Classification of Aurora Borealis Using Svalbard All-Sky Imager Data and EfficientNet Convolutional Neural Network

Kristina Othelia Lunde Olsen

Thesis submitted for the degree of
Master in Space Physics and Space Technology
60 credits

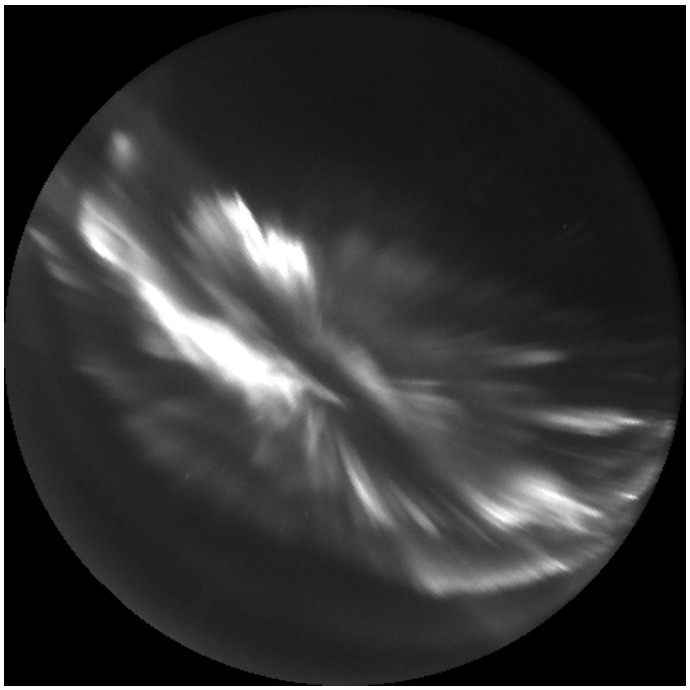
Department of Physics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2022

Classification of Aurora Borealis Using Svalbard All-Sky Imager Data and EfficientNet Convolutional Neural Network

Kristina Othelia Lunde Olsen



Classification of Aurora Borealis Using Svalbard All-Sky Imager Data and EfficientNet Convolutional Neural Network

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

All-Sky Imagers located in the Arctic and Antarctic regions capture images of the sky at regular intervals throughout the winter season. Data from the last decades make up millions of images where auroral researchers have no way of filtering the data without time-consuming manual investigation. We implemented the convolutional neural network family called EfficientNet for automatic classification of All-Sky Imager data. We manually labeled 7,980 images from Ny-Ålesund, Svalbard, into classes based on the appearance of aurora or no visible aurora. As a goal were to classify different aurora shapes, we used the 3 classes *arc*, *diffuse* and *discrete*, while images without detectable aurora were classified as *no aurora*. We found that EfficientNet successfully detected aurora in All-Sky Imager data. Training several EfficientNet models with various hyper-parameters, the highest performing model achieved an classification accuracy of 88% on unseen test data. By aggregating the 3 aurora classes, we archive an binary classification accuracy of 96% on the same test data. The methods shown in this thesis can be applied to data from any auroral All-Sky Imager. We created a data set of 665,865 unlabeled Ny-Ålesund all-sky images (5577 Å and 6300 Å for the same time periods for 2014, 2016, 2018 and 2020) [1], and matched each image to approximate solar wind parameters from NASA's OMNI data [2]. Our model were applied to the data set, and statistical results show that variations in solar wind speed and IMF B_z do not determine the observed aurora shape. Further, our classifier labeled more images as diffuse for the 6300 Å emission line (red aurora), which indicates good predictions. This was expected, as red aurora is a weaker, more diffuse form of aurora. Statistics were also made based on an hourly distribution, where we could observe dayside and nightside aurora. During polar nights, Svalbard is optimal for observing dayside aurora, but we found that the location is probably to high north to observe stronger nightside aurora events, like substorms. Our results for the hourly distributions indicate a (weak) double-peak feature for dayside aurora, which have been observed before [3][4][5]. The feature is strongest for 2020, with one peak around 6-8 local time and a second peak around 14-15 local time (for discrete aurora). Our time points (in magnetic local time) do not match the previous observations, but are +3 hours shifted.

Acknowledgement

I acknowledge use of NASA/GSFC's Space Physics Data Facility's OMNIWeb (or CDAWeb or ftp) service and OMNI data [2]. I acknowledge use of Svalbard All-Sky Imager Data [1], provided by the University of Oslo. I also acknowledge the use of EfficientNet from the CRAI-Nets project [6].

I want to thank my supervisor Lasse Clausen for help and support. Thank you for proofreading and providing valuable feedback on the thesis structure and content. I would also like to say thank you to Bjørn Lybekk for technical support.

To my friends and fellow physicists Frida, Nils, Aron, Erlend, Karianne, Jon and Anna. Thank you for many years together at UiO, for all the reading sessions and social gatherings. Anna, Aron, Frida and Jon, thank you for reading through my thesis and providing great feedback. A special thank you to Jon for help and advice during my struggles with the machine learning aspect of this thesis.

To my family, thank you for the love and support while working on my master's thesis, and for offering the family cabin as a quiet place to work during a long hectic period affected by the Covid-19 pandemic.

Contents

1 Introduction **1**

2 Theoretical Background **2**
2.1 Space weather . . . 2
2.1.1 The solar wind and IMF . . . 2
2.1.2 The solar cycle . . . 3
2.2 The magnetosphere . . . 5
2.3 The Dungey cycle . . . 6
2.4 Aurora . . . 8
2.4.1 The auroral oval . . . 9
2.4.2 Dayside and nightside aurora . . . 10
2.4.3 Aurora colours/emission lines . . . 11
2.5 Geomagnetic substorms . . . 12

3 Machine Learning and Neural Networks **13**
3.1 Feedforward neural network . . . 13
3.2 Activation functions . . . 15
3.3 Propagation, cost function and gradient descent . . . 16
3.4 Convolutional neural network . . . 18
3.4.1 Convolutional layer . . . 19
3.4.2 Pooling layer . . . 20
3.4.3 Fully connected layer . . . 20
3.4.4 Overfitting, dropout layer and data augmentation . . . 21
3.4.5 Imbalanced data . . . 22
3.5 Model evaluation . . . 23
3.5.1 Confusion matrix and F1 score . . . 23
3.5.2 Multi-class confusion matrix and F1 score . . . 24
3.5.3 Classification accuracy . . . 25

4 Instruments and Locations **26**
4.1 Aurora data . . . 26
4.1.1 All-Sky Imager . . . 26
4.1.2 Ny-Ålesund, Svalbard . . . 26
4.2 Solar wind data . . . 27
4.2.1 Advanced Composition Explorer spacecraft . . . 27
4.2.2 Wind spacecraft . . . 27

5 Data and Methodology **28**
5.1 Data . . . 29
5.1.1 Data pre-processing . . . 29
5.1.2 ASI data for CNNs . . . 30
5.1.3 ASI data for aurora predictions . . . 30
5.1.4 High resolution OMNI data . . . 31
5.2 Classes . . . 32

5.3 Data sets . . . 35
5.3.1 ASI data sets for CNNs . . . 35
5.3.2 ASI data sets for aurora predictions . . . 36
5.4 EfficientNet . . . 38
5.5 Training, validation and testing . . . 40
5.5.1 Models and hyper-parameters . . . 40
5.5.2 Neural network training and validation . . . 41
5.5.3 Model testing and evaluation . . . 42

6 Results **43**
6.1 EfficientNet . . . 43
6.1.1 Models . . . 43
6.1.2 Validation predictions . . . 44
6.1.3 AuroraB3 . . . 45
6.1.4 AuroraB3, 2 classes . . . 47
6.2 Prediction results on unlabeled ASI data . . . 48
6.2.1 Distribution of classified ASI data . . . 48
6.2.2 Predictions with equal class probabilities . . . 50
6.2.3 Hourly distribution of classified ASI data . . . 52
6.2.4 Yearly solar wind speed distribution . . . 55
6.2.5 Yearly IMF B_z distribution . . . 59
6.2.6 Hourly B_z-dependent distribution of classified ASI data . . . 62

7 Discussion **69**
7.1 Image labeling and AuroraB3 . . . 69
7.2 EfficientNet AuroraB3 aurora predictions . . . 70

8 Conclusions and future work **72**
8.1 Conclusions . . . 72
8.2 Future work . . . 73

Bibliography **74**

1 Introduction

In space physics, solar wind and the Earth's magnetosphere are two prominent avenues of research. The solar wind originates from the Sun, and consists of plasma and the frozen-in interplanetary magnetic field (IMF). Due to events on the surface of the Sun, there is a constantly changing solar wind that propagates away from the Sun in all directions. The portion of the incoming solar wind that reaches Earth, based on the current properties, may lead to solar wind-magnetosphere coupling. This is when IMF lines reconnect with Earth's magnetic field lines, and the Dungey cycle [7] allows energetic particles from Earth's geomagnetic tail to interact with Earth's ionosphere. The process results in a display of mesmerising light in various colours and shapes in the sky. This phenomenon is known as the nightside aurora, and are a physical phenomena around Earth's geomagnetic poles we can observe with our own eyes, and with ground based cameras. The weaker dayside aurora is caused by (low) energetic particles entering the ionosphere through the polar cusps. The aurora shape and brightness observed from the ground, map to physical processes in the magnetosphere, which allow the study of magnetospheric process.

Every winter, ground-based All-Sky Imagers (ASIs) in the Arctic and Antarctic regions take images of the sky and the aurora light. These images are captured at regular intervals for periods when the Sun is below the horizon. The motivation for this thesis lies in the difficulty to extract the correct information from the ASI data. It is difficult to determine if the imaged sky contains features of aurora, cloud or light pollution. If we can automatically label aurora images according to their shape, we can eliminate the process of manually sort and label images. Data from the last decades make up millions of images that can be used for large-scale image data analysis if they can be automatically labeled.

Previous study on automatic aurora classification achieved an 91% classification accuracy with classes: aurora, no aurora, and cloudy, using Support Vector Machines [8]. Classifications with classes: clear/no aurora, cloudy, moon, arc, diffuse, and discrete (Clausen and Nickisch), have achieved an 82% accuracy with a pretrained deep neural network and a Ridge classifier [9], and an 91% accuracy with a SimCLR model [10]. Other work exclude all ambitious images, and only classify the aurora into subclasses, archiving results around 90% accuracy [11].

Using the same classes as Clausen and Nickisch, we want to use a state-of-the-art convolutional neural network to make an aurora classifier. The network will be trained and tested on data from the Svalbard ASI database [1]. The best performing model will then be applied on a dataset containing 665,865 images from Svalbard ASI database. We use data from years included in the last solar cycle, which is an measurement of the Sun's surface activity. Higher activity, leads to more intense solar wind streams, which affect the solar wind-magnetosphere coupling process. We will also use the IMF B_z component. We will make statistical results to research how the aurora is affected by changes in the solar wind during solar minimum and a solar maximum years.

2 Theoretical Background

This chapter gives a short introduction to the solar wind and the interplanetary magnetic field, before moving on to Earth's magnetosphere. The solar wind-magnetosphere coupling is explained through the Dungey cycle. Lastly, the aurora and geomagnetic substorms are introduced.

2.1 Space weather

Space weather is the varying conditions in the space environment between the Sun, the Earth and the rest of the solar system. The weather phenomena are mainly caused by surface activity on the Sun. Space weather can affect Earth's magnetosphere and ionosphere, and during large space weather events, even human space missions and technology can be impacted. For instance, satellite electronics and on-Earth power grids can be damaged. Our modern society is therefore dependent on knowledge about space weather and to make the necessary precautions to ensure the safety of our technology, and astronauts. Better knowledge can lead to new insights into the science behind the mesmerizing aurora display.

2.1.1 The solar wind and IMF

The Sun is a layered near-perfect sphere consisting mainly of plasma, or electrically charged particles. An important feature is the constantly moving plasma in the convection layer, resulting in electric currents and solar magnetic fields. The outer layer of the Sun's atmosphere is called the Corona. From the Corona, there is a constant stream of charged particles, known as the solar wind, traveling into interplanetary space. Changes in the solar wind are dependent on activity on the surface of the Sun as well as changes in the Corona. In coronal holes, the magnetic field lines are open, and plasma flows gets easier access to the solar wind, powering it up by increasing the velocity and density. The solar wind travels with an average speed of 400 km/s towards Earth [12], but can travel with speeds from 200-800 km/s. It is mainly the solar wind ejected from the Sun's equator, in the Earth-Sun line, that interact with Earth's magnetosphere. Therefore, the condition of the Sun's equator is important. When the magnetic field around the equator is calm, the solar wind is calm (300-600 km/s). When solar activity increases, the magnetic field gets stronger and more erratic, creating more sunspots along the equator. The solar wind speed from active regions like sunspots, create a fast moving solar wind (600-800 km/s).

Closed magnetic field lines create magnetic loops above the Sun's surface. These large structures, known as solar loops and solar prominences, have hot interior plasma flowing along the loops. The solar exit and entrance of these loops can be seen on the surface as dark sunspots. Sunspots appear dark because the area is cooler than the surroundings. The magnetic loops often tangle with other loops because of the constantly moving gasses on the Sun. The tangled up loops stretch and twist, which may lead to magnetic reconnection, see chapter 2.3, and bro-

ken loops. This event, a coronal mass ejection (CME), releases large amounts of plasma into the solar wind, as well as open magnetic field lines. CMEs often occur in active regions with many sunspots. Both CMEs and coronal hole dynamics create shock waves which temporarily interfere with Earth's magnetosphere.

The solar wind largely behaves as an ideal plasma. This means that the solar magnetic field lines are "frozen-in" to the solar wind, being dragged along into interplanetary space [13]. This is referred to as the interplanetary magnetic field (IMF). The properties of the IMF lines are important for the solar wind-magnetosphere coupling, explained in chapter 2.3. The IMF strength **B**, in the GSM coordinate system, has a B_z component in the north-south direction. When the direction is northward, the value is positive, while it is negative when southward. The B_z direction determines the outcome of the magnetic reconnection between the IMF and Earth's magnetic field.

2.1.2 The solar cycle

The solar cycle is a measurement of the Sun's activity, which are determined by the number of sunspots in the Sun's photosphere. A solar cycle starts with a solar minimum, when the sunspot counts are low. When the magnetic field activity increases, so does the number of sunspots, and the solar maximum is when the count is at its highest. This occurs about halfway through the cycle. After the solar maximum, the activity again decreases into a new solar minimum as the cycle ends. Solar observations show that the Sun follows a periodic cycle of 11 years [[14], p. 4]. However, sources do not always agree on the exact year of a solar minimum or maximum, which is why it is more correct to called it a nearly periodic cycle.

The last 13 years of recorded/observed sunspots, by the Royal Observatory of Belgium, are visualized in figure 2.1.1. The graph shows that the most recent solar minimum occurred in 2020, and the last solar maximum occurred in 2014. The 11-year cycle seem accurate this cycle as the graph indicates the previous solar minimum was approximately in year 2009. The sunspot numbers since the mid 1950s are plotted in figure 2.1.2. The graph indicates that the most recent solar cycle was a relatively weak one among the last 6 recorded cycles, with the solar maximum only having approximately half the recorded sunspots as the maximum in the late 1950s.

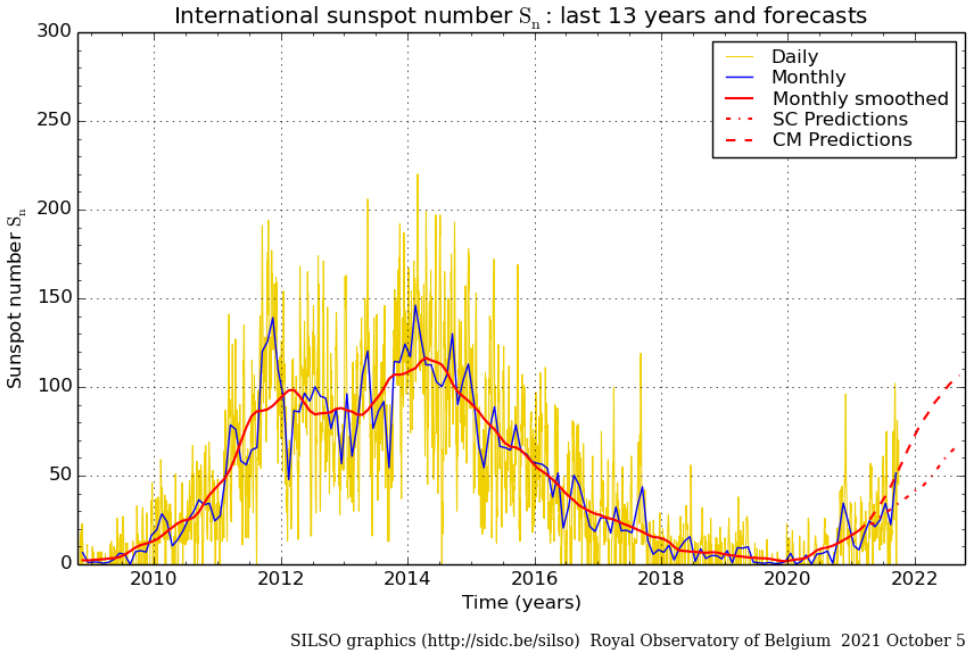


Figure 2.1.1: International sunspot numbers for 2009/2010 until the end of 2021. The graph also display a sunspot forecast for 2022. The graph is from: SILSO data/image, Royal Observatory of Belgium, Brussels [15].

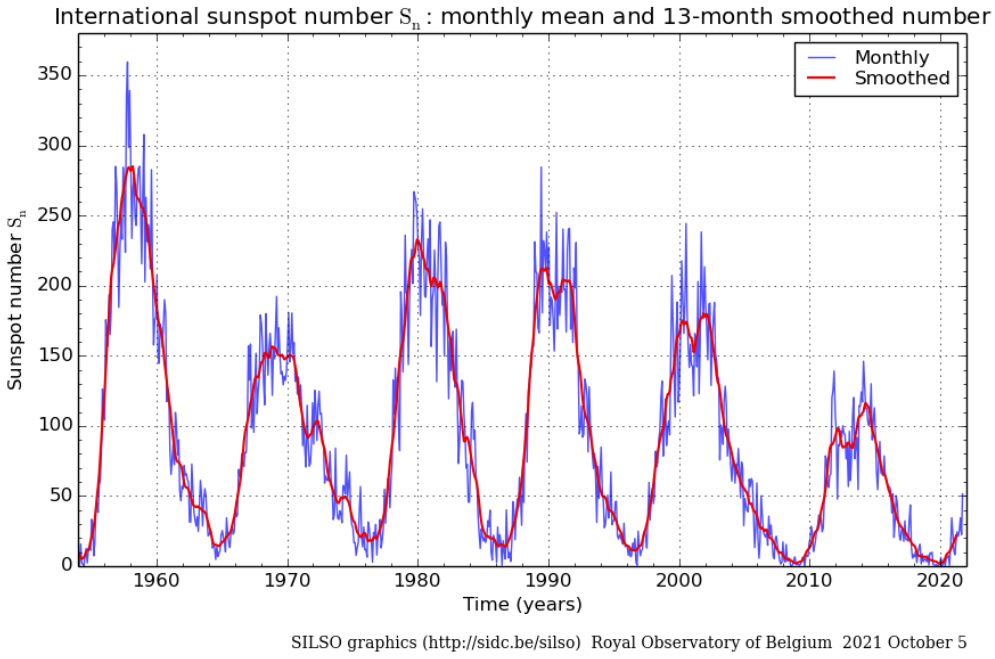


Figure 2.1.2: International sunspot numbers that show the registered sunspots since late 1950s. The graph shows the monthly mean (blue line) number of sunspots, and the 13-month smoothed sunspot numbers (red line). The graph is from: SILSO data/image, Royal Observatory of Belgium, Brussels [15].

2.2 The magnetosphere

Earth's (geo)magnetic field extends into space and protects the planet and its atmosphere against solar wind and cosmic radiation. The field is magnetic dipole approximated, directed northward. Earth's magnetosphere is the region that stretches from Earth's ionosphere out to the extent of the magnetic field. Figure 2.2.1 shows an illustration of the magnetosphere, with incoming solar wind from the Sun. Most of the solar wind heading towards Earth gets deflected and decelerated by the bow shock [14]. The shocked solar wind that passes the bow shock, enters the magnetosheath, the region between the bow shock and the magnetopause. The magnetopause is the outer area of the magnetosphere, creating a boundary between the solar wind plasma and Earth's magnetic field. The magnetopause is also where the solar wind pressure and the pressure from Earth's magnetic field is balanced [14]. The interaction between the solar wind and the magnetosphere distorts the magnetic field, creating a bullet or comet shape. The outer loops of the magnetic field lines get compressed by the solar wind pressure on the dayside of Earth (the side facing the Sun), while they get stretched out into a long magnetotail on the nightside. The dayside has closed magnetic field lines, while the nightside have both open and closed magnetic field lines. Closed magnetic field lines means both ends of the field line is connected to Earth. As the solar wind pressure is dynamic, it puts pressure on the magnetosphere, defining the location of the magnetopause. When the solar wind pressure increases or decreases, the magnetopause is pushed closer or further away from Earth. A polar cusp is where plasma is able to enter Earth's ionosphere.

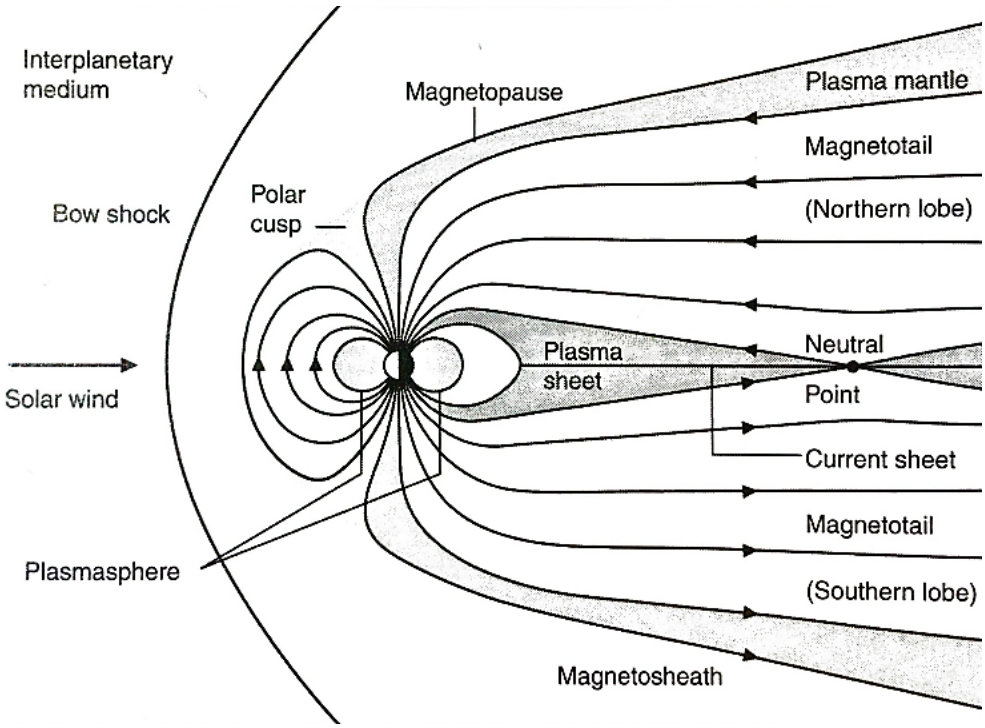


Figure 2.2.1: Illustration of Earth's magnetosphere. Distances and objects are not to scale. After Russel (1972), figure 10.1 [14]

2.3 The Dungey cycle

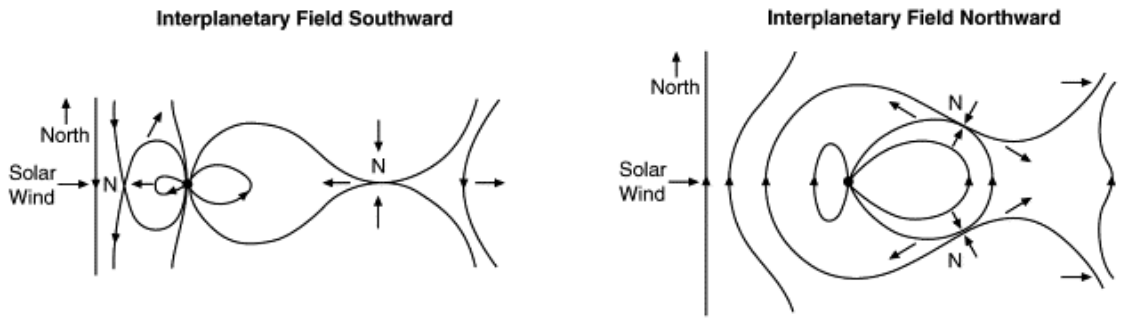


Figure 2.3.1: Illustrations of the magnetic reconnection between the IMF and Earth's magnetosphere for southward IMF (left) [after Dungey, 1961] and northward IMF (right) [after Dungey, 1963] [14]. "N" marks a neutral point where magnetic reconnection occurs.

The Dungey cycle is the process of magnetic reconnection between the IMF and Earth's magnetosphere, or any other body with a magnetic field [Dungey, 1961]. Magnetic reconnection is when two anti-parallel magnetic field lines couple and reconnect, so the interesting mechanisms unfold when the IMF points southward. The reconnection occurs in a neutral point, marked "N" in figure 2.3.1. The reconnection process creates open field lines, connecting IMF lines with Earth's magnetic field. The frozen-in plasma gets access to these new field lines that are connected to the ionosphere above the polar cusps.

The Dungey cycle for the southward solar wind-magnetosphere coupling is illustrated to the left in figure 2.3.1. There is a dayside reconnection at the magnetopause and nightside reconnection in the magnetotail. Figure 2.3.2 by Kivelson and Russel [14] shows a more detailed illustration of the cycle based on Dungeys theory. This illustration also include the field line motions and plasma flow (arrows) across the polar cap. From the supersonic solar wind there is a southward magnetic field line that crosses the bow shock, gets slowed down, and travels across the magnetosheath. The IMF line 1' meet Earth's field line 1 at the magnetopause, where they break up and reconnect (dayside reconnection), creating an open field line. This allows some plasma to follow field-aligned currents into the polar cusps. The field line is then dragged anti-sunwards across the polar cap into the stretched out magnetotail (field line at points 2-4) and wraps around the Earth. At point 5, field lines are stacked up, building up plasma and magnetic flux in the magnetotail. This also occurs for the southern tail (2'-5'), where the field lines have the opposite direction. When reconnection occurs in the magnetotail due to the built up stress (nightside reconnection), field line 6 and 6' of opposite direction connect and reconnect, dividing the plasma flow in the plasma sheet, see figure 2.2.1. One part of the plasma flow/sheet continues away from Earth, carrying field line 7', which now is back to being an IMF line. Line 7 is now a closed terrestrial field line, connected to the ionosphere. The plasma from the sheet flows down the to the polar caps into the ionosphere, creating nighttime auroras (on both hemispheres). After the reconnection in the magnetotail,

the closed field line moves sunwards to the dayside across the auroral zone (7-9).

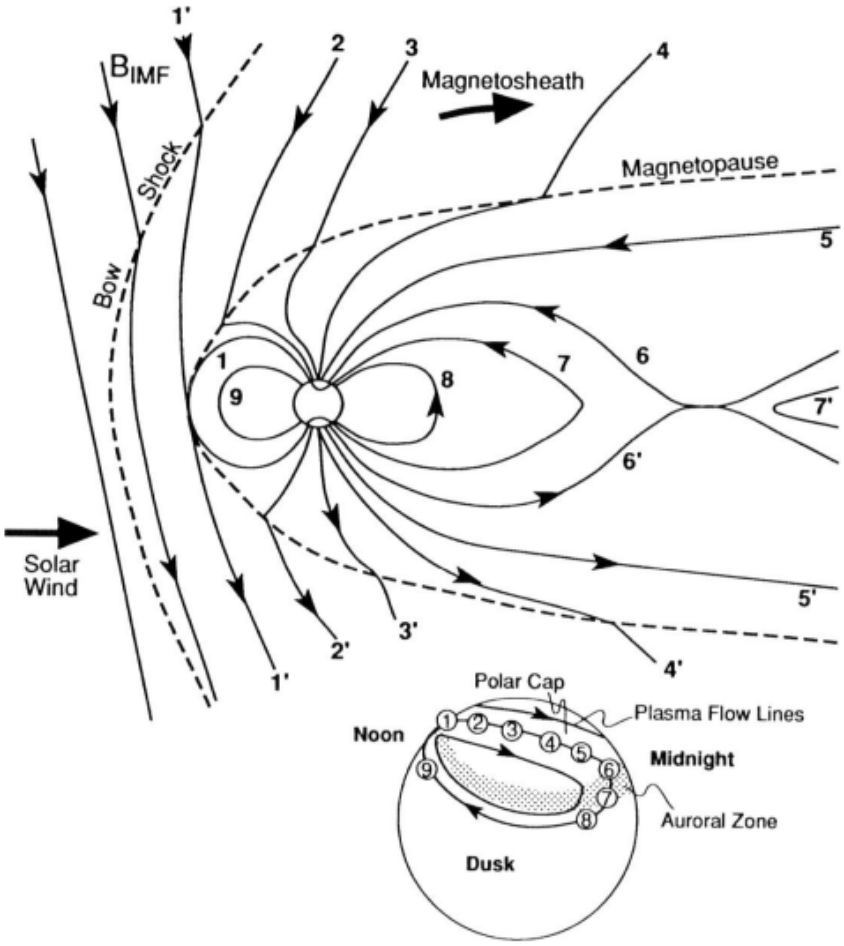


Figure 2.3.2: The Dungey cycle with southward IMF ($B_z < 0$). The illustration shows the interaction between the solar wind and the magnetosphere. After the magnetic reconnection on the dayside (1), the new open field line is dragged anti-sunwards to the nightside, into the magnetotail. The open field line (2-5) reconnects in the magnetotail (6), accelerating plasma down the polar caps. The new closed field line then moves sunwards to start a new cycle (7-9). Distances and objects are not to scale. After Kivelson and Russel, 1995 [14].

The polar cap is the region inside the auroral oval (chapter 2.4.1). The size of the polar cap is defined by the boundary between open field lines and closed field lines, called the open-closed boundary (OCB). The OCB encloses the polar cap, setting the location for the auroral oval. During dayside reconnection, previously closed (high latitude) field lines open, causing the open magnetic flux increases and the OCB to widen equatorward [16]. When the open magnetic field lines reconnect in the magnetotail, they close (and releases the stored magnetic energy into the ionosphere), causing the OCB to contract and move back to higher latitudes [16].

2.4 Aurora

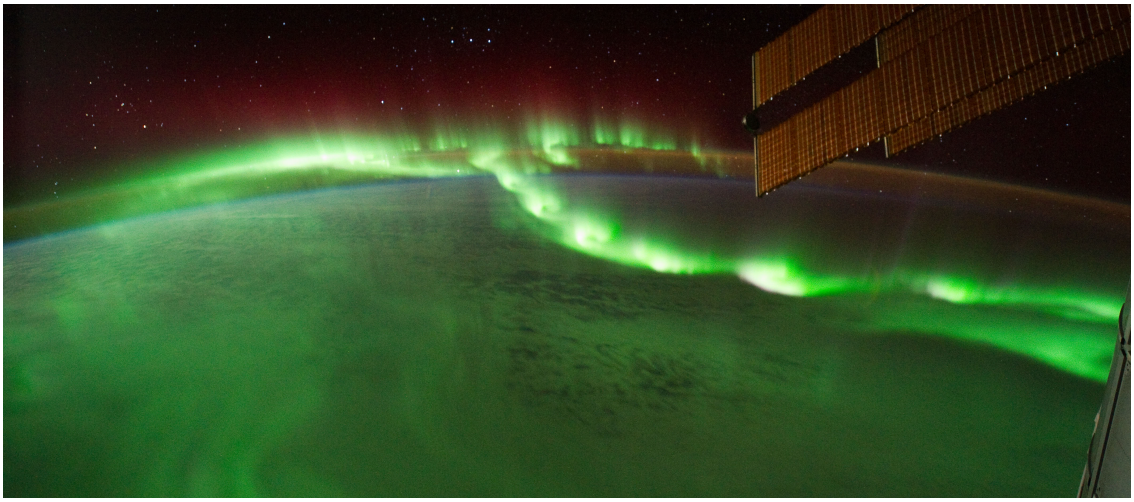


Figure 2.4.1: The aurora as seen from space. The image shows green aurora at lower altitudes and red aurora at higher altitudes. Diffuse aurora can be observed at the lower left and discrete aurora at the top and to the right. "Auroral beads seen from the International Space Station, Sept. 17, 2011 (Frame ID: ISS029-E-6012). Credit: NASA".

The polar light, also known as the Aurora Polaris or simply aurora, is a dynamic light show in Earth's ionosphere that varies in colour, shape, brightness and location. The light phenomenon occur in the auroral ovals around the northern and the southern geomagnetic poles, giving the Latin names Aurora Borealis (northern lights) and Aurora Australis (southern lights). Figure 2.4.1 is an image taken from the International Space Station, showing red and green aurora. The image capture different types/shapes of aurora, with diffuse aurora at the lower left and discrete aurora at the top and to the right. Aurora shapes will be further explained in chapter 5.2, where the classes for the classification are explained.

The ionosphere is a part of Earth's outer atmosphere that is partially ionized by, among others, solar radiation and charged particles. The lower boundary of the region is located about 80 km above the surface of the Earth. Bright aurora is mainly caused by reconnection in the magnetotail, where charged particles from the plasma sheet enter the ionosphere and collide with ionospheric gases. Ionospheric gases receives energy, which leads to excitation of electrons in atoms and molecules. Thereafter, excited electrons return to the ground state, causing photons to be emitted with energy equal to the initially absorbed energy by the atom or molecule. The emitted photons make the auroral light, and the colour of the light is further explained in chapter 2.4.3.

The science behind the auroras was a mystery for a long time, but with the research of Norwegian scientist Kristian Birkeland (1867–1917) the aurora had to let some of its secrets go. Birkeland proposed that charged particles emitted from the Sun produced the atmospheric lights after striking the Earth's magnetic field [[14], p. 346]. The particles would be slowed down by the atmosphere, causing the ionospheric particles to light up due to the energy transfer caused by fric-

tion [17]. Although Birkeland never got to prove his northern light theories, they were confirmed decades later when satellites could perform measurements of the solar wind and Earth's magnetic field [17].

2.4.1 The auroral oval

Aurora occurs in a oval-shaped belt around the geomagnetic pole where magnetic field lines connect with the ionosphere. The belt is known as the auroral oval, illustrated in figure 2.4.2 for the northern and southern hemisphere. The intensity of the auroral oval, the width and location, are dependent on the geomagnetic activity cause by solar wind and IMF properties. When the geomagnetic activity increases, so does the width of the oval. The illustration further shows how the location of the ovals moves (following the OCB) to lower latitudes during a geomagnetic storm or substorm. Only geomagnetic substorm will be explained further, see chapter 2.5, because they are shorter events that lasts a few hours and they can occur quite frequently. The illustration also shows that the oval is slightly shifted around the geomagnetic pole. This effect is caused by the solar wind pressure on the magnetic field, slightly shifting the oval towards the nightside [18]. The all-over auroral oval is narrower on the dayside than the nightside. The widest region is located at (local) midnight at the nightside, a feature the illustration fails to show. The oval is always oriented with local noon towards the Sun, meaning the oval is not rotating, but the Earth is rotating below the oval. All aurora data for this thesis is gathered from Svalbard, which is located on the southern edge of the auroral oval in the northern hemisphere.

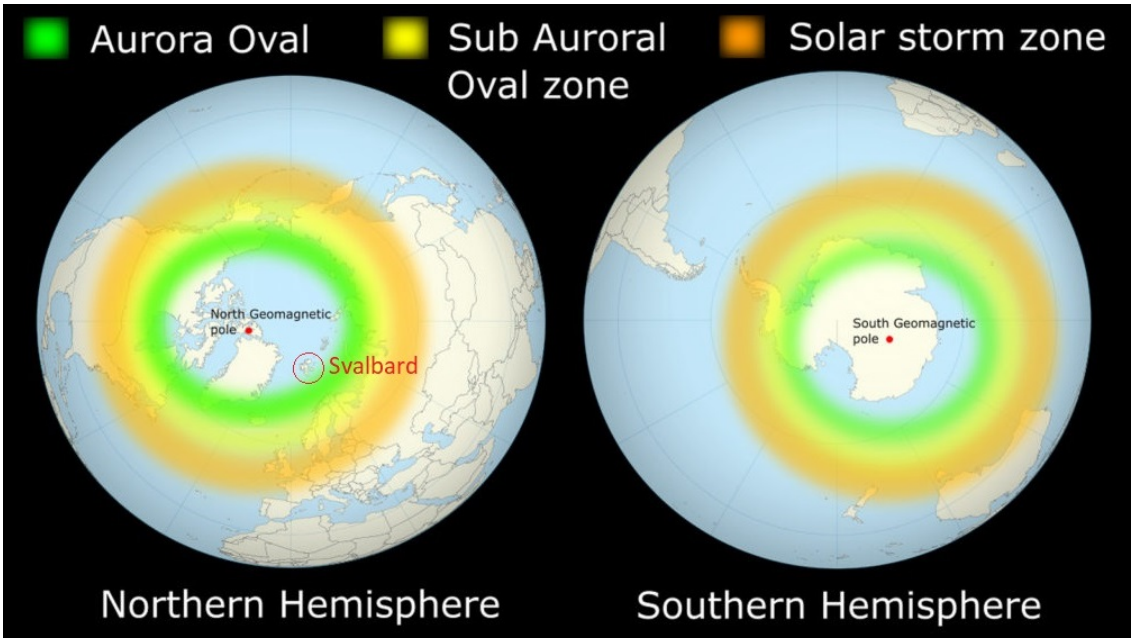


Figure 2.4.2: The auroral ovals for the northern and southern hemispheres, with markings of Svalbard and the geomagnetic poles. The oval latitudes are not exact, but gives an idea of the extent of the auroras during different solar activity intensities. Illustration by © William Copeland [19], with Svalbard edit.

2.4.2 Dayside and nightside aurora

Aurora can be divided into dayside and nightside aurora. Nightside aurora is a result of the Dungey cycle explained in chapter 2.3. The (accelerated) high-energy particles from the plasma sheet allows them to reach far down into the ionosphere. The collisions at lower altitudes, 100-200 km, means the aurora is dominated by green light. Nightside aurora have more discrete shapes, from a single arc to fast moving pattern covering the night sky. Nightside aurora can be observed at various latitudes based the incoming solar wind conditions, as the OCB expands and contracts. During average conditions, quiet auroral arcs are common, while an increase in solar wind speed and density (southward IMF) cause the auroral oval to widen and relocate southward. Svalbard is therefor not the most ideal location to observe nightside aurora during strong auroral events like substorms. A quiet aurora means that the shape and movement, as well as the brightness, only experience minor changes.

As seen from figure 2.2.1, a polar cusp is a dip in the magnetosphere where the magnetosheath plasma has direct access to the ionosphere. It is field-aligned currents (Birkeland currents) following the magnetic field lines than funnels the plasma into the ionosphere through the cusps, creating dayside (cusp) aurora [20]. The polar cusps existence are independent of the direction of the IMF [21], but the size and location are not. The slowed solar wind/magnetosheath particles have less energy than the particles causing the nightside aurora. As the particles mainly interact with oxygen atoms at altitudes from above 200 km, dayside aurora mainly consist of red light. The low-energy particles are long-lived, with a lifetime of 110 seconds. When the red photon finally radiates, the aurora have moved. The result is that the detailed structures of the aurora become blurred and lead to diffuse red aurora. This make the dayside aurora a weak display compared to the vibrant colors and shapes of the nightside aurora. Dayside aurora also have some green, and sometimes blue-violet, aurora present below the red layer, which creates more discrete shapes.

Both aurora mechanisms feeds the auroral oval with energetic particles, but at different locations on the oval. The cusp aurora is a smaller belt on the dayside of the auroral oval, illustrated in figure 2.4.3. Magnetotail reconnection feeds the nightside of the auroral oval in a broader belt than the cusp aurora. It has been found that the location of the center of the cusp is located at an average invariant latitude 80.3° at noon, and 78.7° at 0800 and 1600 MLT (Magnetic Local Time) [22]. The latitude varies about $\pm 6°$ based on solar wind conditions, and the width of the cusp widens slightly for increased solar wind conditions. With an increased northward IMF value, the center of the cusp barely move, but with an increased southward IMF value ($\sim 10nT$), the center move toward equator to about 73° [22].

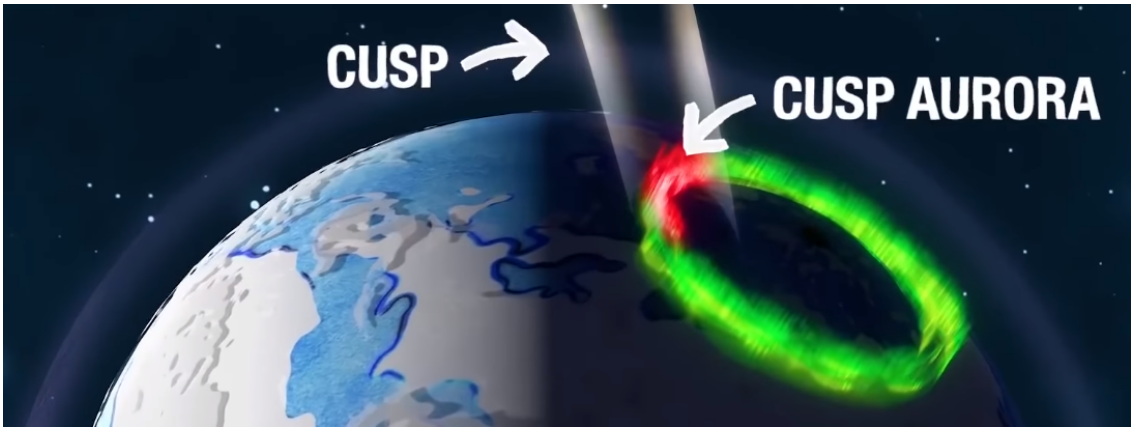


Figure 2.4.3: An illustration of the cusp and cusp aurora [23]. Cusp aurora appear in a smaller belt on the day side of Earth. Because of the tilt of the Earth, locations like Svalbard have periods with polar nights, allowing the cusp aurora to be studied as the Sun is below the horizon.

2.4.3 Aurora colours/emission lines

The main distinctive colours of the aurora are dependent on the altitude and the composition of the ionosphere. Red aurora occur at altitudes 200 km or above, where nitrogen molecules and oxygen atoms collide with energetic particles. This process mainly emits photons/light at 6300 Å for oxygen and higher wavelengths for nitrogen. Green-yellow aurora comes from collisions with atomic oxygen, and occurs at lower altitudes around 100-200 km. This is the strongest emission line and has a wavelength of 5577 Å. Green aurora is also the most striking because the human eye is more sensitive to green light. Blue-violet aurora comes from ionized nitrogen, and only occurs during intense events. The strongest emission line has a wavelength of 4278 Å.

Table 2.4.1 shows some details over the most common aurora colours. For green aurora, the energy is higher, and therefor the aurora appears at a lower altitude than red aurora [18]. At the bottom of the aurora, the colour often looks blue-violet, where some dark red light is mixed in. It is also normal to see yellow aurora near the top, between the green and red aurora.

Table 2.4.1: Colours of the aurora light at various altitudes. Energetic particles react/collide with atmospheric oxygen [O] and nitrogen [N]. The altitudes are not exact, various sources states different to-from limits.

Colour	**Wavelength [Å]**	**Reaction with**		**Altitude [km]**
Red	6300	O	Atom	$\sim 200 \rightarrow$
Green-Yellow	5577	O	Atom	$\sim$ 100 - 200
Dark Red	6500-6800	N_2	Molecule	$\sim$ 80 - 100
Blue-Violet	4278	N_2^+	Ion	$\sim$ 80 - 100

2.5 Geomagnetic substorms

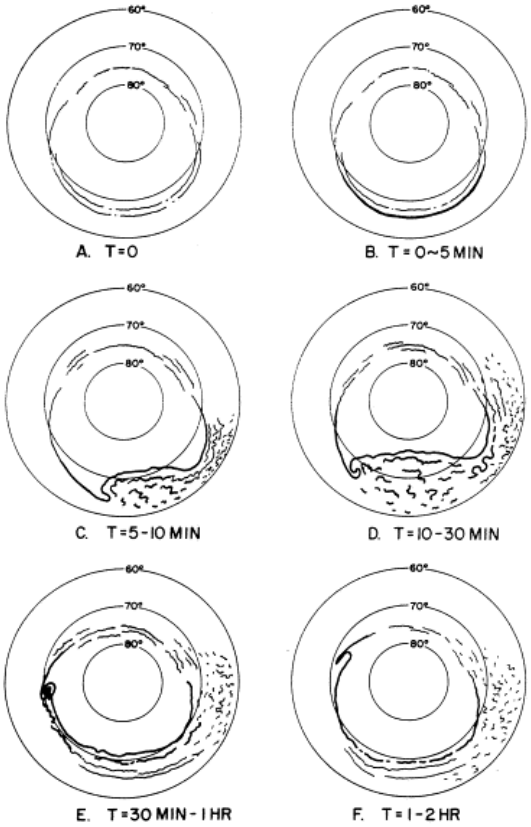


Figure 2.5.1: Auroral substorm temporal evolution. Drawing by S.-I. Akasofu [24].

Substorms are a result of the solar wind-magnetospheric reconnection. Figure 2.5.1 illustrates a substorm based on observations done by All-Sky Imager data from different locations around the hemisphere [24]. A substorm is divided into three phases; the growth phase, the expansion phase and the recovery phase [25].

The growth phase begins when dayside reconnection (southward IMF) occurs at the magnetopause. During this phase it is an increase of geomagnetic flux in the magnetotail, the OCB expands, and it is the phase before the aurora becomes active. There is a quiet auroral arc that drift equator wards, as seen in figure 2.5.1, ”A”. The active part of the substorm, the expansion phase, starts when nighttime reconnection occurs. This is when the auroral arc suddenly brightens, as seen in ”B” in the figure. During this phase the geomagnetic disturbance in the ionosphere increases. The phase lasts about 30 minutes, and there is a rapid expansion of the polar cusp size, meaning the auroral oval widens and moves to lower latitudes. As seen in the figure (”C”-”E”), the auroral display explodes, forming moving arcs and discrete aurora. The recovery phase, when nightside reconnection dominates, ends the substorm and can last for about an hour. During this phase the plasma sheet recovers, the OCB contracts and the aurora display quiets down into diffuse aurora. After the substorm, the OCB is back to the initial state, and quiet auroral arcs reappears, if any.

3 Machine Learning and Neural Networks

Machine Learning (ML), a subfield of Artificial Intelligence, allows computer algorithms to learn through data observations without explicit instructions. The algorithms learn to detect meaningful patterns and improve automatically through experience [26]. ML is divided into two main field: supervised and unsupervised learning. In supervised learning, a ML model learns from pre-labled data. In unsupervised learning, a ML model learns from unlabeld data where the output is unknown. This thesis uses supervised learning, more specifically, the subcategory classification. In classification the goal is to make a model that can predict class labels on unseen data based on previous observations.

Neural network (NN) is one of the most used ML methods in computer science today. NN are supervised machine learning techniques that are inspired by the biological neural system in the brain, where neurons (are modeled) to send information in form of mathematical functions between layers. These networks can be trained to solve arbitrary complex problems, such as speech recognition, object detection in images and finances. A widely used network for image classification is a convolutional neural network (CNN) [27]. The CNN architecture is specifically designed for image processing as the network directly processes pixel data as input and learns how to extract features like shapes, textures and patterns to detect objects [28]. Other ML methods need to preprocess the image to extract these features. To further explain CNNs, we will use the basic feedforward neural network (FFNN) to explain the main building blocks of neural networks. The chapter is based on lecture notes by Morten Hjorth-Jensen [28].

3.1 Feedforward neural network

The FFNN, or multilayer perceptron, consists of three basic components: the input layer, one or more hidden layers and the output layer. Each layer is made up by a number of neurons, or nodes. Each neuron in a layer is connected to all the neurons in the next layer, making it a fully connected network, see figure 3.1.1. The input layer, holding the input neurons, represents the real valued inputs from the data given to the network. The output layer, holding the output neurons, gives the predicted values after the network has processed the input. The network dimension is defined by the depth (the number of layers) and the width (the number of neurons in a hidden layer) [29].

The connections between neurons means the output from one layer is used as input in the next layer. Each connection is represented by this input, as well as a respective weight variable w and an added bias b. The connections in a FFNN can only process information in a forward direction, meaning the neurons can't form a cycle [27].

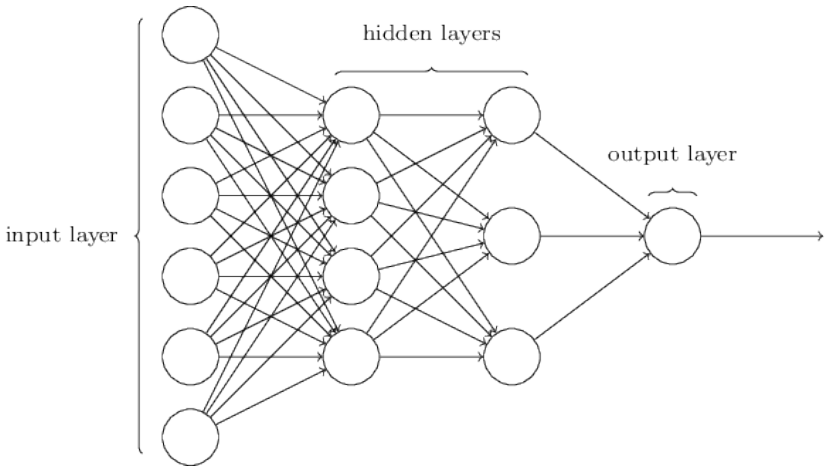


Figure 3.1.1: Fully connected feedforward neural network architecture with two hidden layers [27]. The circles represent neurons and the arrows represent the connection between neurons in different layers. All the arrow heads are connected to a neuron in the next layer, meaning information is only processed in one direction (forward).

When constructing a mathematical model for the network, we start by examining the case for a single hidden layer and n input neurons. The weighted linear sum z_j for each neuron j is given by

$$z_j = \sum_{i=0}^{n-1} w_{ij} x_i + b_j, \tag{1}$$

where x_i is the data input for input neuron i. The weight goes from the i-th neuron to the j-th neuron in the hidden layer and b_j is the bias for the j-th neuron. The resulting z_j is the input argument to the activation function σ (see chapter 3.2) for output y_j.

$$y_j = \sigma(z_j) = \sigma\left(\sum_{i=0}^{n-1} w_{ij} x_i + b_j\right). \tag{2}$$

For a deeper network with multiple layers, the equation for an arbitrary layer l is given by

$$y_j^l = \sigma(z_j^l) = \sigma\left(\sum_k w_{jk}^l y_k^{l-1} + b_j^l\right), \tag{3}$$

where y_j^l is the activation of the j-th neuron in the l-th layer. The sum goes over the k neurons in the previous layer $l-1$. The input activation y_k^{l-1} is the output activation of the previous layer [30]. The expression can be generalized for a network with an arbitrary number L of fully connected layers, indexing $l = 1, ..., L$. Introducing vector and matrix notation, we let vectors $\mathbf{y}^l$ and $\mathbf{b}^l$ be the output and biases for layer l. σ^l is the activation function for layer l. We also introduce the weight matrix W^l that contain all the weights between every neuron from layer $l-1$ to neurons in layer l. Thus the activation from (3) becomes [27]

$$\mathbf{y}^l = \sigma^l(\mathbf{z}^l) = \sigma^l\left(W^l \mathbf{y}^{l-1} + \mathbf{b}^l\right). \tag{4}$$

The final output of the process can be expressed as [28]

$$\hat{\mathbf{y}} = \mathbf{y}^L = \sigma^L \left(W^L \sigma^{L-1} \left(W^{L-1} \left(... \sigma^1 \left(W^1 \mathbf{x} + \mathbf{b}^1 \right) ... \right) + \mathbf{b}^{L-1} \right) + \mathbf{b}^L \right). \quad (5)$$

Note that input $\mathbf{x} = \mathbf{y}^0$, where $\mathbf{x}$ is the independent input layer from the data. The shape of W^l is an $N_{l-1} \times N_l$ matrix and $\mathbf{y}^l$ and $\mathbf{b}^l$ are $N_l \times 1$ column vectors.

3.2 Activation functions

The universal approximation theorem [31] loosely states that a neural network with one or more hidden layer can give outputs that approximate any real-valued function to an arbitrary degree/accuracy. This is the case as long as the network has any "squashing" activation function [29]. The Sigmoid function, defined as (6), is a good option and makes sure the network can output non-linear functions of the input x [28]. The outputs of the function are values between 0 and 1.

$$\sigma_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

The rectified linear unit (ReLU) activation function has become recently more popular, and is often used with convolutional neural networks. ReLU is defined as

$$\sigma_{\text{ReLU}}(x) = max(0, x), \quad (7)$$

and was proposed by Nair and Hinton in 2010 [33]. It is known for its simplicity. ReLU is equal to x if $x \geq 0$ and equal to 0 if $x < 0$, while the differentiation outputs 0 or 1. The simplicity of the function gives faster computations and have showed improved performance results compared to deep NN using the Sigmoid activation function [34][35].

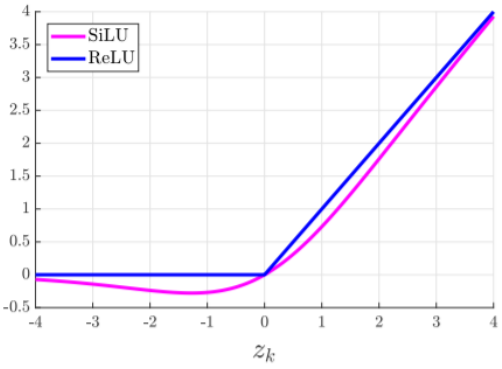


Figure 3.2.1: The shape of the SiLU and the ReLU activation functions [32].

More recently, a variation called the Sigmoid-weighted linear unit (SiLU) activation function has given even better results than Sigmoid, and is included in this thesis because of its use in the EfficientNet models. SiLU is the Sigmoid activation function multiplied with its input [32], defined as

$$\sigma_{SiLU}(x) = x \cdot \sigma_{\text{sigmoid}}(x). \quad (8)$$

When researchers have tested various activation functions on ImageNet [36], SiLU have performed similarly or better than the go-to ReLU when comparing classification accuracy [32][37]. In figure 3.2.1 we can see the different shapes of SiLU and ReLU. SiLU have a smoother shape, and allows some negative weights, while ReLU sets all negative weights to zero [32]. This leads to more active neurons, and a slightly more computational expensive option, when using SiLU.

For classification problems it is normal to add the Softmax activation function in the last layer. This function outputs values between 0 and 1 and the outputs also sum up to 1. These final outputs can therefore be interpreted as the network probabilities for a class to be true or false. The softmax function is derived by

$$\sigma_{\text{softmax}}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \tag{9}$$

where $\mathbf{x}$ is the input vector and x_i are the input vector elements. The sum is the normalization term, ensuring that the output values sums up to 1 for each class K in the classification problem. If we have multiple output classes, but only want to view the class with the highest probability, the argmax function [38] can be added to extract the class index with the highest probability.

3.3 Propagation, cost function and gradient descent

To make a neural network perform well the network has to be optimized, or trained. This is done by minimizing a cost function by optimizing the weights and biases for the network. The method used for this purpose is the so-called backpropagation combined with gradient descent. This chapter is based on chapter 2 of the book *Neural Networks and Deep Learning* by Nielsen (2015) [27].

From chapter 3.1 we know that the final network output for a FFNN can be expressed by matrix-vector multiplications. The initial information from the input $\mathbf{x}$ propagates through the network by forward propagation and outputs an prediction $\hat{\mathbf{y}}$ [29]. The output neuron with the highest activation is the prediction the network assumes as the best match for the corresponding input. The prediction is then compared with the true target $\mathbf{y}$, or the ground truth. The degree of error between them is calculated by the cost function C, also called the loss function, which tells the computer how well the network performs. A normal cost function to use is the mean squared error (MSE) derived as

$$C_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \text{mean}[(\mathbf{y} - \hat{\mathbf{y}})^2] \tag{10}$$

For this thesis, the so-called cross-entropy function was used, as it is a go-to function when working with CNN classification problems. The cross-entropy is derived as

$$C_{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_i \log \hat{\mathbf{y}}_i + (1 - y_i) \log(1 - \hat{\mathbf{y}}_i), \tag{11}$$

where K is the number of outputs/classes [30].

The weights and biases in the network are randomly initialized, which most likely results in a large error between the prediction and the true target. To reduce the error, we use gradient descent, or stochastic gradient descent, to minimize the cost function. The goal is to increase the value for the correct output neutron, and decrease the values for the incorrect output neurons. To calculate

the gradients based on the cost function for the gradient descent algorithms we have to use the backpropagation algorithm (Rumelhart et al., 1986) [30][39]. With backpropagation, information from the cost function can propagate backwards to calculate gradients by the partial derivatives

$$\frac{\partial C}{\partial w_{jk}^l} \qquad \text{and} \qquad \frac{\partial C}{\partial b_j^l}. \tag{12}$$

The weight parameters and bias associated with every neuron are updated by gradient descent methods

$$w_{jk}^l = w_{jk}^l - \eta \frac{\partial C}{\partial w_{jk}^l}, \qquad b_j^l = b_j^l - \eta \frac{\partial C}{\partial b_j^l}, \tag{13}$$

where η is the learning rate. Next we introduce an intermediate quantity δ_j^l. This quantity is the error in the j-th neuron in the l-th layer, and z is the weighted sum as introduced in (3).

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \tag{14}$$

For the output layer, in matrix-based form, the error is given by

$$\delta^L = \nabla_{\hat{\mathbf{y}}} C(\mathbf{y}, \hat{\mathbf{y}}) \odot \sigma^{L\prime}(\mathbf{z}^L). \tag{15}$$

$\nabla_{\hat{\mathbf{y}}} C(\mathbf{y}, \hat{\mathbf{y}})$ is the derivative of the loss function with respect to the output prediction and $\odot$ is the Hadamard product. $\mathbf{z}^L$ is the input for the activation function from (5). The error for a layer l can be expressed by

$$\delta^l = \left((W^{l+1})^T \delta^{l+1} \right) \odot \sigma^{l\prime}(\mathbf{z}^l), \tag{16}$$

where $(W^{l+1})^T$ is the transpose of the weight matrix in layer $l+1$ and $\mathbf{z}^l$ is the input for the activation function from (4). This means we need to know the error in the $l+1$ layer, which would be the output layer as the first calculation. The transpose of W propagate the error backwards to the output of layer l. The Hadamard product $\odot \sigma^{l\prime}(\mathbf{z}^l)$ propagates the error backwards through the activation function in layer l. Backpropagation gives us a way to calculate the rate of change of the cost function with respect to any weight parameter and biases in the network. The rate of change is given by the two equations

$$\frac{\partial C}{\partial w_{jk}^l} = y_k^{l-1} \delta_j^l \qquad \text{and} \qquad \frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{17}$$

The process can be thought of as when the gradients are calculated, we take a small step in the negative direction to close in on a local minimum of the cost function, to minimize the error between the output and the ground truth. This process is performed layer by layer, but the input layer is not updated with new

parameters as it contains the input data. The universal approximation theorem guarantees that there exist a weight and bias combination that results in an arbitrarily small error. However, it does not offer any method on how that combination can be found. For that reason, gradient descent based algorithms coupled with backpropogation is used.

3.4 Convolutional neural network

A convolutional neural network is an important deep learning algorithm that uses raw image data directly to extract features in the image. The CNN automatically learns which image features are important for predicting the correct class. CNNs are a subset of neural networks and they are mainly used for image related analysis.

Similar to a FFNN, a CNN architecture consists of an input layer, hidden layers and an output layer. The fundamental hidden layers are the convolutional layer and the pooling layer, as well as an activation function. The architecture of a basic CNN can be seen in figure 3.4.1, where we see that the network uses multiple convolutional layers and pooling layers. Different convolutional layers will enable the network to detect and learn different features (such as: edges, colors or shapes/objects). The network will make sense of these features while training on a set of images with a known ground truth. A CNN can also include layers like a normalization layer and a dropout layer. At the end of the network architecture there is a fully-connected layer, also known as a dense layer, with a softmax activation function that outputs the network predictions. The reason we add a fully-connected layer is that not all neurons in one layer are fully connected to all neurons in the previous layer, unlike from a FFNN [28].

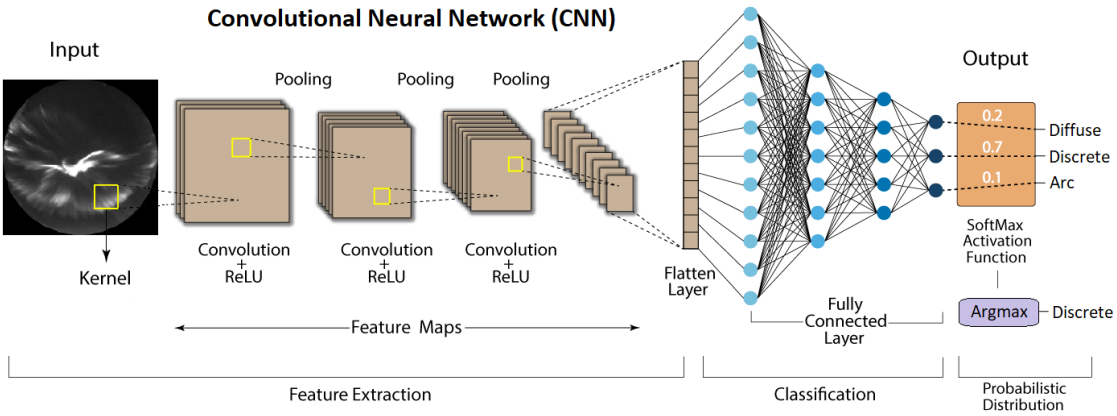


Figure 3.4.1: Illustration of a typical CNN architecture. The network use convolutional layers + ReLU functions and pooling layers for feature extraction. The output feature maps are flattened into a single array and serves as input for multiple fully connected layers who classifying the input. At the end, the network uses a softmax activation function to output class probabilities. Image Source: Google Images. The illustration is slightly altered by changing the input image and outputs.

3.4.1 Convolutional layer

The most important layer in a CNN is the convolutional layer. The main job of this layer is to extract features from the input image, or feature map, by scanning over the input with a smaller kernel, or filter. The feature map is represented as a three-dimensional grid, where each grid value represents a pixel value [29]. The height and width of the grid are the image dimensions, and the depth (different from network depth) is represented by the image RGB channels. The kernel is systematically applied left to right and top to bottom, as visualized in figure 3.4.2. A normal kernel size is 3x3 or 5x5 pixels, and the kernel moves with a stride of n-pixels. One kernel location above the input map calculates a single value for the three-dimensional output feature map, due to element-wise multiplication. Note that the figure shows a two-dimensional example, e.g. a grey image with only one depth channel. The bottom example in the image shows an input image with zero padding. This padding preserves the input size and helps preserving features located at the edges.

This process outputs a new feature map that highlights the areas the kernel feature correlates to features in the image. Normally, the convolutional layer have multiple kernels, each detecting different types of features. This means that the layer creates a stack of new output images/feature maps. Lastly, the output is passed through a non-linearity, like the ReLU activation function, before becoming the input of the next layer in the network.

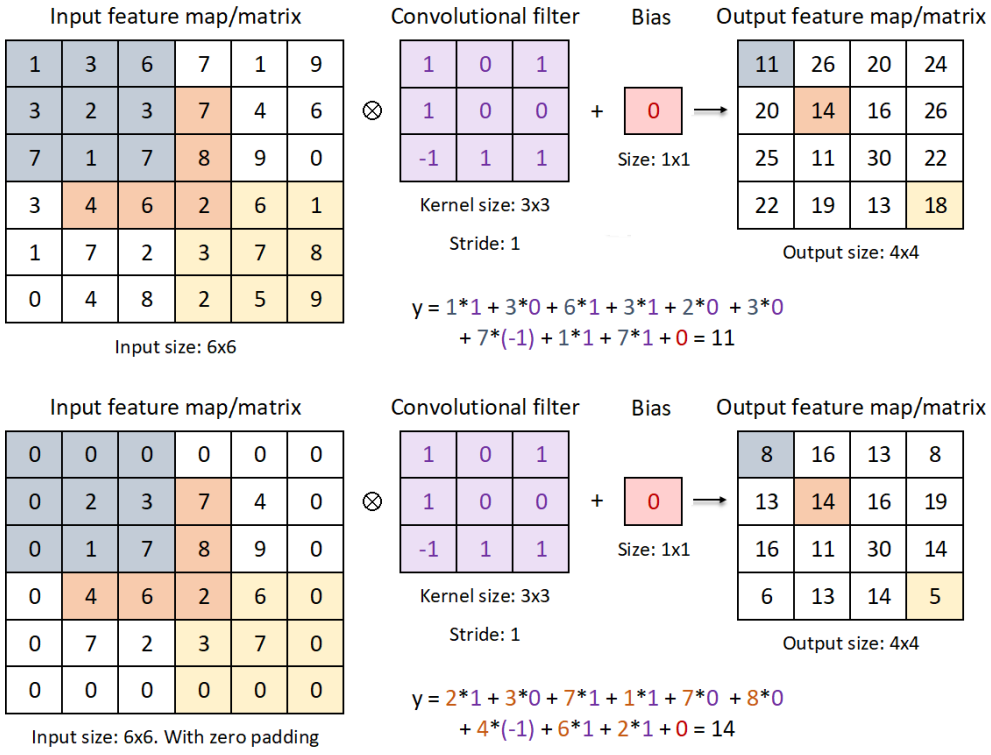


Figure 3.4.2: The convolutional layer with one input and one output channel, kernel of size 3x3 and stride of 1. The upper output calculation is for $y_{1,1}$ and the lower calculation is for $y_{2,2}$, where the output is with zero padding to preserve the input image shape.

3.4.2 Pooling layer

The purpose of the pooling layer is to reduce the spatial size of the feature maps while keeping the most important feature information intact. This process reduces the total amount of network parameters and thus results in fewer parameters the network needs to learn. This results in less computational time and the memory constraint is also reduced. The visualisation in figure 3.4.3 shows how average pooling, min pooling and max pooling works on a 4x4 input matrix with a 2x2 kernel matrix and stride of 2. The average pooling layer calculates the average of each patch covered by the kernel, while min/max pooling keeps the lowest/highest value in a region. Average and max pooling are the two most commonly used.

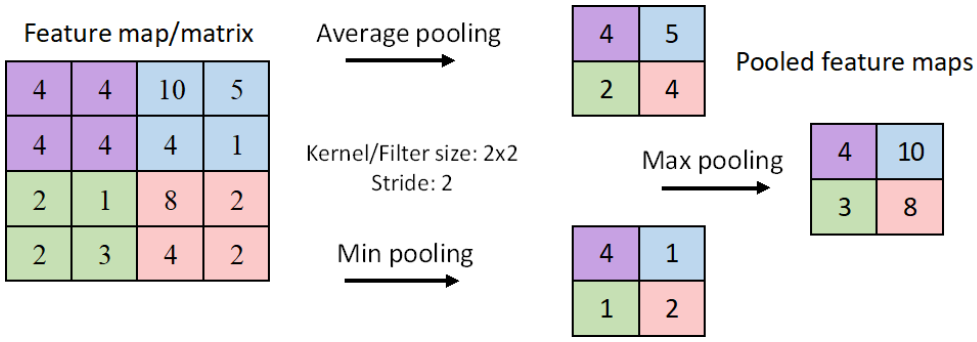


Figure 3.4.3: Illustration of average pooling, min pooling and max pooling. Kernel matrix size of 2x2 and stride of 2.

The pooling layer is normally added after the convolutional layer and activation function in the CNN architecture, but it does not have to be added every time. A downside of using pooling layers are that important feature information can be lost in the transformation, since the original matrix size is reduces by removing (feature) values.

3.4.3 Fully connected layer

After the feature extraction performed by the multiple layers of convolutions and padding, the CNN needs to perform classification. The classification part is performed by fully connected layers, making them the last layers of the CNN. As the name suggests, the main purpose of these layers is to connect all the neurons in the network, just like a feedforward neural network.

The feature maps from the previous output layer first gets flattened into a one-dimensional feature array, before being used as the input for the fully connected layers. Now, every value in the fully connected layer gets a say in what the final output of the classification will be. By using the softmax activation function at the end, the model's predicted outputs are presented as a probability distribution. This distribution holds the probability score (between 0 and 1) for each possible class label the input image can have.

3.4.4 Overfitting, dropout layer and data augmentation

Overfitting is when the network performs well on the training data, but not on the unseen validation/test data. This often occurs if the network trains too long. The network stops learning which features are important for the various classes, but memorizes the training data instead.

One way to detect overfitting is to follow the training vs. validation loss (or error) during a number of epochs. Figure 3.4.4 illustrates what underfitting, overfitting and the optimal solution looks like. We know the network fits to closely to the training set when the validation loss increases, and the training and validation losses diverge. Finding the optimal solution can be difficult, and is dependent on finding a good learning rate for the network to efficiently reduce loss. A good optimizer that handles the learning rate for a CNN is the gradient-descent based algorithm ADAM (Adaptive moment estimation) [40]. Another way to detect overfitting is to compare the validation accuracy and test accuracy. We do not want the test accuracy to be significantly lower than the validation accuracy.

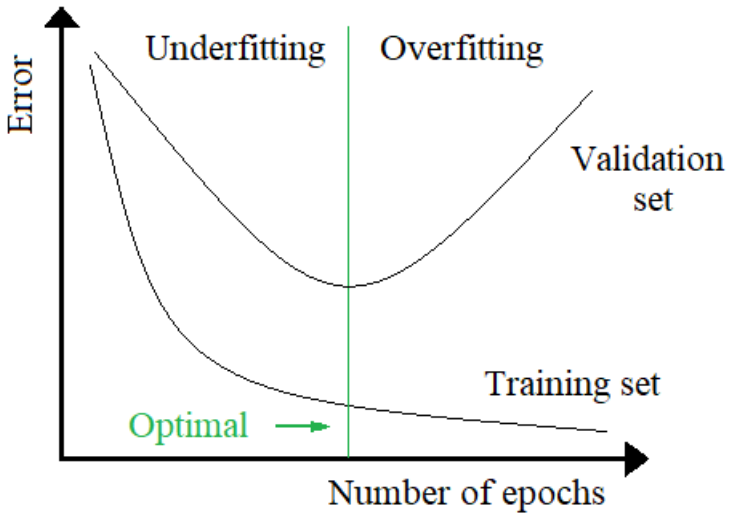


Figure 3.4.4: Error/Loss as a function of epochs. The graph shows which areas that increase the chance of underfitting and overfitting. The optimal point/area is in between.

Two techniques to prevent overfitting when building a CNN are dropout and data augmentation.

When implementing a dropout layer in the network, a percentage of random neurons are temporarily switched off during the training process. This means that the connections between neurons are switched off as well. The network learning should in theory improve, because the chance of memorizing specific training set features instead of learning features are reduced. Dropout is normally used with/after the fully connected layer.

Data augmentation is a method of artificially increasing the size of the training data without adding new data. Instead, by implementing random data transformations, we increase the data with slightly modified versions of existing data.

This prevents overfitting by creating more variation in the training set, reducing the model's chance of focusing to much on a few specific features. Examples of data augmentation are image rotating, flipping, scaling, or padding. These are related to image position. Colour augmentation can also be used, where for example the image contrast or brightness can be altered.

3.4.5 Imbalanced data

When working with classification problems, it is normal for the amount of data for each class to differ. With an imbalanced dataset, the accuracy is biased toward the dominant class. If e.g. 95% of the data is class D, the model will get a 95% accuracy if all samples are classified as class D. This model is not able to correctly predict the non-dominant class, making it useless. For this thesis, we looked at two methods to solve this problem.

The first method tackles the problem by oversampling with Pytorch's weighted random sampler.The second method is to use the weight parameter in Pytorch's Cross-Entropy loss function. Both methods uses the same re-scaling weights based on the sizes of the classes. For the first method, the class weight has to be assigned to every image in a tensor, while the second method used the four weights directly (weight=[$w_{\text{class x}}$, $w_{\text{class y}}$, $w_{\text{class z}}$, ...]).

Both methods helps to reduce overfitting, and they were tested during network training. A weight is calculated by

$$w_{\text{class x}} = \frac{\text{Size of the largest class}}{\text{Size of class x}} \tag{18}$$

3.5 Model evaluation

To evaluate the performance of the classifier some useful metrics are introduced in the following sub-chapters.

3.5.1 Confusion matrix and F1 score

A confusion matrix describes the performance of the classifier by counting the model predictions and arranging them in a matrix according to the actual class. The matrix is therefore used on a data set where the true classes are known, e.g. in supervised learning. For this thesis, the confusion matrix will be used on the validation and test sets of the pre-labeled images.

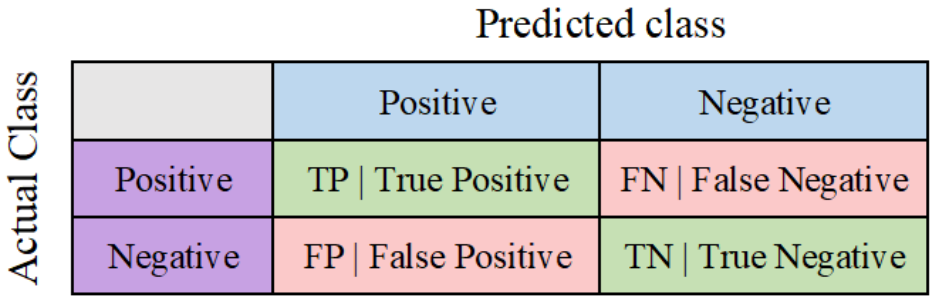


Figure 3.5.1: Confusion matrix for a binary classification problem.

Figure 3.5.1 shows a confusion matrix for two classes; Positive and Negative. The confusion matrix tells us that the classifier indicates the class correctly if we have *True*, and incorrectly if we have *False*. The four outputs of the confusion matrix is the TP, TN, FP (type 1 error) and FN (type 2 error).

TP The nr. of correct predictions, when the actual class was Positive.

TN The nr. of correct predictions, when the actual class was Negative.

FP The nr. of wrong predictions when the actual class was Positive.

FN The nr. of wrong predictions when the actual class was Negative.

From the output, we can calculate the precision, recall and F1 scores [41]. The precision score is defined as the ratio between the true Positives and the total number of Positives predicted by the classifier [42]. In other words, the precision score is how often the predicted Positives are actually correct.

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{19}$$

The recall score, or sensitivity, is defined as the ratio between the true Positives and the true Positives plus the false Negatives [43]. In other words, the recall score is how often the observed Positives are actually predicted correctly.

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{20}$$

The F1 score is calculated to evaluate the classifiers accuracy based on the precision and recall, as the harmonic mean [44].

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{21}$$

All three scores have output between 0 and 1, where 1 is a perfect score, and can easily be implemented by using scikit-learn [45]. The F1 score is very useful for classification problems with imbalanced classes, where classification accuracy can be misleading, see chapter 3.4.5 and 3.5.3. By maximizing the F1 score we find a good balance between the precision and recall.

3.5.2 Multi-class confusion matrix and F1 score

When we have a classification problem with multiple classes, like classifying aurora shapes, the metric calculations have to be done for each class. The scores are calculated for the chosen class against all the confusion matrix outputs of the remaining classes.

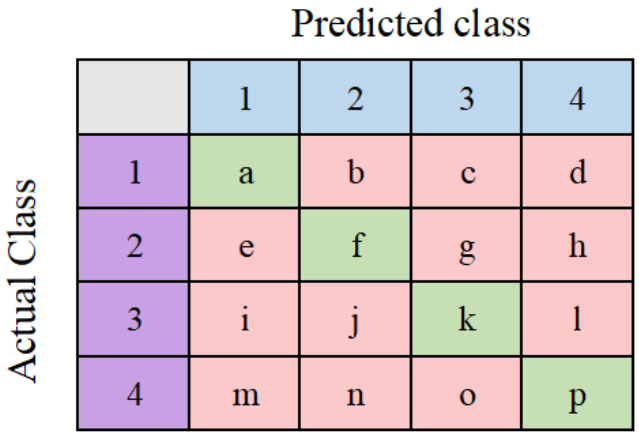


Figure 3.5.2: Confusion matrix for a multi-class classification problem. a, f, k and p are the True Positive values.

The single class precision score is calculated as the ratio between the class TP value and the class TP value plus the remaining values of the (actual) class row. The single class recall score is calculated as the ratio between the class TP value and the class TP value plus the remaining values of the (predicted) class column. The confusion matrix in figure 3.5.2 will further help to explain the calculations for class 1 by using (19) and (20);

$$\text{Precision[class=1]} = \frac{a}{a + (b + c + d)}$$
$$\text{Recall[class=1]} = \frac{a}{a + (e + i + m)}$$

The F1 score is calculated for each class by equation (21) and inserting the precision and recall for that class. The overall F1 score for the classifier is calculated

as an weighted average. This may result in a score not between precision and recall [44]. When using a weighted F1 score, each class is weighted by the number of samples of that class. Weighted averages can also be calculated for precision, recall and accuracy.

3.5.3 Classification accuracy

The classification accuracy measures the performance of the classifier. The score is the ratio between correctly predicted classes/targets t_i and total predictions made n, as defined in equation (22) [28]. The score is a intuitive metric, but is not considered the best for data sets that have imbalanced classes. This makes the classifier biased and and the classifier struggles to correctly predict minority classes. A perfect score equals to 1.

$$\text{Accuracy} = \frac{\sum_{i-1}^{n} I(t_i = y_i)}{n} \tag{22}$$

From the equation we have that the correctly predicted targets is the same as the model output y_i. I is the indicator function, where

$$I = \begin{cases} 1, & t_i = y_i \\ 0, & t_i \neq y_i \end{cases} \tag{23}$$

It is normal to see Top-5 accuracy scores in scientific articles and model comparison, normally with the ImageNet data set. The Top-5 score is the accuracy score for the actual class to be one of the five highest predicted classes. For this thesis, the conventional version of accuracy, the top-1 accuracy is used. This means when making a prediction, only the class with the highest probability output is accounted for.

4 Instruments and Locations

4.1 Aurora data

4.1.1 All-Sky Imager

When studying auroral activity and collecting data in form of images, the area of interest is the entire sky from horizon to horizon. To achieve the wanted effect, a 180 degree field-of-view is required. These images are taken by the All-Sky Imager (ASI), a ground-based camera with a circular fish-eye lens.

The instrument is light sensitive, pointing upwards, and protected by a transparent dome. Important components are the photon counter and the filter wheel [46]. The filter wheel holds one or more filters of certain emission wavelengths characteristic for auroral activity. The intensity/brightness of the aurora is measured by the photon counter. Due to the relative low brightness, images is taken with a 2 second exposure time, and only when the sky is dark [46]. The ASI takes an image with the different filters at different times. During 1 minute, the ASI takes four images with the 5577 Å filter, but only two images with 6300 Å filter.

Rain, snow and cloudy weather are the largest disadvantages for ASIs. Observations are also not possible when the moon is visible in the field-of-view, as the moonlight is to bright.

4.1.2 Ny-Ålesund, Svalbard

An ideal location for collecting aurora data is Svalbard. Therefore our data is collected from the ASI located close to the Sverdrup Research Station in Ny-Ålesund. The data and ASI is maintained by the University of Oslo.

The ASI has a Keo Sentry 4ix Monochromatic Imager from Keo Scientific. The camera takes 512x512 pixel images, and is equipped with 5577 Å and 6300 Å filters on a single-filter wheel [46]. Svalbard is considered an arctic desert, and therefor has low yearly precipitation. The seasons on Svalbard are characterized by the midnight sun and the dark period [47]. The dark period is when the Sun is located below the horizon, and two important periods are the astronomical twilight and polar nights. During the polar nights from early November to early February, the only light illuminating the sky is caused by auroras or the moon. This makes Svalbard the perfect place to observe dayside cusp aurora. The astronomical twilight expands the period (mid-to-late October to mid February), with a twilight effect during mid-day [48][47]. The climate and lighting conditions on Svalbard are due to the high latitude at 78.92°N (geomagnetic (AACGM) 76.24°N) [46].

4.2 Solar wind data

4.2.1 Advanced Composition Explorer spacecraft

The Advanced Composition Explorer (ACE), figure 4.2.1, is a spacecraft part of the NASA Explorers program. ACE observes the IMF and the solar wind, as well as energetic particles originating from regions outside our solar system [49]. A part of the ACE mission is to examine and collect data on the solar wind, and transmit the data to Earth. With the data, it is possible to investigate the solar wind (plasma) interaction with Earths magnetosphere and ionosphere. Real-time data can also be used for space weather forecasting and warnings of solar storms. ACE is orbiting the L_1 Lagrange point, an unstable point along the Earth-Sun line where the two gravitational forces between the masses are balanced [50]. ACE has been operating since 1998, and is predicted to continue its mission until 2024 [49].

4.2.2 Wind spacecraft

The Wind spacecraft, figure 4.2.2, is part of the Global Geospace Science program and is a solar wind laboratory [51]. Similar to ACE, Wind measures solar wind properties before it reaches and interacts with Earths magnetosphere. The IMF is also measured. The data is collected and transmitted to Earth. Wind has been operating at the L_1 Lagrange point in the Earth-Sun system since 2004 [51]. In this orbit, ACE has enough fuel to continue its mission until 2074 [52].

Figure 4.2.1: The ACE spacecraft. Measures solar wind and magnetic fields. Credit: NASA/B. Dunbar [53]

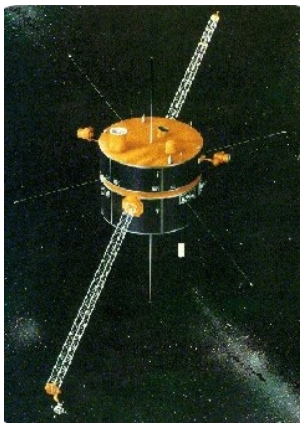


Figure 4.2.2: The Wind spacecraft. Measures solar wind properties. Credit: NASA/L. B. Wilson [54]

5 Data and Methodology

This chapter describes the data, classes and methodology used for this thesis. We started the thesis by manually labeling all original ASI data based on their features, and create a training database for the convolutional neural networks. The data described in chapter 5.1 include Ny-Ålesund ASI data from Svalbard All-Sky Imager Data [1] and OMNI data from NASA [2]. Since all our ASI data comes from Ny-Ålesund, we only work with northern lights images, which in general will be referred to as aurora. Chapter 5.2 gives an overview and examples of the four classes the aurora is divided into.

The next sections describe how the network was selected and trained by highlighting some of the implementation details. The final sections describe how the network was used to predict the class of unseen aurora images and statistics based on these predictions.

All programming was done in the programming language Python. For implementation, we used the Anaconda platform. Table 5.0.1 gives an overview of the main software and hardware specifications used. We trained all networks on a Nvidia RTX2080Ti GPU. To make a thesis that is not code based, all important scripts can be found in the following GitHub repository. Larger data sets are available from Dropbox.

- GitHub: https://github.com/kristinaothelia/AuroraCNN
- Dropbox: https://www.dropbox.com/sh/3pt4b8aeoh1xb4w/AADmpwfDE_1btXlbJjBsglUra?dl=0

Table 5.0.1: The main software and hardware specifications.

Name	Version	Description	
Software			
Python	3.8.5	Programming language	[55]
Anaconda	4.9.1	Python distribution platform	[56]
NumPy	1.20.0	Used for implementation	[57]
Matplotlib	4.3.4	Used for data visualization	[58]
Scikit-Learn	1.0.2	Used for model analysis	[59]
PyTorch	1.7.0	Open source ML library	[60]
CUDA	11.3	Used for PyTorch, GPU	
Hardware			
CPU		Intel Core i5-5200, Windows 10, 4 GB RAM	
GPU		Nvidia RTX2080Ti, Linux, 11 GB memory	

5.1 Data

The overview of available data at the ASI data site shows many days without entries. Missing ASI data from time points are mainly caused by the camera not operating. This could be caused by light pollution or if the moon was located in the camera field-of-view. Missing data in the OMNI data set are replaced by variations of ”9”-s combinations, see data explanation as mentioned above.

5.1.1 Data pre-processing

The camera used in the ASI has a 512x512 px resolution. Due to the camera lens, there is a black border around the area of interest in the image. The black border is removed during the calibration phase, and the final images on the ASI data site have a lower resolution, see table 5.1.1.

When the data is downloaded from the site, the images have very low intensity, making them look almost black. The images are therefore transformed using a Python script, 1, with Numpy's percentile method. The process brightens the image and brings out the different weather conditions and auroral shapes.

Algorithm 1 Scaling image intensity

img: raw input array
imgS: scaled output array

1: $\text{imgS} = \frac{\text{img}}{\text{np.percentile(img, 99)}}$
2: $\text{imgS}[\text{imgS} > 1] = 1$
3: $\text{imgS} = (\text{imgS}\ (2^{16} - 1)).\text{astype(np.uint16)}$

OMNI data are gathered directly from NASA and how the data is processed are explained at NASA's OMNIweb [61].

5.1.2 ASI data for CNNs

The data used for training a CNN model, were received in separate turns creating four sets of data, as listed in table 5.1.1. Data set 1 was received first, by a random draw of images from the data library. Further into the thesis, the data amount was increased by 6000 images, data set 2, 3 and 4. Green and red aurora data have almost identical timestamps. Since data set 1 and 2 were not generated at the same time, the dates do not exactly match. Generating the data sets at different times also resulted in a few overlapping images. Thus, the total data consists of 7980 images.

There is no definite way the data should be split during network training. For this thesis, we first extracted approximately 10% of the data for each class, creating a test set of unseen data (6: ASI_test). The remaining $\sim$ 90% makes up the data for training the classifier (5: ASI_train_valid).

Table 5.1.1: List of data sets consisting of All-Sky Imager data [1] of red and green Aurora Borealis above Ny-Ålesund, Svalbard.

Set	Name	Nr. of images	Resolution [px]	Wavelength [Å]
1	Green1	2000	$471x471$	5577
2	Red1	2000	$469x469$	6300
3	Green2	2000	$471x471$	5577
4	Red2	2000	$469x469$	6300
	Combined green and red aurora data sets			
5	ASI_train_valid	7182	$469x469$, $471x471$	5577, 6300
6	ASI_test	798	$469x469$, $471x471$	5577, 6300

5.1.3 ASI data for aurora predictions

ASI data of unseen aurora borealis to be classified by the CNN model were downloaded from the data library. The data chosen were from the Ny-Ålesund station, in 5577 Å and 6300 Å, as listed in table 5.1.2. Data for January, November and December from the years 2014, 2016, 2018 and 2020 were downloaded. Only these three months were chosen because of the the light conditions on Svalbard, as explained in chapter 4.1.2. All images went through a simple transformation, as explained by algorithm 1. The year 2014 and 2020 were chosen because they are solar maximum and solar minimum, respectively. The years 2016 and 2018 were included to have more data to compare statistical results with. The reason for the different data set sizes comes from the ASI taking twice as many 5577 Å images during a given time interval.

Table 5.1.2: Red and green Ny-Ålesund ASI data. Both data sets include available data for January, November and December from the years 2014, 2016, 2018 and 2020. All data are downloaded from the Svalbard All-Sky Imager Data website.

Set	Name	Nr. of images	Wavelength [Å]
1	ASI_red[1,2]	221 004	6300
2	ASI_green[1,2]	441 861	5577

1: Insert wanted wavelength and year in the web link
http://tid.uio.no/plasma/aurora/nya6/wl/yr
2: For January 2014, the data is from camera nya4, hence
http://tid.uio.no/plasma/aurora/nya4/wl/2014/

5.1.4 High resolution OMNI data

OMNI data is solar wind magnetic field and plasma data in a 1-min interval. Each entry is made up of a large number of parameters, which are time-shifted to the Earth's bow shock nose [61]. The OMNI data sets are generated at NASA OMNIWeb: High Resolution OMNI [2]. NASA gathers the data from spacecrafts like ACE and Wind. For this thesis, data sets for 2014, 2016, 2018 and 2020 were generated, with only a few selected parameters, including time point, B_z parameters, speed and density. To reduce the size of the data sets more, only data for the chosen months were included. The main purpose of this was to make the Python code run faster when matching OMNI data to ASI data.

Table 5.1.3: High-Resolution 1-min OMNI data sets [61].

Set	Name	Year	Months	Spacecraft[1]
1	omni_min2014	2014	Jan, Nov, Dec	51, 71, 99
2	omni_min2016	2016	Jan, Nov, Dec	51, 71, 99
3	omni_min2018	2018	Jan, Nov, Dec	51, 71, 99
4	omni_min2020	2020	Jan, Nov, Dec	51, 99

1: 51: Wind, 71: ACE, 99: Missing data

Further information about the data can be found under *One min and 5-min solar wind data sets at the Earth's bow shock nose, 4b* at OMNIWeb [61].

5.2 Classes

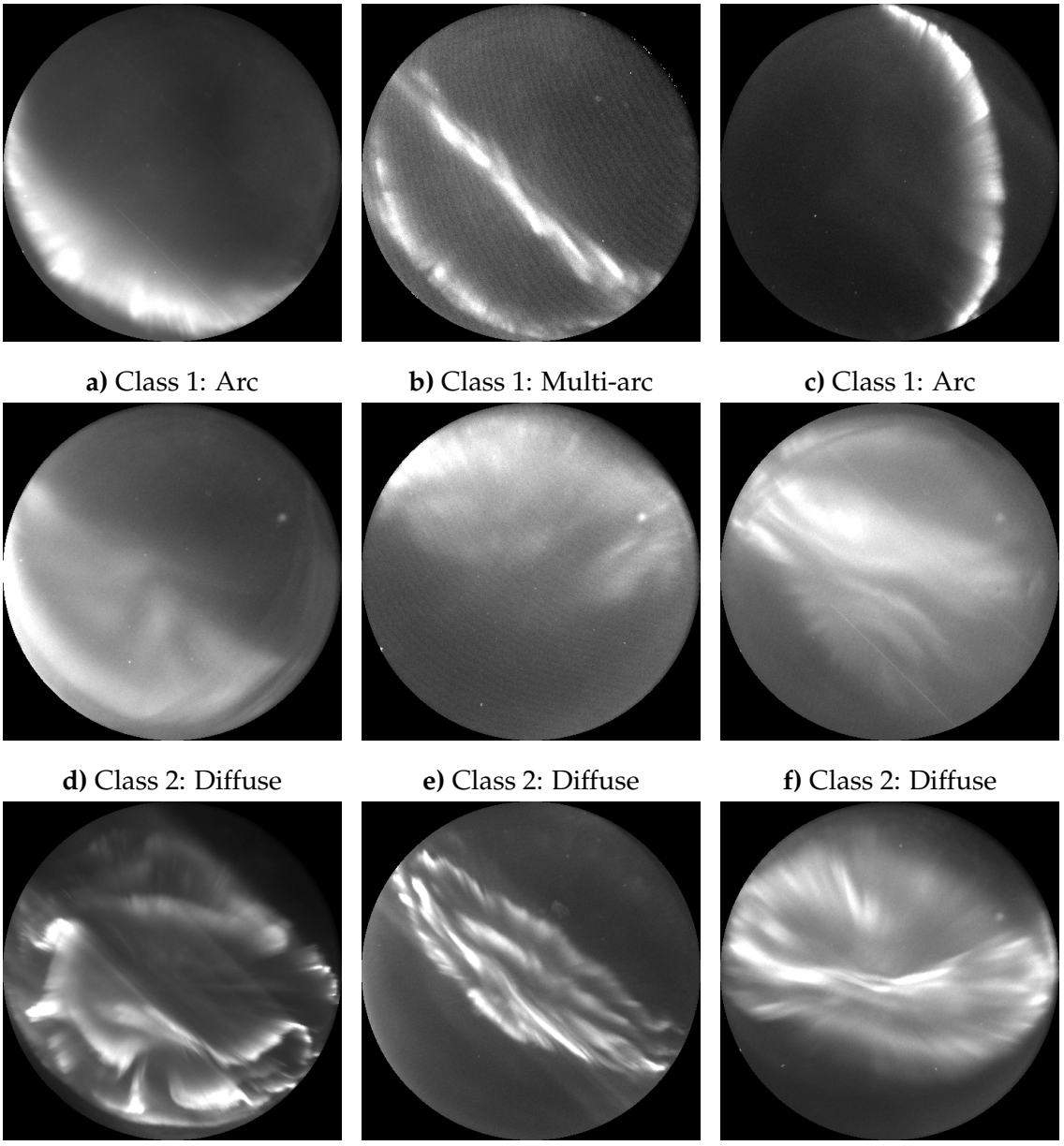
The images from the ASI_train_valid and ASI_test data, see chapter 5.1.2, need assigned class labels to act as the training, validation and test data for the supervised learning done by the CNN classifier. For the four classes $y \in [0, 1, 2, 3]$ in table 5.2.1, we have the named labels No Aurora, Arc, Diffuse and Discrete, respectively. The classes used for this thesis are slightly altered from the six classes defined by Clausen and Nickisch [9]. Compared to Clausen and Nickisch, we combined their *cloudy* and *clear/noaurora* into a single class, No Aurora. Their class for *moon* were not used.

Table 5.2.1: Description of the four classes $y \in [0, 1, 2, 3]$, which are based on Clausen and Nickisch definition [9].

Class y	**Label**	**Explanation**
0	No Aurora	No auroral activity. The image shows a clear sky, where stars and planets can be visible (dependent on the light intensity), or the sky is dominated by clouds or fog. The camera dome can be contaminated, for example by water or snow.
1	Arc	This label is used for images that show one or multiple bands of aurora that stretch across the field-of-view; typically, the arcs have well-defined, sharp edges.
2	Diffuse	Images that show large patches of aurora, typically with fuzzy edges, are placed in this category. Structured, but not well-defined. The auroral brightness is of the order of that of stars.
3	Discrete	The images show auroral forms with well-defined, sharp edges, that are, however, not arc like. Well-defined structures/shapes. The auroral brightness is high compared to that of stars.

Each image from the ASI data are labeled with one of the four classes by manually inspecting each image. They were inspected in black and white. Figure 5.2.1 and 5.2.2 contains example images of various (weather) conditions and which class they were assigned. Note that all the images in figure 5.2.1 were relatively easy to assign a single class. A good amount of the data are images with weak aurora display and/or images that could fit multiple classes.

Figure 5.2.1 shows three images for each of the classes arc, diffuse and discrete. In panels **a)-c)** there are images of arcs, where **b)** shows an example of a multi-arc. Panels **d)-f)** show example images of diffuse aurora and panels **g)-i)** show examples of discrete aurora.



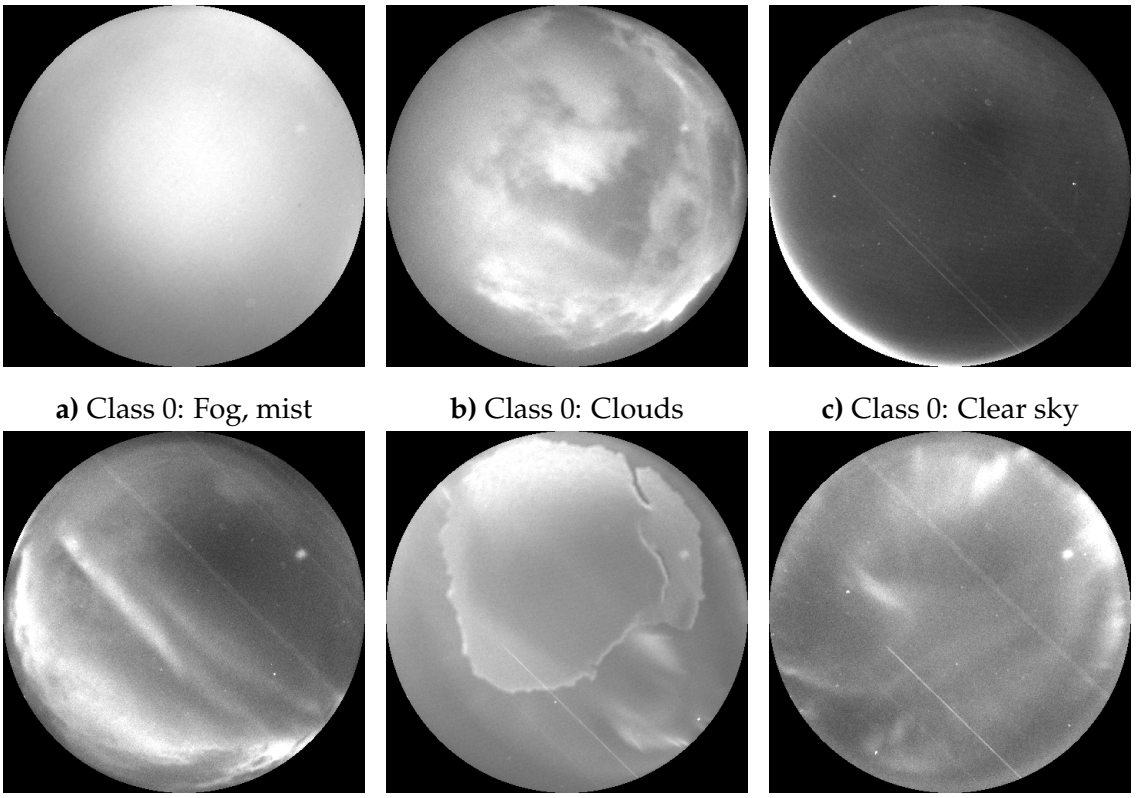
a) Class 1: Arc **b)** Class 1: Multi-arc **c)** Class 1: Arc

d) Class 2: Diffuse **e)** Class 2: Diffuse **f)** Class 2: Diffuse

g) Class 3: Discrete **h)** Class 3: Discrete **i)** Class 3: Discrete

Figure 5.2.1: ASI data of classes Arc, Diffuse and Discrete. Various forms of arcs are in panel **a)-c)**, diffuse in **d)-f)** and discrete in **g)-i)**.

Figure 5.2.2, panel **a)-c)**, shows examples of images placed in the no aurora class. This includes clear sky and weather conditions like fog and clouds. The images in panel **d)-f)** show examples where multiple conditions are met, or correct labeling is difficult. Panel **d)** show an example where we have clouds and aurora. Panel **e)** show an image with aurora detected, but where the dome is contaminated. In this case the class was set to diffuse. Panel **e)** and **f)** are examples of images that are difficult to assign a single class.



a) Class 0: Fog, mist

b) Class 0: Clouds

c) Class 0: Clear sky

d) Class 1: Clouds and two arc-like bands

e) Class 2: Some aurora with dome contamination

f) Class 2: Aurora image hard to label correctly

Figure 5.2.2: ASI data of classes No Aurora, Arc and Diffuse. No aurora occurs in panel **a)-c)**, and the weather conditions are foggy, cloudy and clear sky, respectively. The clear sky image has some light pollution that can be misinterpreted as an arc by the classifier. In panel **d)**, the weather condition is partly cloudy and some (multi) arcs can be detected. As we want to detect aurora, the image was classified as Arc. The images in panel **e)-f)** were labeled Diffuse. Panel **e)** shows an image with detectable aurora, but where the dome is contaminated.

In practice, the threshold between classes was hard to define when labeling images. The most dominant class feature ultimately determined the assigned label. Images that contains detectable aurora, at least by the human eye, and fog/-clouds, were set to an aurora class. Most of these images were set to diffuse because of dominating clouds hiding much of the aurora features. This means the actual aurora above the clouds may be arcs or discrete. Very dim and/or scattered aurora was also labeled diffuse.

5.3 Data sets

All data sets for this thesis are in JSON (JavaScript Object Notation) format.

5.3.1 ASI data sets for CNNs

The labeled data from ASI_train_valid and ASI_test, from chapter 5.1.2, were made into JSON data sets by using the python script **data_to_json.py**.

- ASI_train_valid.json, ASI_test.json

The training set was further split 80/20 into training and validation containers directly in Python. This was accomplished with a split function that randomly splits the data set. After some network training with unsatisfactory results, we had a new run through of the data using **correct_classes.py**. The class labels were manually corrected or approved image by image. The new information was directly saved to the JSON file.

Each entry in the data sets contains information about the image as shown in the example below. When testing the model and making predictions, a new JSON data test set is made with updated score values.

- ASI_test_predicted_efficientnet.b3.json


```
{
    "image_path": "Aurora_test\nya6_20191206_112953_5577_cal.png",
    "wavelength": "5577",
    "timepoint": "2019-12-06 11:29:53",
    "shape": [
        469,
        469
    ],
    "label": "diffuse",
    "human_prediction": True,
    "score": {},
    "solarwind": {}
}
```


The data distribution between the four classes is listed in table 6.2.1. ASI_CNN is the distribution between the total 7980 images from the combined ASI_train_valid and ASI_test data, while the ASI_train container represents only the training images. ASI_train contains $\sim$ 72% of the total data. We can see from the table that our data is imbalanced, where the no aurora class consists of 41% of the data. Arc is the smallest class, with 10%. Diffuse contains 21% and discrete contains 28% of the data.

The imbalance problem was corrected by using oversampling/class weighting for the training data set. ”Weights1” is the weights used to get balanced classes. ”Weights2” was used to highlight the importance of the underrepresented aurora classes. We used weights1 x1.5 for the three aurora classes, and let the no aurora class be. When training the network for two classes, the single class Aurora is a combination of the three aurora classes arc, diffuse and discrete. ”Weights3” was used when training with two classes.

Table 5.3.1: Data distribution (# of images) for the ASI_CNN and ASI_train data sets used for CNN training. ASI_train contains $\sim$ 72% of the data. Weights1 is to even out the unbalanced classes and Weights2 is for raising the importance of the aurora features during training, with x1.5 compared to Weights1. The weights are for balancing the training set. The validation and test sets are not weighted.

Set	No aurora	Arc	Diffuse	Discrete	Total
ASI_CNN	3280	823	1649	2228	7980
Distribution	41%	10%	21%	28%	100%
ASI_train	2332	602	1203	1610	5747
Weights1	1.0	3.9	1.9	1.5	
Weights2	1.0	5.8	2.9	2.2	
ASI_train, 2 classes	2332		3415		5747
Weights3	1.5		1.0		

5.3.2 ASI data sets for aurora predictions

The data from ASI_red and ASI_green, from chapter 5.1.3, are made into unlabeled data sets made for predictions and statistics (ASI_R.json and ASI_G.json). Different to the data sets used for network training and testing, these sets include information about the solar wind and IMF. The most important OMNI parameters for this thesis is the north-south magnetic field strength component B_z, named "B_z, nT (GSM)" and solar wind speed ("Speed, km/s").

As we wanted to match an aurora image to the associated OMNI parameters, we encountered a problem. The image of the aurora and the associated B_z value is not the B_z value at the identical time in the OMNI data set. Therefore, we had to get creative to match (time shift) the data. For daytime/dayside aurora (no Dungey cycle), the time the solar wind uses from the bow shock nose to the ionosphere above the magnetic poles are in the span of minutes. This is based on an average solar wind speed, and the distance of approximately 90,000 km from the bow shock to Earth [62]. For this thesis, we defined dayside aurora for hours between 06-17. Therefore, the mean of the B_z timepoint $\pm$3 minute time interval were added as the parameter values. The standard deviation was also calculated and exported to the "solarwind" dictionary in the JSON files.

For nighttime/nightside aurora (with Dungey cycle), the time the energetic particles use from the bow shock nose to the ionosphere are longer than for the dayside. For this thesis, we defined nightside aurora for hours between 18-05. This exact time is dependent on multiple variables, and hard to calculate exactly. Therefore we used the B_z timepoint for one hour before the aurora timepoint, as

well as a mean over half an hour (timepoint $\pm$15 minutes). The time-shift of one hour was chosen because of the temporal evolution of substorms, see chapter 2.5. If the recorded B_z values were missing, the entry was dropped, giving fewer points to calculate the mean and standard deviation. However, if all values during the time interval were missing, the no-data number was added (a combination of 9's, dependent on the OMNI data parameter). Calculating the mean and standard deviation were done for all parameters and added to the JSON file.

The next step is using the model to add a class label ("label") and predicted class scores ("score"). To keep the original data set unchanged, we made new datasets with these prediction results. A file entry from the dataset with predictions made looks like the example below. At the end, we have two JSON files for the same data. One with and one without a class label and prediction scores.

- ASI_R.json, ASI_R_predicted_efficientnet-b3.json
- ASI_G.json, ASI_G_predicted_efficientnet-b3.json


```
{
    "image_path": "/Master/T_DATA_green/nya6_20141222_142652_5577_cal.png",
    "wavelength": "5577",
    "timepoint": "2014-12-22 14:26:52",
    "shape": [
        469,
        469
    ],
    "label": "diffuse",
    "human_prediction": false,
    "score": {
        "aurora-less": 0.017253359779715538,
        "arc": 0.04337286576628685,
        "diffuse": 0.9202288389205933,
        "discrete": 0.01914495974779129
    },
    "solarwind": {
        "Bz, nT (GSM)": "9.72",
        "Bz, nT (GSM), SD": "0.16",
        "Speed, km/s": "375.09",
        "Speed, km/s, SD": "0.60",
        "Proton Density, n/cc": "6.27",
        "Proton Density, n/cc, SD": "0.70"
    }
},
```

5.4 EfficientNet

When designing a convolutional neural network for the thesis, we ended up with unsatisfactory accuracy results on self-made networks. We therefore looked into well-known CNNs and ended up with using the state-of-the-art network called EfficientNet. EfficientNet is a CNN that consists of a family of scaled networks, where the network complexity increases from the baseline network (EfficientNet-B0). EfficientNet was released open source by Google Brain in 2019, with the paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" [63]. The EfficientNet information and results are from this paper.

One of the main advantages of using EfficientNet is to take advantage of the effort made to efficiently scale up the baseline network of the family, gaining higher classification accuracy while keeping the number of parameters low. Instead of only scaling up e.g. the number of layers, EfficientNet uniformly scales the network depth (number of layers in the network), width (number of channels/feature maps in each layer) and resolution (input image) by using fixed scaling coefficients [63]. The architecture of EfficientNet-B0, the baseline network, is described by table 5.4.1 [63]. EfficientNet uses the SiLU activation function, see eq. (8).

Table 5.4.1: EfficientNet-B0 (baseline network) architecture [63].

Stage	Operator	Resolution	#Channels	#Layers
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

A way to see how good an image classification model performs is to test it on ImageNet. ImageNet is a very large database of about 14 million hand-labeled images over 20 000 different classes/categories. A subset ILSVRC (Russakovsky et al., 2015. [36]) is widely used for testing models, with approximately 1.2 million labeled test images over 1000 different classes/categories.

The EfficientNet performance results on ImageNet (ILSVRC) showed how the scaling method lead to better accuracy and efficiency than existing CNN models. EfficientNet-B0 achieved an Top-1 accuracy of 77.1%, with 5.3M parameters 0.39B FLOPS (Floating Point Operations Per Second). Compared to DenseNet-169 (Huang et al., 2017. [64]) with 76.2% accuracy, have 14M parameters and 3.5B FLOPS. The scaled up model, EfficientNet-B7, achieved state of the art Top-1 accuracy of 84.3%, with only 66M parameters and 37B FLOPS. In comparison, CNN model GiPipe (Huang et al., 2018. [65]) with the same accuracy, had 8.4x as many parameters [63]. Figure 5.4.1 shows EfficientNet models Top-1 accuracy as a function of parameters compared to other CNN models.

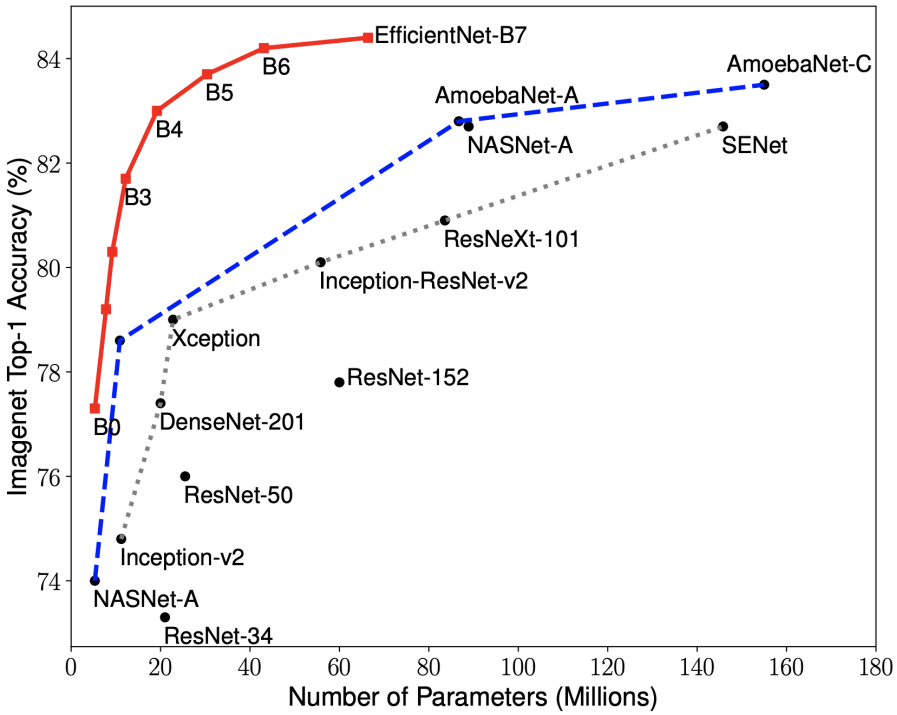


Figure 5.4.1: Model size vs. Top-1 accuracy on ImageNet [63].

Checking the leaderboard for "Image Classification on ImageNet" (19.9.21 [66]), EfficientNet based models are no longer ranking as the best, but they are doing very good based on accuracy vs number of parameters. In #6, Meta Pseudo Labels - EfficientNet-B6-Wide, has 390M parameters, while better ranking models have 1470M-2440M parameters. The Top-1 accuracy differ less than 1%.

5.5 Training, validation and testing

5.5.1 Models and hyper-parameters

EfficientNet was implemented through the CRAI-Nets project [6]. We chose to mainly focus on EfficientNet-B3 as this model have a good accuracy-parameter ratio, providing a usable training time. Table 5.5.1 lists the dropout rate, width coefficient, depth coefficient and image resolution for for each model.

Table 5.5.1: The EfficientNet family [63] parameter coefficients for models B0 to B7.

Model	Image resolution [px]	Width coefficient	Depth coefficient	Dropout rate
EfficientNet-B0	224	1.0	1.0	0.2
EfficientNet-B1	240	1.0	1.1	0.2
EfficientNet-B2	260	1.1	1.2	0.3
EfficientNet-B3	300	1.2	1.4	0.3
EfficientNet-B4	380	1.4	1.8	0.4
EfficientNet-B5	456	1.6	2.2	0.4
EfficientNet-B6	528	1.8	2.6	0.5
EfficientNet-B7	600	2.0	3.1	0.5

Before training the CNN, the input data has to be transformed. Torchvision's transforms function were used for this purpose, where the data are first made into numpy float objects and then torch tensors. The data augmentation technique of randomly rotating the data was implemented to reduce overfitting. We used torch's interpolation function to interpolate the data, where modes bilinear and bicubic were tested. The size parameter is set to the image resolution associated with the model, see table 5.5.1. The data were standardized.

The various hyper-parameters used during training are listed in table 5.5.2. The number of parameters are dependent on the network, and does not change when altering hyper-parameters. During training, the full training dataset has to be passed through the network. As the dataset is to large to be passed through as one unit, it is divided into smaller parts called batches. These batches are fed to the network. The batch size is the number of training images in one batch. When all batches have been through the network, more specifically done one forward and one backward pass, we have fulfilled one epoch. The gradient decent method upgrades the weights and biases through backpropagation for every epoch. Therefore, we need multiple epochs to minimize the cost function and optimize the learning.

All network configurations were trained for 200 or 300 epochs and various batch sizes were tested. A smaller batch size requires less memory space and we

quickly found out we were constricted by the GPU memory when training EfficientNet-B4, even with really small batch sizes. For EfficientNet-B3, memory issues arose when using a batch size of 64.

Table 5.5.2: Tested hyper-parameters during training for EfficientNet models. For PyTorch's StepLR learning rate scheduler, the learning rate decays by γ every step size epochs.

Model	#Params. [mill]	Batch size	Epochs	Step size	Learning rate	γ
EfficientNet-B2	7.7	8, 16, 24, 32	200, 300	50, 75	0.1, 0.01	0.1, 0.5
EfficientNet-B3	10.7	8, 16, 24, 32	200, 300	50, 75	0.1, 0.01	0.1, 0.5
EfficientNet-B4[1]	17.6	8, 16	200, 300	50, 75	0.1, 0.01	0.1, 0.5

1: Batch size > 16 gave GPU memory error.
Note: With the Adam optimizer and a learning scheduler, we use an initial learning rate that decreases with a factor g after x (step size) epochs.

5.5.2 Neural network training and validation

To create a model that classifies the data as desired, the model has to be trained. This is a step-wise process that iterates through the data numerous times to optimize the weights and biases in the network. Before training starts, the data is split 80/20 into training and validation sets, and undergoes training and validation transformations as mentioned above. For this thesis, the difference between the two transformation setups is that the validation data are not randomly rotated. Next, the chosen EfficientNet model has to be initialized, the initial learning rate set and the cross-entropy loss function has to be created. Weighing up for imbalanced classes can be done by passing weight parameters to the loss function, or use torch's Weighted Random Sampler (WRS) directly when creating torch data loaders. With WRS, we do not use the weights as four class weights, but an individual weight for each image in the data. The image weight is the is set by the associated label, meaning every image labeled arc, will receive the arc weight.

For each epoch, the network is fed one batch of data at a time. The data gets passed through the network and the loss is calculated based on the differences between the target (ground truth) and the predicted values. This loss, or error, is backpropagated through the network by the Adam optimizer (with learning rate input and default parameters). This is when the weights and biases are updated. The optimizer uses the learning rate to decide how fast the network should learn.For each epoch, we calculate the mean loss by the losses from each batch iteration. The process is repeated until all training data have gone through the network once, and the training for one epoch is completed. Next, validation is performed for the same epoch. This is to check how the model is actually

performing, by seeing if the model is underfitting or overfitting the data. Validation is basically the same process as training. The difference is that the data batches now come from the smaller validation set, and weights and biases are not updated. The model validation accuracy is also calculated. When validation is done, we move on to the next epoch and repeat the training and validation processes. When a model achieved a higher validation accuracy than the current highest validation accuracy, the model was saved.

We used a learning scheduler to implement a learning rate decay during training. The scheduler takes the Adam optimizer and the two hyper-parameters, step size and γ, as input. This learning rate decay helps fine tune the model to get closer to a loss-function minimum and optimize the learning. The main learning rate used when fine tuning models was an initial learning rate $lr = 0.01$, with a decay factor of γ every step size epochs.

To visually determine how the network model was evolving, the mean training loss, mean validation loss and mean validation accuracy for each epoch was calculated and stored. When all epochs were done, we could plot the result to see the loss and accuracy development. Sometimes the validation accuracy were highest after the model started to overfit, thus saving a bad model. If epochs near the ”sweet spot” between underfitting and overfitting had an high accuracy, we tried early stopping to test the model further.

5.5.3 Model testing and evaluation

Models with good potential, e.g. good validation accuracy, were further tested on completely unseen data for the network, the ASI_test set. We use the same data transformation as for the validation data. The model predicts and sets a class label for each image, and we use the result to compare with the ground truth. By this, we can calculate the test accuracy, as well as precision, recall and F1 score. We also create a confusion matrix to see where the model predicts correctly and wrong. Note that the metric scores and CM were also generated when performing validation, and the metrics and CM plot were saved for the model with the highest validation accuracy. This made it possible to compare all metrics between the validation and test result of the model.

6 Results

In this chapter we present the results for the network training, validation and testing, details about the highest performing EfficientNet model (AuroraB3) and statistical prediction results made by AuroraB3 on unseen and unlabeled data.

6.1 EfficientNet

6.1.1 Models

Many models were trained when testing different network parameters. Table 6.1.1 contains some of the models that achieved the best performance results. From the table we can see that the combination of an initial $lr = 0.01$, step size = 75 and $\gamma = 0.1$ gave good results. We can see that for EfficientNet-B3, a batch size of 24 proved to give good results.

Model 2, marked in table 6.1.1, was chosen as the best model to use further for this thesis. The model was weighted for prioritizing aurora features for the three aurora classes, while the no aurora class was kept at its original size. This model, further referred to as *AuroraB3*, achieved the best results by adding the weight parameters to the Cross-Entropy loss function. AuroraB3 was trained to an validation accuracy of 0.90, and achieved an test accuracy of 0.88 on unseen ASI data. Model 3, with mode Bicubic, resulted in overall better training and validation results, but because of slightly more overfitting, the test accuracy became slightly lower than for AuroraB3. We saw a trend where models achieved a very low diffuse test accuracy, and a very high no aurora test accuracy, when using balanced classes. Models with aurora weighted classes achieved a more balanced per-class accuracy.

Table 6.1.1: Metrics and network parameters for four selected EfficientNet models. Precision, recall and F1 scores are for the validation results. Accuracy are for both the validation results and the test results on unseen ASI data. Footnote 1-4, see section 3.4.5.

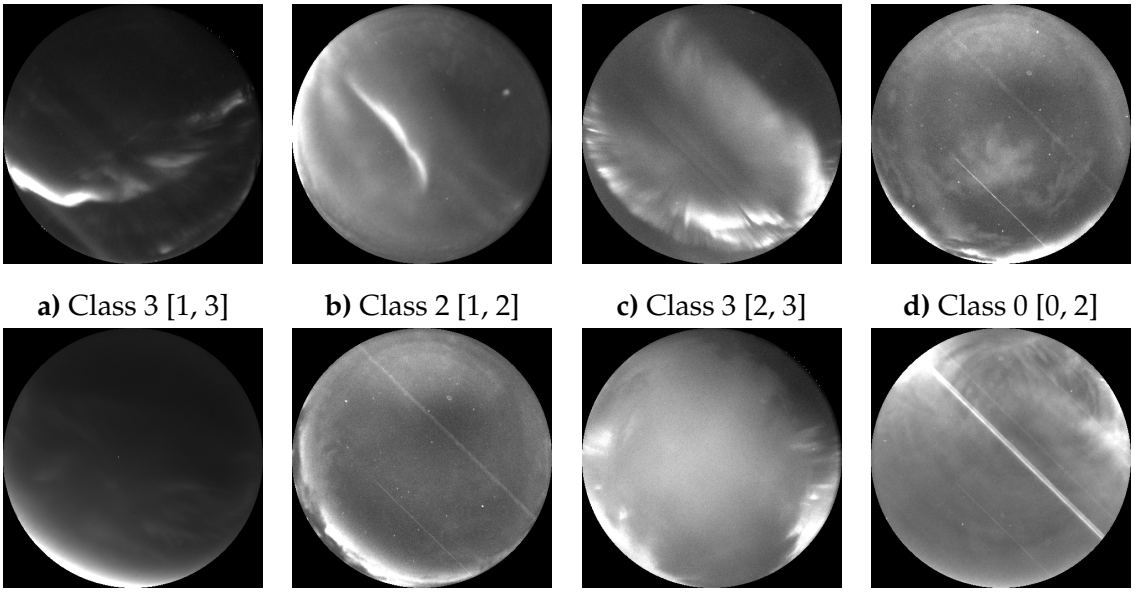
Model	Batch size	Step size	Lr. rate	g	Precision score	Recall score	F1 score	Accuracy Valid.	Accuracy Test
1: B3[1,3,6]	16	75	0.01	0.1	0.90	0.90	0.90	0.90	0.88
2: B3[2,3,5]	24	75	0.01	0.1	0.91	0.90	0.90	**0.90**	**0.88**
3: B3[1,3,6]	24	75	0.01	0.1	0.91	0.91	0.91	0.91	0.87
4: B4[2,4,5]	8	75	0.01	0.1	0.90	0.89	0.90	0.89	-

1: Balanced classes
2: Aurora weighted classes
3: Weights added to Pytorch's Cross-Entropy loss function
4: Pytorch's Weighted Random Sampler
5: Pytorch's interpolation mode: Bilinear
6: Pytorch's interpolation mode: Bicubic

6.1.2 Validation predictions

After training various models, we had a quick look at some of the images that were predicted badly, or wrong, during validation. The panel captions in figure 6.1.1 tells which class the model classified the image as, as well as the classes the model was struggling between.

The model struggled between two classes for panel **a)-d)**, where the scores were divided almost equally, and divided between three classes for **e)-f)**. Panel **f)** in figure 6.1.1 is a good example of an image that is hard to label correctly, as it includes clouds, arc-like light in form of light pollution and a weak veil that can be light fog or diffuse aurora. This would possibly have been easier to detect using real-colored ASI data. Panel **g)** and **h)** are examples of a mismatch between the model prediction and the manual labeling. Panel **g)** shows aurora covered by clouds, where the model predicted diffuse aurora. By the human eye, it looks like discrete aurora is hiding behind the clouds. The image in panel **h)** is predicted as an arc, but its CCD bleeding misinterpreted as an arc.



a) Class 3 [1, 3] **b)** Class 2 [1, 2] **c)** Class 3 [2, 3] **d)** Class 0 [0, 2]

e) Class 2 [0, 1, 2] **f)** Class 0 [0, 1, 2] **g)** Class 2: Aurora covered by clouds **h)** Class 1: Wrong, CCD bleeding

Figure 6.1.1: Images the model found hard to label with high certainty. Panel **a)-d)** all have a probability distribution that are approximately equal between two classes, these are stated inside the brackets of each image. Panel **e)** and **f)** are the same but between three classes. Panel **g)** shows aurora covered by clouds, where the model predicted the image to include diffuse aurora. By human eye, it looks like discrete aurora is hiding behind the clouds. The image in panel **h)** is predicted as including an arc, but its CCD bleeding misinterpreted as an arc. *Reminder: [0: No Aurora, 1: Arc, 2: Diffuse, 3: Discrete].*

6.1.3 AuroraB3

The normalized confusion matrix, figure 6.1.2, for AuroraB3 was made to gain a better understanding of the model performance during the training and validation phase. We observe that the model performs well for the no aurora, arc and discrete classes, where the model correctly predicted the class 93%, 92% and 90% of the times, respectively. The class-accuracy drops to 83% for the diffuse class. We observe that the two most common class-combinations that the model gets confused by are diffuse/discrete (6%+6%) and arc/diffuse (7%+4%). The confusion matrix in figure 6.1.3 shows the model performance on unseen ASI data from data set ASI_test. The highest score for the correctly predicted classes, is the arc class with 93%. model predicts 89% correctly for both the no aurora and the discrete class, but only 84% for the diffuse class. The class-combinations AuroraB3 is confused about, commonly include the diffuse aurora class.

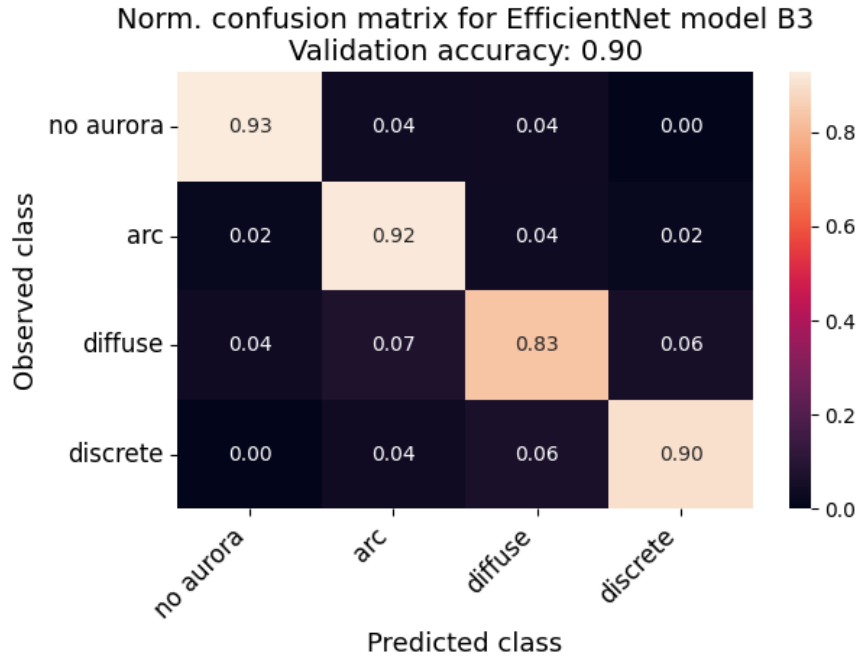


Figure 6.1.2: A normalized confusion matrix for the trained AuroraB3 model on a 20% validation portion of the ASI_train_valid data set. The diagonal is the percentage of correctly predicted labels, the per-class accuracy. The off-diagonal elements are mislabeled.

The graph in figure 6.1.4 shows the validation accuracy, validation loss and training loss as functions of epochs for the network training that resulted in AuroraB3. The save point (epoch 162), marks the epoch where the model with the highest achieved validation accuracy was saved. We observe that the training loss decreases for the first 150 epochs, while the validation loss flattens out with small fluctuations, after the learning rate reduction at epoch 75. After the additional learning rate reduction at epoch 150, both losses are relatively stable around 0.17 for training and 0.27 for validation. The validation accuracy is more or less stable after epoch 75. Up to epoch 75 there are larger fluctuations and an increase in accuracy. The network has some overfitting tendencies after around epoch 150.

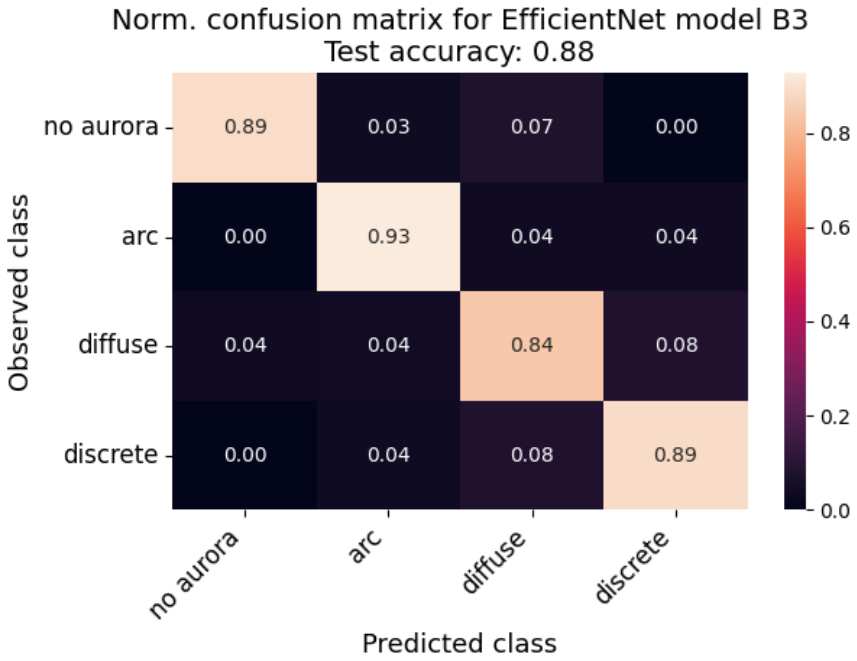


Figure 6.1.3: A normalized confusion matrix for the trained AuroraB3 model on the unseen ASI_test data. The diagonal is the percentage of correctly predicted labels, the per-class accuracy. The off-diagonal elements are mislabeled.

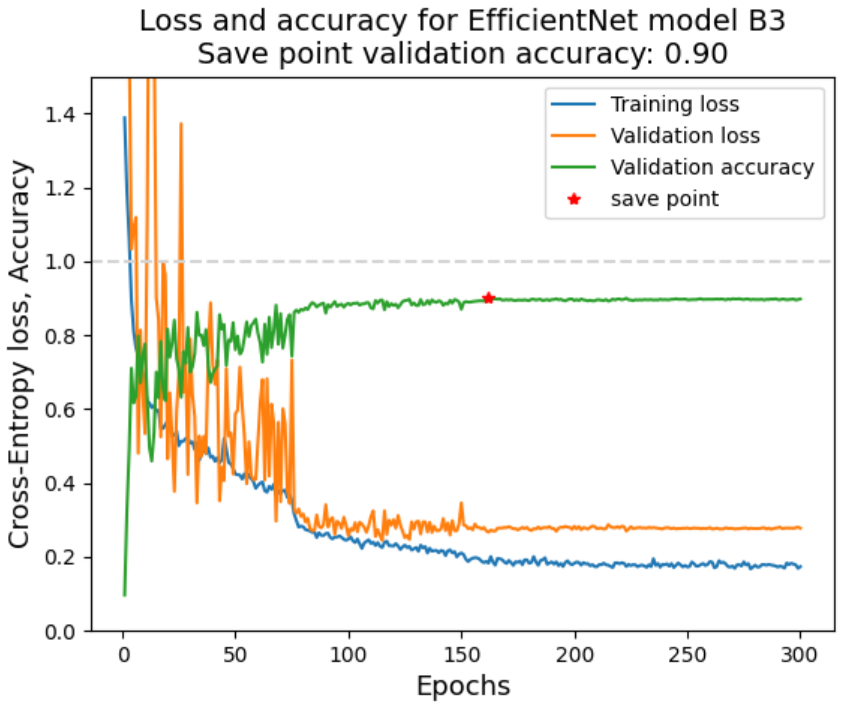


Figure 6.1.4: Validation accuracy, validation and training cross-entropy loss as functions of epochs for the network. The save point marks the epoch (162) where model AuroraB3 was saved. At the save point, the training loss is 0.18 and the validation loss is 0.27.

6.1.4 AuroraB3, 2 classes

The same parameters as for AuroraB3 in the previous chapter were also used to train a model for two classes. The confusion matrix in figure 6.1.5 shows that the model predicted both classes very well, with identical true positives as the test accuracy of 96%. The train/validation loss graph can be seen in figure 6.1.6. The graph shows that the model did not overfit the data significantly, and that the training and validation loss are relatively small. We can also see that the validation accuracy is stable around 0.96 for the last 100 epochs. The weighted precision, recall and f1 score are all 0.96.

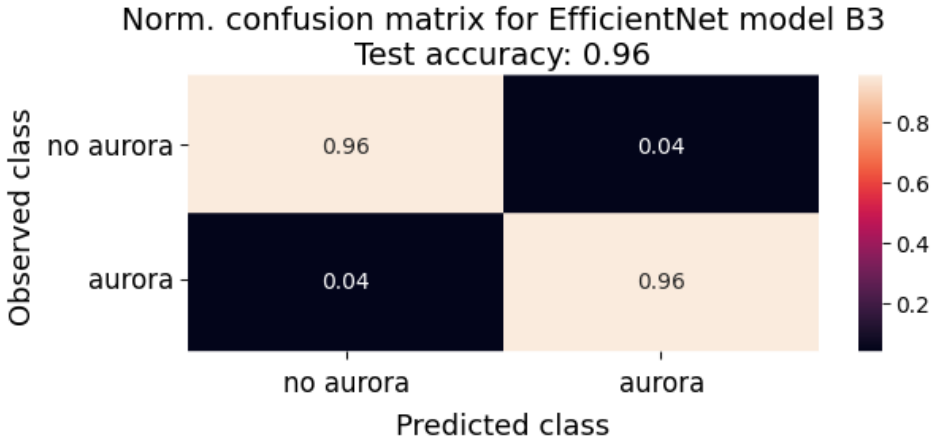


Figure 6.1.5: A normalized confusion matrix for the two class B3 model with identical training parameters as AuroraB3. The CM were made on the unseen data ASI_test.

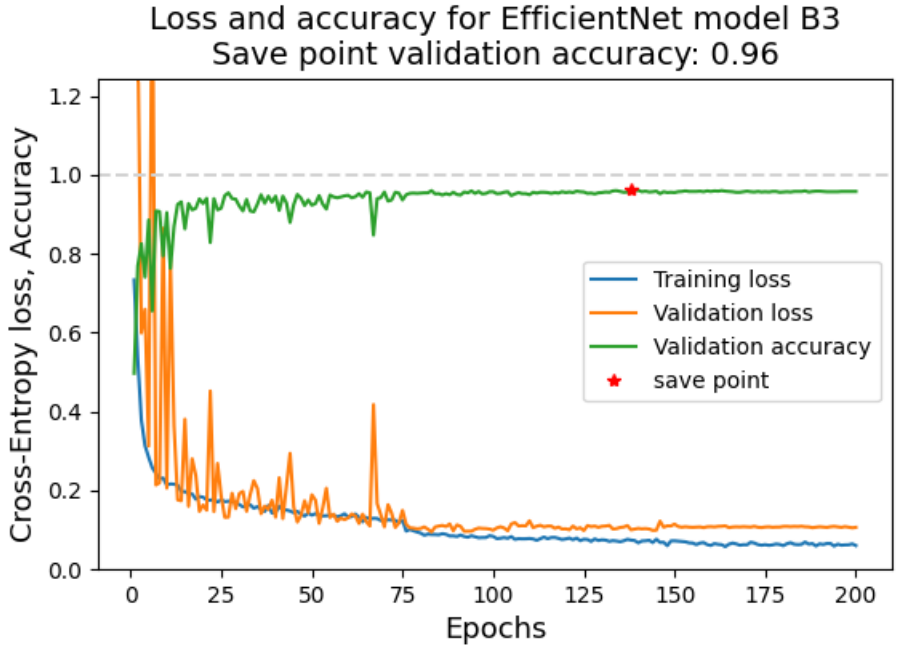


Figure 6.1.6: Validation accuracy and validation and training cross-entropy loss as functions of epochs. The save point marks the epoch where the model was saved.

6.2 Prediction results on unlabeled ASI data

When AuroraB3 was decided to use for the rest of the thesis, we used the model on the ASI_G.json and ASI_R.json datasets (with OMNI data). These datasets do not have pre-labeled images (ground truth), so when making predictions, both a class label and class prediction scores were saved to a new dataset, as mentioned in chapter 5.3.2.

The process of making statistical results started with downloading the green aurora ASI data for years 2014 and 2020. The images were downloaded and pre-processed, as described in chapter 5.1.1. The image information was saved to the ASI_G.json file, before OMNI data were matched to the dates and updated in the file. When the datasets were made, we used AuroraB3 to predict the class of each image. Later, data for 2016 and 2018 were downloaded. This was done to easier observe statistical differences between years closer to a solar minimum or maximum. These results were merged with the 2014 and 2016 results. Lastly, the same process was made for the total amount of red aurora images. The reason for making predictions on only parts of the desired data at a time, was the amount of data. The total of 665,865 images needed more storage space than we had available. Therefore, we downloaded, predicted and deleted the data in three turns. We could now make statistical results of the predicted data.

6.2.1 Distribution of classified ASI data

In terms of programming, when finding a general distribution, we simply count the number of entries/images that were classified to a specific class when we iterated through the a dataset. By making an "if statement", the distribution could be made for separate years or months by matching the wanted time period to the entry timestamp. We also made error bars that indicate the prediction accuracy. This is an artificial error, that only show the error compared to the other entries. This was done by adding an error score based on the prediction score. If the highest prediction score was greater or equal to 0.9, a low error of $\varepsilon = 0.1$ was added. If score$\in [0.5, 0.9)$, we used $\varepsilon = 0.4$, and scores below 0.5 got $\varepsilon = 0.8$.

The total number of images from the ASI_green and ASI_red data, that have been predicted and classified by AuroraB3, are listed in table 6.2.1. The table also list the number of images that have been classified with one of the aurora labels.

The distribution of classified images is shown in figure 6.2.1 and 6.2.2, for green and red aurora, respectively. For green aurora, the bar chart shows that the no aurora class generally have the highest count. If we exclude the no aurora class, we see that the discrete class is the most dominant for all years, with a peak with 39.5% for 2016. This makes up 55.3% of the predicted images if we only consider the aurora classes. The arc class is the least dominant class for all years except 2016, where the predicted data contains 19% arcs, compared to 12-15% for the other years. Diffuse aurora increases the two years before and during solar minimum, with 21-22%, compared to 13-16% for the two years during and after

Table 6.2.1: Total number of images for each year, with the Aurora column that shows the number of images that were predicted to one of the three aurora classes.

Year	Data	Total	Aurora
2014	ASI_green	112 000	61 000
	ASI_red	58 000	35 000
2016	ASI_green	124 000	89 000
	ASI_red	62 000	46 000
2018	ASI_green	83 000	55 000
	ASI_red	41 000	27 000
2020	ASI_green	119 000	81 000
	ASI_red	59 000	43 000

solar maximum.

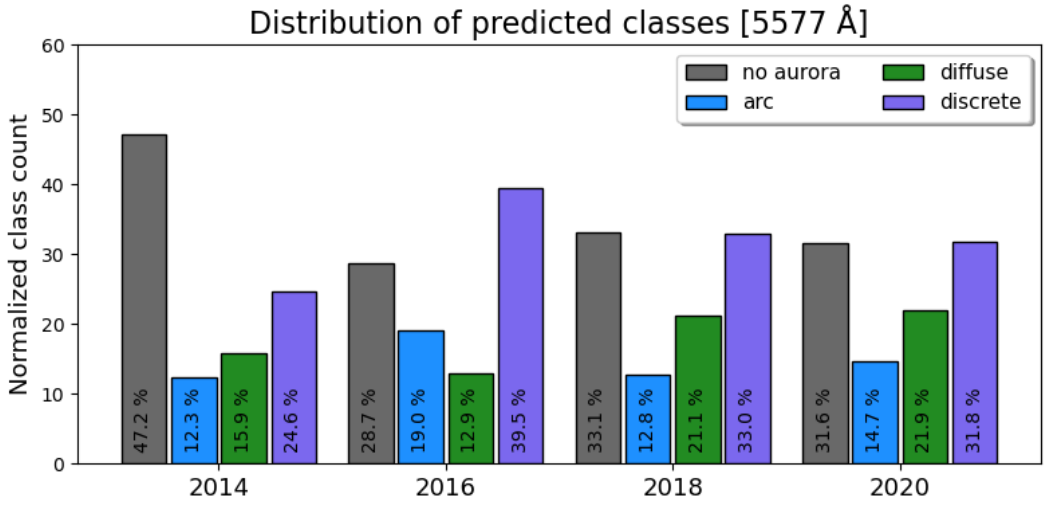


Figure 6.2.1: Distribution of classified 5577 Å ASI data for years 2014, 2016, 2018 and 2020. Each year only include data for January, November and December.

For the classification distribution of the red aurora images, we see one clear difference from the green aurora. That is the roughly 10% increased of diffuse aurora for 2014, 2016 and 2018. For 2014 there is a relative large decrease for the no aurora class, while it stays more stable for 2016 and 2018. The predicted arc events are the least dominant for all years. For year 2020, the occurrence of discrete aurora have increased by approximately 6%, while the discrete class have decreased for all the other years by 1-5%.

Monthly distribution plots can be found on GitHub, under e.g. stats/Red/b3/Pie.

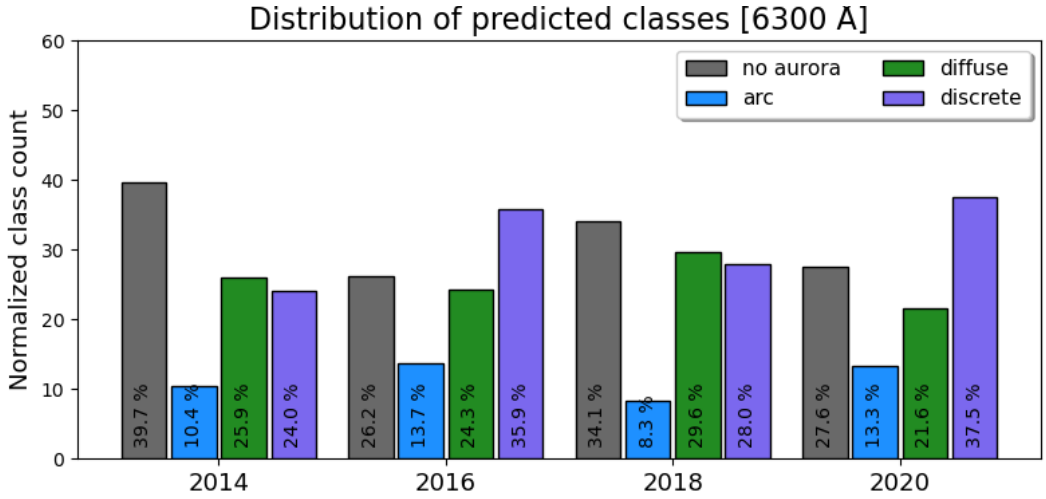


Figure 6.2.2: Distribution of classified 6300 Å ASI data for years 2014, 2016, 2018 and 2020. Each year only include data for January, November and December.

6.2.2 Predictions with equal class probabilities

When the model predicts the class of an image, it gives a prediction probability score to each possible class. The class with the highest score is the Top-1 class and the class with the second highest score is the Top-2 class, and so on. Sometimes, the model struggles to predict an image with a clear result, meaning a Top-1 class with a single high score. When this happens, we thought it would be interesting to research which class combinations the model struggled the most with. By iterating through the dataset, we added entries with the Top-1, Top-2 and Top-3 class label and scores, to a dictionary if the two or three highest scores were equal, with a fairly large 10% error margin. This process was done separately for two or three equal scores. Below are some dictionary examples of entries we were looking for

- {Arc: 0.46, Discrete: 0.45}
- {No aurora: 0.51, Diffuse: 0.47}
- {No aurora: 0.34, Diffuse: 0.32, Arc: 0.31}

From the two dictionaries, we made tables for all class combinations, and counted the number of occurrences for each combination. This was done separately for red and green aurora. Table 6.2.2 was made to highlight which class-combinations the model struggles with between two equal scores. For green aurora, the two classes diffuse and discrete, is the class-combination which the model struggles the most. This combination makes up for 32.3% of the cases. Second, the no aurora and diffuse class combination makes up for 26.2%.

The same statistical results are found for red aurora, where no-aurora and diffuse makes up 25.8% of the data set, while diffuse and discrete makes up 38.0%. The combination no aurora and discrete has the lowest account for both green and red aurora, with only 0.8% and 0.4%, respectively.

Table 6.2.2: Class predictions on ASI_green and ASI_red data with two equal class outputs. The distribution does not take into account whether the prediction is correct or not. The total percentage of ASI_green data was 2.3%, and 2.5% for the total ASI_red data.

Class a	**Class b**	**Equal prob.** [5577 Å]	**Equal prob.** [6300 Å]
no aurora	arc	11.0 %	10.8 %
no aurora	diffuse	**26.2** %	**25.8** %
no aurora	discrete	0.8 %	0.4 %
arc	diffuse	15.1 %	16.6 %
arc	discrete	14.5 %	8.3 %
diffuse	discrete	**32.3** %	**38.0** %

The table does not take into account the order of the Top-1 class and the Top-2 class, just that the probabilities are similar. Even with a fairly generous limit, only approximately 2.5% of the total data falls under the category with two equal probability scores. Most of the data the model predicts a class for, have a high Top-1 score. The percentage of images that have a Top-1 class with a prediction score higher than 85%, makes up $\sim$ 81% for ASI_green and $\sim$ 79% of ASI_red.

For three equal probabilities, two class-combinations stands out. The combination of no-aurora/arc/diffuse and arc/diffuse/discrete. For ASI_green, the two combinations account for $\sim$ 33% and $\sim$ 52%, respectively. The images that have this kind of prediction scores only makes up 0.07% of the total data. For ASI_red, the two combinations account for $\sim$ 47% and $\sim$ 49%, respectively. This make up 0.06% of the data set. Text files with results can be found on GitHub under stats.

6.2.3 Hourly distribution of classified ASI data

In terms of programming, the hourly class distribution was made similar as the process in chapter 6.2.1. The hourly distribution was made by extracting the hour from the entry timestamp and adding an error as described in chapter 6.2.1. The error was now added to a dictionary that holds 24 (hour) keys. The error value and an error count for that hour were added as a value list. The hourly distribution was extracted class for class, year for year, for both red and green aurora. This was done so that we could make subplots to see how the various aurora classes behaved on average for all four years. The normalized plots were made by counting the entries within an hour, divided by all entries. This means the sum of all four classes, each year, sums up to 1. Plots were also made for each separate month, available on GitHub.

In figure 6.2.3 we have an hourly overview of the class distribution for green aurora. The y-axis shows the class occurrence for each hour (local time), normalized on the total amount of data for each year. In the upper panel, we can see that arcs appeared mostly during nighttime, apart from year 2016. 2016 have a peak around noon, as well as high nighttime occurrence. In the second and third panel, we observe that diffuse and discrete aurora were common from early morning to afternoon. Both classes have little activity between $\sim$17/18-24 local time. This time period is part of the defined nighttime, and we also observe from the fourth and fifth panel that almost no aurora is imaged during this time. The last panel is a combination of the three aurora classes. The panel indicates that for 2014, most aurora was observed from hours 04-12. The two later years, had a more even distribution of aurora occurrences during daytime. The data for 2020 shows a very clear two-peak distribution, with a occurrence dip around noon. The error bars are based on the prediction probability scores, where the higher top-1 class score gives a lower error. So, the error gives an indication if the (average) prediction for a specific hour is correct. We can see from figure 6.2.3 and 6.2.4 that the prediction uncertainties are clearly larger for the diffuse class. We can also observe that the error bars are slightly larger for the arc class for red aurora compared to green aurora.

The statistical results for the red aurora classification in figure 6.2.4 share many of the same tendencies as for the green aurora. As expected from earlier distribution presentations, we see an increase in the occurrence of diffuse aurora. This is the case for all four years, with a small peak around hours 08-10 for 2018 and 2020. For the discrete aurora, we see an even clearer two-peak distribution for dayside aurora, compared to the green aurora, for 2018 and 2020. The main peak occur around 06-08 local time, while the second peak occur around 15 local time.

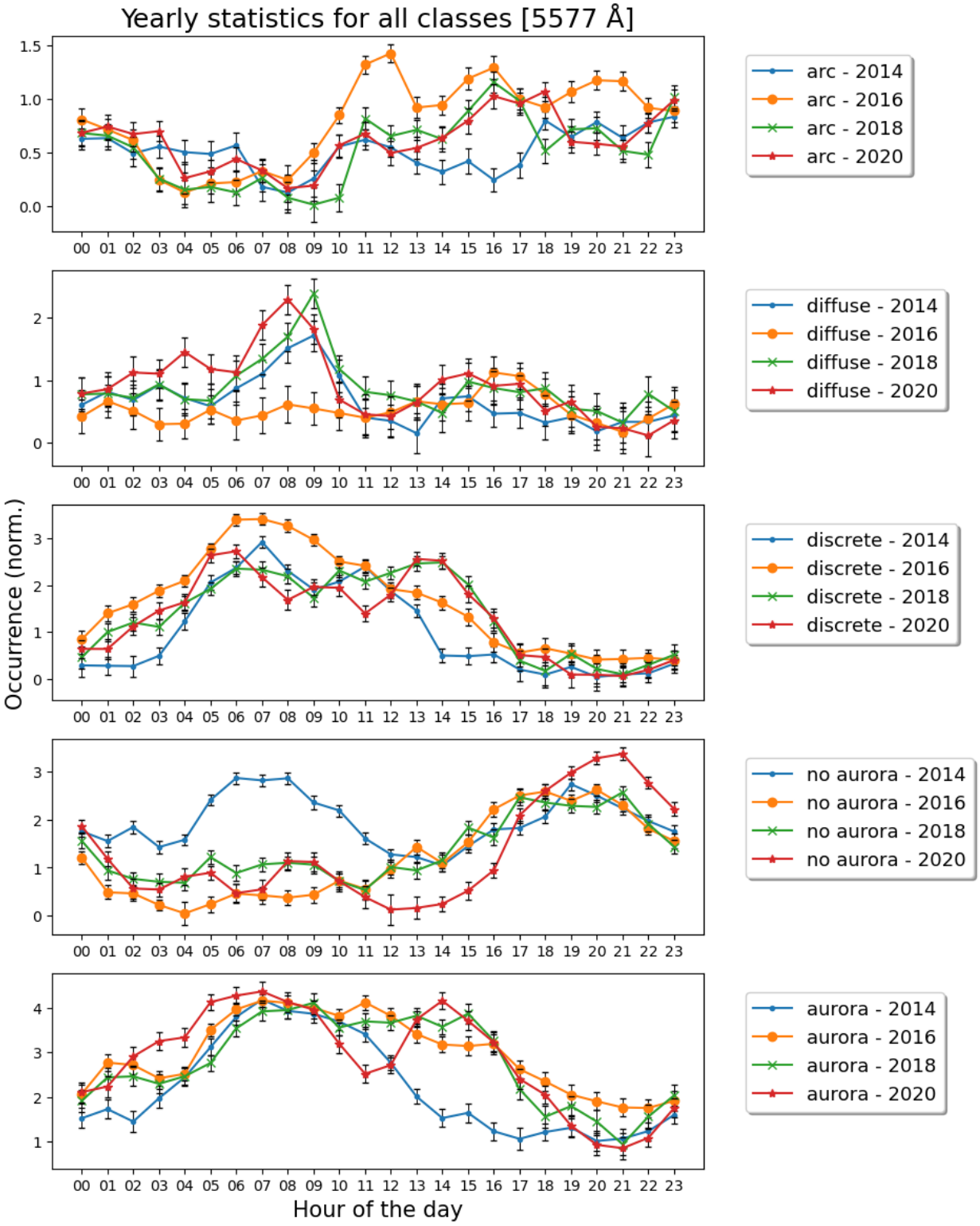


Figure 6.2.3: Hourly distribution from the ASI_green data. The distributions show the normalized aurora count for each (local) hour of the day for each year of data.

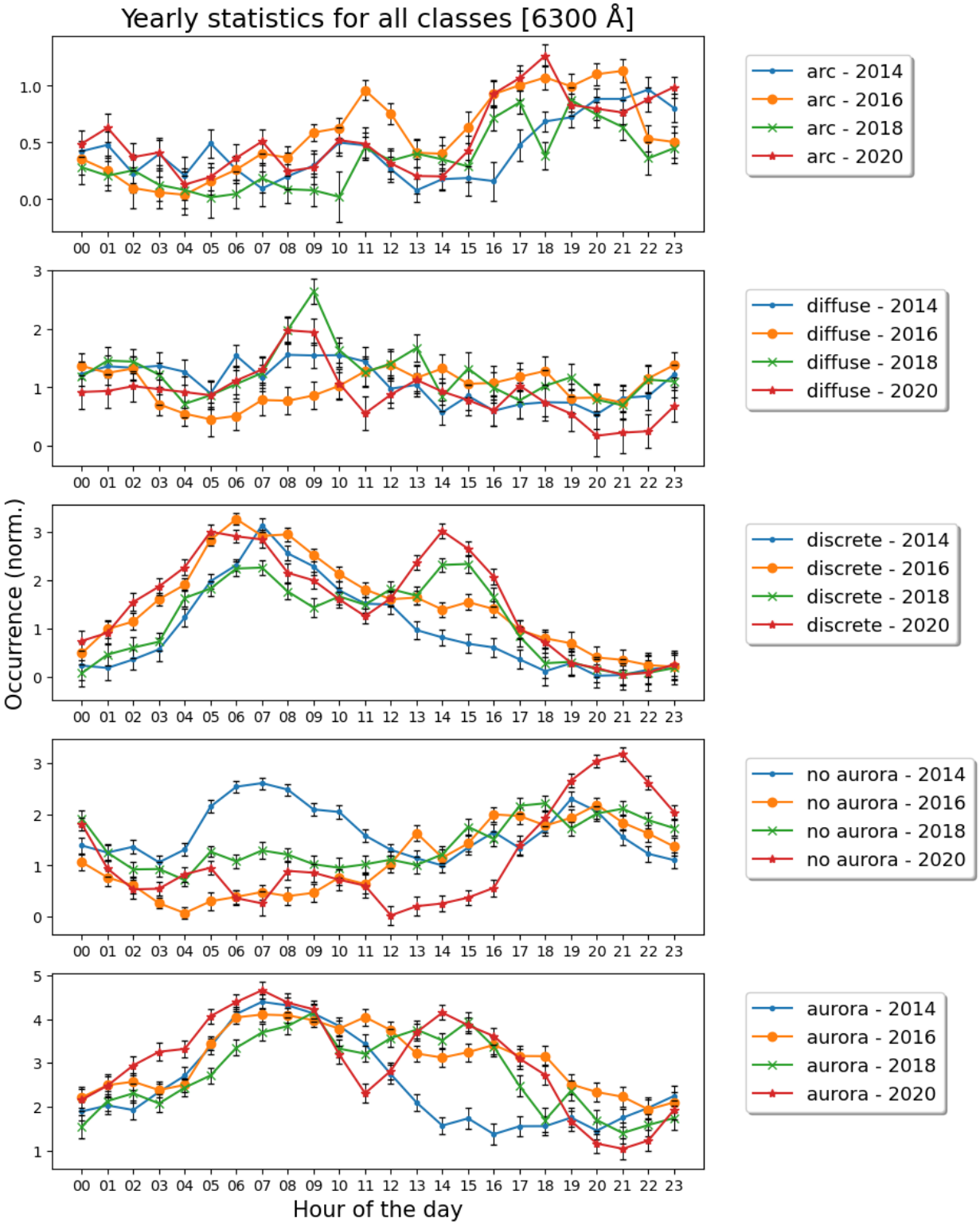


Figure 6.2.4: Hourly distribution from the ASI_red data. The distributions show the normalized aurora count for each (local) hour of the day for each year of data.

6.2.4 Yearly solar wind speed distribution

The programming setup of creating and visualizing the distribution for the solar wind B_z component (chapter 6.2.5) and the speed were made almost identical. Which solar wind parameter the distribution was made for, could easily be changed by editing a function input between B_z and speed. The two parameters also used different bin sizes, as the speed parameter values are larger than the B_z values. A difference is that the B_z distribution is visualized separately for daytime (06-17 local time) and nighttime (18-05 local time). As mentioned in chapter 5.3.2, with daytime aurora and no Dungey cycle, the average B_z value should be more accurate and have a smaller standard deviation.

The function we made goes through the dataset entry by entry. If a year parameter was activated, meaning we want the distribution for a specific year, a simple if statement was used to filter out the desired images by the timestamp. The next if statement matches the image label with the desired class. If they match, the OMNI parameter and standard deviation parameter were added to a dictionary. When the function finished iterating through the entire dataset, dictionaries for classes no aurora, arc, diffuse, discrete and the additional classes aurora (arc, diffuse and discrete combined) and All (all four classes combined) were made. If an OMNI parameter were a *no data*-value, the entry was dropped.

The next step was to visualize the distribution. This was done by creating bins and using Numpy's histogram function. The function outputs the histogram values and bin edges. For B_z-distribution, we used bins $B_z \in [-20, 20, 41]$nT, making a bin width of 1 nT. For speed-distribution, we used bins $Speed \in [200, 800, 31]$ km/s, making a bin width of 20 km/s. With a bin width $\neq 1$, we had to normalize the histogram output values even with "Density=True" in np.histogram(). The last step was to plot the histogram values as a function of the bins. Based on the dictionaries used, we could plot the distributions for separate classes or the entire dataset. We also added error bars made out of the associated standard deviations for each bin. The error bars are horizontal, along the B_z- or speed values/bins. An error bar wider than the bin width means the mean value could be so off that the actual value could lie in one of the neighbouring bins.

Figure 6.2.5 shows the solar wind speed distribution for all four years, independent of the emission line. For 2014, the measured solar wind speeds are between 300-600 km/s (slow wind), with an average speed around 400 km/s. About 30% of the data belong to the highest range with speeds between 340-380 km/s. The distribution for 2018 and 2014 are very similar, but 2018 have some occurrences of fast solar winds (600-800 km/s). 2018 have the highest peak of speeds between 380-400 km/s. 2016 have measured solar wind speeds between 240-720 km/s. The range is more evenly distributed, with only a 10% occurrence for a speed around 550 km/s as the highest peak. 2016 is the year with the highest occurrence of fast solar winds. The year of the solar minimum is 2020, and we observe an average solar wind speed around 320 km/s. Speeds between 300-360 km/s belong to approximately 70% of the data.

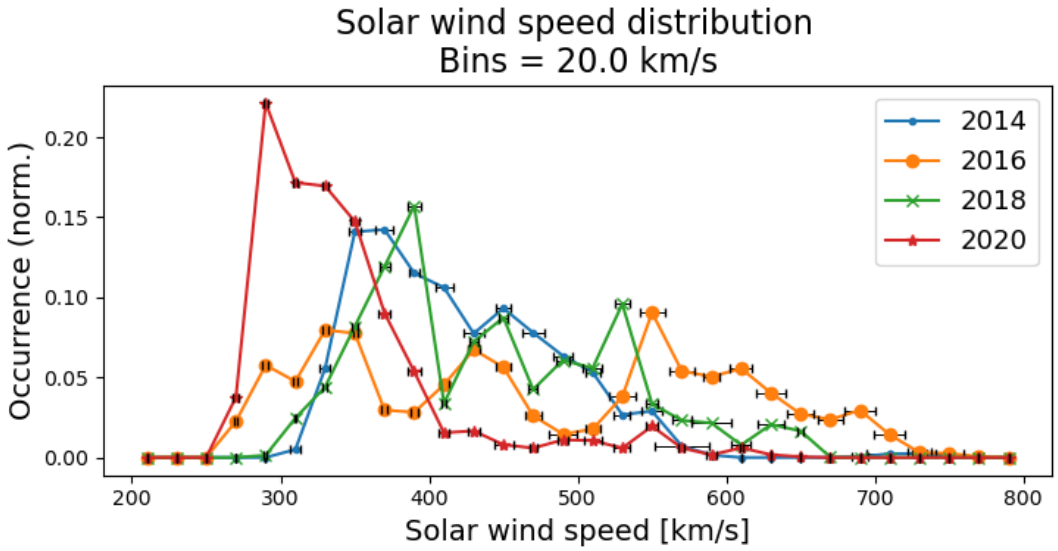


Figure 6.2.5: Solar wind speed distribution for the time points for all four years in our data set. The distribution for all classes under one, are identical for both emission lines. Each point is a histogram bin that spans for 20 km/s.

The error bars are fairly small, indicating that the mean speed of the images belonging to a bin width are quite accurate. Overall, we see a slightly larger error for 2014 and 2016. We also observe a larger error when the speed is higher.

Figure 6.2.6 and 6.2.7 are the solar wind distribution graphs for emission lines 5577 Å and 6300 Å, respectively. The graphs shows the speed distribution for each individual class. The main thing we observe, is that the general distribution shape of the two plots follows the all-over distribution shape from figure 6.2.5.

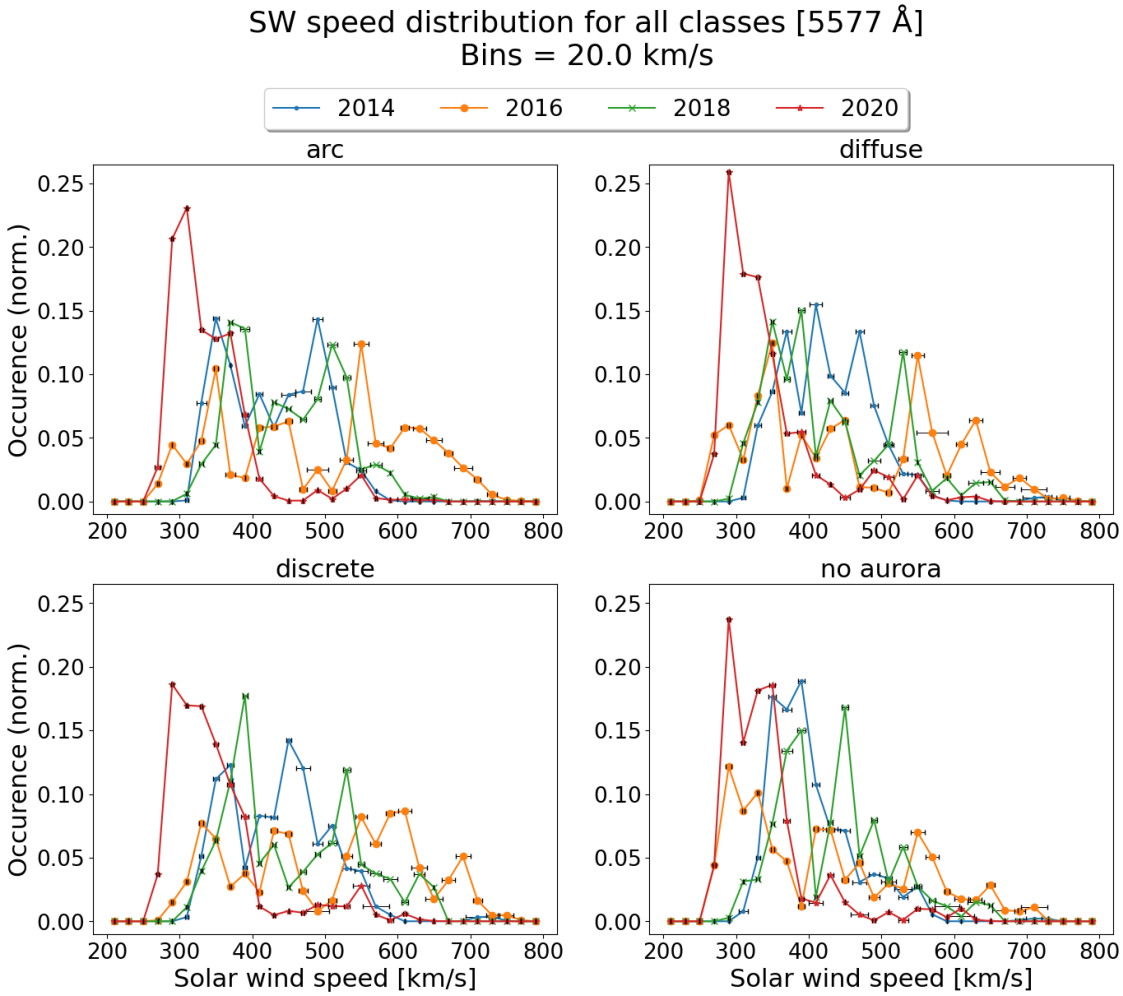


Figure 6.2.6: Solar wind class distribution for ASI_green data. Each panel show the yearly distribution for each year, and each point is a histogram bin that spans 20 km/s.

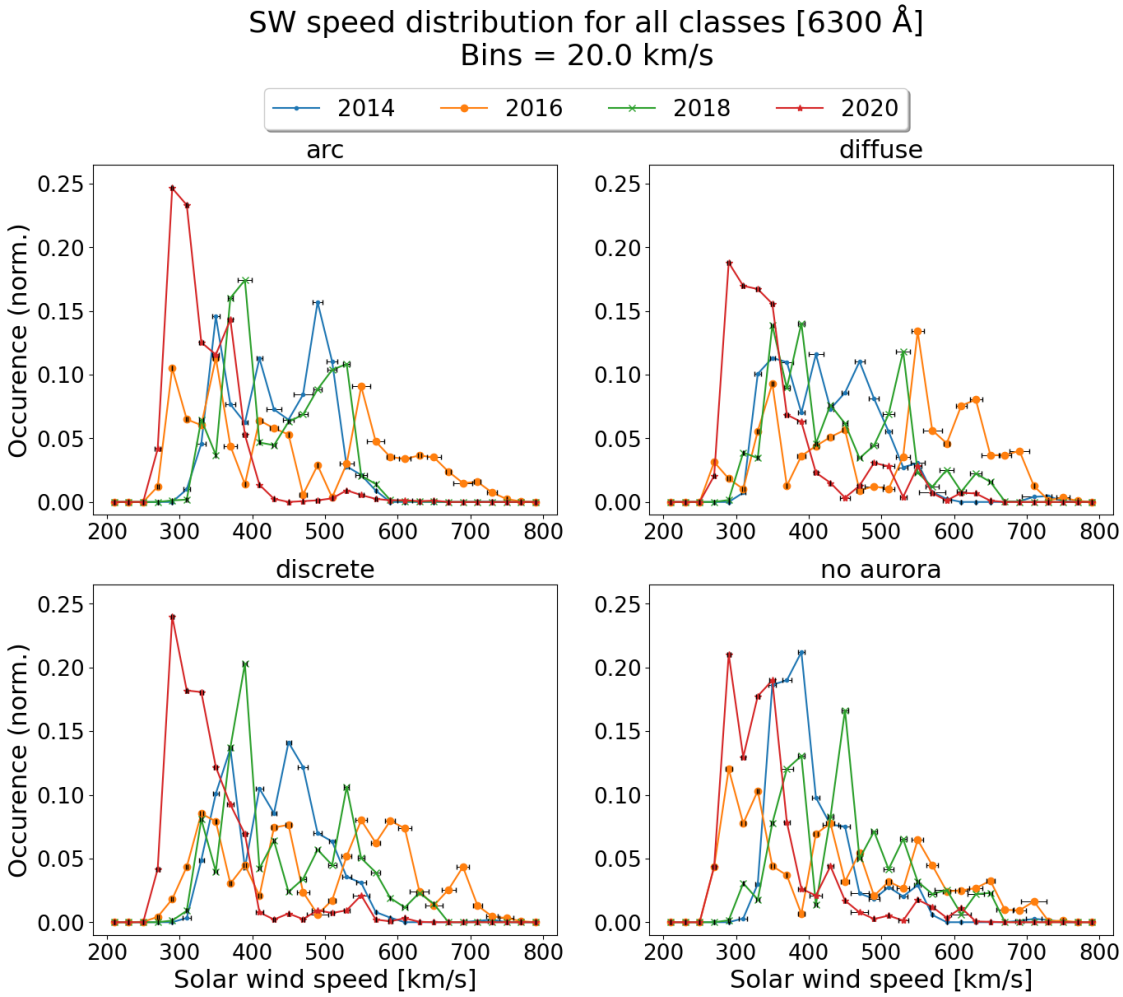


Figure 6.2.7: Solar wind class distribution for ASI_red data. Each panel show the yearly distribution for each year, and each point is a histogram bin that spans 20 km/s.

6.2.5 Yearly IMF B_z distribution

The programming basics for the following results are explained in chapter 6.2.4. In figure 6.2.8-6.2.10 we have the separate distributions of the southward ($B_z < 0$) and westward ($B_z > 0$) B_z (GSM) solar wind component for daytime and nighttime aurora. The text information printed in the plot are the percentage of data that have a positive or negative associated B_z value. This is also divided for daytime and nighttime aurora.

We start by looking at the graphs for 2014 and 2020 separately. We excluded 2016 and 2018 as they are very similar as the distribution in figure 6.2.10. These, and the plots for red aurora can be found on GitHub. The plots for red aurora was excluded from the report as they are almost identical to the distribution for the green aurora. We do not pay a lot of attention to the no aurora class, as the B_z values can lead to misleading interpretations. In hindsight, the no aurora class should have been divided into clear and cloudy. This is because we wont know if there was aurora activity behind the cloudy weather, and can't draw any conclusions about the relationship between the B_z value and no aurora occurrence.

2014

- **arc** | We observe that the distribution have more negative than positive B_z values, where roughly $B_z \in [-8, 5]$ nT for both daytime and nighttime. The peak for daytime aurora lies evenly around the center, while the center is slightly shifted towards the negative B_z range for nighttime aurora.
- **diffuse** | We observe more positive B_z values than for arc, where we have a number of B_z values with $\sim$ 15% occurrence. The main peaks of the distribution lies on the positive side of the B_z range.
- **discrete** | The distribution have evenly distributed occurrences for positive and negative B_z for nighttime aurora, while its slightly shifted towards the positive side for daytime aurora. The range of the main distribution lies within $B_z \in [-6, 6]$ nT.

2020

- **arc** | We observe that the distribution have more negative than positive B_z values, where roughly $B_z \in [-6, 5]$ nT for both daytime and nighttime aurora. The main occurrence lies around $B_z \in [-3, 3]$ nT. Approximately 60% of the entries have $B_z < 0$.
- **diffuse** | The graph shows quite evenly distributed occurrences around $B_z \in [-5, 6]$ nT. The range for the highest occurrences are a little narrower. The highest occurrence, of approximately 25%, for a single B_z bin is for B_z =1-2 nT.
- **discrete** | Approximately 60% of the entries have $B_z < 0$, and range $B_z \in [-5, 6]$ nT. The distribution is very similar to the one for arc.

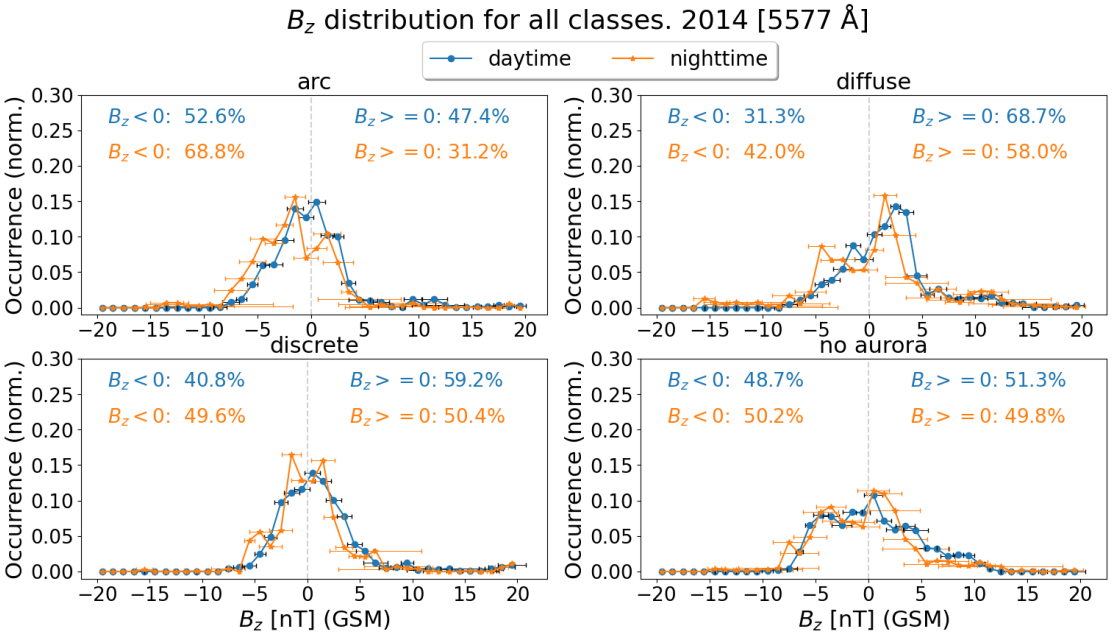


Figure 6.2.8: Distribution of positive and negative B_z values for all classes in 2014. Bin width of 1nT. The blue, circular-pointed line shows the distribution of daytime aurora, while the orange, star-pointed line shows the distribution of nighttime aurora.

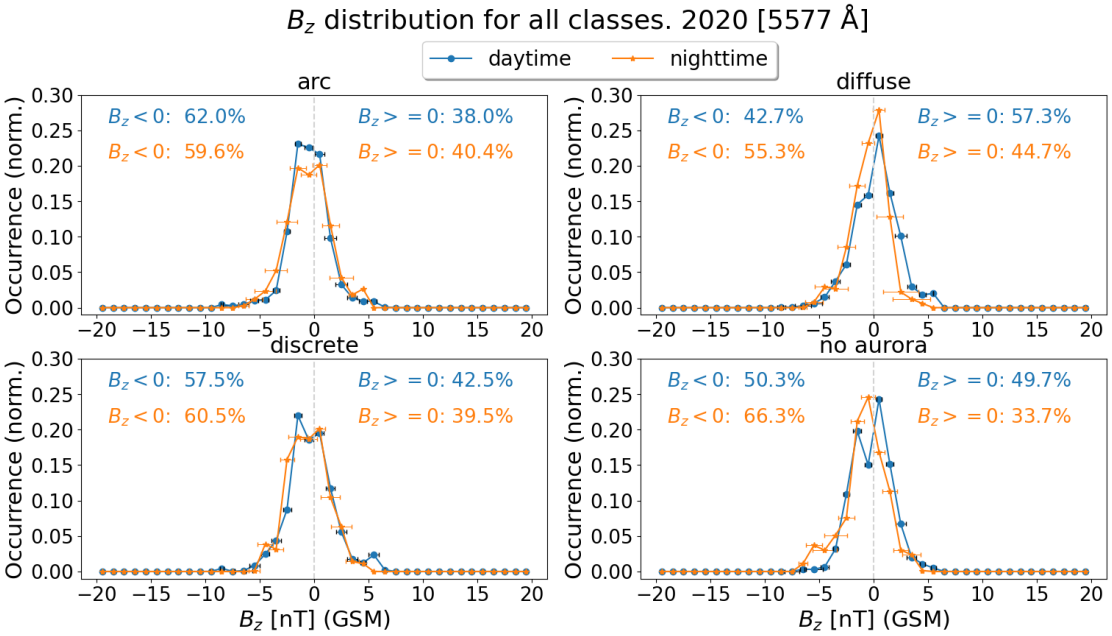


Figure 6.2.9: Distribution of positive and negative B_z values for all classes in 2020. Bin width of 1nT. The blue, circular-pointed line shows the distribution of daytime aurora, while the orange, star-pointed line shows the distribution of nighttime aurora.

Figure 6.2.10 shows the average distribution for all years combined. All classes have an approximate normal distribution around $B_z = 0 \pm 2$ nT. Arc (and no aurora) have more occurrences of a negative B_z value than a positive, while it is the other way around for diffuse and discrete. We also observe that the range for the three aurora classes are $B_z \in [-6, 6]$ nT.

2014 stands out in form of having a broader distribution of data as a function of B_z values, compared to the other years. For 2014, the occurrence for a B_z value lies at maximum 16%, while 2020 have peaks where a B_z value represents 20-28% of the data. An observation is that the closer we are to solar minimum, the narrower the distribution gets.

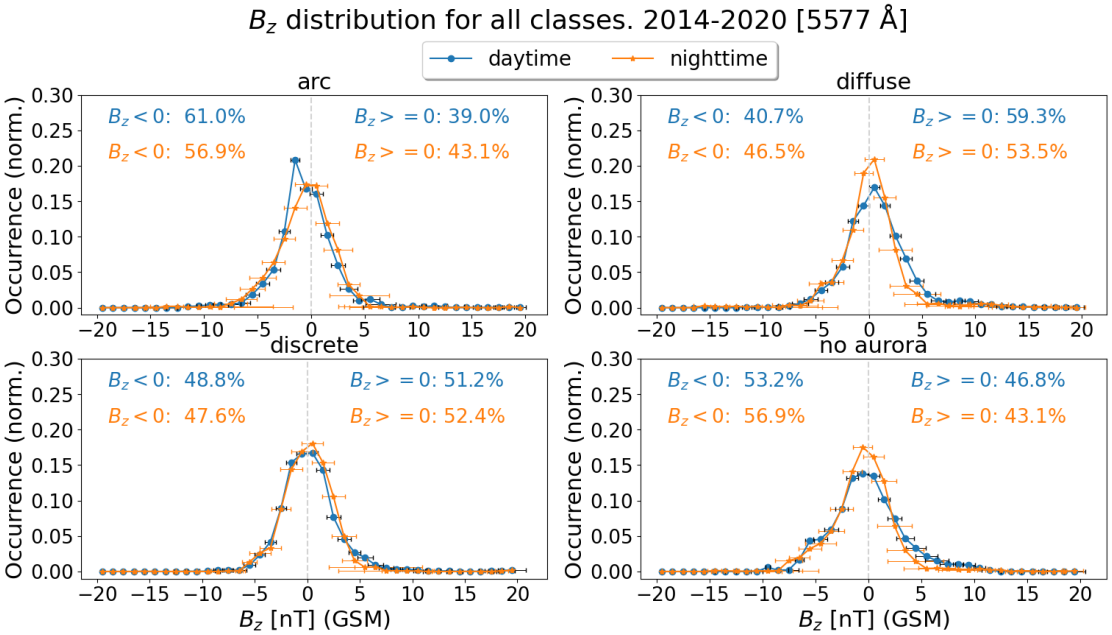


Figure 6.2.10: Distribution of positive and negative B_z values for all classes. Bin width of 1nT. The blue, circular-pointed line shows the distribution of daytime aurora, while the orange, star-pointed line shows the distribution of nighttime aurora.

The error bars have something in common for all years, they are much wider for nighttime aurora, indicating entries may not be placed in the correct bin. The larger uncertainty comes from the calculating the average B_z value over a larger time interval during dayside reconnection.

6.2.6 Hourly B_z-dependent distribution of classified ASI data

The statistical results for the hourly B_z distribution were made similarly to the hourly distribution in chapter 6.2.3. The result is basically the same, but the graphs shows separate results for southward ($B_z < 0$) and northward ($B_z > 0$) IMF. The method however works the same, but the extraction of the entry hour timestamp was saved to two individual lists for a negative or positive B_z value. The two lists were plotted as a function of hours.

Figure 6.2.11-6.2.14 shows the hourly distribution of classified ASI data based on the associated B_z value for each entry. The figures are for every two years from 2014 to 2020, for green aurora. Each graph shows the normalized class occurrence, per hour, for a year of data.Plots for red aurora can be found on GitHub under stats/Red/b3. These were excluded from the report as they did not indicate large variations from the green aurora distribution.

For 2014, figure 6.2.11, we see that the aurora takes the form of arcs throughout the day, but they are mostly appearing as nighttime aurora. Discrete and diffuse aurora are more common during the day, where we have a peak around 08-09 local time for diffuse aurora. A smaller peak appears around 15 local time for $B_z > 0$. For discrete aurora, there are similar amount of events for both positive and negative B_z, mainly between 05-13. Again, we wont focus to much on the no aurora class, but we observe an even distribution of daytime and nighttime aurora for $B_z > 0$. For $B_z < 0$, we observe two peaks around the boundary for daytime and nighttime aurora.

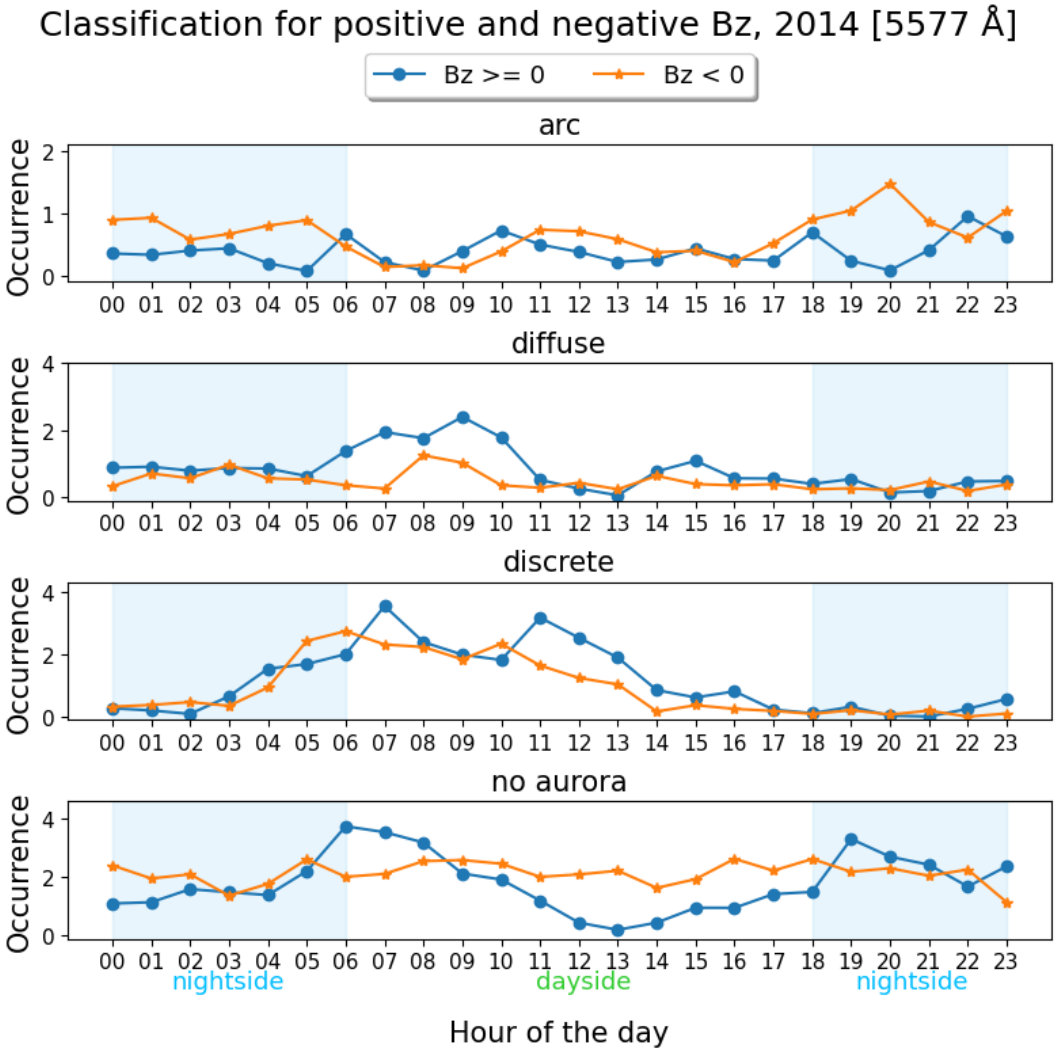


Figure 6.2.11: Positive and negative B_z classification for all classes, for January, November and December 2014. The y-axis show the normalized occurrence.

For 2016, we observe that the diffuse class have very even, low count, through the day. We only see one peak around 16-18 local time. Discrete aurora dominates from early hours to afternoon, while arcs are more common from afternoon to early hours. This distribution is more or less equal for red aurora, with the exception of an all over higher count of diffuse aurora. This happens to also be the case for 2014 and 2018. For 2020, the occurrence tendencies are similar, but with an all over higher count of discrete aurora, and lower for diffuse.

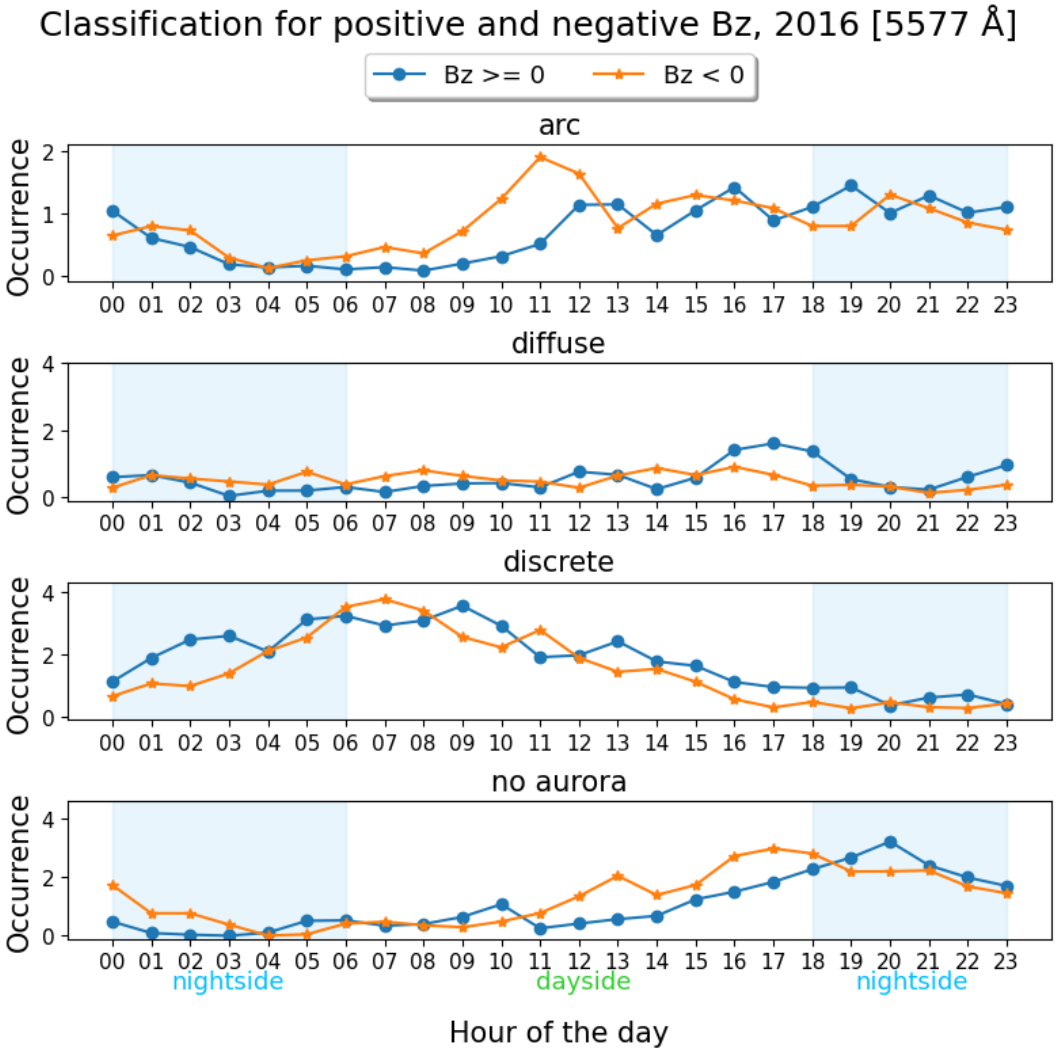


Figure 6.2.12: Positive and negative B_z classification for all classes, for January, November and December 2016. The y-axis show the normalized occurrence.

For 2018, we see arc with high occurrence except for the time between 03-10 local time. There is an higher peak for southward B_z around 11-12 local time. Diffuse again have a peak around 08-10 local time, for both southward and northward B_z. For northward, we also see an increase around 15-18. Discrete have a peak around 06-08 for southward B_z and a peak around 12-14 for northward B_z. Again, we observe that images are classified as no aurora from afternoon to midnight.

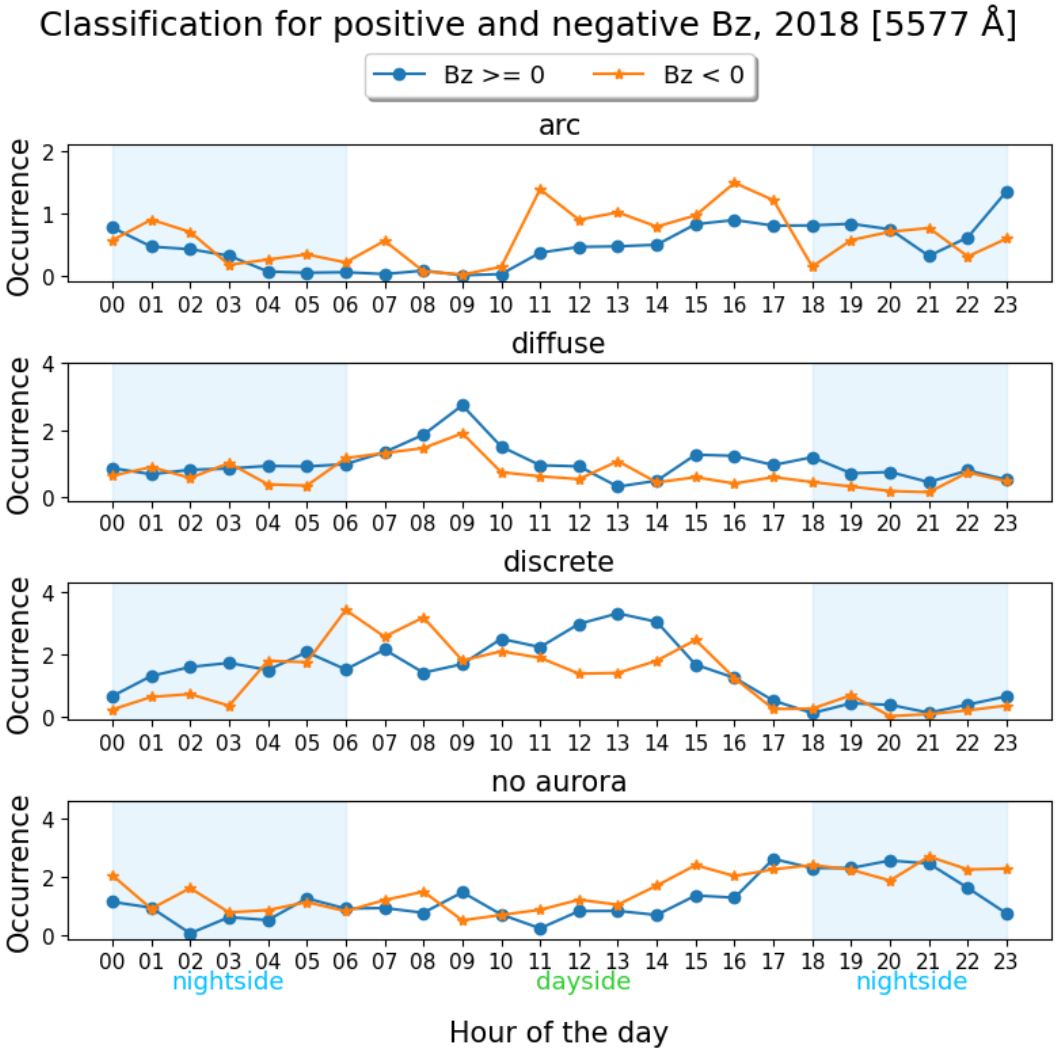


Figure 6.2.13: Positive and negative B_z classification for all classes, for January, November and December 2018. The y-axis show the normalized occurrence.

For 2020, just like the previous years, we have close to none discrete aurora from afternoon (∼17-18) to midnight, while we in this period have aurora arcs as the dominant aurora events. The diffuse aurora have a significant peak for $B_z > 0$ around 07-09 local time. We also observe an peak around 15-17. Discrete continues with a two-peak distribution, with the main peak at 05 and the secondary peak at 13-14. These peaks are one hour later for $B_z > 0$.

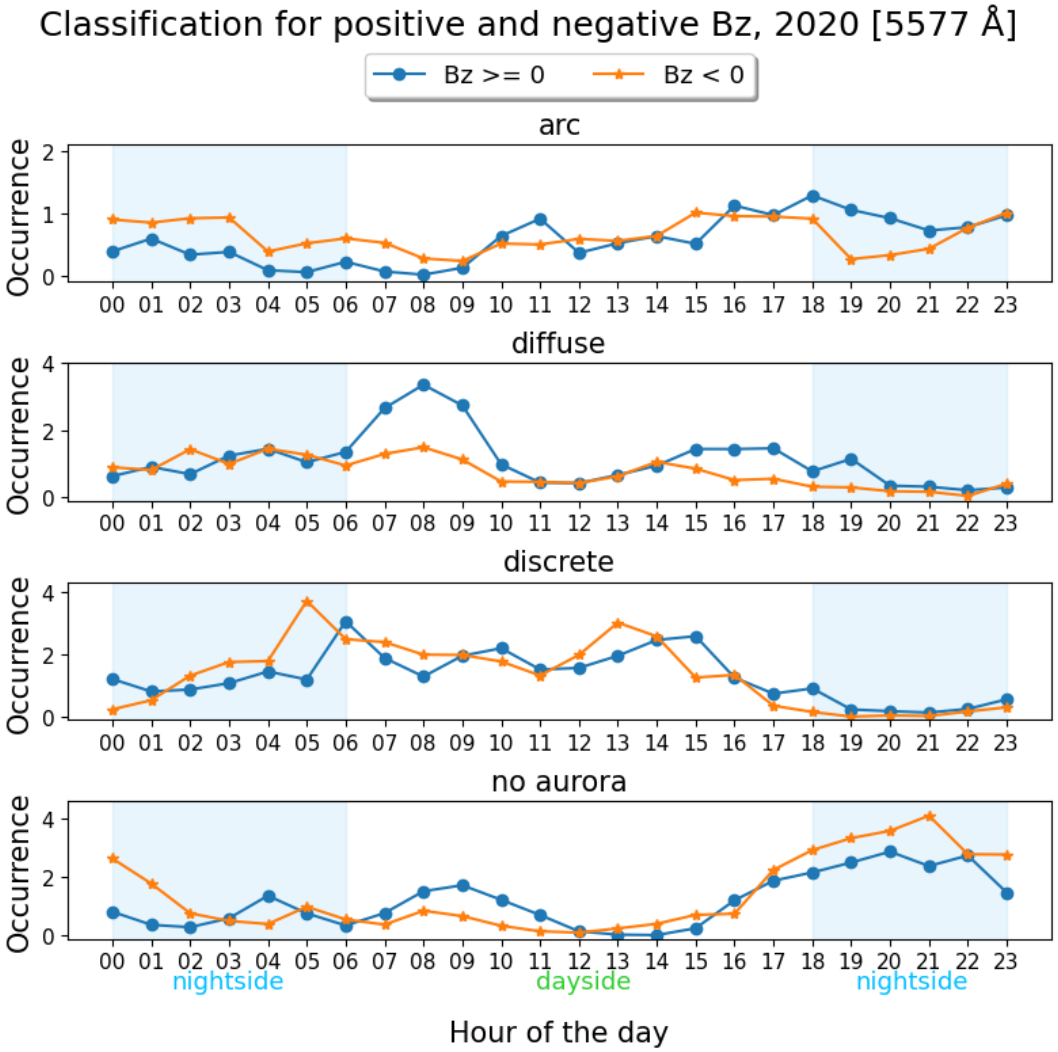


Figure 6.2.14: Positive and negative B_z classification for all classes, for January, November and December 2020. The y-axis show the normalized occurrence.

Figure 6.2.15 and 6.2.16 are the distribution for all years combined, for green and red aurora, respectively. The green auroral arcs have the highest occurrence from 11 to 01, while the occurrence are lower during morning. The southward B_z have the clearest peak around 11-12 local time. The occurrence of the red auroral arcs are lower during the day, compared to green arcs. The occurrence is also a little lower during night/morning for nighttime arcs. Overall, when we have a high occurrence of arcs for a timepoint/hour, the bin data only counts for about 1% of the total data. This is the same for a normal occurrence of diffuse aurora, where high occurrence can on average account for approximately 2%. The green diffuse aurora show a two-peak distribution, for northward B_z, with the main peak around 08-09 local time, and the secondary peak from 15-17 local time. We see the same tendency for southward B_z, but they are not as dominant. The two-peak distribution is not as clear for red diffuse aurora, but the main peak is there for northward B_z. The occurrence of diffuse red aurora are all-over a little higher than for green diffuse aurora.

Discrete aurora show a two-peak distribution for both B_z directions for red aurora, with the main peak around 07-08 local time. The secondary peak appears around 14-15 local time. The peaks appears an hour shifted, where the peaks for southward B_z discrete red aurora occur an hour earlier. The two-peak distribution is not as clear for the green diffuse aurora. For southward B_z we have the main peak, but the distribution for the northward B_z is high for daytime discrete aurora.

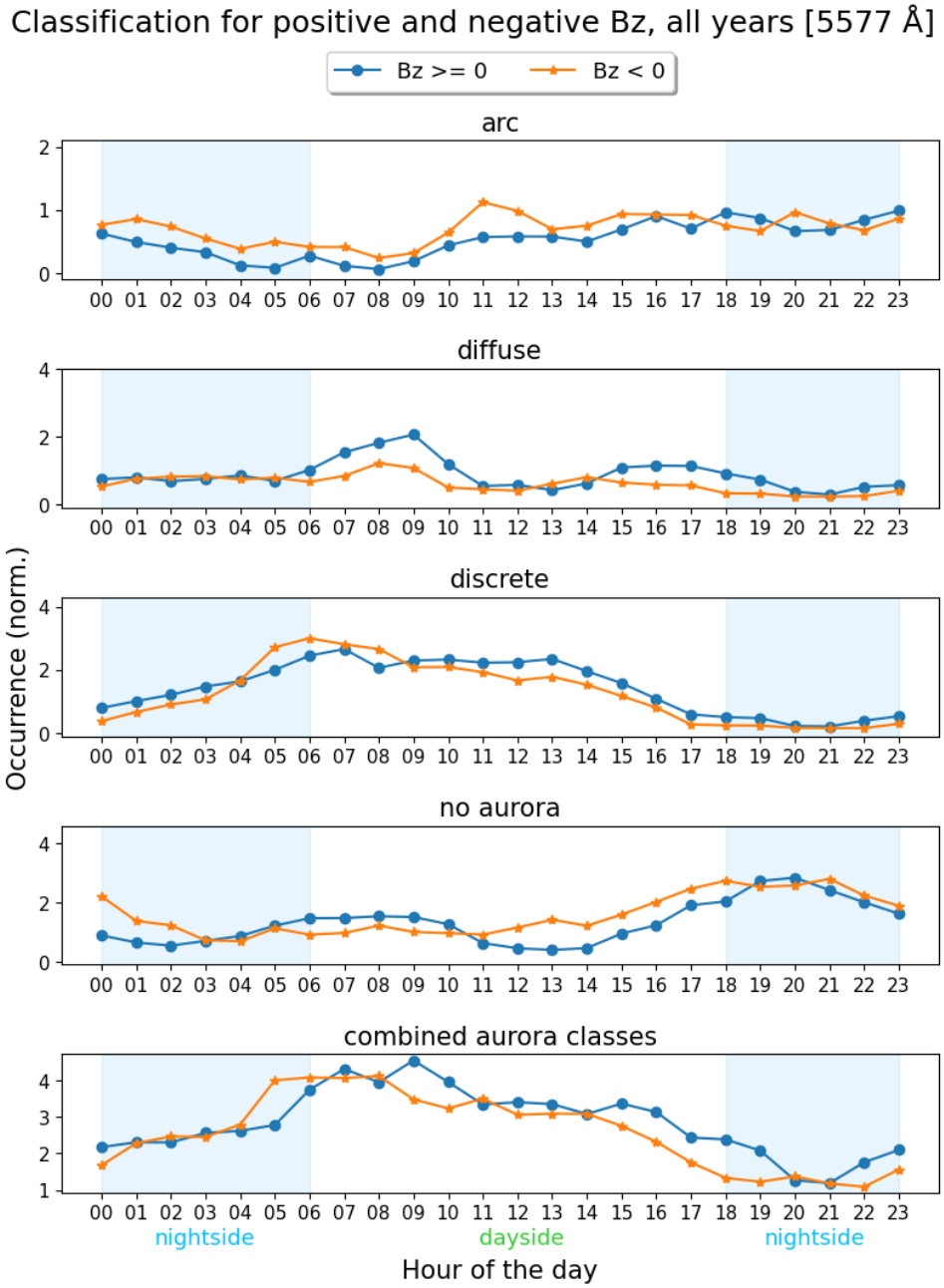


Figure 6.2.15: Positive and negative B_z classification for all classes, for January, November and December combined for all four years.

The no aurora class stands out with high occurrence during nighttime, for images

taken with both the red and green filter. The highest rounded peak around 19-21 local time, is slightly higher for images taken with the green filter.

For the combined auroral classes in the last panel of the two figures, we see that the aurora occurrence follow the same distribution, but it is shifted for approximately one hour between northward and southward B_z. The main peak is still dominant at 06-08 for the southward B_z, and 07-09 local time for the northward B_z. The aurora occurrence is on average for the four years low during night time, when we would expect to see nighttime aurora.

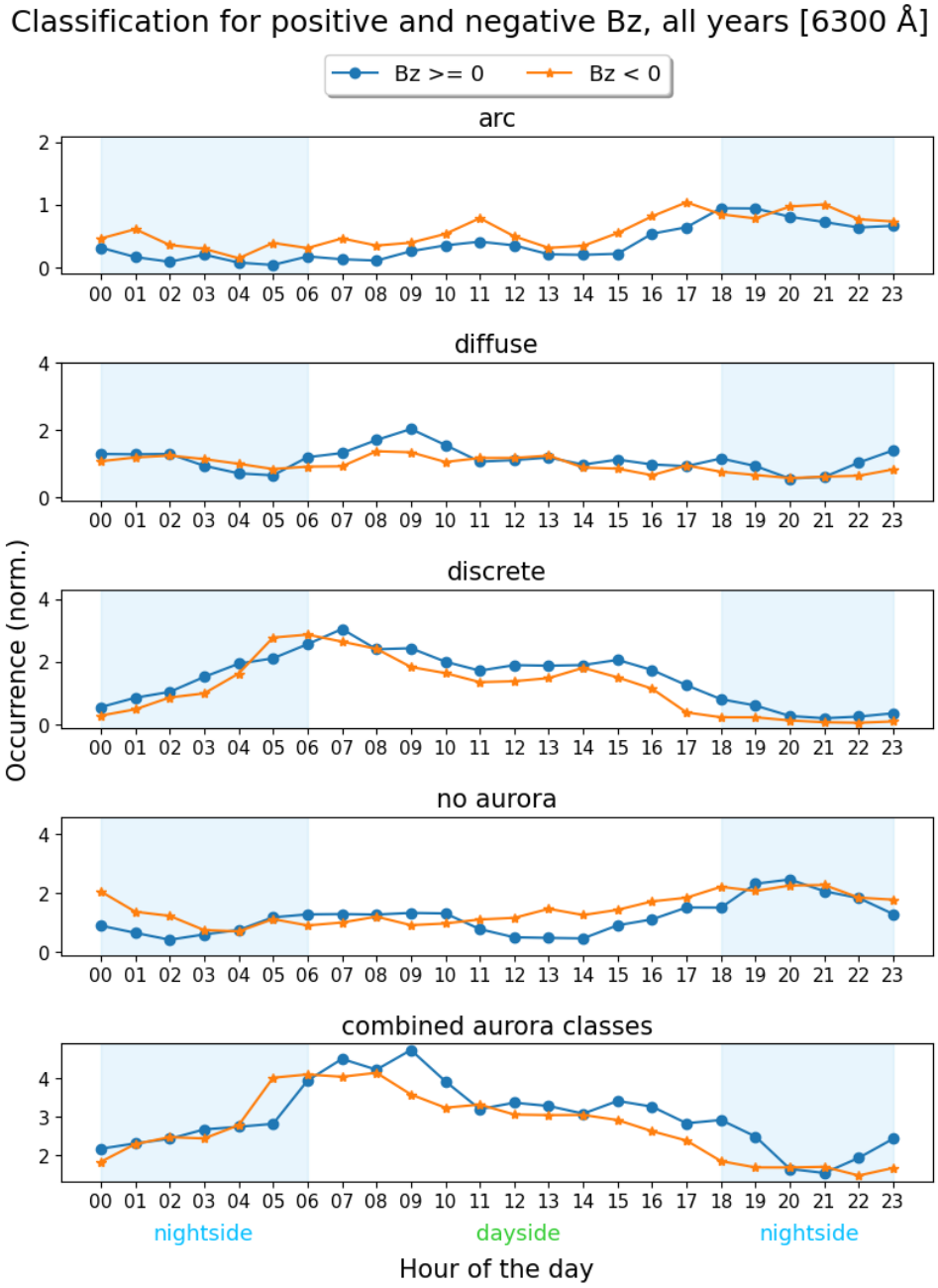


Figure 6.2.16: Positive and negative B_z classification for all classes, for January, November and December combined for all four years.

7 Discussion

Our study uses the state-of-the-art EfficientNet in an attempt to improve past results in automatic aurora classification [8][9][11]. Our EfficientNet AuroraB3 classifier was trained on manually labeled ASI data from Ny-Ålesund, Svalbard. On a 4-class classification problem (*no aurora*, *arc*, *diffuse* and *discrete*), AuroraB3 achieved a classification accuracy of 88% on unseen ASI data. Aggregating the 3 aurora classes, AuroraB3 archived a 96% classification accuracy on the same unseen test data.

AuroraB3 was applied on 665,865 unlabeled images from the same data source as the training and testing data. Our statistical results of the predicted labels, and their associated solar wind parameters, shows that variations in solar wind speed and IMF B_z do not determine the observed aurora shape. The most striking factor of the hourly distribution, were that nightside/nighttime aurora seams to be missing for diffuse and discrete aurora. Further, the no aurora class have many entries, showing that the images for the timepoints are labeled. From 15-00 local time, we mainly observe arcs when aurora is present.

7.1 Image labeling and AuroraB3

AuroraB3 achieved an validation accuracy of 90% and an test accuracy of 88%, with classes: no aurora, arc, diffuse and discrete. It is expected for the test accuracy to be slightly lower since AuroraB3 was tested on unseen data (Aurora_test). This indicate some overfitting by the network, which were expected from the train-validation loss evolution from figure 6.1.4. The small overfitting trend were not viewed as problematic, and the accuracy decrease of only 2% confirms this. When we only distinguished between the binary classification aurora and no aurora, the accuracy increased to 96%. An important step to achieve this accuracy was to correct for imbalanced classes. When we also weighted the three aurora classes a little more important than the no aurora class, the confusion matrix 6.1.3 shows an fairly even per-class accuracy. Without increasing the importance of the aurora features, the classifier tended to overfit the no aurora class at the expense of the diffuse class accuracy.

The largest error in not achieving an higher accuracy arise from misclassification between classes. The misclassification probably originate in the training data, where ambiguous images are wrongly labeled by humans. The confusion matrices from chapter 6.1.3 and table 6.2.2, shows that misclassification between no aurora/diffuse and discrete/diffuse are more common than for other class-combinations. Aurora belonging to these classes, are also the most common to be labeled ambiguously by the classifier. Image labeling was especially a problem between diffuse and discrete aurora, as the aurora takes all forms and levels of brightness, blurring the boundary between them. Sometimes, it also proved difficult to distinguish between no aurora and diffuse, e.g. between a thin layer of high altitude clouds and weak aurora shapes, or between fog/mist and diffuse glow-like aurora. Another problem that made labeling difficult, was that an im-

age could often include different types of aurora. Diffuse aurora was often seen with either an arc or discrete aurora. Many images that contains no aurora, had some light pollution in the form of arc-like light because of e.g. moon rising. The images we labeled were from random time points. It would be an advantage to label time-series of images, as it would be easier to distinguish between aurora classes and unclear weather conditions.

Although some previous work did achieved an higher accuracy ($\sim$91%) [8][11][10], the data and classes are often quite different. This makes it difficult to compare results directly. Previous work show a wide range of studies using different numbers and definitions of classes, from 2-3 broad classes including clouds to specific aurora sub-classes [8][9][67]. Another step some studies [11] take for achieving an greater accuracy is to only use non-ambiguous images, excluding e.g. images with light pollution and weather pollution. Creating a clean data set of well-defined aurora images. It is understandable to archive an greater accuracy if the classifier is trained and tested on these well-defined classes, as the classifier have clear features combined to each label. Kvammen et al. [11] achieved an 92% accuracy with ResNet-50 when excluding clearly ambiguous auroral forms and only using aurora sub-classes. The motivation behind creating a clean data set is to avoid confusing the network during training, we chose not to do this as real, untouched data the classifier is designed for include all these types of pollution's and ambiguous images. We believe the prediction results on the unseen data could included incorrect predictions hard to catch.

7.2 EfficientNet AuroraB3 aurora predictions

Statistical results were made from image labels and their associated solar wind parameters after AuroraB3 was applied on 665,865 unlabeled images. This thesis main focus have been creating a labeled data set, and training and testing various CNN models. The statistical results are only based on the occurrence of different types of aurora, based on the predicted labels. The results cannot say anything specific about location of e.g. the polar cusp or the position of the aurora in the imaged sky. The result can not say anything about the intensity/brightness or the dynamic of the aurora. This is because we are labeling single images. If we had used time-series, we could have made results e.g. for a substorm period.

Similar distributions between green and red aurora is expected, as the images are of the same aurora/time periods. What we observe from the different distributions is that red aurora have more entries classified as diffuse. This would indicate green aurora at lower altitudes show more discrete/clear shapes, and that the red aurora at a higher altitude start to loose these distinct shapes. This is accurate with the theory explained in chapter 2.4.2. It also indicates that the model predicts the labels correctly.

The solar wind and B_z distributions show that closer to solar minimum, we have a slower average solar wind speed (chapter 6.2.4) and a narrower B_z distribution (chapter 6.2.5). The same distributions that show the variations between classes, indicate that solar wind properties do not effect the shape of the aurora, as they

all follow the trend of the average distribution. Since our results indicates the intensity of the solar wind speed or direction of the IMF is an unsignificant source of determine the shape of the aurora, other more relevant factors need to be in play. This could be magnetospheric processes or ionospheric currents, which have not been a part of this study.

2014 is the period closest to the solar maximum, it would be expected for this year to display the highest solar wind speeds, but both 2016 and 2018 have an higher average SW speed. This could maybe be explained by coincidences of Earth's location compared to sunspot locations. The increased solar wind originate from sunpots, but the sunspot has to face the Earth during eruption for the fast solar wind speed to interact with Earth's magnetosphere. Solar minimum, year 2020, have an average low solar wind speed, but we did not observe less diffuse and discrete aurora. The low solar wind speed could have caused weaker substorms, keeping the OCB at a higher latitude, making the aurora visible in Ny-Ålesund.

Previous research on the statistical distribution of aurora occurrence, showing two distinct dayside aurora emission regions, have been done with satellite data (electron acceleration events and ultraviolet image observation) [3][4], and by three emission-line ASIs (4278 Å, 5577 Å and 6300 Å) [5]. The two regions are located in the prenoon and postnoon sectors of the aurora oval, and the "midday gap" is observed. The observations find the highest occurrence peak, the "hot spot" or "bright spot", at 14-15 magnetic local time (MLT). The weaker "warm spot" were found at around 6-9 MLT for two observations [5][3], and at 10 MLT for the ultraviolet image observation (1700 Å) [4]. These spots are located around 75° magnetic latitude (MLAT) [4].

We can also observe a double-peak feature for (some) diffuse/discrete dayside aurora from the distribution plots. Dayside aurora shows occurrence peaks in diffuse and discrete aurora at one or two recurring time points, around 06-09 and 14-16 local time. This means our observed peaks matches their, but for local time. When converting our time points to MLT, our observed peaks are approximately shifted +3 hours.

When we investigate the hourly distribution results, we observe that arcs occurs mainly from midday to midnight. Our high occurrence of images labeled no aurora during nighttime, for all four years, indicates the reason is caused by something other than e.g. clouds. The reasonably explanation is that Svalbard is located to high north to capture all the diffuse and discrete aurora caused by stronger events like substorms. Substorm will widen the OCB, and lower the aurora oval center equatorward. Since substorms are a result of the Dungey cycle, it explains why the absence of aurora occur for nightside aurora only.

8 Conclusions and future work

8.1 Conclusions

The time consuming task of classifying and labeling aurora images is an obstacle for auroral researchers in performing large-scale analysis of ground-based data. With convolutional neural networks constantly being improved and reaching new benchmarks (on ImageNet) it shows great promise for automatic aurora classification. We labeled 7,980 images from an All-Sky Imager located in Ny-Ålesund, Svalbard. Our EfficientNet-B3 convolutional neural network classifier, although trained and tested on data that include ambiguous images, achieved an accuracy of 88% on unseen data. The most important method we used for reducing overfitting was to correct for our imbalanced dataset. As many previous studies train their networks on datasets that exclude all or some of the ambiguous images, and only score about 3% better, we would conclude that our classifier does as designed. We classified arcs with an per-class accuracy of 93%, while the difficult diffuse aurora class achieved an 84% accuracy.

A new set of data, containing the available data for 2014, 2016, 2018 and 2020 (Jan, Nov and Dec) for both 5577 Å and 6300 Å emission, were constructed from Ny-Ålesund All-Sky Imager data. The images were matched with solar wind parameters from NASA's OMNI data. We used the total of 665,865 unlabeled images for a statistical analysis after applying our classifier on the data. Previous studies [3][4][5] have presented a double-peak feature for dayside aurora. The dominant "hot spot" (14-15 MLT) and the weaker "warm spot" (around 6-9 MLT) peak are divided by the "midday gap" at local noon. Our results indicate the same double-peak feature, but the intensity of the peak are switched, and the times are shifted approximately +3 hours for MLT. Without knowing more about the location of the polar cusp, it's hard to conclude something from these results.

The main result we can conclude with from the statistical analysis is that Svalbard is perfect for observing dayside aurora, at least with ASIs. The occurrence of dayside aurora are not affected much by the variations in solar wind parameters. This can be explained by the constant stream of charged particles into the polar cusp. We can also say that observing strong aurora event on Svalbard can be a letdown, as the discrete aurora is visible on the sky on locations further south. There are however still an high occurrence of arc and some diffuse aurora. Further, our classifier labeled more images diffuse for the 6300 Å emission line, which were expected and hoped for, as red aurora have a weaker, and more diffuse display because of the time it takes the photon to be emitted.

8.2 Future work

Many different adaptations, tests, and experiments can still be performed for automatic aurora classification. Some ideas for future work within automatic classification of auroral images include multi-labeling images, creating a benchmark test dataset of auroral images and classifying time-series of data.

- After labeling the original data for this thesis, it is clear that a labeling system with multiple assigned labels (ranged by priority) would be beneficial. This would hopefully make classification easier for ambiguous data.
- Researchers are not agreeing on how auroras should be classified, but having one set of classes would enable researchers to easier compare classification results. If there was an agreement of classes, a larger labeled dataset could be composed, including all-sky images from various locations. This set could then be used for both training and testing of new classification methods.
- To make statistics directed more towards aurora dynamics or auroral event intensity, classification of time series could be a good experiment, using recurrent neural networks.
- A more computational expensive model could be trained with a stronger GPU.

For our own results, a further step could be to include the solar wind B_y component. We would also split the no aurora class into clear and cloud. Another step we could take would be to test the model on data from other ASI's. To test the accuracy for a different camera and different sky location. A test set with labeled images would have to be composed/provided.

Bibliography

[1] University of Oslo. *Svalbard All-Sky Imager Data: Availability*. URL: http://tid.uio.no/plasma/aurora/index.html (visited on 10/13/2021).

[2] NASA OMNIWeb. *OMNIWeb: High Resolution OMNI*. URL: https://omniweb.gsfc.nasa.gov/form/omni_min.html (visited on 11/10/2021).

[3] Patrick T. Newell, Kevin M. Lyons, and Ching-I. Meng. "A large survey of electron acceleration events". In: *Journal of Geophysical Research: Space Physics* 101.A2 (1996), pp. 2599–2614. DOI: https://doi.org/10.1029/95JA03147.

[4] K. Liou et al. "Synoptic auroral distribution: A survey using Polar ultraviolet imagery". In: *Journal of Geophysical Research: Space Physics* 102.A12 (1997), pp. 27197–27205. DOI: https://doi.org/10.1029/97JA02638.

[5] Z.-J. Hu et al. "Synoptic distribution of dayside aurora: Multiple-wavelength all-sky observation at Yellow River Station in Ny-Ålesund, Svalbard". In: *Journal of Atmospheric and Solar-Terrestrial Physics* 71.8 (2009), pp. 794–804. ISSN: 1364-6826. DOI: https://doi.org/10.1016/j.jastp.2009.02.010.

[6] Jon Nesvold and Jon Andre Ottesen. *CRAI-Nets*. CRAI at Oslo University Hospital, Aug. 2021. URL: https://pypi.org/project/crainets/0.1.1b0/.

[7] J. W. Dungey. "Interplanetary Magnetic Field and the Auroral Zones". In: *Phys. Rev. Lett.* 6 (2 Jan. 1961), pp. 47–48. DOI: 10.1103/PhysRevLett.6.47. URL: https://link.aps.org/doi/10.1103/PhysRevLett.6.47.

[8] J. Rao et al. "Automatic Auroral Detection in Color All-Sky Camera Images". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.12 (2014), pp. 4717–4725. DOI: 10.1109/JSTARS.2014.2321433.

[9] Lasse B. N. Clausen and Hannes Nickisch. "Automatic Classification of Auroral Images From the Oslo Auroral THEMIS (OATH) Data Set Using Machine Learning". In: *Journal of Geophysical Research: Space Physics* 123.7 (2018), pp. 5640–5647. DOI: 10.1029/2018JA025274.

[10] Jeremiah W. Johnson et al. "A Contrastive Learning Approach to Auroral Identification and Classification". In: *CoRR* abs/2109.13899 (2021). URL: https://arxiv.org/abs/2109.13899.

[11] A. Kvammen et al. "Auroral Image Classification With Deep Neural Networks". In: *Journal of Geophysical Research: Space Physics* 125.10 (2020), e2020JA027808. DOI: https://doi.org/10.1029/2020JA027808.

[12] Dr. David H. Hathaway. *The Solar Wind*. Aug. 2014. URL: https://solarscience.msfc.nasa.gov/SolarWind.shtml.

[13] M.J. Owens and R.J. Forsyth. "The Heliospheric Magnetic Field". In: *Living Rev. Sol. Phys.* 10.5 (2013). URL: https://doi.org/10.12942/lrsp-2013-5.

[14] C.T. Russell, J.G. Luhmann, and R.J. Strangeway. *Space Physics: An Introduction*. Cambridge University Press, 2016. ISBN: 9781107098824.

[15] SILSO World Data Center. "The International Sunspot Number". In: *International Sunspot Number Monthly Bulletin and online catalogue* (). URL: http://www.sidc.be/silso/.

[16] G. L. Siscoe and T. S. Huang. "Polar cap inflation and deflation". In: *Journal of Geophysical Research: Space Physics* 90.A1 (1985), pp. 543–547. DOI: https://doi.org/10.1029/JA090iA01p00543.

[17] Yngve Vogt. "The king of northern lights". In: *Appolon* (2017). URL: https://www.apollon.uio.no/english/articles/2017/birkeland_english.html.

[18] Jan A. Holtet. *Nordlys i Store norske leksikon*. Retrieved: 08.12.2021. URL: https://snl.no/nordlys.

[19] William Copeland. *What are the northern lights?* Sept. 2020. URL: https://nordnorge.com/en/artikkel/what-are-the-northern-lights/.

[20] M. Hatfield (NASA's Goddard Space Flight Center). *Science on the Cusp: Sounding Rockets Head North*. Nov. 2018. URL: https://www.nasa.gov/feature/goddard/2018/science-on-the-cusp-sounding-rockets-head-north.

[21] C.T Russell. "The polar cusp". In: *Advances in Space Research* 25.7 (2000). Proceedings of the DO.1 Symposium of COSPAR Scientific Commission D, pp. 1413–1424. ISSN: 0273-1177. DOI: https://doi.org/10.1016/S0273-1177(99)00653-5. URL: https://www.sciencedirect.com/science/article/pii/S0273117799006535.

[22] X. W. Zhou et al. "Solar wind control of the polar cusp at high altitude". In: *Journal of Geophysical Research: Space Physics* 105.A1 (2000), pp. 245–251. DOI: https://doi.org/10.1029/1999JA900412.

[23] R. Saunders and E. Cunningham. *Why is Earth leaking into space?* Video edited by M. Popovski. (Image timepoint: 3:59). URL: https://www.youtube.com/watch?v=T9RomHPVv04&t=273s&ab_channel=PrimalSpace.

[24] S.-I. Akasofu. "The development of the auroral substorm". In: *Planetary and Space Science* 12.4 (1964), pp. 273–282. ISSN: 0032-0633. DOI: https://doi.org/10.1016/0032-0633(64)90151-5.

[25] R. L. McPherron, C. T. Russell, and M. P. Aubry. "Satellite studies of magnetospheric substorms on August 15, 1968: 9. Phenomenological model for substorms". In: *Journal of Geophysical Research (1896-1977)* 78.16 (1973), pp. 3131–3149. DOI: https://doi.org/10.1029/JA078i016p03131.

[26] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.

[27] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[28] Morten Hjorth-Jensen. *Data Analysis and Machine Learning: Neural networks, from simple perceptron to deep learning*. https://compphysics.github.io/MachineLearning/doc/pub/NeuralNet/pdf/NeuralNet-minted.pdf. pp. 1-12. Oct. 2019.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Chap. 6 and 9. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[30] Pankaj Mehta et al. "A high-bias, low-variance introduction to Machine Learning for physicists". In: *Physics Reports* 810 (2019), pp. 49–60. DOI: https://doi.org/10.1016/j.physrep.2019.03.001.

[31] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314.
[32] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. *Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning*. 2017. arXiv: 1702.03118 [cs.LG].
[33] Vinod Nair and Geoffrey Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair". In: vol. 27. June 2010, pp. 807–814.
[34] Xavier Glorot, Antoine Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: vol. 15. Jan. 2010.
[35] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.
[36] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
[37] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: 1710.05941 [cs.NE].
[38] PyTorch. *torch.argmax*. URL: https://pytorch.org/docs/stable/generated/torch.argmax.html#torch.argmax (visited on 02/20/2022).
[39] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: https://doi.org/10.1038/323533a0.
[40] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].
[41] Scikit-learn. *sklearn.metrics.confusion_matrix*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html (visited on 12/16/2021).
[42] Scikit-learn. *sklearn.metrics.precision_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html (visited on 12/16/2021).
[43] Scikit-learn. *sklearn.metrics.recall_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall.html (visited on 12/16/2021).
[44] Scikit-learn. *sklearn.metrics.f1_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (visited on 12/16/2021).
[45] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
[46] University of Oslo. *Svalbard All-Sky Imager Data: Technical Information*. URL: http://tid.uio.no/plasma/aurora/tech.html (visited on 10/13/2021).
[47] Rolf Stange. *Midnight sun and polar night*. Mar. 2021. URL: https://www.spitsbergen-svalbard.com/spitsbergen-information/midnight-sun-polar-night.html.
[48] Kamilla Pedersen. *Nå drar sola for vinteren*. Nov. 2013. URL: https://www.yr.no/artikkel/na-drar-sola-for-vinteren-1.11368304.
[49] NASA Science: Solar System Exploration. *ACE*. Retrieved: 05.11.2019. URL: https://solarsystem.nasa.gov/missions/ace/in-depth/ (visited on 11/10/2021).

[50] Neil J. Cornish. *What is a Lagrange Point? NASA* Solar System Exploration. URL: https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/ (visited on 11/22/2021).
[51] Lynn B. Wilson. *Wind Spacecraft*. *NASA* Goddard Space Flight Center. URL: https://wind.nasa.gov/index.php (visited on 11/10/2021).
[52] Susannah Darling and Lina Tran. *25 Years of Science in the Solar Wind*. *NASA*. URL: https://www.nasa.gov/feature/goddard/2019/25-years-of-science-in-the-solar-wind (visited on 11/23/2021).
[53] Brian Dunbar. *Advanced Composition Explorer*. *NASA*. URL: https://www.nasa.gov/ace (visited on 11/10/2021).
[54] H. Zell. *ACE, Workhorse Of NASA's Heliophysics Fleet, Is 15*. *NASA* Goddard Space Flight Center. URL: https://wind.nasa.gov/images/WIND.jpg (visited on 11/22/2021).
[55] G. van Rossum. *Python tutorial*. Tech. rep. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI), May 1995.
[56] Anaconda Inc. *Anaconda Software Distribution*. 2022. URL: https://anaconda.com.
[57] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.
[58] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.
[59] Scikit-learn. *Machine learning in Python*. URL: https://scikit-learn.org/stable/.
[60] PyTorch. *An open source machine learning framework.* URL: https://pytorch.org/ (visited on 08/22/2021).
[61] Joe King, GSFC/SPDF Natalia Papitashvili, and Inc. ADNET Systems. *High-Resolution OMNI data set*. URL: https://omniweb.gsfc.nasa.gov/html/HROdocum.html (visited on 11/10/2021).
[62] V. Lobzin and V. Krasnoselskikh. *Cluster reveals the reformation of the Earth's bow shock*. *ESA*. May 2007. URL: https://sci.esa.int/s/8rkEg6A.
[63] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].
[64] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). URL: http://arxiv.org/abs/1608.06993.
[65] Yanping Huang et al. "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism". In: *CoRR* abs/1811.06965 (2018). URL: http://arxiv.org/abs/1811.06965.
[66] Papers with Code. *Image Classification on ImageNet*. URL: https://paperswithcode.com/sota/image-classification-on-imagenet.
[67] Qiuju Yang et al. "Extracting Auroral Key Local Structures From All-Sky Auroral Images by Artificial Intelligence Technique". In: *Journal of Geophysical Research: Space Physics* 124 (May 2019). DOI: 10.1029/2018JA026119.