points X_j within a certain radius ϵ. These relationships of neighbourhood are represented by a graph G. The nodes correspond to the points X_i and the edge's weights correspond to the Euclidean distance between X_i and X_j. If X_k and X_l are not neighbours, the edge is initialized to ∞. The second step consists in computing the geodesic distances. For each pair of points (X_i, X_j), we compute the geodesic distance $d_G(i, j)$ based on the graph G by applying the shortest path algorithm. Thanks to the previously described steps, we can create a matrix of the distances. The last step is the dimensionality reduction. The original dataset, $X = \{X_1, ..., X_n\}$, corresponds to a set of points located in a space with d dimensions. The idea is to project this data in a new space with $d'(d' < d)$ dimensions. This new representation is named Y and $Y = \{Y_1, ..., Y_n\}$. δ_{ij} corresponds to the distance between X_i and X_j. d_{ij} corresponds to the distance between Y_i and Y_j. The aim is to find the configuration for which the distances δ_{ij} and d_{ij} are as similar as possible. It is usually impossible to find a configuration with $d_{ij} = \delta_{ij}$ for each i and j. There are comparison criteria which are invariant under affine translations. The following value is one of the possible criteria :

$$J_{ee} = \frac{\sum_{i<j}(d_{ij}\delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

We want to find a configuration which minimise these criteria. We can use a gradient descent. But, in fact, we can use linear algebra. The first step consists in computing the square distances matrix. Then, the distances are centered. If the dataset contains n points, we define the following matrix :

$$J = I_{n\times n} - \frac{1}{n}1_{n\times n}$$

with I the identity matrix and 1 a matrix of ones

We can compute the matrix of centered distances :

$$B = -\frac{1}{2}JP^2J$$

Then, we extract the eigenvalues and the eigenvectors from the matrix B. We keep the d' greatest eigenvalues λ_i and the associated eigenvectors V_i. The last step consists in finding the new coordinates of the data set $Y = \{Y_1, ..., Y_n\}$. We compute D, the matrix containing these coordinates, as follows :

$$D = V_{d'}L_{d'}^{\frac{1}{2}}$$

with $L_{d'}^{\frac{1}{2}}$ the diagonal matrix containing the d' greatest eigenvalues

and $V_{d'}$: the matrix containing the d' associated eigenvectors

9.3.2 Locally Linear Embedding (LLE)

If the density is great enough, we can consider that the dataset is locally approximately linear [15]. The first step consists in K nearest neighbours of each point X_i. Then, we can express each point on the basis of its neighbourhood with linear coefficients. The reconstruction errors are computed as follows :

$$E(W) = \sum_{i=1}^{n} ||X_i - \sum W_{ij} X_j||^2$$

$$with\ W_{ij}\ :\ the\ coefficients\ of\ the\ matrix\ which\ correspond\ to\ the$$

$$weight\ of\ the\ contribution\ of\ the\ point\ j\ for\ the\ reconstruction\ of\ the\ point\ i$$

We are looking for the matrix W whose elements minimise the reconstruction error. If X_i is not in the neighbourhood of X_j, we choose $W_{ij} = 0$. To be invariant under translation, we have to meet the following condition :

$$\sum_{j=1}^{n} W_{ij} = 1$$

Eventually, based on the weights previously found, we look for the points $Y = \{Y_1, ..., Y_n\}$, which allow to minimise the following reconstruction error :

$$E(Y) = \sum_{i=1}^{n} ||Y_i - \sum_{j \neq i} W_{ij} Y_{ij}||^2$$

The solution can be found thanks to eigenvalues and eigenvectors, like for the ISOMAP algorithm.

9.4 Annex D : Classification

9.4.1 Bayesian classifier

The Bayesian classification is based on the Bayes theorem. Let C be the class depending on the features F_i ($for\ 1 < i < n$). We can write :

$$P(C|F_1, ..., F_n) = \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)}$$

Because the denominator does not depend on C and the feature's values are given, we only focus on the numerator. We have the following equality :

$$p(C)p(F_1, ..., F_n|C) = p(C)p(F_1|C)p(F_2, ..., F_n|C, F_1) = p(C)p(F_1|C)p(F_2|C)...p(F_n|C)$$

If each feature is independent from the other ones, we can write :

$$p(F_i|C, F_j) = p(F_i|C)$$

Therefore, we obtain the following equality :

$$p(F_1, ..., F_n|C) = p(F_1|C)p(F_2|C)...p(F_n|C) = \Pi_{i=1}^{n} p(F_i|C)$$

And eventually :

$$p(C|F_1, ..., F_n) = \frac{1}{Z} p(C) \Pi_{i=1}^{n} p(F_i|C)$$

$$With\ Z\ : a\ constant\ value$$

The prior probabilities of the different classes can be computed assuming that the classes have equal probabilities or by estimating the probabilities of each class on the basis of the available data.

To estimate the parameters of the probability distribution related to a specific feature, we have to assume the type of the distribution. We assume that we have a normal distribution.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_{i\mu})^2$$

To estimate the most probable class cl, we use the maximum a posteriori (MAP) estimation method.

$$cl = argmax_c p(C = c) \Pi_{i=1}^{n} p(F_i = f_i|C = c)$$

If we assume that all classes have the same probability, we can write :

$$p(F_i = f_i|C = c) = f_{g,v}(x) = \frac{1}{\sqrt{2\pi\sigma_{g,v}^2}} e^{\frac{-1}{2\sigma_{g,v}^2}(x-\mu_{g,v})^2}$$

This method only requires to know the mean and the variance of the different features for each class. These parameters do not require lots of training data to be estimated. However, this method gives poorer results than a random forest algorithm.

9.5 Annex E : Image samples of the created subclasses

9.5.1 Cloudy sky with auroras

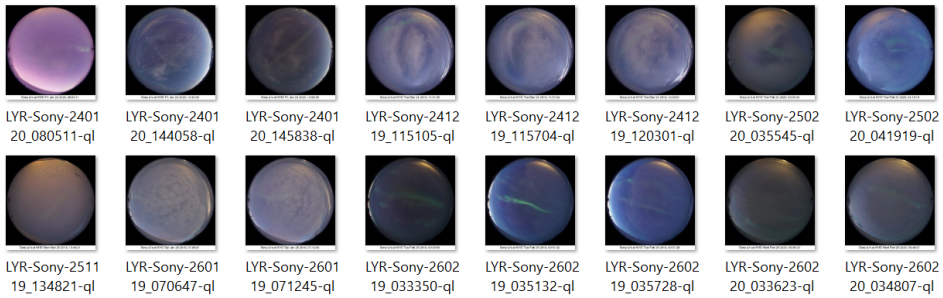

Figure 9.4: Blue images containing auroras and clouds

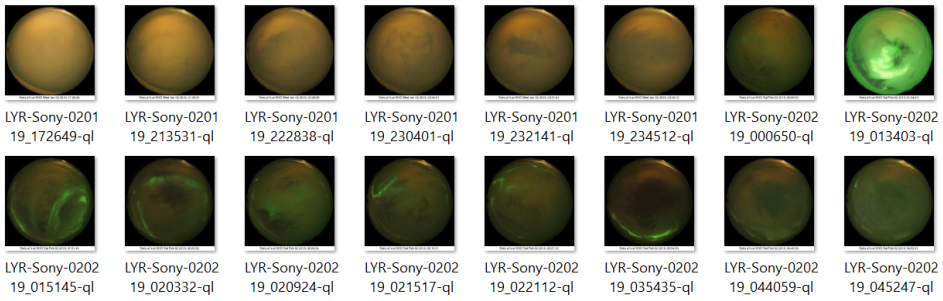

Figure 9.5: Brown and green images containing auroras and clouds

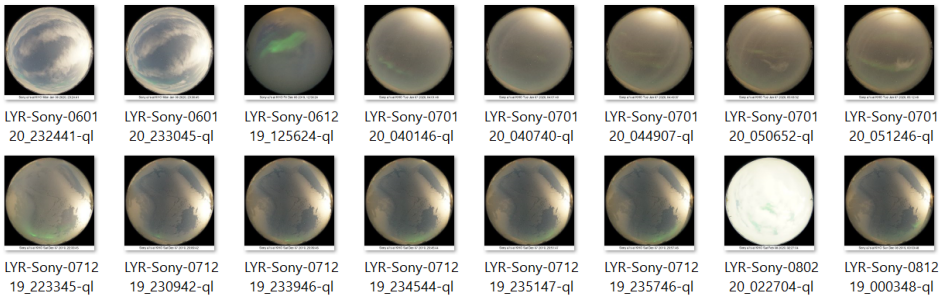

Figure 9.6: Gray images containing auroras and clouds

9.5.2 Cloudless sky with auroras

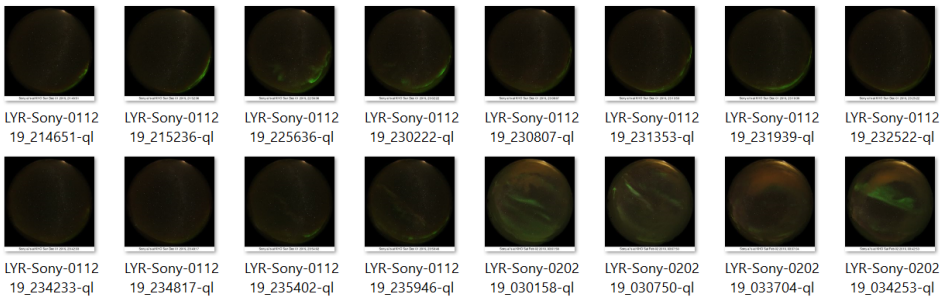

Figure 9.7: Black images containing auroras but no cloud

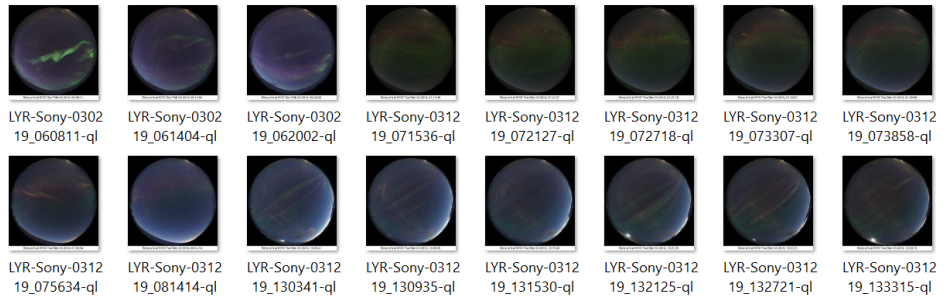

Figure 9.8: Dark blue images containing auroras but no cloud

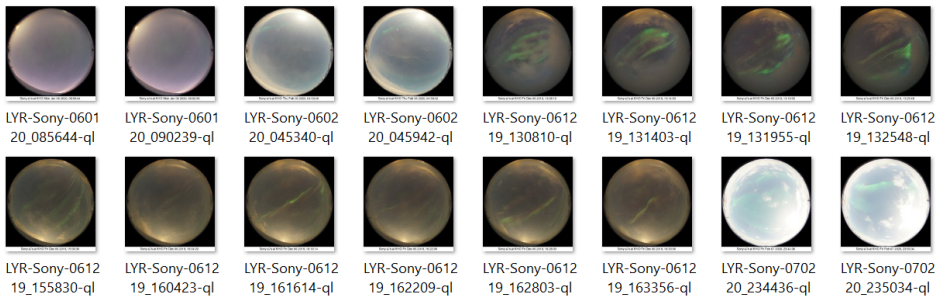

Figure 9.9: Gray images containing auroras but no cloud

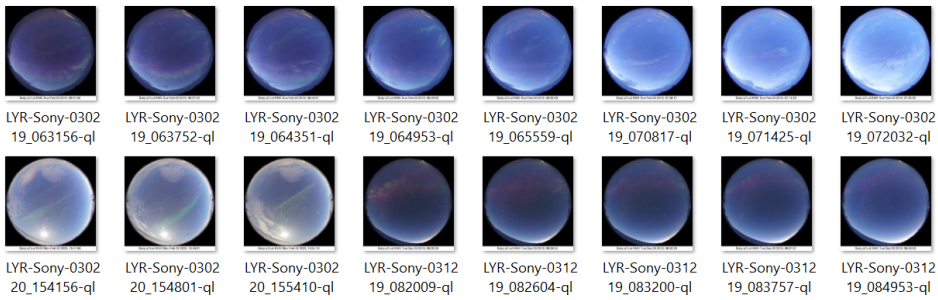

Figure 9.10: Blue images containing auroras but no cloud

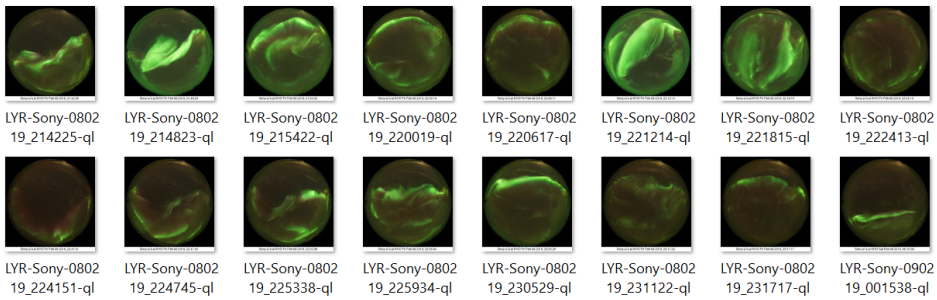

Figure 9.11: Black and green images containing auroras but no cloud

9.5.3 Cloudy sky without aurora

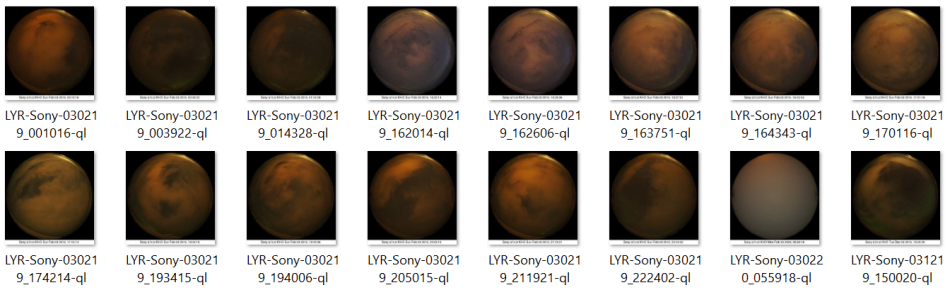

Figure 9.12: Black and brown images containing clouds but no aurora

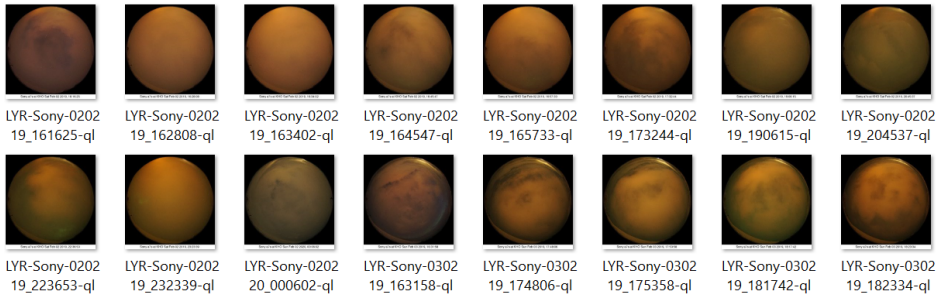

Figure 9.13: Brown images containing clouds but no aurora

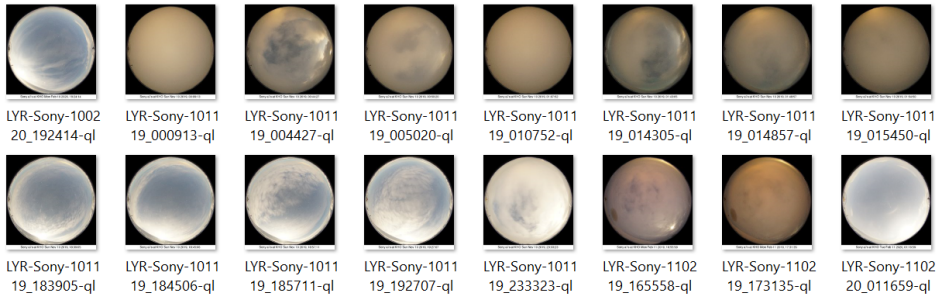

Figure 9.14: Gray images containing clouds but no aurora

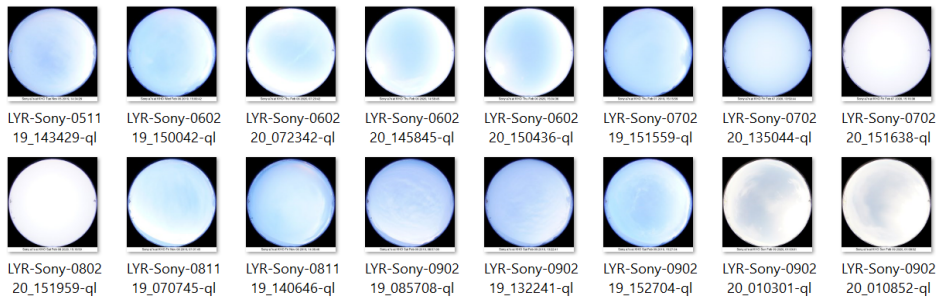

Figure 9.15: Light blue images containing clouds but no aurora

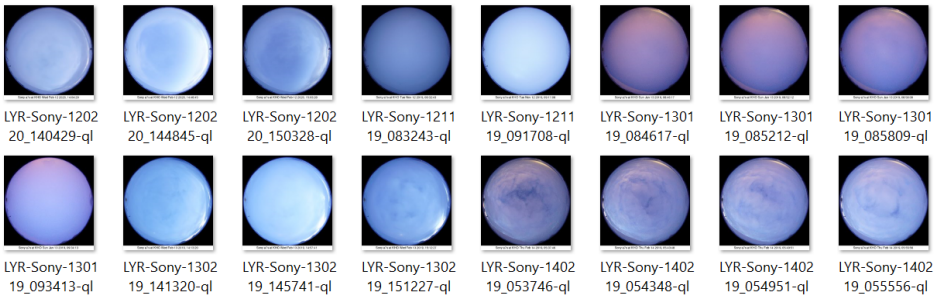

Figure 9.16: Blue images containing clouds but no aurora

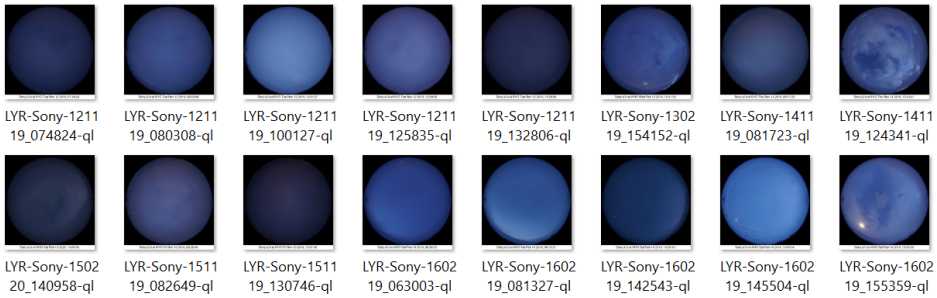

Figure 9.17: Dark blue images containing clouds but no aurora

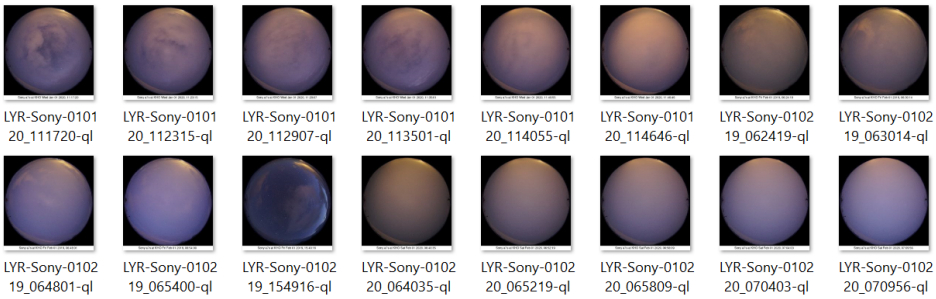

Figure 9.18: Gray and blue images containing clouds but no aurora

9.5.4 Cloudless sky without aurora

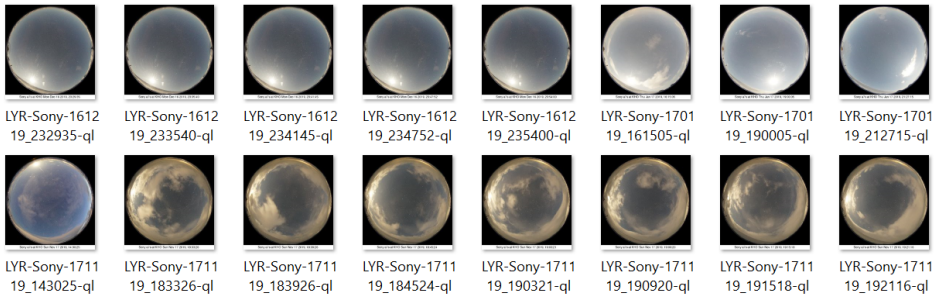

Figure 9.19: Gray images containing no cloud and no aurora

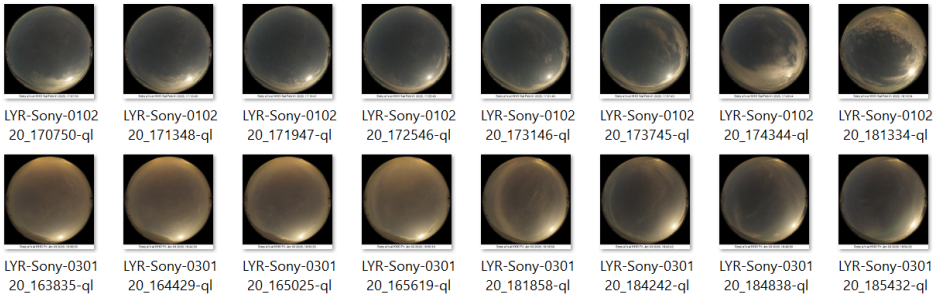

Figure 9.20: Dark gray images containing no cloud and no aurora

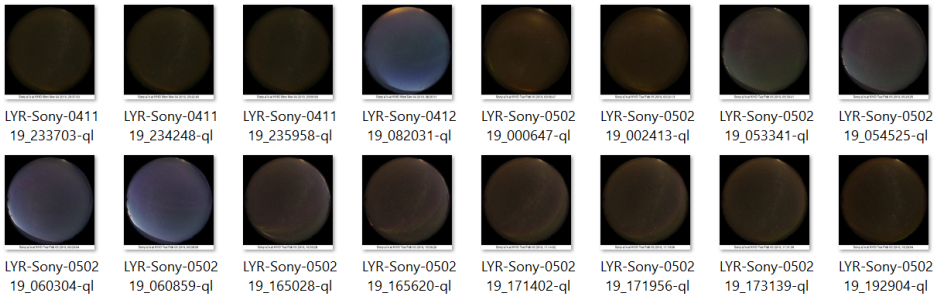

Figure 9.21: Dark blue and black images containing no cloud and no aurora

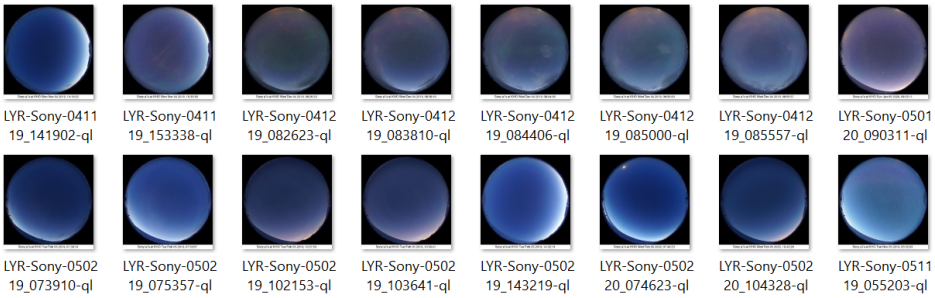

Figure 9.22: Dark blue images containing no cloud and no aurora

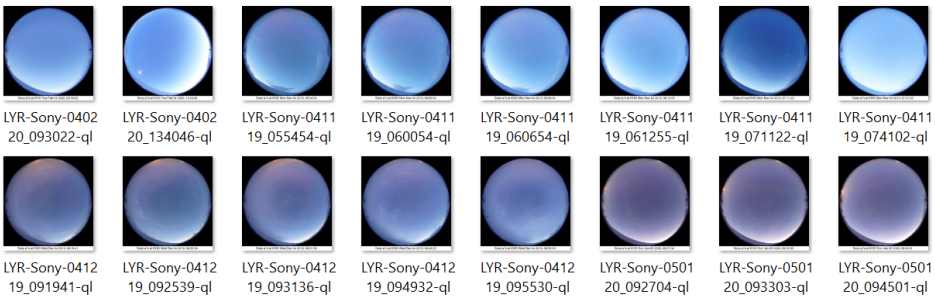

Figure 9.23: Blue images containing no cloud and no aurora

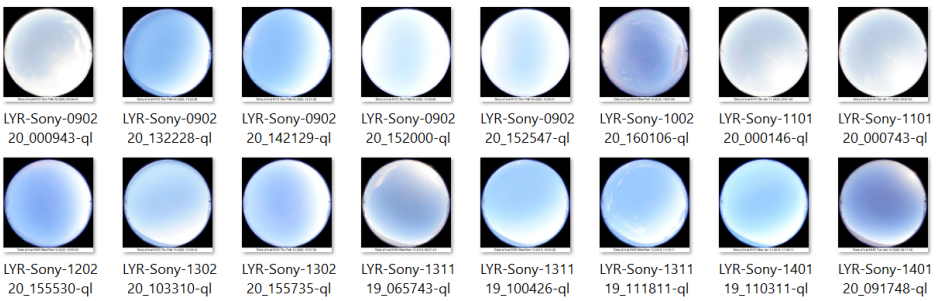

Figure 9.24: Light blue images containing no cloud and no aurora