NTNU Trondheim
Norwegian University of Science and Technology
Department of Engineering Cybernetics

PROJECT DESCRIPTION SHEET

Name of the candidate: Sondre Foslien

Field of study: Cybernetics and Robotics

Thesis title (Norwegian): Maskinlæring for klassifisering av isfjell/skip i satellittbilder

Thesis title (English): Machine learning for classifying iceberg/ship in satellite images

Background
Ocean surveillance is today done with satellite pictures, inspected by human operators. They use their extensive domain knowledge to classify the objects on the images, including icebergs and ships. This is a time consuming process, it is not perfect and is not scalable as more and more satellites is placed into orbit and more and more pictures arrive. As Equinor moves their operation towards the Barents sea and outside Newfoundland, it gets more critical to spot potentially hazardous icebergs early. A potential solution for this is to employ some machine learning technique.

Statoil, together with partner C-CORE, have hosted a competition on Kaggle.com where the international machine learning community was challenged to come up with the best classifier to distinguish icebergs from ships from satellite images. For this competition, a unique dataset of 5000 targets has been made available. The focus and challenge of this project is the investigation and understanding of the most successful machine learning approaches used during the Kaggle competition on this particular problem and dataset.

Work description
1. Perform a background and literature review to provide information and relevant references on:
    - The problem itself
    - The production of the dataset and the satellite technology used
    - State-of-the-art non-machine learning techniques for classifying satellite images
    - State-of-the-art machine learning techniques for classifying satellite images

2. Analyzing the different techniques proposed on the Kaggle forums and found in the top 3 solutions, particularly those related to the dataset and mitigation of problems with the dataset. Describe state of the art, expected benefits and drawbacks with respect to the particular problem/dataset considered.

3. Select and implement some techniques used to address challenges in the dataset.

4. Evaluate results on the dataset, draw comparisons with results obtained during the Kaggle competition.

Specifications
The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts. and not be longer than 40 A4 pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, project specification, list of symbols and acronyms, table of contents, introduction and background, problem formulations, scope and delimitations, main body with derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The report shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this project description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

Start date: 1 September 2018 **Due date:** As specified by the administration.

Supervisor: Francesco Scibilia
Co-advisor(s): Massimiliano Ruocco

Trondheim, _____

_____
Francesco Scibilia
Supervisor

Table of Contents

Table of Contents iv

List of Figures vi

Abbreviations vii

1 Introduction 1
- 1.1 The problem . . . 1
- 1.2 Kaggle . . . 2
- 1.3 The dataset . . . 2
- 1.4 Motivation . . . 3

2 Background 5
- 2.1 Machine learning . . . 5
  - 2.1.1 Neural networks . . . 6
- 2.2 Remote sensing . . . 7
  - 2.2.1 Radars . . . 7
- 2.3 State-of-the-art iceberg-ship classification . . . 12
  - 2.3.1 Non-machine learning . . . 12
  - 2.3.2 Machine learning . . . 13

3 Dataset problem mitigation techniques 15
- 3.1 Image augmentation . . . 15
  - 3.1.1 Flipping and flopping . . . 16
  - 3.1.2 Height and width shifts . . . 17
  - 3.1.3 Zooming . . . 18
  - 3.1.4 Rotation . . . 18
  - 3.1.5 Contrast, color and brightness . . . 19
- 3.2 Pseudo labelling . . . 19
- 3.3 Test-time augmentation . . . 20
- 3.4 Lee filtering . . . 22

3.5 Discrete Fourier Transform 22
3.6 Autoencoders 23
3.6.1 Convolutional autoencoders 23

4 Own testing **25**
4.1 Method 25
4.2 Image augmentation 26
4.2.1 Results 27
4.3 Pseudo labelling 29
4.3.1 Results 29
4.4 Test-time augmentation 31
4.4.1 Results 31

5 Discussion **33**
5.1 Image augmentation 33
5.2 Pseudo labelling 34
5.3 Test-time augmentation 35

6 Conclusion and future work **37**

Bibliography **38**

List of Figures

1.1 An example image from the dataset, including both HH and HV polarization. The image illustrates a ship. 2

2.1 A figure illustrating a typical layout for a neural network. Figure courtesy of Glosser.ca (2013) 6
2.2 A figure showcasing the two different sensor types. To the left is an active sensor, where a radar satellite emits a signal and gets a reflection back. The middle figure is a radar that measures emissions from the earth. And the rightmost figure is a radar measuring the reflections of the sun from the earth. All icons courtesy of Icons8 (n.d.). 8
2.3 Figure showing the azimuth, including ground range, nadir, slant range and flight path 9
2.4 Figure illustrating the incidence angle along with slant range, ground range and nadir. 9
2.5 A figure illustrating the concept of polarization, with the horizontal and the vertical axis marked as H and V. 10
2.6 An example of how the range resolution works. The figure illustrates target A and B and the radar. There is two different reflected electromagnetic waves, one from each of the targets. Since the tail of the first reflection is ahead of the head of the second reflection, it is possible to recognize two distinct scatters. 11

3.1 The figure shows the first original M followed by six M's stroke warped different ways. Courtesy of Yaeger et al. (1996) 15
3.2 A sample image from the data-set before and after flipping verticially. . . 17
3.3 A sample image from the data-set after shifting 8 pixels along the x-axis and 8 pixels along the y-axis. 17
3.4 A sample image from the data-set and a version zoomed in 20% and a version zoomed out 20%. 18

3.5 A sample image from the data-set and a version rotated a random degree between 0 and 30 counterclockwise. 19
3.6 A plot of the function $f(x) = x\log(x)$ 20
3.7 An illustration presenting the concept of autoencoders, encoder and decoder. Figure is courtesy of Valkov (2017). 23

4.1 A figure illustrating the VGG16 architechture. Modified from Blier (2016) 26
4.2 A graph illustrating the different classification losses between the four different augmenting techniques and the baseline. The baseline result is set to 1, and the other losses are scaled accordingly. 27
4.3 A graph illustrating the classification results when combining different augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly. 28
4.4 A graph illustrating the classification results on the CIFAR10 dataset for different augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly. 28
4.5 A graph illustrating the classification results when using pseudo labelling for a 70/30 test/train split . The baseline result is set to 1, and the other losses are scaled accordingly. 30
4.6 A graph illustrating the classification results when using pseudo labelling for a 30/70 test/train split . The baseline result is set to 1, and the other losses are scaled accordingly. 30
4.7 A graph illustrating the classification results when using different test-time augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly. 31
4.8 A graph illustrating the classification results when using different combinations of test-time augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly. 32

5.1 Two example figures from the Equinor/C-CORE dataset and the CIFAR10 dataset 33

Abbreviations

SAR	=	Synthetic Aperture Radar
RBG	=	Red, Green and Blue
CNN	=	Convolutional Neural Network
ILSVRC	=	ImageNet Large-Scale Visual Recognition Challenge
Radar	=	Radio Detection And Ranging
ESA	=	European Space Agency
RAR	=	Real Aperture Radar
GMES	=	Global Monitoring for Environment and Security
SM	=	Stripmap
IW	=	Inferometric Wide Swath
EW	=	Extra-Wide Swath
WV	=	Wave mode
CFAR	=	Constant False Alarm Rate
SCR	=	Signal-to-clutter
S-AIS	=	Space-based Automatic Identification System
QD	=	Quadratic Discriminant
SVM	=	Support Vector Machines
MNIST	=	Modified National Institute of Standards and Technology
MAP	=	Maximum A Posteriori
TTA	=	Test-Time Augmentation
DFT	=	Discrete Fourier Transform
DBSCAN	=	Density-Based Spatial Clustering of Applications with Noise
CIFAR	=	Canadian Institute for Advanced Research

Chapter 1

Introduction

This chapter will introduce the problem at hand. This includes section 1.1 which will present the problem, and steps done to solve it. Section 1.2 will give the reader a insight into the Kaggle platform, which is a requirement to understand the problem. Section 1.3 will present the dataset which will be used for this project, and section 1.4 will give an insight into why this is an interesting problem to solve.

1.1 The problem

In January of 2018, Equinor, together with partner C-CORE, hosted a competition on Kaggle.com. In this competition, the international machine learning community was challenged to make the best classifier for discriminating between icebergs and ships in satellite Synthetic Aperture Radar (SAR) imagery. Many different techniques were applied, by the 3343 international teams who entered the competition. The focus and challenge of this project will be the investigation and understanding of the most successful machine learning approaches used during the Kaggle competition on this particular problem and dataset. This will be used to build a foundation for a master thesis, where a more complete classifying solution will be implemented.

This project will contain:

1. A background and literature review of the problem at hand, including satellite imagery, machine learning and the state-of-the-art techniques for classifying targets in satellite SAR images

2. A review of the different techniques used in the Kaggle competition, particularly those related to the dataset.

3. An implementation of some of those techniques and a evaluation on the given dataset.

1.2 Kaggle

Usmani (2017) gives an introduction to Kaggle, which will be presented here. Kaggle is a crowd-sourced platform made to attract, train and challenge data scientist to solve data science and machine learning problems. A lot of data scientists are only theorists who rarely get to practice their art before being employed. The Kaggle team want's to challenge that, by providing a platform for hosting real-life data science and machine learning competitions. The competitions can be anything from projects with only an educational purpose, to genuine problems from real-life companies. Participation is incentivized by rewards which can be anything from job offers to monetary rewards. The most remarkable were Heritage Heath, who offered $\$3\ 000\ 000$ in prize money for solving their problem. Many of the prize pools hover around the $\$10\ 000$ to $\$50\ 000$ range.

The Kaggle community is big, with over one million members, having submitted over four million models to different competitions (Usmani (2017)). Of these four million submissions, $47\ 799$ was submitted to the Equinor/C-CORE competition, with an average of 514 submissions each day. This makes it the most popular image based competition of all time, and the seventh most popular of all Kaggle competitions (C-CORE (2018)).

1.3 The dataset

The dataset selection process is described in C-CORE (2018). The dataset is a state-of-the-art dataset that consists of Sentinel-1 imagery, mostly acquired along the east coast of Newfoundland and Labrador in 2017. With some additional supplements from other parts of the word, the final dataset contains 2553 icebergs and 2488 ships. The targets were classified using expert analysts.

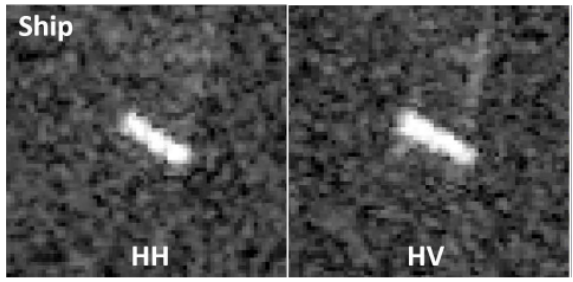


Figure 1.1: An example image from the dataset, including both HH and HV polarization. The image illustrates a ship.

Each sample is a 75×75 two channel image containing the HH and HV channel from the Sentinel-1 SAR imagery. This is different from RGB images, as each pixel represents the intensity of the reflected signal the earth surface produces when illuminated by a energy beam. This will be discussed further in section 2.2. The images were collected using Sentinel-1 Interferometric Wide Swath mode. There is no georeferencing information, but it contains metadata about the incidence angle of the target chip. An example image is

shown in figure 1.1. The images were subject to several constraints: Each image should contain a target in open ocean, with only one target per image and no image borders or ambiguities. This resulted in over 5000 images like what's shown in figure 1.1

1.4 Motivation

Satellite imagery is today widely used for ocean surveillance, where a human operator is typically responsible for the image analysis. They use their extensive domain knowledge to classify the objects in the images, including icebergs and ships. This is a time consuming process, it is not perfect nor is it scalable as more satellites are placed into orbit and more pictures are produced. If Equinor wants to move their operation towards the Barents sea and outside Newfoundland, it gets more critical to spot potentially hazardous icebergs early. The introduction of machine learning used for image classification in the last decade rises the question if it could be used for a more effective surveillance of icebergs. The motivation for this project is to explore this idea, looking at what's been done in the field earlier and what were done under the Kaggle competition hosted by Equinor and C-CORE in 2018.

Chapter 2

Background

This section present the background theory needed for the rest of the project, and puts the work done in relation to earlier research. This includes section 2.1 which presents background theory related to machine learning and image classification, section 2.2 presenting remote sensing and radars, as well as a review of current state-of-the-art techniques for discriminating between icebergs and ships in section 2.3.

2.1 Machine learning

Alpaydin (2010) gives an introduction to machine learning. According to him, it is when a machine learns by itself to solve a problem. This is typically used when the algorithm needed to solve a problem is unknown. Then we make the machine design the algorithm instead. A typical example of this is classifying e-mail as spam or not spam. The problem is known, the input is known and the desired output is known. But the algorithm mapping the input to the correct output is unknown. Instead a computer can be used to infer the algorithm. In many machine learning cases it is extracted from large amounts of data. Machine learning is normally divided into three categories: supervised learning, unsupervised learning and reinforcement learning. Unsupervised learning uses lots of data, but no ground truth to extract some sort of pattern that can be useful. Reinforcement learning interacts with some environment to learn the optimal way of controlling it based on some sort of reward function. Supervised learning, uses large amounts of data and a ground truth to find a mapping from the input data to the corresponding ground truth. Supervised learning will be the main focus of this report.

It is worth noting that Kaggle competitions are a machine learning competition, so any machine learning algorithm can be used. For further reading into the machine learning world, the reader is directed to Alpaydin (2010), Raschka (2015) or any other book on the subject. By far, the most heavily used machine learning algorithm in the competition were neural networks. Therefore, neural networks will be a focus of this report and will be presented in the next section.

2.1.1 Neural networks

Neural networks are the most frequently used algorithm in the Kaggle competition. This is probably due to it's historically high performance in image classification tasks. This can, at least partly, be attributed to the invention of convolutional neural networks (CNN), a type of neural network. CNN will be presented later.

According to Wu (1992), neural networks are heavily inspired by the human brain. The neural network is based on a large amount of processing units, often called neurons, which are connected in such a way that they can learn from data they experience. Normally, the network is organized in layers, where a neuron has a connection to each of its predecessors in the earlier layer, as can be seen in figure 2.1. The first layer is a input layer, which is where the network is given its input. The last layer is the output layer, which produces the output. Finally, we have the hidden layer, which is used to add complexity so the network is able to find complex patterns. You can have any number of hidden layers. Each neuron produces a output signal based on the sum of the weighted inputs from the last layer:

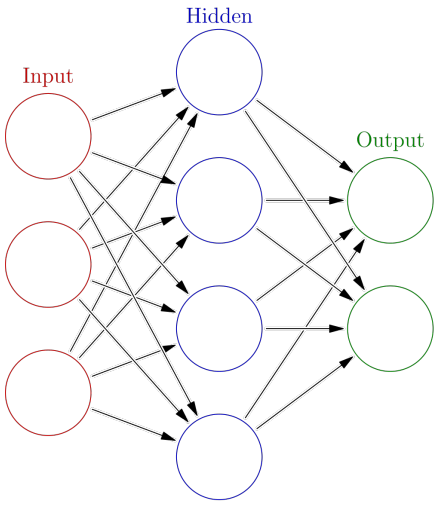


Figure 2.1: A figure illustrating a typical layout for a neural network. Figure courtesy of Glosser.ca (2013)

$$y_i = f(\sum_{j=1}^{n} w_{ji} x_j) \tag{2.1}$$

where y_i is the output, w_{ji} is the weight of the connection from neuron j in the last layer to neuron i in the current layer, x_j is the output of neuron j in the previous layer and f is a nonlinear activation function. The weights w_{ij} are adjusted based on an algorithm called backpropagation. A loss function (i.e. mean squared error, binary cross entropy etc) uses the network output and the ground truth to calculate some loss value. The gradient of this loss function is calculated and used to determine how to adjust each weight in order to minimize the loss.

$$\Delta w_{ij} = -\eta \frac{\partial L(y, x)}{\partial w_{ij}} \tag{2.2}$$

where Δw_{ij} is the adjustment of w_{ij}, η is a constant between 0 and 1 determining the network learning rate and $L(y, x)$ is the loss function given a ground truth y and a network output x.

For more information about neural networks and anything related, the reader is suggested reading Goodfellow et al. (2016), Wu (1992), LeCun et al. (2015) or any other

introduction litterature to neural networks.

Convolutional Neural Networks

CNN's was a major breakthrough for neural network applications in image processing. A problem with densely connected neural networks, is that the amount of learnable parameters quickly rises to enormous amounts with increasing input size and hidden layer sizes. A 28×28 pixel image results in $28 \cdot 28 = 784$ input neurons. A small 30 neuron hidden layer would then result in $784 \cdot 30 = 23520$ learnable parameters.

To alleviate this problem the idea of CNN's were proposed. It is based on two basic ideas: local receptive fields and shared weights. Instead of connecting every neuron in the last layer to every neuron in the next, a small, local receptive field of neurons (for example 5×5 neurons) are connected to a single neuron in the next layer. This local receptive field moves with a certain stride between each step. These 5×5 weights are also shared, which further cuts down on the number of parameters. This is especially well suited for images, since the shared weights will act like a filter, looking for the same patterns over the entire image. It exploits the spatial invariance of an image. To learn more about CNN's Goodfellow et al. (2016) is a good bet, but also Nielsen (2015).

CNN's sprung into popularity for image classification purposes in 2012 when AlexNet, a CNN architecture, (Krizhevsky et al. (2012)) and the SuperVision-team from University of Toronto won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) with an error of 15.4% (Russakovsky et al. (2015)). Second place had a 26.2% error rate. The SuperVision team was the first to use CNN's successfully in this contest, and it was the first major proof that CNN's worked. In the 2013 ILSVRC a majority of the contestants used CNN's for classification. This 2012 competition is seen as the breakthrough for CNN's, which has been immensely popular ever since.

2.2 Remote sensing

There exists two ways to collect spatial data. The first is called ground-based methods, which is when one does in situ measurements and perform land surveys. Opposite to these methods, is remote sensing methods. Remote sensing methods uses a sensor to acquire image data, which can be derived into information and a limited representation of the real world. This means that the information is collected by a device that's not in contact with the objects being measured. The remote way of collecting is sometimes the only applicable, and is cost effective and scalable compared to the ground-based methods (Kerle et al. (2004)). It was therefore used as the preferred sensing method for iceberg/ship surveillance.

2.2.1 Radars

One of the primary techniques of remote sensing is radars, which is an acronym for **Ra**dio **D**etection **A**nd **R**anging. Radars can either have passive or active sensors. (Kerle et al. (2004))

The active radar has it's own energy source. It emits a controlled beam of energy to the surface, and measures the amount of energy that was reflected back. Since it's not dependent on some external source of energy, it works at all hours of the day. (Kerle et al. (2004))

The passive radar uses natural sources of energy, which it is designed to receive and measure. This could be emissions made by the earth or radiation from the sun. This sensor is often referred to as a optical sensor. It is easy to interpret for a human, as the optical sensor represents the earth the same way as the eye does. At the same time, it has the disadvantage that it is highly dependant on uncontrollable external energy sources. An example of this is the illumination conditions from the sun. (Kerle et al. (2004))

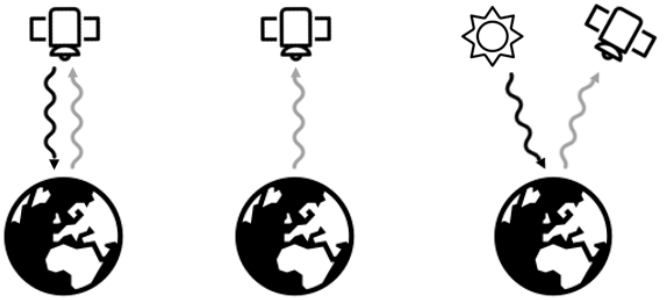


Figure 2.2: A figure showcasing the two different sensor types. To the left is an active sensor, where a radar satellite emits a signal and gets a reflection back. The middle figure is a radar that measures emissions from the earth. And the rightmost figure is a radar measuring the reflections of the sun from the earth. All icons courtesy of Icons8 (n.d.).

The radar equation

Lusch (1999) gives the radar equation as

$$P_R = P_T(\sigma^0 A)\left(\frac{G^2\lambda^2}{(4\pi)^3 R^4}\right) \tag{2.3}$$

where P_R is the power returned, P_T is the power transmitted by the radar system, σ^0 is the radar scattering coefficient, A is the area of the resolution cell of the radar system, G is the gain of the antenna, λ is the wavelength of the transmitted signal and R is the distance from the antenna to the target. From this, it can be inferred that there is three factors that influence the strength of the backscattered received energy (Kerle et al. (2004)):

1. The inherent radar properties, λ, G, A and P_T
2. The imaging geometry, which has an effect on σ^0 and R.
3. The object being imaged, and it's orientation

Radar geometry

This section will present some radar specific geometry and give name to relevant angles and directions.

Azimuth is the direction parallel to the radar flight path. (ESA (2014)) See figure 2.3. Here one can see the radar indicated, the flight path and the following azimuth. In satellite images the azimuth is the the along-track direction of the image.

The incidence angle is the angle between the the radar beam from the radar and a normal to the intercepting surface (ESA (2014)). This is illustrated in figure 2.4. The incidence angle has an effect on the resulting image: a larger incidence angle normally gives a weaker reflection.

The same figure (2.4) also shows slant range, which is the distance to a particular object of interest from the radar antenna, ground range which is the distance along the ground to the object of interest from the nadir, which is the single point directly below the radar on the surface of the earth (ESA (2014)).

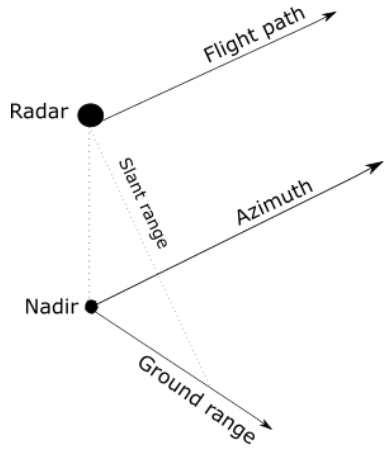


Figure 2.3: Figure showing the azimuth, including ground range, nadir, slant range and flight path

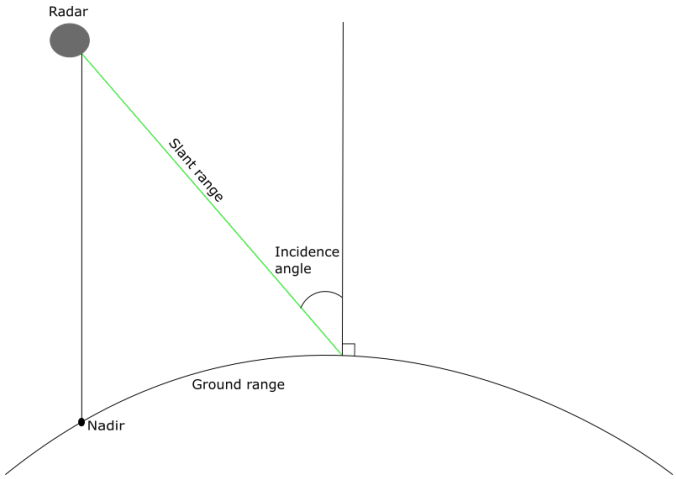


Figure 2.4: Figure illustrating the incidence angle along with slant range, ground range and nadir.

Polarization

Radars often are active remote sensing devices, which means that they send electromagnetic energy and wait for a reflection. Since they have an active sensor, the user has to decide the orientation the transmitted energy. One also have to choose which backscatter orientation should be received. The orientation can have a big effect on the resulting image (Kerle et al. (2004)). Any angle can in principle be used, but in reality only vertical and

horizontal is used (Lusch (1999)). This gives four possibilities:

- *HH*: Send and receive horizontally
- *HV*: Send horizontally and receive vertically
- *VH*: Send vertically and receive horizontally.
- *VV*: Send and receive vertically

The concept of polarization is illustrated in figure 2.5, where the blue wave has a horizontal polarization and the red wave has a vertical polarization.

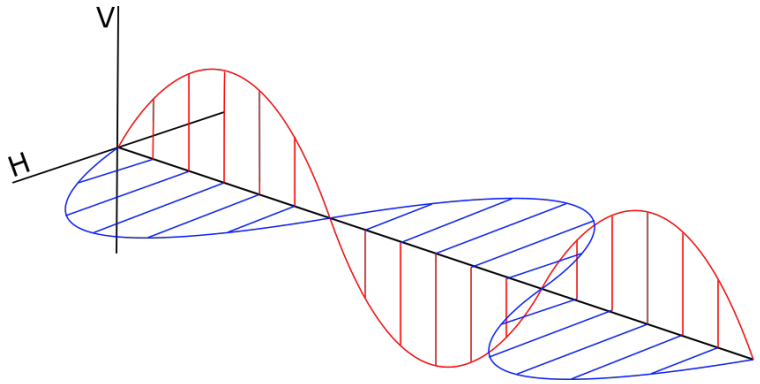


Figure 2.5: A figure illustrating the concept of polarization, with the horizontal and the vertical axis marked as H and V.

Imaging mechanisms

van Zyl (2011) presents two different mechanisms by which radars produce images: real aperture radar (RAR) and synthetic aperture radar (SAR). To understand the difference, one needs to understand how the resolution of a radar is achieved. The resolution of a radar is defined by the smallest separation between features which can be distinguished. If two targets are separated by a distance of x_r in the slant range direction, the difference in timing of the backscatter is given by

$$\Delta t = 2\frac{x_r}{c} \tag{2.4}$$

where c is the speed of light. The radar waves are normally transmitted in pulses with effective pulse time length of τ. Two targets can be distinguished from each other if the reflection of the first object is fully received before the reflection of the second object has arrived, see figure 2.6 for an example. This gives the range resolution of a radar

$$x_r = \frac{c\tau}{2} \tag{2.5}$$

Therefore, one can increase the resolution of a radar by doing shorter bursts of energy. However, this comes at the expense of making the signal-to-noise ratio worse. This is

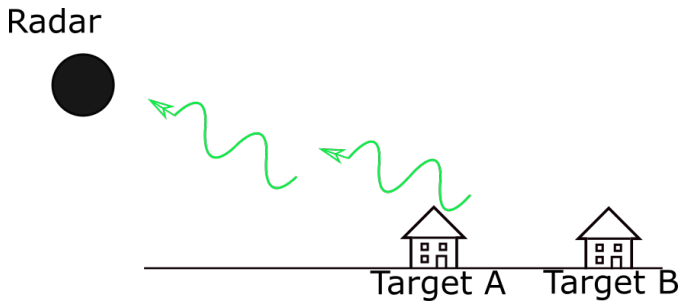


Figure 2.6: An example of how the range resolution works. The figure illustrates target A and B and the radar. There is two different reflected electromagnetic waves, one from each of the targets. Since the tail of the first reflection is ahead of the head of the second reflection, it is possible to recognize two distinct scatters.

equal for both SAR and RAR imaging. Where they differ is in the way they achieve the azimuth resolution.

According to van Zyl (2011) the resolution of a real aperture radar relies on the width of the antenna beam. It is determined by the size of the antenna and the distance to the target. The antenna beam width is given by the equation

$$\theta_a \approx \frac{\lambda}{L} \tag{2.6}$$

where θ_a is the beam width, λ is the wavelength and L is the length of the antenna. The resulting illumination in the azimuth direction from the antenna beam becomes

$$x_a \approx R\theta_a \approx \frac{\lambda R}{L} \approx \frac{\lambda h}{L \cos\theta} \tag{2.7}$$

where x_a is the illumination distance in azimuth direction, h is the distance from the radar to the nadir, and θ is the look angle. This means that you either need to have a short distance between the radar and the target, a very small wavelength or a massive antenna to get fine resolution. Either of these solutions have big drawbacks. Therefore, one normally exploits the concept of synthetic aperture.

Synthetic aperture radars increases the aperture synthetically. This is achieved by utilising the azimuth motion of the radar to use several backscatters of the same object to simulate a longer antenna (Kerle et al. (2004)). van Zyl (2011) describes how two targets at two different azimuth positions will be at different angles compared to the aircraft velocity vector, giving different Doppler frequencies. This can be used to separate targets in the azimuth direction.

Sentinel-1

The dataset assembled by C-CORE comes from the Sentinel-1 satellites. Sentinel-1 is a constillation of two sattelites, Sentinel-1A and Sentinel-1B, launched in 2014 and 2016 (Pelich et al. (2015)). Sentinel-1 is a part of Copernicus, previously known as GMES

(Global Monitoring for Environment and Security). They are designed to monitor the entire world's land masses, including the sea-ice and icebergs. Each radar has a 12 day repeat cycle, with a 180° phase difference, which means they can provide images of the same objects every six days (Pelich et al. (2015)). This makes them ideal for continuous monitoring of land masses. The Sentinel-1 carries a single C-band synthetic aperture radar (SAR), which can operate in four acqustition modes: Stripmap (SM), Interferometric Wide swath (IW), Extra-Wide swath (EW) and Wave mode (WV) (ESA (n.d.a)). While EW is the mode intended for marine use, IW was used to collect data for our set. EW has a large swath width of 400km, but IW has higher resolution at 5 by 20 meters compared to 20 by 40 (ESA (n.d.a)).

2.3 State-of-the-art iceberg-ship classification

Separating icebergs and ships in satellite images is not a new problem, it has been exhaustively researched. This section will present the state-of-the-art techniques used for this problem today.

2.3.1 Non-machine learning

In this section some of the non-machine learning approaches for discriminating between icebergs and ships will be presented. Most of the research presented here comes from C-CORE. C-CORE is a research-based advisory services and technology solutions company heavily specialised in remote sensing, and also the maker of the dataset used in the Kaggle competition.

Howell et al. (2004) suggests two different methods for discriminating between ships and icebergs: the HH/HV area ratio and the HV signal-to-clutter ratio. They collected SAR images of 20 icebergs and 19 ships, and used the CFAR (Constant False Alarm Rate (Bunch and D.Fierro (1992))) algorithm to detect the targets for both the HH and HV channel. For the HV channel, only 4 of the 19 icebergs were detected, while 18 of the 19 ships were detected. It was also noted that the 4 icebergs that was detected in the HV channel, all had a greater response in the HH channel. Based on these findings they investigated both HH/HV area ratio and HV SCR. The area ratio gave a 97% accuracy, while HV SCR had an accuracy of 92%. It is worth keeping in mind that this is with a very small dataset, and the authors noted that the majority of the ships were big supply ships and that more testing should be done using smaller vessels.

Howell et al. (2006) proposes to maximize the a posteori probabilities from Bayes' rule in order to classify targets. The maximum likelihood Gaussian classifier were used to model the probability of the target belonging to either class. SAR images for both the HH and HV channel is used. From the images an exhaustive set of target features is extracted: HH and HV SCR, HV/HH area ratio, HH mean amplitude, major axis, and many more. In total 28 different features. Then three different feature selection algorithms are used to optimize the algorithm: sequential forward selection, genetic algorithm and a limited exhaustive search. They all achieve very close to the same result, with a classification accuracy of $93\pm0.5\%$. Exhaustive search is the best at 93.5%, but at a larger computational

cost. Compared to Howell et al. (2004) this test is done on a much larger dataset with ships of all sizes, and the results might therefore be more comparable to real world performance.

English et al. (2013) presents a way to use the Space-based Automatic Identification System (S-AIS) for collecting ground truth to improve existing discrimination algorithms, and also to use the S-AIS data for iceberg surveillance. As seen in Howell et al. (2004) and Howell et al. (2006) there is developed highly accurate algorithms for discriminating between ships and icebergs, but they are never 100% accurate. Therefore English et al. (2013) proposes to use S-AIS to assist on confusing targets. They present two main hurdles to overcome for S-AIS to be used for this purpose. Firstly, in high ship density areas AIS messages might collide and some signals might get dropped. Still, this is not considered a big limitation, since regions that are frequented by icebergs have small ship densities. Another problem is synchronisation of S-AIS and SAR data, since they come from different satellites. A dead reckoning approach is proposed and tested, where one assumes that the ship will continue to hold it's current heading and speed since the last S-AIS update. This works to a satisfying degree, but is predicted to get much better as the RADARSAT Constellation Mission gains traction, as they contain both SAR imaging and S-AIS on the same satellite.

2.3.2 Machine learning

This section will present some of the machine learning approaches to discriminating between icebergs and ships in the literature today. There has been done a lot of research in this field in the last few years, and this section will present some of the approaches the author finds most promising.

According to Power et al. (2013), C-Core has since 2004 investigated a series of well known classification methods for the task of discriminating ships and icebergs. This includes linear discriminant, quadratic discriminant (QD), neural networks, k-nearest neighbour and support vector machines. They trade blows, mostly dependent on the dataset at hand, but QD has been shown to outperform the others for normally distributed datasets. QD was first explored by Howell et al. (2008) along with sequential forward selection for feature selection. QD uses a training set of known targets to build a quadratic discriminant, which is then used on new samples to discriminate based on distance to discriminant. The paper concludes with a discrimination result of 95% using the HH/HV dataset. This is also recommended further as the channels to use for iceberg/ship classification, due to the increased iceberg backscatter in the HH channel, and big difference in iceberg/ship backscatter in the HV channel.

Convolutional neural networks have already been tested for classifying icebergs and ships by Bentes et al. (2016), with very promising results. A very small CNN with only two layers was used to classify 128×128 pixel images as either icebergs or ships. It did so with a accuracy of 98%. This paper does not specify what polarization was used for the SAR-images, as the TerraSAR-X StripMap can be in both single and dual polarization mode. The paper uses a very limited dataset of only 345 unique targets, does not use a validation set for validating the performance of the CNN and uses a very small test set for final testing. In addition to this, the dataset is expanded using reflections and rotations so it contains a total of 600 samples before it is split into 90% training and 10% testing. This will give a artificially high test accuracy, since there will be a severe data-leakage between

the train-set and the test-set. This is something that could be improved, in order to get a result that is a bit closer to what can be expected in real life.

Jun et al. (2017) published a report detailing their work on the Kaggle dataset from the competition. They used three different machine learning techniques and compared them. This included a SVM on the histogram of gradients, a simple four layer CNN and the more complex CNN architecture, ResNet50. The dataset is not presented explicitly, but is of the exact same size as the Kaggle dataset, and the competition is mentioned in the paper. It therefore seems safe to assume it's the same. Some data preprocessing is tested, using Lee-filtering to remove noise and doing image augmentation. When compred to Bentes et al. (2016), this report avoids doing some of the same mistakes. The dataset is split before doing augmentation. It is tested with a larger neural network. A larger dataset is used, and a larger portion of the data is saved for testing. The SVM is shown to overfit heavily, has the worst result of them all at 70%, but seems to improve from the Lee filtering. The simple CNN as a accuracy of 88% with no statistically significant difference with or without filtering. ResNet has the highest accuracy at 96%, but does worse when the filter is applied, maybe due to the filter removing features. For some reason, not explained by Jun et al. the image augmentation is not applied to the dataset used for training the small CNN, only for ResNet. This could have improved the performance of the small CNN.

In addition to this research related to classifying SAR image targets, there has also been done a lot of research related to target detection using machine learning (Hwang et al. (2017), Cozzolino et al. (2017) and Wang et al. (2017)). Target detection is the first step in the pipeline from raw image to classified targets. So even though target detection is not strictly relevant to the problem presented in this project, it shows the usefulness of machine learning in the field, and it might be pointing to a future with a single machine learning algorithm from raw SAR image to fully classified targets.

Chapter 3

Dataset problem mitigation techniques

This section will present some of the dataset problem mitigation techniques suggested by the Kagglers on the Kaggle forum. Section 3.1 to 3.6 will present some of the technique suggested, how it would affect the dataset and why it helps mitigate our problems.

3.1 Image augmentation

A typical problem when trying to use neural networks for any purpose is the data-set being of inadequate size. You always want more data, preferrably more quality data, and you are never satisfied. The more data you have, the better your algorithm will generalise to new, unseen data. Creating the data-set is normally very labour intensive. Manually expanding the data-set with more data is in the best case, expensive or, in the worst case, infeasible. This is the reason why image augmentation is a natural choice for this problem. Collecting and labelling large amounts of SAR images is very costly, and expanding the dataset using image augmentation at no cost is therefore a very attractive option.

Image augmentation has roots back in 1996 (Yaeger et al. (1996)). Here two Apple engineers, Yaeger and Lyon, alongside Webb presented a state-of-the-art ANN for classifying handwritten characters on the Apple Computer's Newton MessagePad. One of the main innovations was what they called stroke warping. Stroke warping produced small changes in skew, rotation, and x and y linear and quadratic scalings to the training data.

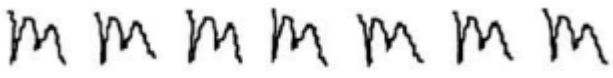


Figure 3.1: The figure shows the first original M followed by six M's stroke warped different ways. Courtesy of Yaeger et al. (1996)

The act of augmenting images have been picked up by multiple entities since 1996. There have been proposed many different augmenting techniques, all with the single goal to increase the machine learning algorithm performance. Researchers at Microsoft were the first to use elastic distortions on the MNIST dataset. Elastic distortions are made to simulate the uncontrolled oscillations of the hand muscles, dampened by inertia. This technique along with the usage of a early version of a CNN's, which was not all that normal at the time, gave the highest performance on the MNIST data-set at that time. (Simard et al. (2003))

Other techniques has also been proposed throughout the years, including but not limited to:

- Extracting smaller patches from the original image (zooming) (Krizhevsky et al. (2012))
- Flipping the images horizontally (Krizhevsky et al. (2012))
- Adding random changes to contrast, brightness or colour (Howard (2013))
- Altering the scale of the images, using both lager and smaller images than the original (Howard (2013), Wu et al. (2015))
- Vignetting, making the outer parts darker than the centre of the image (Wu et al. (2015))
- Lens distortion, for example barrel or pincushion distortion (Wu et al. (2015))

It is important to adjust the augmentation techniques used, based on the dataset that needs expanding. Elastic distortion works very well for hand drawn digits, since it imitates hand motions. It would not perform well on the problem of classifying icebergs and ships, since hand motions have no relation to SAR-images. However, other augmenting techniques are often used by the participants in the Kaggle competition. In the next section, the most frequently used will be presented. Later, in section 4 of the paper, there will be done an attempt at empirically finding the effect of some of these techniques.

3.1.1 Flipping and flopping

This entails flipping around an axis, either x, y or both at the same time. It given by the following equation:

$$B_{i,j} = \begin{cases} A_{rows-i-1,j} & \text{if flip around x} \\ A_{i,cols-j-1} & \text{if flip around y} \\ A_{rows-i-1,cols-j-1} & \text{if flip around x and y} \end{cases} \tag{3.1}$$

where $cols$ and $rows$ are the number of columns and rows in the image, A is a matrix representing the original image and B is the flipped image. This makes a new picture which can be saved and appended to the training set. The result of this kind of flip can be seen in figure 3.2.

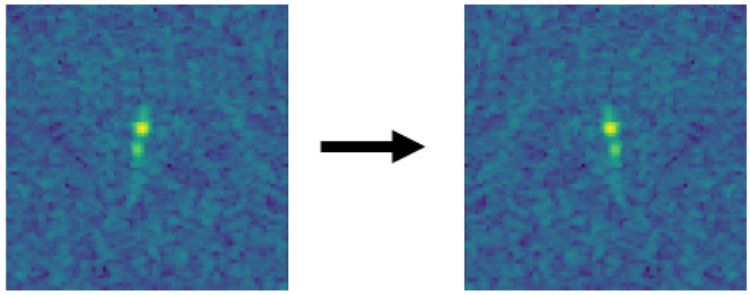


Figure 3.2: A sample image from the data-set before and after flipping verticially.

In theory, images of icebergs or ships should be invariant to flips both along the horizontal and vertical axis. Whether or not it makes the classifier work better highly depends on the type of classifier that is used. There were various reports from the Kagglers, some reported better performance overall with both flipping and flopping. Some reported no effect by flopping but better score when flipping. Some reported better score flipping images for the VGG16 network, but not for other network architectures. Later experimentation will have to show.

3.1.2 Height and width shifts

Height and width shifts are to translate the image either along the x-axis or along the y-axis. This has been used in several of the submitted solutions. An example of a height and width shift can be seen in figure 3.3.

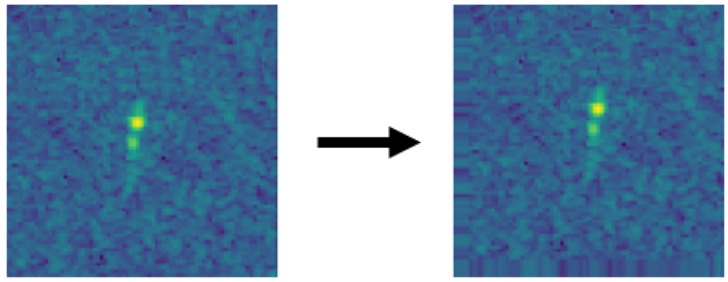


Figure 3.3: A sample image from the data-set after shifting 8 pixels along the x-axis and 8 pixels along the y-axis.

After the shifting of the image, it still contains a object that could be either an iceberg or a ship. The only difference is that it is no longer placed in the middle. This can give a more robust classifier.

A interesting point is what one should do about pixels outside the boundary of the images. When the picture is translated along the x-axis towards the right, the left part of the image becomes undefined. This part of the image needs to be filled in with something.

There is different strategies for this. Most of the contestants have chosen to fill in the pixels with the whatever the nearest neighbour contains. This is fine for this dataset, since almost everything except the centre object is considered noise, and will most likely not help classifying the object. But, strictly speaking, this is not known for a fact. The water reflections might contain different reflections based on whether the target is a ship or an iceberg. If the content in the periphery of the image is very important for the classifier, one would have to tackle this problem in a bit more sophisticated way. Examples of this could be to mirror the edge on the left of the picture. Another approach is to make the image ”wrap” around, making whatever is moved out of the boundary on the right reappear on the left. One might also have to reevaluate if the images should be translated at all if the edges are that important.

3.1.3 Zooming

Zooming is also a technique often used. This technique exploits the fact that it should not really matter if the picture is taken up close, or from far away. A zoom example is shown in figure 3.4.

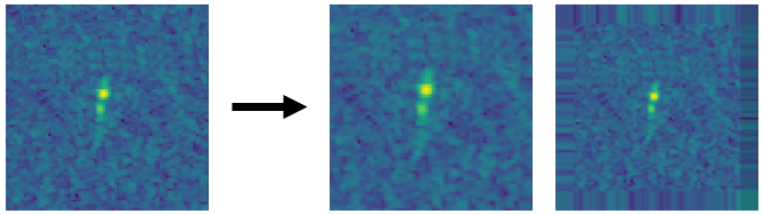


Figure 3.4: A sample image from the data-set and a version zoomed in 20% and a version zoomed out 20%.

Again, the problem of what to do with the periphery arises, when the image is zoomed out. One can employ the same techniques as used on the image translation problem, but since all data is kept inside the picture, you won't have the problem of loosing information. When zooming in you will loose some data.

3.1.4 Rotation

Rotation exploits the invariance the images has to small rotations. Rotation is shown in figure 3.5.

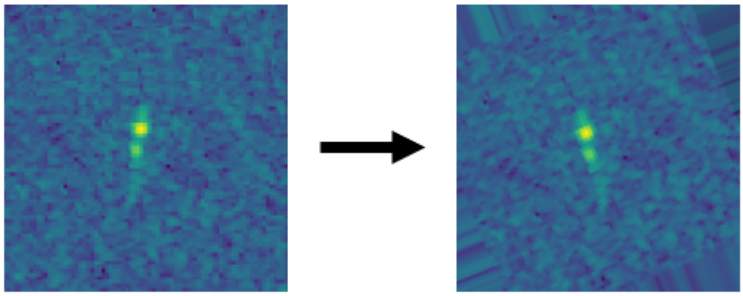


Figure 3.5: A sample image from the data-set and a version rotated a random degree between 0 and 30 counterclockwise.

Once again you have the problem of filling in the new regions of the picture. You also loose a bit for information from the edges, and you have to figure out how many degrees the image can rotate to still give a benefit to the training of the classifier. Too much rotation, and you might end up with something that does not resemble a real-life SAR-image of a iceberg or ship.

3.1.5 Contrast, color and brightness

It were proposed by some kagglers to adjust the contrast, color and brightness of the images. This were suggested to not work as well due to the properties of the images in the dataset. Pixel contrast and pixel brightness is the main properties of our images and an adjustment of them might skew the data too much. Adjusting the color is not a possibility since the images consists of reflection intensities and not color values. This is proprietary to our dataset, and these techniques can work well in other situations with other types of data.

3.2 Pseudo labelling

Pseudo labelling is a way of training a neural network in a semi-supervised fashion. It works well in situations where you have a large amount of unlabelled data and labelling is cost intensive. In a nutshell, pseudo labelling consists of two training stages. First you train you network on the labelled data you have access to. Then this trained network is used to classify all the unlabelled images. The network output can be interpreted as how confident the network is that the input is or isn't of that category. The pictures the network is the most confident about is added to the training set, and are given the labels the network has predicted (the pseudo labels). Then the network is retrained using the new, expanded dataset.

According to Lee (2013), pseudo labelling has the same effect as entropy regularization. Pseudo labelling and entropy regularization works by favouring low-density separation between classes, which is a common assumption for using semi-supervised learning. This means that the decision boundary should lie in low-density regions, and pseudo la-

belling will push the boundary towards these regions to increase generalization of the network.

Lee (2013) gives the entropy of the unlabelled data as

$$H(y|x') = -\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} P(y_i^m = 1|x^{'m}) \log P(y_i^m = 1|x^{'m}) \tag{3.2}$$

where n' is the number of unlabelled samples, C is the number of classes, y_i^m is the unknown label of the mth sample, and $x^{'m}$ is the mth sample input. Conditional log-likelihood is plotted in figure 3.6. This shows that in order to minimize H, $P(y_i^m|x^{'m})$ should either be close to 0 or to 1. Pseudo labelling gives the samples with the lowest uncertainty, meaning $P(y_i^m|x^{'m})$ is either as low or as high as possible. This finally means: pseudo labelling minimizes the entropy.

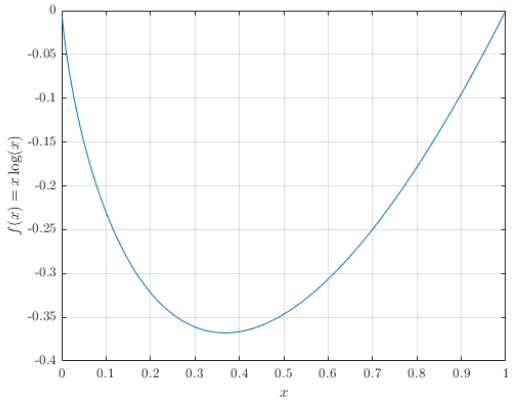


Figure 3.6: A plot of the function $f(x) = x\log(x)$

The MAP (maximum a posteriori) estimate is given by (Grandvalet and Bengio (2005))

$$C(\theta, \lambda) = L(\theta) - \lambda H(y|x') \tag{3.3}$$

where $L = \log P(y^m|x^m;\theta)$ is the conditional log-likelihood, H is the entropy as defined in (3.2) and θ is some parameter which parametrizes $P(y^m|x^m;\theta)$. L is only dependent on the already labelled samples, while H is given by the unlabelled samples. By maximizing the MAP estimate, we can make sure that the model generalizes well. This can be done either by maximizing the conditional log-likelihood or minimizing the entropy of the unlabelled data. Given that pseudo labelling minimizes the entropy, it should also make sure that the model trained with the unlabelled data generalizes as well as possible.

3.3 Test-time augmentation

Test-time augmentation (TTA) is closely linked to image augmentation, but is not as well known by the machine learning community. It follows mostly the same tactic, using image

augmentation to expand the dataset. But this time it's not the training set that is expanded, it's the test-set. Combinations of predictions for multiple transformed versions of a single test image is used to help improve performance. To understand how this could help it is necessary to look at an image acquisition model.

According to Wang et al. (2018) the image acquisition model can be formulated as

$$X = \mathcal{T}_\beta(X_0) + e \tag{3.4}$$

where X_0 is the true representation of the image, $\mathcal{T}_\beta$ is a transformation operator parametrized by β and e is some noise that's added to the image. This all results in a transformed image X. Let's assume that all the transformations that are applied to X_0 is reversible. This gives

$$X_0 = \mathcal{T}_\beta^{-1}(X - e). \tag{3.5}$$

Further, Wang et al. (2018) explains how deep learning connects to this. If $f(\cdot)$ represents the neural network, parametrized by θ, Y should be inferred from X by

$$Y = f(X; \theta). \tag{3.6}$$

X is one of many possible transformed versions of the underlying image X_0. Using X directly for inference can lead to a biased result. Therefore X_0 is used for inference

$$Y = f(X_0; \theta) = f\left(\mathcal{T}_\beta^{-1}(X - e); \theta\right) \tag{3.7}$$

Instead of finding a prediction for X, the distribution of Y is computed

$$P(Y) = P\left(f\left(\mathcal{T}_\beta^{-1}(X - e); \theta\right)\right), \text{where } \beta \sim P_\beta, e \sim P_e \tag{3.8}$$

To obtain a final prediction, we compute the expectation of Y.

$$E(Y) = \int yP(y)dy = \int_{\beta \sim P_\beta, e \sim P_e} f\left(\mathcal{T}_\beta^{-1}(X - e); \theta\right) P(\beta)P(e)d\beta de \tag{3.9}$$

This is very computationally expensive, so instead $E(Y)$ is estimated using Monte Carlo simulations

$$E(Y) \approx \frac{1}{N}\sum_{n=1}^{N} y_n = \frac{1}{n}\sum_{n=1}^{N} f\left(\mathcal{T}_{\beta_n}^{-1}(X - e_n); \theta\right), \text{where } \beta_n \sim P_\beta, e_n \sim P_e \tag{3.10}$$

where N is the total number of simulations. As can be seen by inspecting the equation, this is equal to TTA. We acquire an image, sample β_n and e_n, transform our acquired image and infer on that. As $N \to \infty$ this should approach the true label Y, given that $f(\cdot)$ a true mapping from $X_0 \Rightarrow Y$.

3.4 Lee filtering

Lee filtering was proposed on Kaggle multiple times, but probably most notable in Jun et al. (2017). Lee filtering is a despeckling filter, well suited for SAR images, as they often are contaminated by noise. Lee filtering was first proposed by Lee (1980). He suggested that the speckle acts as a multiplicative noise, and proposed a filter that minimized the mean squared error (Mahdavi et al. (2017)). Therefore it is also known as the MMSE filter.

Mahdavi et al. (2017) presents the Lee filter as having the following form.

$$\hat{R}(x,y) = W(x,y) \cdot I(x,y) + (1 - W(x,y)) \cdot \bar{I}(x,y) \tag{3.11}$$

where $\hat{R}(x,y)$ is the estimated intensity of pixel (x,y), $I(x,y)$ is the observed intensity of (x,y) and $\bar{I}(x,y)$ is the average intensity of all pixels in a local window around (x,y). $W(x,y)$ is a weighting parameter, weighting the observed intensity vs the average intensity, depending on how heterogeneous or homogeneous the region are. In homogeneous regions W is small, and therefore the estimated pixel intensity will be heavily affected by the average intensity of the local window, while for heterogeneous regions it will be the opposite.

Jun et al. (2017) showed that Lee filtering worsened the result for all CNN solutions. A possible explanation proposed by the author of this project is that the SAR-images that were collected already were processed to reduce speckle. The Sentinel-1 images that were collected were level 1 Ground Range Detected images, as described in ESA (n.d.b). These images are multi-looked, which is a data collection technique where one use multiple images of the same target to average out speckle (Franceschetti and Lanari (2016)).

3.5 Discrete Fourier Transform

Some users reported an increased performance by using the discrete Fourier transform (DFT) on the images to create two additional bands for a total of 4 channels. The Fourier transform gives the image in the frequency domain, while the input is in the spatial domain. The two-dimensional DFT for a square image of size $N \times N$ is given by (Gonzalez and Woods (2006))

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi(\frac{ki}{N} + \frac{lj}{N})} \tag{3.12}$$

where $f(i,j)$ is a pixel in the original image, while $F(k,l)$ is the fourier transformed image.

Even tough it was reported to give a good response, it was decided not to pursue any further in this report. This is due to that the result was only reported to improve when the model trained using the DFT was ensambled with other models. The study of ensambling techniques is outside the scope of this project, therefore DFT in connection with ensambling it is left as future work.

3.6 Autoencoders

Goodfellow et al. (2016) presents a description of the autoencoder. They say it is a neural network, trained to copy it's input. Internally there is a hidden layer, generating a code of the input. One can say that the network consists of two parts: an encoder $h = f(x)$ and a decoder $r = g(h)$. This is illustrated in figure 3.7.

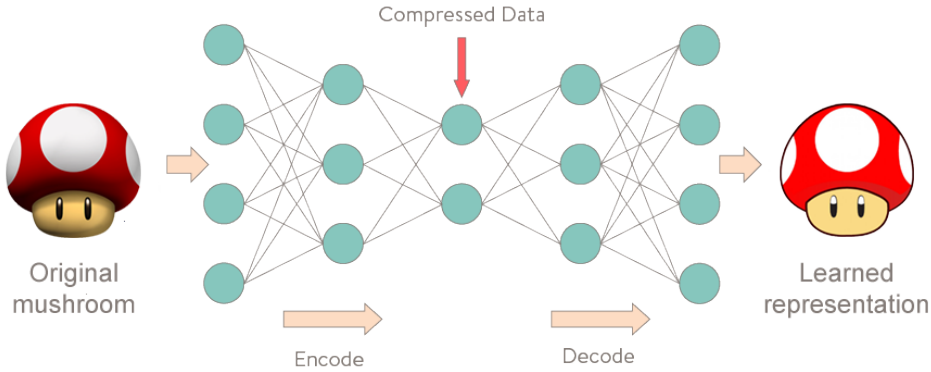


Figure 3.7: An illustration presenting the concept of autoencoders, encoder and decoder. Figure is courtesy of Valkov (2017).

The learning process is described by minimizing the loss function $L(x, g(f(x)))$. Here L is a loss function, penalizing $g(f(x))$ for being dissimilar to x. The most used autoencoder is called the undercomplete autoencoder. Here the hidden layer h is smaller than the input and output, and the network is forced to extract the most salient features from the dataset.

Erhan et al. (2010) experimented with pre-training using denoising autoencoders, which is an autoencoder trained at denoising a corrupted version of its input. They showed that using this autoencoder as a starting point, adds robustness to deep architectures. Comparing networks seeded with an autoencoder, to a network not seeded using an autoencoder, gave vastly different endpoint local minimas. It also seemed to have a regularizing effect, but different to and performing better than L_1/L_2 regularization. In addition to this, even though the pre-trained networks had a higher training loss for large enough layers, they seemed to generalize better than their non-pre-trained counterparts. The pre-trained effect is present also for very large datasets, indicating that the starting point is very important.

3.6.1 Convolutional autoencoders

There was suggested on the Kaggle forums to also experiment with convolutional autoencoders. They were first presented by Masci et al. (2011). Convolutional autoencoders are similar to autoencoders, the difference lies in that the network uses convolutional layers instead of densely connected layers. This makes the network discover localized features that repeat themselves in the input. The reconstruction is then a linear combination of basic image patches.

Both autoencoders and convolutional autoencoders are something that should be explored, but will not be in this report, due to time constraints. It would also require a different testing methodology, as the weights no longer could have been initialized as the ImageNet weights. The reason for using ImageNet weights is discussed later in section 4.1.

Chapter 4

Own testing

This chapter will contain tests done of the techniques presented in chapter 3 on the C-Core/Equinor dataset. Section 4.1 will introduce the method used for testing. Section 4.2 to 4.4 contains tests of a selection of the techniques presented in chapter 3. They will be compared to a baseline result and how they perform on other datasets.

4.1 Method

In this section the methodology for testing the different techniques will be presented. To be able to compare the different techniques, a standard testing pipeline will be developed. To achieve reproducibility a set of free and publicly available architectures and tools will be used.

The deep learning framework of choice will be Keras (Chollet et al. (2015)), due to the fact that is was used by a majority of the contestants in the Kaggle competition and is the framework the author has the most experience with. The network architecture will be based on VGG16, which was first presented in Simonyan and Zisserman (2014) and is illustrated in figure 4.1. VGG16 is a very complex network, and a simpler network might give higher classifier performance, but it was extensively used by the top contestants and is well known in the deep learning image recognition community. Since design of the classifier is outside the scope of this project, it was chosen as an appropriate network architecture. The weights are chosen to be the pre-trained weights from the ImageNet dataset (Deng et al. (2009)), as given by Chollet et al. (2015). The top layers consists of one 256 neuron dense layer with ReLU activation, followed by a Dropout layer with a dropout-rate of 0.5 and finished with a single output neuron with sigmoid activation. The top layers are kept trainable, so the network can be fine tuned using the C-CORE/Equinor dataset.

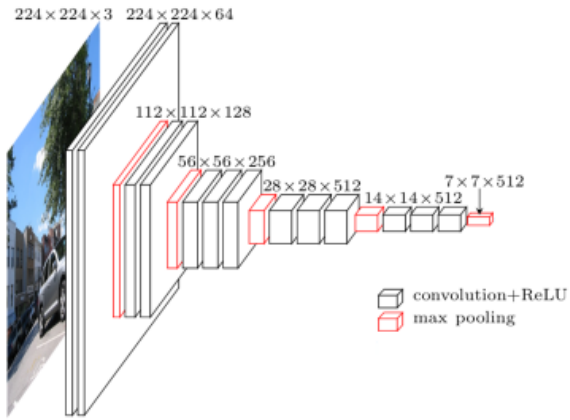


Figure 4.1: A figure illustrating the VGG16 architechture. Modified from Blier (2016)

During training the optimal number of epochs will be determined used 5-fold cross validation. This will be determined individually for each technique, due to the fact that modifications to the dataset might result in a dataset requiring more or less training time to converge. The original dataset will be split 70-30 between training- and test-set. The results will be evaluated using binary crossentropy, as it is defined by Chollet et al. (2015), on the test-set. This loss function was chosen since it was also used to make the leaderboard in the Kaggle competition.

4.2 Image augmentation

For image augmentation it was decided that the Keras image preprocessing function *ImageDataGenerator* should be used. It can be used in conjunction with the Keras *fit* function to generate batches of augmented images. The generator makes batches of a specified size where a random set of augmentations were applied to the images. What augmentations the generator can use are specified by the programmer. The generator also allows you to specify what should happen to pixels outside the boundary of the images, as talked about in section 3.1.2. To narrow down the amount of hyperparameters to experiment with, it was decided to use the mode *'reflect'*, but this is something that could have been experimented with.

Further, a decision was made to test only one augment at a time, for a series of augmentation ranges. From cross-validation it will be decided which augmentation range yields the best result, and at how many epochs of training. This would then be used to predict on the test-set to get a final measure of performance. Continuing off this, the techniques which were deemed to have a positive impact on the prediction accuracy, will be tested in conjunction to see if the performance gain will compound. Further, to be able to say something about which properties of the dataset makes the image augmentation techniques work as they do, the techniques will also be tested on another dataset, namely CIFAR-10 (Krizhevsky (2012)).

A choice had to be made with regards to what types of augments that should be tested.

Due to the specific properties of the dataset, it was decided to explore rotation, width and height shifts, zooming, flipping and flopping the images. The results those experiments is presented in the following section.

4.2.1 Results

Firstly, to be able to put the following results into context, a baseline result for the VGG16 architecture was made. This was done using the technique presented in 4.1, without applying any special technique to the dataset.

Continuing, rotation, as talked about in section 3.1.4, were tested for different rotation degrees. This included $5, 10, 15$ and 20 degrees. The cross validation gave the best results for rotation = $10°$.

Width and height shift, as described in section 3.1.2, were then tested. The images were shifted $5\%, 10\%, 15\%$ and 20% of the total picture width/height. Shifting the images 15% of the total size gave the best results.

Then zooming, as described in section 3.1.3, were tested. The images were zoomed both in and out $5\%, 10\%, 15\%$ and 20% of the total picture size. Zooming the images 15% of the total size gave the best results.

Finally, flipping and flopping the images were tested. It was tested for flipping, flopping and both flipping and flopping. Only flopping gave the best result.

The best working augmentations as determined by the cross validation were then trained a final time before begin tested on completely unseen test-data. The final test results compared to the baseline is presented in figure 4.2.

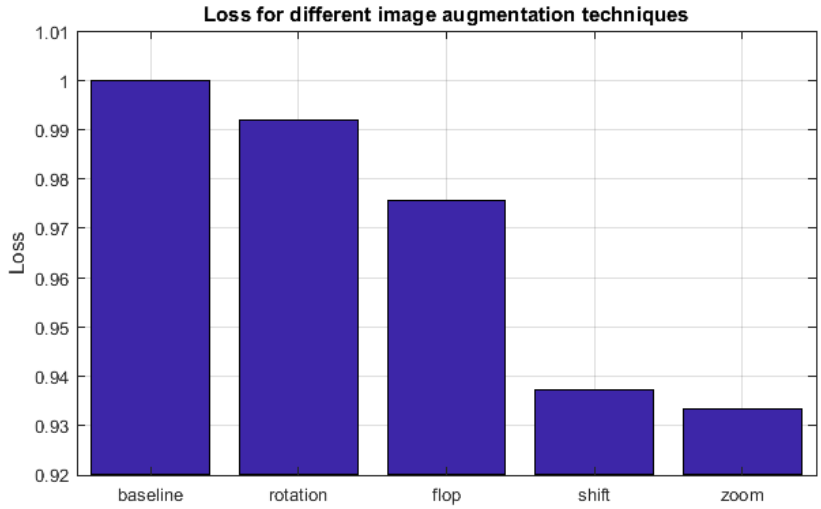


Figure 4.2: A graph illustrating the different classification losses between the four different augmenting techniques and the baseline. The baseline result is set to 1, and the other losses are scaled accordingly.

Continuing, the best augmentation parameters were then tried in combination with each other to see if the effect would compound. The results can be seen in figure 4.3. It was not experimented with rotation, due to the poor performance.

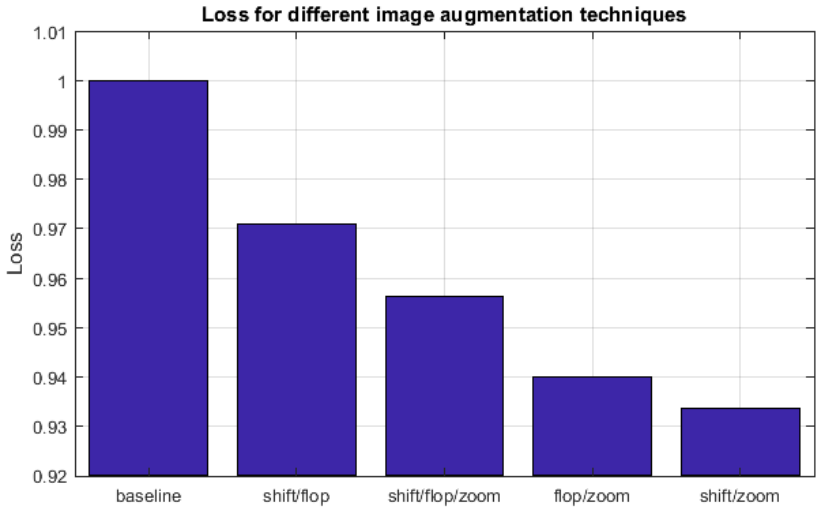


Figure 4.3: A graph illustrating the classification results when combining different augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly.

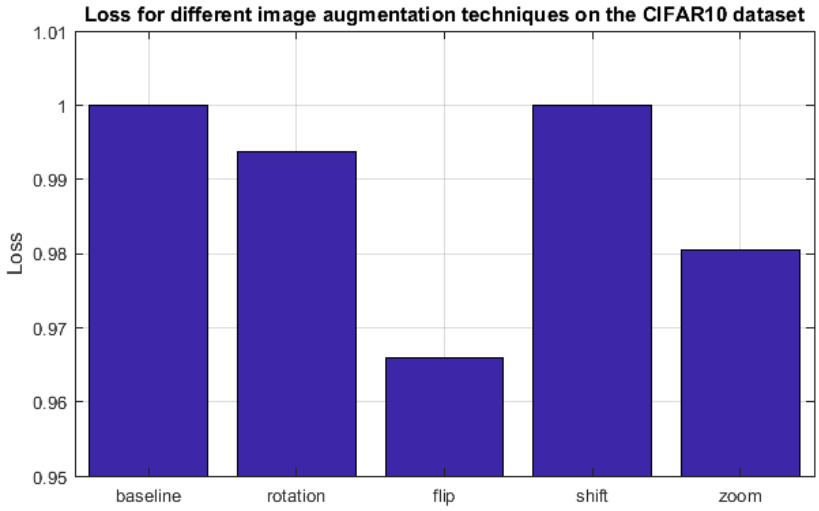


Figure 4.4: A graph illustrating the classification results on the CIFAR10 dataset for different augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly.

The exact same procedure was done on the CIFAR10 dataset. The hypothesis was that comparing the effect on both datasets will give further insights into why some augments work while others don't. Those results can be seen in figure 4.4. Due to time constraints and in order to keep the complexity of the comparison low, there were not done any experimenting with compounding augmentations on the CIFAR10 dataset.

Continuing, there was an attempt at clustering the different data using a clustering algorithm called DBSCAN, which can be read more about in Ester et al. (1996). The idea was to test augmentations on different clusters, seeing if it had any different effects. Unfortunately, DBSCAN only produced very small clusters. There was hard to see why the algorithm was putting certain images into the same cluster, making it a challenge to gain any useful insight by augmenting just one (or a few) of the many clusters. It was therefore decided to not pursue this further.

4.3 Pseudo labelling

For pseudo labelling the baseline model will be used as a starting point. This model will be used to classify each picture in the test-set. Then, all the pictures the model seems fairly certain about will be included in the training set, with the tag the baseline model predicted. This threshold will be empirically found, due to the fact that we want to make sure that we still have a fair number of sample in the test set for testing in the end. This is then trained again using cross validation to determine the optimal number of epochs.

This will be done for two different train/test split ratios: 70/30, as the image augmentation case, and 30/70. The 30/70 split is to if there is a big difference in the effect of pseudo labelling depending on how much training data you have, and how much unlabelled data you have. Pseudo labelling was heavily used in the Kaggle competition, but this might be due to the fact that they had to work on a 30/70 train/test split. Therefore it is done two tests in this report, so the result can be compared between the "normal" train/test split of 70/30, and the Kaggle-split of 30/70.

4.3.1 Results

The final testing results can be seen in figure 4.5 and figure 4.6.

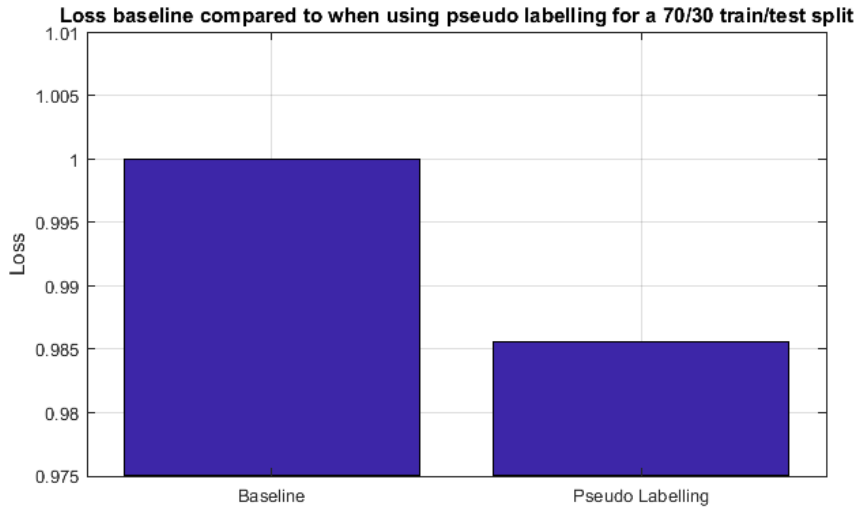


Figure 4.5: A graph illustrating the classification results when using pseudo labelling for a 70/30 test/train split . The baseline result is set to 1, and the other losses are scaled accordingly.

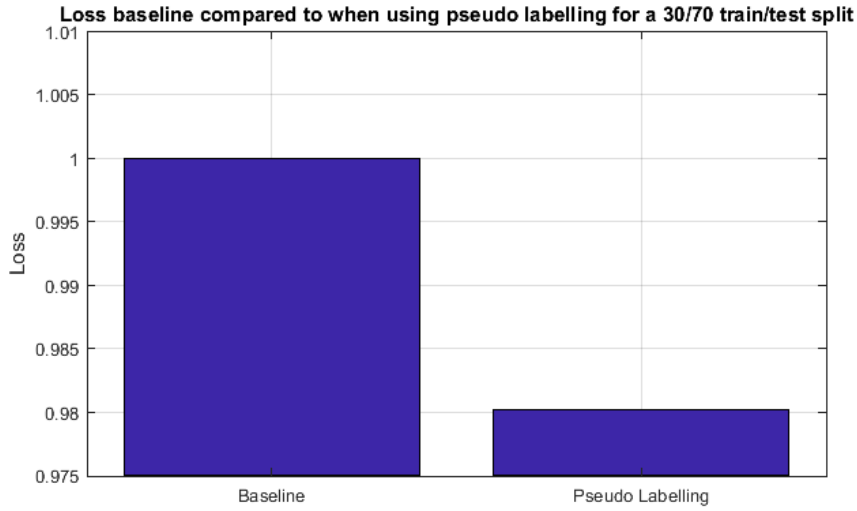


Figure 4.6: A graph illustrating the classification results when using pseudo labelling for a 30/70 test/train split . The baseline result is set to 1, and the other losses are scaled accordingly.

4.4 Test-time augmentation

The testing of test time augmentation will largely follow the same approach as the testing of image augmentation. This is due to the fact that it is basically image augmentation applied at a different place in the deep learning pipeline. What's important is to avoid the pitfall of testing the augmentation techniques at the test-set. This could result in overfitting the test-set, and we would get a too high end test result, not representative of the general performance. Therefore, it was decided to test the augmentation parameters at the validation data, once for each of the 5 folds. Since this approach requires a more careful matching of images and their augmented counterparts, the Keras *ImageDataGenerator* won't be used, and a self implemented solution will be used instead. Once again, the approach of testing one augment at a time, following by trying multiple of the best performing augmentations at the same time were also done this time around. Then the best performing combination were tested on the test-set.

4.4.1 Results

The results for the best single test-time augmentations can be seen in figure 4.7. The best results were for rotation $= 20°$, only horizontal flip, zooming in 20%, 5 pixels of width shift and 5 pixels of height shift.

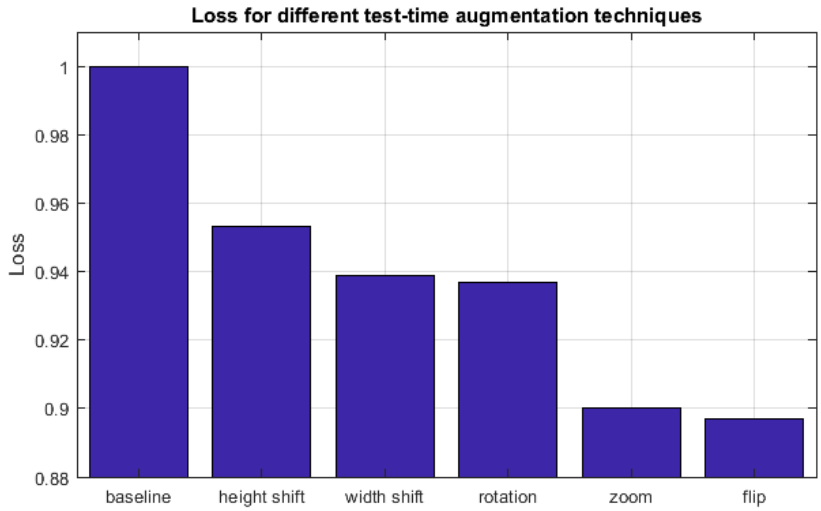


Figure 4.7: A graph illustrating the classification results when using different test-time augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly.

Continuing, some of the augments were tried in pairs of two, seeing if the performance gain would compound. Zooming, rotating and flipping were also tested in combination, as they were the three best performing augments. Those results can be seen in figure 4.8.

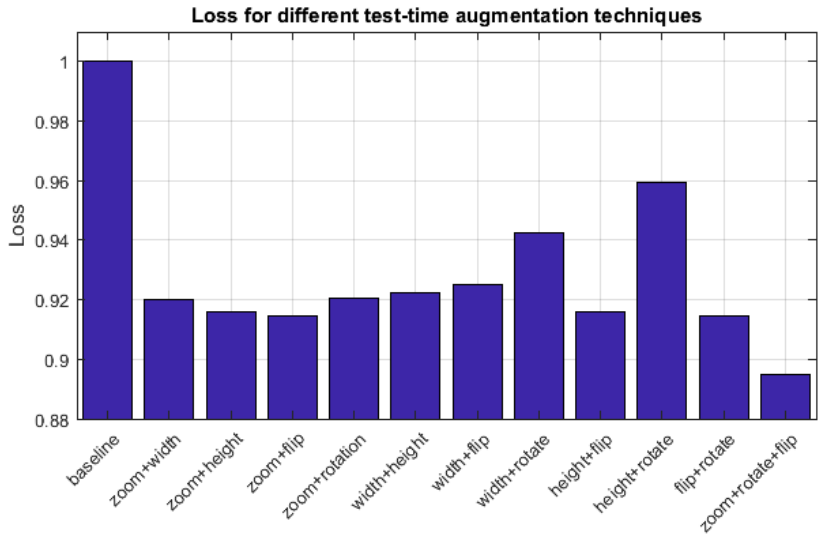


Figure 4.8: A graph illustrating the classification results when using different combinations of test-time augmentation techniques. The baseline result is set to 1, and the other losses are scaled accordingly.

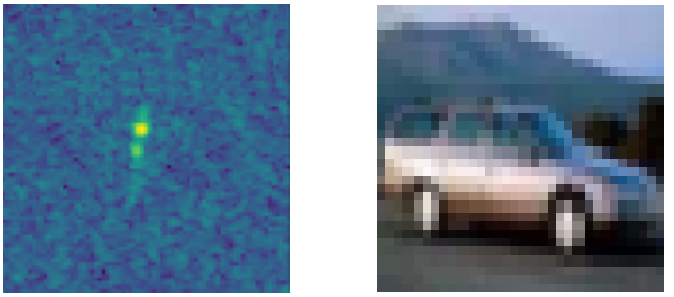
Chapter 5

Discussion

5.1 Image augmentation

Image augmentation gave promising results. Giving the best performance boost was zooming, which gave a classification boost of 7%. On the other hand rotating the pictures gave poor results, with about 1% better classification. Even though this is a gain in performance, one have to consider if this is worth the effort, given that it introduces a higher computational complexity and, to be fair, might not even be a statistically significant reduction.

To explain why certain augments worked and why others didn't, requires looking at other datasets as well. This will give a better basis when trying to spot trends. Therefore, what follows will be a comparison between the CIFAR10 dataset and the Equinor/C-CORE dataset with regards to which augmentations that gave good results.



(a) **(b)**

Figure 5.1: Two example figures from the Equinor/C-CORE dataset and the CIFAR10 dataset

Consider the two pictures in figure 5.1, which are random examples from both the Equinor/C-CORE dataset and the CIFAR10 dataset. For this section it is worth keeping in mind the difference of CIFAR10 images and the C-CORE/Equinor dataset. While the first one is a set of optical images, the last one is SAR satellite imagery. The biggest difference

between augmenting the CIFAR10 and Equinor/C-CORE dataset is in their response to shifting images. Looking at these two examples, it is easy to see why. The images in the CIFAR10 datset is much better located, containing almost zero border around the object of interest. Shifting the image will almost always result in loosing crucial information for classification. On the other hand, the images from Equinor/C-CORE can be shifted at least 40% of the picture before the target leaves the image. It is not given that this won't impact our classification performance, due to the fact that the reflected water around the object might contain useful information.

Continuing, they both have a good response to zooming. One might start wondering why CIFAR10 performs better with zooming, since it didn't perform well with shifting the images. This is probably due to the fact that the images are zoomed both in and, more importantly, out. Zooming out does not result in any information loss. One can clearly see why the Equinor/C-CORE images might benefit from zooming, both in and out. Zooming will result in a larger or smaller target, which in turn simulates a larger or smaller iceberg/ship.

Interestingly, they both responded differently to flipping and flopping. While the iceberg/ship images responded best to only flopping, CIFAR10 responded to flipping. This is highly surprising to the author, who believed that both would respond best to flopping. Flipping, for example the CIFAR10 image in figure 5.1, would result in a upside down car. Which, in the authors head, should not improve classification accuracy for upright cars. Iceberg/ship responded the best to flopping, which essentially means to mirror the image around the y-axis. Flipping would result in a upside down ship or iceberg, since the images are captured at a certain incline, and not straight down. Therefore, a flipped image does not represent something that might appear in the test-set.

Lastly, rotating didn't work for either of the datasets. This could be due to some of the same reasons as mentioned in the last paragraph. The images are taken at a incline, a rotation might distort them, resulting in something not representative of what's in the test-set.

Interestingly, when trying to combine multiple augmentation techniques, the results did not compound at all. The worst performing was when shifting was combined with flopping. The best results were achieved when combining zooming and shifting, but they still were worse than only zooming. One might question what's going on here? A hypothesis is that when adding multiple augmentations on top of each other, the performance suffer due to the combination of augments becomes too much. When a ship is flopped, or a ship is zoomed in, it still resembles a ship. When you both zoom and flop at the same time, it might loose some of it's ship resemblance. Another hypothesis is that the performance gained by flopping an image are closely related to the performance gained when for example shifting an image. That would mean that adding multiple augments won't have any big effects.

5.2 Pseudo labelling

Pseudo labelling did not give as good results as were reported in the Kaggle competition. The 30/70 train/test split achieved a slightly better performance gain compared to 70/30,. Still, the gain is small, and the results not very different. They both gave about a 2%

increase in performance.

The first mystery to solve is why there were no big difference between the two splitting techniques. The 30/70 split had a lot more data to do pseudo labelling on, and should therefore give a large increase in performance. On the other hand, the 70/30 split have more data to train on, and should therefore result in a model that's better at predicting on unseen data. This means that the 30/70 split does a worse job of classifying unseen images, and therefore the pseudo labels are more often wrong as well. These two effects, the larger amount of pseudo labelling data for the 30/70 split, and the better accuracy of the 70/30 split model, seems to offset each other, so the effect becomes almost the same.

Another mystery is why this were reported to give a big boost in performance for the Kagglers. First of, one can not exclude the possibility that some of what was said on the forums wasn't true. On the Kaggle platform, there is this concept of public and private leaderboards. On the public leaderboard the algorithms are only tested on a small part of the test set. On the other hand all results are available to everyone during the competition. The private leaderboard only becomes accessible to the contestants when the competition is finished, and here every algorithm is tested on the entire test-set. It could be that pseudo labelling worked fine for the public leaderboard, but when the private was published a lot of the models using heavy pseudo labelling fell down a lot of places without it being reported on the forums. There was quite a big upset when the private leaderboard was published, so it could be possible.

A second explanation could lay in the fact that many Kagglers had baseline models that performed much better than the baseline used in this project. The performance of the baseline were not the main focus of this project, so the baseline could have been much better. A better classification result for small amount of training data would lead to a more significant performance gain for the 30/70 split model.

A third explanation is in how the dataset were presented. For the Kagglers, the dataset were split 30/70 (C-CORE (2018)), and the contestants had no information about the 70 % of the data that were saved for testing. Unlike in this project, where we could choose to use the actual labels instead. For the Kagglers it is clearly beneficial to at least get the 2% increase in classification performance, compared to nothing at all.

5.3 Test-time augmentation

Test-time augmentation gave good results. The best effect was by flipping the pictures, which gave an 10% decrease in loss, zooming gave a decrease of 10%, rotation and width shift 6% and height shifting about 5%. This is clearly the best results of any the techniques that have been tested in this project. This were a surprise to the author, as this were pretty far from the most discussed technique on the Kaggle forums. In the initial read-trough of the forums it were only noticed once. None of the top 3 solutions reported using TTA in their end documentation, but second place winner Beluga said it was something he wanted to test but didn't find time to. This experimentation indicates that a loss reduction could have been possible if using TTA for Beluga and other contestants not using TTA.

The best working augments are pretty equal to that of image augmentation, except for one big difference. That's in the flipping/flopping of images. For image augmentation there were no performance increase when flipping the images, but there were a big

one when flopping the images. For TTA there were some increase for flopping, but the biggest increase of all were gained by flipping the images. This means that classifying a upside-down version of the target alongside the original gives better accuracy than only the original. This raises many questions, like why did flipping not work for image augmentation before training, but works so well after, during the classification process? For the time being, the author does not have the answer to this. It might require further testing. An idea is to try to do augmentation at both ends at the same time, and see how the classification responds to that. That way, the network should be trained at recognising the for example upside down version of an image, so when it is exposed to it at the end it should classify it even better.

Looking further into the TTA results, the combination of multiple augmentations did not perform as well as hoped. None of the techniques gave a compounded effect, and more often than not the effect were worsened. Look at *zoom+width* as an example. In combination they gave a relative loss of 0.92. At their own, zoom gave 0.90 and width shift gave 0.94. This tells us something about how zooming increased the performance compared to width shifting. Ideally, zooming would give increased performance on, for example, 10 images, and width shifting on 6 completely different pictures. Then the effect would compound and the accuracy would increase. The real world experiments shows that zooming might for example affect 10 pictures, width shifting the same 10 images, but also negatively affect 4 images. Then the effect of adding width shifting to zooming is only negative compared to only zooming. This is a simplified example, but illustrates what might happen. *zoom+height* also does not compound, but gives better results than *zoom+width*, even though height shifting originally gave worse results than width shifting. This indicates that height shifting has a positive effect on more different images to zooming than width shifting.

Chapter 6

Conclusion and future work

In this project we have looked at how satellite images are made and what's been done in the field of iceberg/ship discrimination already. Further, we have looked at machine learning and how it could be used to solve the problem of classifying icebergs and ships. We have studied different techniques for mitigating dataset shortcomings, proposed during the Kaggle competiton hosted by Equinor and C-CORE. These have been presented and discussed, and a few were chosen for further testing on a state-of-the-art dataset compiled by C-CORE and Equinor. This testing showed impressively good results for some techniques, like test-time augmentation, while lackluster performance for others, like pseudo labelling. These experiences will be saved and used in a following master thesis, where a more complete classifier will be built.

When working with this project, a few ways to continue exploring the project became apparent. The first way is related to image augmentation, and the fact that adding multiple augments did not give a compounded performance gain. Because of this, it could be interesting for further work to look into ensambling a few of the single augmented models, and see how that performs compared to those trained using multiple augmentations at the same time. Then one could truly see if the performance gain can be compounded, or if the gains are closely related to each other. One could also try different augmentations than the ones used in this project. For example one could could experiment with shearing, the fill-mode for what to happen with the edges of the pictures, or maybe use larger or different values for the augments already tested.

Looking at what pseudo labelling could be used for in future work, it could be interesting to see how a model would respond to large amounts of unlabelled data. In this experimentation we were removing samples from an annotated test-set, only to give the samples pseudo labels and place them in the training set. If there exists a large collection of unlabelled iceberg/ship images, or if it could be quite easily constructed, it would be very interesting to see how the model would respond to pseudo labelling and training on those samples.

Further work on TTA could be what's described earlier: testing using both image augmentation on the training data and the test data at the same time. This could give interesting

and even better results. There is also always possible to test more augmentations. Other augmentation techniques, or the same but with different parameters.

Machine learning is an ever evolving field of study, with a lot of unexplored potential use-cases. The case of classifying satellite images is clearly a task well suited for machine learning, even though a lot of research is still required. Using the power of Kaggle and crowd sourcing a solution has given a lot of potential ideas towards a useable solution. It is only a matter of collecting and structuring the information.

Bibliography

Alpaydin, E., 2010. Introduction to Machine Learning, 2nd Edition. The MIT Press.

Bentes, C., Frost, A., Velotto, D., Tings, B., June 2016. Ship-iceberg discrimination with convolutional neural networks in high resolution sar images. In: Proceedings of EUSAR 2016: 11th European Conference on Synthetic Aperture Radar. pp. 1–4.

Blier, L., 2016. A brief report of the heuritech deep learning meetup #5. https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/.

Bunch, J. R., D.Fierro, R., 1992. A constant-false-alarm-rate algorithm. Linear Algebr Appl. 172, 231–241.

C-CORE, 2018. Statoil kaggle competition services, report R-17-077-1386, Revision 2.0.

Chollet, F., et al., 2015. Keras. https://keras.io.

Cozzolino, D., Martino, G. D., Poggi, G., Verdoliva, L., July 2017. A fully convolutional neural network for low-complexity single-stage ship detection in sentinel-1 sar images. In: 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). pp. 886–889.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. pp. 248–255.

English, J., Hewitt, R., Power, D., Tunaley, J., April 2013. Ice-sais — space-based ais and sar for improved ship and iceberg monitoring. In: 2013 IEEE Radar Conference (RadarCon13). pp. 1–6.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., Bengio, S., Mar. 2010. Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11, 625–660.
URL http://dl.acm.org/citation.cfm?id=1756006.1756025

ESA, 2014. 5.5 geometry glossary. https://earth.esa.int/handbooks/asar/CNTR5-5.html, accessed: 2018-09-24.

ESA, n.d.a. Instrument payload. https://sentinel.esa.int/web/sentinel/missions/sentinel-1/instrument-payload, accessed: 2018-11-15.

ESA, n.d.b. level 1. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/product-types-processing-levels/level-1, accessed: 2018-12-07.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96. AAAI Press, pp. 226–231.
URL http://dl.acm.org/citation.cfm?id=3001460.3001507

Franceschetti, G., Lanari, R., 2016. Synthetic Aperture Radar Processing, Second Edition, 2nd Edition. CRC Press, Inc., Boca Raton, FL, USA.

Glosser.ca, 2013. Artificial neural network with layer coloring.
URL https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Gonzalez, R. C., Woods, R. E., 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. The MIT Press.

Grandvalet, Y., Bengio, Y., 2005. Semi-supervised learning by entropy minimization. In: Saul, L. K., Weiss, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 17. MIT Press, pp. 529–536.
URL http://papers.nips.cc/paper/2740-semi-supervised-learning-by-entropy-minimization.pdf

Howard, A. G., 2013. Some improvements on deep convolutional neural network based image classification. CoRR abs/1312.5402.

Howell, C., Mills, J., Power, D., Youden, J., Dodge, K., Randell, C., Churchill, S., Flett, D., July 2006. A multivariate approach to iceberg and ship classification in hh/hv asar data. In: 2006 IEEE International Symposium on Geoscience and Remote Sensing. pp. 3583–3586.

Howell, C., Power, D., Lynch, M., Dodge, K., Bobby, P., Randell, C., Vachon, P., Staples, G., July 2008. Dual polarization detection of ships and icebergs - recent results with envisat asar and data simulations of radarsat-2. In: IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium. Vol. 3. pp. III – 206–III – 209.

Howell, C., Youden, J., Lane, K., Power, D., Randell, C., Flett, D., Sept 2004. Iceberg and ship discrimination with envisat multipolarization asar. In: IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium. Vol. 1. p. 116.

Hwang, J.-I., Chae, S.-H., Kim, D., Jung, H.-S., 2017. Application of artificial neural networks to ship detection from x-band kompsat-5 imagery. Applied Sciences 7 (9).
URL http://www.mdpi.com/2076-3417/7/9/961

Icons8, n.d. Free icons.
URL https://icons8.com/icons

Jun, L., Atharva, P., Dhruv, S., December 2017. Iceberg-ship classifier using sar image maps. Tech. rep., Stanford University.

Kerle, N., Janssen, L., Huurneman, G., Bakker, W., Grabmaier, K., van der Meer, F., Prakash, A., Tempfli, K., Gieske, A., Hecker, C., Parodi, G., Reeves, C., Weir, M., Gorte, B., Horn, J., Pohl, C., Ruitenbeek, F., Woldai, T., 01 2004. Principles of remote sensing : an introductory textbook. Principles of remote sensing : an introductory textbook.

Krizhevsky, A., 05 2012. Learning multiple layers of features from tiny images. University of Toronto.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12. Curran Associates Inc., USA, pp. 1097–1105.
URL http://dl.acm.org/citation.cfm?id=2999134.2999257

LeCun, Y., Bengio, Y., Hinton, G., 05 2015. Deep learning. Nature 521, 436–44.

Lee, D.-H., 07 2013. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. ICML 2013 Workshop : Challenges in Representation Learning (WREPL).

Lee, J., March 1980. Digital image enhancement and noise filtering by use of local statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2 (2), 165–168.

Lusch, D. P., 1999. Introduction to microwave remote sensing. Center for Remote Sensing and Geographic Information Science Michigan State University.

Mahdavi, S., Salehi, B., Moloney, C., Huang, W., Brisco, B., 07 2017. Speckle filtering of synthetic aperture radar images using filters with object-size-adapted windows. International Journal of Digital Earth.

Masci, J., Meier, U., Cireşan, D., Schmidhuber, J., 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In: Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I. ICANN'11. Springer-Verlag, Berlin, Heidelberg, pp. 52–59.
URL http://dl.acm.org/citation.cfm?id=2029556.2029563

Nielsen, M. A., 2015. Neural Networks and Deep Learning. Determination Press.

Pelich, R., Longépé, N., Mercier, G., Hajduch, G., Garello, R., July 2015. Performance evaluation of sentinel-1 data in sar ship detection. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). pp. 2103–2106.

Power, D., Yue, B., Youden, J., December 2013. Summary of previous research in iceberg and ship detection and discrimination in sar. Tech. rep., C-Core.

Raschka, S., 2015. Python Machine Learning. Packt Publishing.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115 (3), 211–252.

Simard, P. Y., Steinkraus, D., Platt, J. C., Aug 2003. Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. pp. 958–963.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556.
URL http://arxiv.org/abs/1409.1556

Usmani, Z.-u.-h., 2017. Kaggle for Beginners: with Kernel Code. Gufhtugu Publications.

Valkov, V., 2017. What to do when data is missing? - part ii.
URL http://curiousily.com/data-science/2017/02/02/what-to-do-when-data-is-missing-part-2.html

van Zyl, J., 2011. Synthetic Aperture Radar Polarimetry. JPL Space Science and Technology Series. Wiley.

Wang, G., Li, W., Ourselin, S., Vercauteren, T., 2018. Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation. arXiv preprint arXiv:1810.07884.

Wang, Y., Wang, C., Zhang, H., Nov 2017. Combining single shot multibox detector with transfer learning for ship detection using sentinel-1 images. In: 2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA). pp. 1–4.

Wu, B., Dec 1992. An introduction to neural networks and their applications in manufacturing. Journal of Intelligent Manufacturing 3 (6), 391–403.
URL https://doi.org/10.1007/BF01473534

Wu, R., Yan, S., Shan, Y., Dang, Q., Sun, G., 2015. Deep image: Scaling up image recognition. arXiv preprint arXiv:1501.02876.

Yaeger, L., Lyon, R., Webb, B., 1996. Effective training of a neural network character classifier for word recognition. In: Proceedings of the 9th International Conference on Neural Information Processing Systems. NIPS'96. MIT Press, Cambridge, MA, USA, pp. 807–813.
URL http://dl.acm.org/citation.cfm?id=2998981.2999095