Evaluation and implementation of a Fully Convolutional Neural Network for Iceberg Classification

Author: Patrick Schneider

Subject: Kernel-Based Machine Learning and Multivariate Modelling

Professor in charge: Marta Janira Castellano Palomino

Date: 08.01.2019

1. General

1.1 Abstract

Drifting icebergs present threats to navigation and activities in areas such as offshore of the East Coast of Canada. Currently, many institutions and companies use aerial reconnaissance and shore-based support to monitor environmental conditions and assess risks related to icebergs. However, in remote areas with particularly harsh weather conditions, these methods are not feasible, and the only viable monitoring option is via satellite. This report illustrates a fully convolutional neural network model that automatically identifies if a remotely sensed target is a ship or iceberg. Several different learning rate implementations, activation functions and architecture types were evaluated on the prediction outcome and behavior. This dataset has been collected by C-CORE.

1.2 Introduction

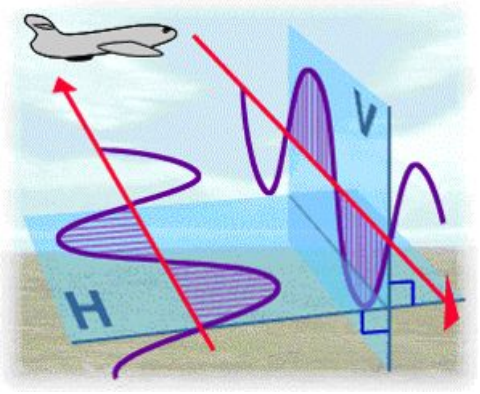
The international energy company Statoil has worked with companies like C-CORE. C-CORE have been using satellite data for over 30 years and have built a computer vision based surveillance system. To keep operations safe and efficient, Statoil is interested in new ideas using machine learning to more accurately detect and discriminate against threatening icebergs as early as possible.

1.2.1 Background Technology

The remote sensing systems which are used to detect icebergs are installed on satellites over 600 kilometers above the Earth. The Sentinel-1 satellites monitor land and ocean. The satellite captures images of the Earth's surface at a given location and a given instant in time. The C-Band radar operates at a frequency that sees through darkness, rain, cloud and fog.
Satellite radar works in much the same way as aircraft radar. It sends a signal off an object and records the echo. Then that data is translated into an image. An object will appear as a bright spot because it reflects more radar energy than its surroundings. But strong echoes can come from anything solid - land, islands, sea ice, as well as icebergs and ships. The energy reflected back to the radar is referred to as backscatter.

When the radar detects an object, it can not tell an iceberg from a ship or any other solid object. The object needs to be analyzed for certain characteristics - shape, size and brightness - to determine the type of object. The surrounding area, in this case the ocean, can also be analyzed or modeled. Many things affect the backscatter of the ocean or background area. High winds will generate a brighter background. Conversely, low winds will generate a darker background. The Sentinel-1 satellite is a side looking radar, which means it sees the image area at an angle. The ocean will be darker at a higher angle.

In this specific application the radar polarization needs to be considered, which is how the radar transmits and receives the energy. Sentinel-1, can transmit and receive in the horizontal and vertical plane (dual-polarization image).



The components are:

- HH (transmit/receive horizontally)

- HV (transmit horizontally and receive vertically).

Those components play an important role in the object characteristics, since objects tend to reflect energy differently. Easy classification examples are seen below. These objects can be visually classified.

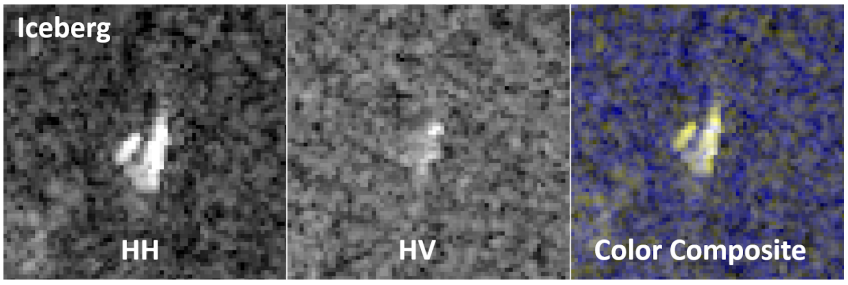


Figure 3. Example visualization of iceberg. (Left) Band1, (middle) Band2, (right) composition

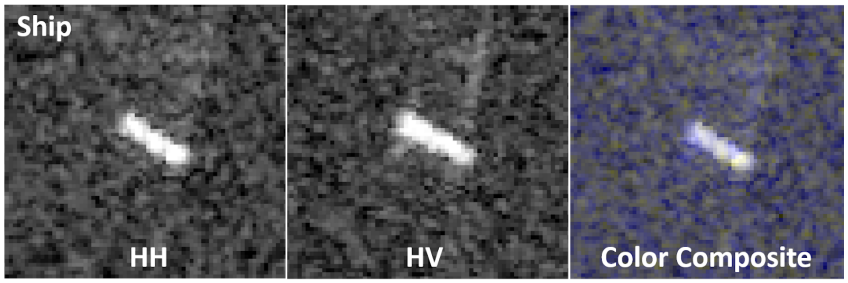


Figure 4. Example visualization of ship. (Left) Band1, (middle) Band2, (right) composition

Below are hard classification examples:

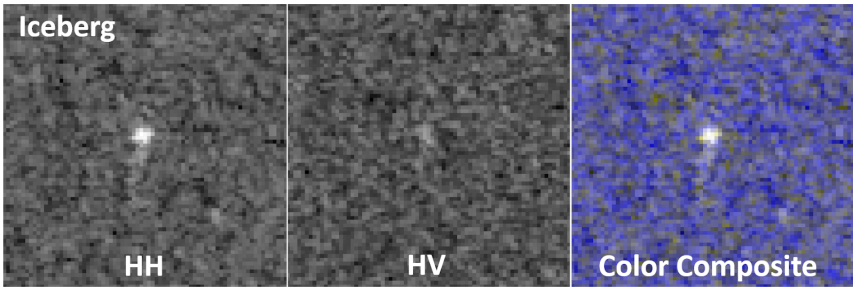


Figure 5. Difficult visualization of iceberg. (Left) Band1, (middle) Band2, (right) composition

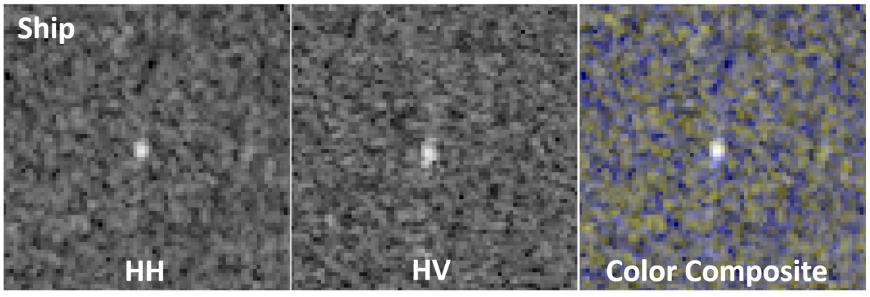


Figure 6. Difficult visualization of ship. (Left) Band1, (middle) Band2, (right) composition

1.2.2 Data set

The complete data set is composed of a labeled train and unlabeled test set. The unlabeled test set will be not used in the experimental stage. The test set can be used to compete in the Kaggle competition to retrieve a performance metric.

The data is presented as train.json and test.json. The files consist of a list of images with the following fields:

- **id** - the id of the image
- **band_1, band_2** - the flattened image data. Each band has 75x75 pixel values in the list(5625 elements). These are float numbers with unit being dB. Band 1 and Band 2 are like earlier explained signals characterized by radar backscatter produced from different polarizations at a particular angle. The polarizations correspond to HH (transmit/receive horizontally) and HV (transmit horizontally and receive vertically).
- **inc_angle** - the incidence angle of which the image was taken. This field has missing data marked as "na". Those images with "na" incidence angles are all in the training data to prevent leakage [reasoning of competition host]. First model evaluations have shown that the improvement where not significant with inc_angle, that's why I decided to take them out in the training. Note: In the competition are solutions that worked successfully with the inc_angle in a more sophisticated manner.
- **is_iceberg** - the target variable, set to 1 if it is an iceberg, and 0 if it is a ship.

1604 images in the training set. Each image is represented in 5625 elements from band 1, 5625 elements from band 2, which are flattened images of data, and inc angle, the incidence angle of the image. In summary: 1604 images with 11,251 features.
The bands are reshaped into 75x75 pixel matrices, and normalizing dB frequency to a scale between 0 and 1.

1.3 Previous Work

Previous works on Synthetic aperture radar (SAR) images include using multivariate approaches, support vector machines (SVM) and Convolutional Neural Networks (CNN) for classification. A two-class maximum likelihood model can classify ships with an accuracy of 93% [4] by maximizing the posteriori probability obtained from bayes rule. A study conducted by the German Aerospace Centre show that CNNs can outperform SVMs, by achieving an F1 score of 97%.
Spatial FCM clustering method were used to detect coastlines in FCM images [5]. Although the data set is smaller, the idea can be used to detect the edge of the object in the center of the image. A filter can be applied to remove the background noise. Moreover, shape properties can be exploited to build the model. The combination of HV and HH bands, can be used for feature extraction as shown by [5].
Convolutional Neural Networks are leading the way in image classification problems. The AlexNet algorithm [6], won the 2012 ILSVRC and the publication is regarded as one of the

most influential publications in deep learning. SAR images are associated with speckle noise, that affect their quality. Spatial filters reduce the speckle noise at the expense of image details and edges. Methods have been developed to suppress speckle with multi-look processing [8], filtering methods [9], wavelet-based despeckling methods [10], lock-matching 3D algorithm [11] and Total Variation methods [12].

Other noteworthy:
[13] proposes the application of Convolutional Neural Networks (CNN) for ship-iceberg discrimination in high resolution TerraSAR-X StripMap images. The CNN model is compared with a SVM and the final results indicate a superior classification performance of the proposed CNN method, which is interesting in the context of this subject and related work on the expertise of this project.

[14] shows how to examine the training validation/test loss function for subtle clues of underfitting and overfitting and suggests guidelines for moving toward the optimal balance point. Then discusses how to increase/decrease the learning rate/momentum to speed up training, which was an helpful indicator for this project.

2. Own Work

2. 1 Theory

2.1.1 Convolutional Neural Network

Input Layer:
This layer takes as a input of three bands with each 75 pixel.

Output Layer:
The output layer gives an estimate on the probability for the two classes (ship or iceberg).

Convolution layers:
These layers create a convolution kernel that is convolved with the layer input over a single spatial dimension to produce a tensor of outputs to the next layer.

Pooling layer:
Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layers operate on each feature map independently.
In this project exclusively max-Pooling and global-Average-Polling is used.

- Max-Pooling: Takes the max over the steps too but constrained to a pool_size for each stride.
- global-Average-Polling: GAP averages every value of (x,y) coordinate in 1 feature map into 1 value, then send this value to softmax function for classification.

2.1.2 Adaptives learning rates

The simplest and probably most used adaptation of learning rate during training are techniques that reduce the learning rate over time. These have the benefit of making large changes at the beginning of the training procedure when larger learning rate values are used. Afterwards decreasing the learning rate so that a smaller rate with smaller training updates are made to the weights in a the later stage of the training process.With that, good weights can be learned early and fine tuned later.

Popular learning rate schedules are the following (which are also considered in the experiments):

- Decrease the learning rate gradually based on the epoch (**time based decay**)
- Decrease the learning rate using punctuated large drops at specific epochs (**step decay**)
- Decrease the learning rate based on an exponential function (**exponential decay**)

2.1.3 Optimization methods

Stochastic gradient descent is the default optimization method in the keras framework and performs a parameter update for each training example x(i) and label y(i).

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

The algorithm calculates an exponential moving average of the gradient and the squared gradient. The parameters beta1 and beta2 control the decay rates of these moving averages.
The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before, then calculating bias-corrected estimates.

alpha	Learning rate/step size. Represents the proportion that weights are updated (e.g. 0.01). Larger values like 0.3 result in faster initial learning before the rate is updated. Smaller values like 1.0E-5 slows learning down during training
beta1	The exponential decay rate for the first moment estimates (e.g. 0.9)

beta2	The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient like in this project.
epsilon	Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

Table 1. Coefficient overview and explanation for ADAM

Other tested optimization techniques:

Adaptive Gradient Algorithm (AdaGrad)	maintains a learning rate for each parameter that improves performance on problems with sparse gradients
Root Mean Square Propagation	maintains for each parameter a learning rate that are adapted based on the average of recent magnitudes of the gradients for the weight - e.g. how quick it changes. This means the algorithm does well on noisy problems (online and non-stationary).
Adagrad	Adagrad is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the updates.[3]

Table 2. Other tested optimization methods and their explanation

2.1.4 Activation functions

The activations were tested with tanh, sigmoid and the ReLU function. In the following the main differentiation of those function in the context of this project.

Early reasoning: When a gradient gets backpropagated through the tanh and sigmoid functions, they will zero out the gradient if the activation is high.
E.g. if the output of a layer is 5, the derivative of tanh at x=5 is flat (gradients saturating). The outputs of layers in the forward pass are too high for these types of activations to backprop a meaningful gradient, and thus the network can not learn.
This takes place within a single layer activation. That means that it can happen for a deep network as well as for a one-layer neural network. ReLU prevents this issue from happening by providing a nonlinear function that doesn't saturate. That means that the derivative is always just 1 for x > 0. It acts as an identity function in backpropagation.

2.1.5 Architecture

The final solution was inspired by the full convolutional networks for 2D segmentation architecture. Source: [15]
The python output can be seen in Appendix A.

It takes the input image of size 75x 75x 3 (RGB image)

Built using:
5x Convolution layers (used only 3x3 size)
4x Max pooling layers (used only 2x2 size)
1x Global Average Pooling
5 x Zero padding(1x1)

Description of layers:

- Zero padding(1x1) in front every layer
- Batch normalization for the first 4 convolutional layers

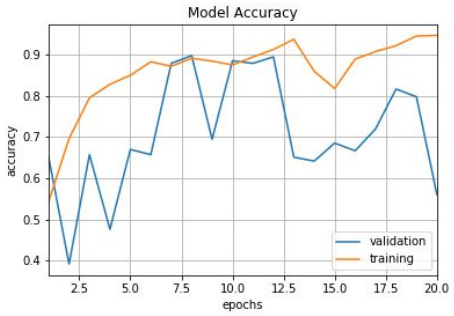
Input image: 75x75x3

Convolution using 32 filters + Max pooling (37x37x32)
Convolution using 64 filters + Max pooling (18x18x64)
Convolution using 128 filters + Max pooling (9x9x128)
Convolution using 128 filters + Max pooling (4x4x128)
Convolution using 2 filters + Global average Pooling (4x4x2)

Output layer with Softmax activation on two nodes

2.2. Experiments

Baseline result:



The baseline represents the described FCN architecture with a constant learning rate and SGD.

In the picture it can be observed that the training is quite constant, while the validation fluctuate a lot and gets no stability.

Learning annihilation:

The **step decaying** learning showed slightly better results and decreased the learning on the midway of the 20 epochs.

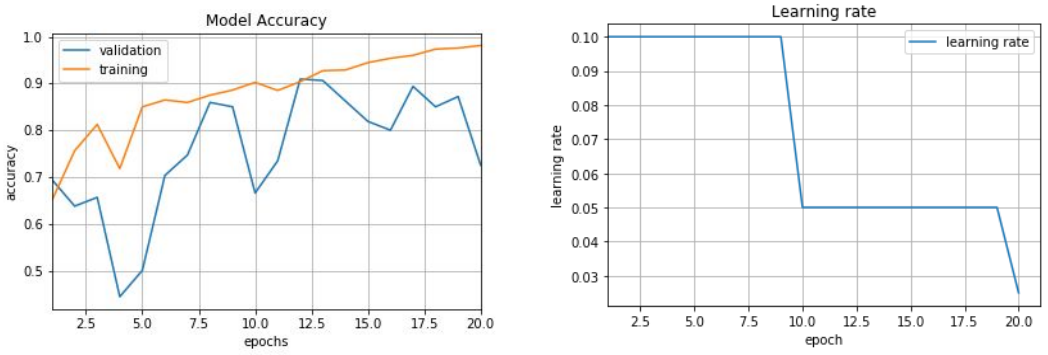


Figure 8. Left) Step decaying learning result, (right) learning rate evolution over epochs

The best result got archived with the exponential decay function illustrated below.

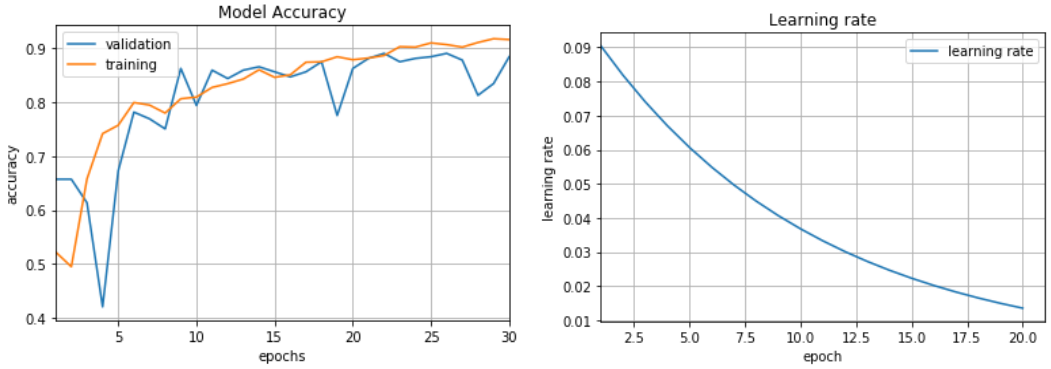


Figure 10. Left) Step exponential decaying learning result, (right) its learning rate evolution over epochs

The learning rate gradually decreased over each epoch and the validation accuracy was near the training accuracy at any time.

Batch size:

General observation was that a better performance was achieved early on with a bigger batch size, because an epoch gets fit on more data in a run.b6

Architectures and Activation function:

To test the activation functions the decision fell on a smaller network. The tanh function in the original FCN did not propagate the error through the network properly. In a 2 layer CNN the difference in error propagation on the derivative became more obvious (see figure 12). The relu function had a better optimization on the validation set and achieved after 30 epochs the better accuracy but was prone to overfitting. The tanh activation seemed to have not such a quick optimization on the validation set, but seemed to be more stable. It would have been interesting to see the optimization on several more epochs.

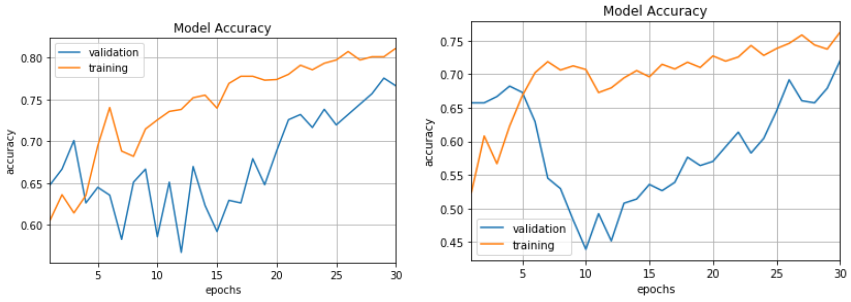


Figure 12. layer CNN with (left) relu activation, (right) tanh activation

Overall performance on validation set

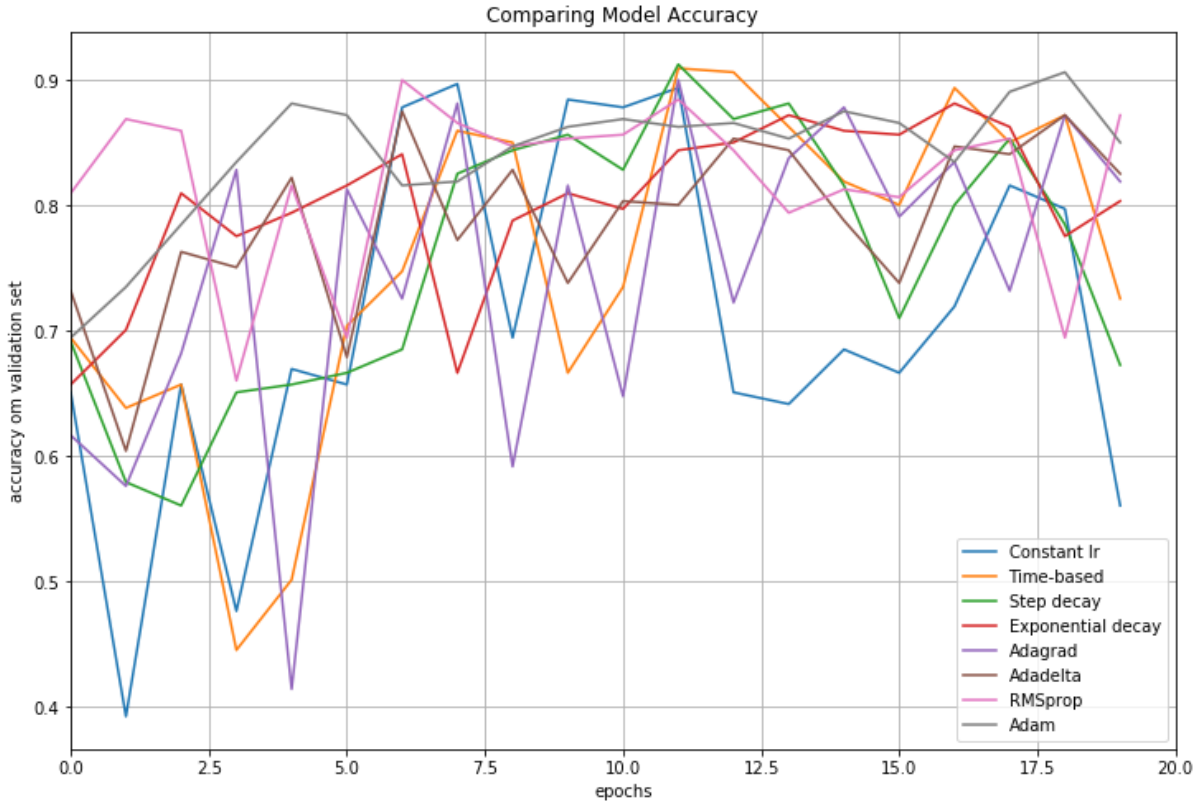


Figure 13. Comparison of all used methods on a 20 epochs range

This picture shows the performance of the training on the validation set accuracy of all the tested methods. **Adam** had in most of the observed experiments the best overall fit. This plot illustrates well of how important it is to choose the right methods for learning and training optimization. The **constant learning** rate showed no reliable learning over increasing epochs. I followed the assumption that most used method (excluded constant lr) find their convergence over time in the same accuracy level. The interesting part is to find a method that converges early to save computing power.

3. Discussion

This section gives a general project *conclusion* of the results. The *critical assessment* section discusses challenges and important improvements in an experimental environment to prove a hypothesis. The *future work* section discusses improvements that might lead to better results and need to be evaluated.

3.1 Conclusion

The illustrated FCN architecture archived an accuracy of 92,68% in the 32 epoch with ADAM and exponential decay learning. Except for constant learning rate, it seemed that in a proper hyperparameter setup all of the used methods find in later epochs the convergence of an accuracy of +91%. Further did the learning rate configuration show that a big LR leads to big improvements in each epoch but holds the risk of overfitting. Most of the fitting was initiated on 0.1 LR and showed high variance over the epochs. That's why it might be recommended to implement early stopping rounds and test the optimization on smaller LR (e.g. <0.01) with the right learning decay method. The input consisted of 3 channels (band1, band2 and the difference of band1/band2), where the pre-processing only considered the rescaling and normalization of the images. With this relatively straightforward implementation and the hint of using already established FCN architectures, the challenge of classifying icebergs and ships was surprisingly simple solved in an early approach.

3.2 Critical Assessment

Accuracy alone will not make the best indicator [Analysis of confusion matrix]
A false negative results in ships being unnecessarily rerouted, causing delays and financial loss, while a missed detection can be catastrophic and lead to a crash. In a real life models these two cases should be considered independently.

Architecture
The reasoning on choosing this architecture was hard to explore, but seems to represent the spirit of deep learning. Only 3 different architectures were tested with relevant results (accuracy over >0.78). The winning architecture was the presented one in 2.1.5. The other architecture were a "slim" version of the winning one and due to the missing time, computing power and intuition not in detail explained.

Needed computing power
I was warned that this is not a simple project. I especially realized this in the experimental stage. The experiments took a quite long time to process (30 minutes of one fit on 20 epochs on my notebook). The solution for me was to rent computing power on AWS. AWS provides a data science environment with Jupyter notebook and Tensorflow. To make the process of experimenting more exhaustive, time efficient and cost efficient, a solution would have been

to create a AWS cluster and rent cheap but powerful spot price instances to automatically deploy them in the computing cluster for relatively little cost.

The interesting behavior of the different learning rate scheduling function was visible starting at epoche 25+.

Activation functions - backpropagation speed difference
In the experiments it became clear that the ReLU function provided a quicker result with an accuracy of 0.92 after 20 epoches. Tanh and sigmoid increased the performance not as quick in 20 epoches (0.82) but more stable, where it would have been interesting to see the result with more than 40 epoches on all functions.

For an experimental environment, several runs should have been tested on different seeds to proof a hypothesis.

3.3 Future work

Preprocessing
Another improvement can be the preprocessing of the data. In this report the incidence angle was not considered which may influences the result.
The incidence angle can affect the intensity of the satellite readings. Other approaches considered the angle but the theory of satellite imaging technologies was above my understanding at this stage.

Transfer learning on pre trained model
Assumption: There are pre trained models out there for coastal satellite imaging that could be used for transfer learning with a small LR on this data set.

Dropout layer
The effects of using a dropout layer to reduce overfitting was not explored in this project and could be checked on future work.

Cyclic Learning implementation
Can be found successfully implemented in the code but there was no time left to parameterize and analyze it in a useful manner.

Architectures
There are several more sophisticated architectures that made their name in satellite image processing like the VGG-16 architecture. Further, expert opinions in the scene point on Squeeze-and-Excitation Networks (SEnets) to archive further improvements. The implementation of SENets introduces a building block for CNNs to improve channel interdependencies at almost no computational cost. It does this by adding parameters to each channel of a convolutional block so that the network can adaptively adjust the weighting of each feature map.

References:

[1] https://sentinel.esa.int/web/sentinel/missions/sentinel-1/instrument-payload [05.01.2019]

[2] Carlos Bentes, Anja Frost, Domenico Velotto, Björn Tings - Ship-Iceberg Discrimination with Convolutional Neural Networks in High Resolution SAR Images https://elib.dlr.de/99079/2/2016_BENTES_Frost_Velotto_Tings_EUSAR_FP.pdf [05.01.2019]

[3] Sashank J. Reddi, Satyen Kale, Sanjiv Kumar -On the Convergence of Adam and Beyond, 2018

[4] C. Howell et al. “A Multivariate Approach to Iceberg and Ship Classification in HH/HV ASAR Data”.
In: 2006 IEEE International Symposium on Geoscience and Remote Sensing (2006).

[5] Gholamreza M. Mohammad A. “A level set based method for coastline detection of SAR images”. In:
IEEE (2017).

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: Communications of the ACM 60.6 (2017),

[7] Kaiming Xiangyu Shaoqing Jian. “Deep Residual Learning for Image Recognition”. In: IEEE (2015).

[8] Jean-Marie Nicolas and Frdric Adragna. “The Principles of Synthetic Aperture Radar”. In: Processing
of Synthetic Aperture Radar Images (2010), pp. 25–55.

[9] Victor S. Frost et al. “A Model for Radar Images and Its Application to Adaptive Digital Filtering of
Multiplicative Noise”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4.2
(1982)

[10] Hua Xie, L.e. Pierce, and F.t. Ulaby. “SAR speckle reduction using wavelet denoising and Markov
random field modeling”. In: IEEE Transactions on Geoscience and Remote Sensing 40.10 (2002)

[11] Guangyi Chen et al. “Adaptive video denoising using block matching 3-D filtering”. In: 2012 25th
IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (2012)

[12] Jose M Bioucas-Dias and Mario A T Figueiredo. “Multiplicative Noise Removal Using Variable Splitting
and Constrained Optimization”. In: IEEE Transactions on Image Processing 19.7 (2010)

[13] Li Jun, Parulekar Atharva, Samant Dhruv: Iceberg-Ship classifier using SAR Image Maps http://cs229.stanford.edu/proj2017/final-reports/5244144.pdf [05.01.2019]

[14] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay: https://arxiv.org/abs/1803.09820 [05.01.2019]

[15] Jonathan Long, Evan Shelhamer, Trevor Darrell: Fully Convolutional Networks for Semantic Segmentation; UC Berkeley https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf [05.01.2019]

Appendix A - Architecture

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 75, 75, 3)	0
zero_padding2d_1 (ZeroPaddin	(None, 77, 77, 3)	0
conv2d_1 (Conv2D)	(None, 75, 75, 32)	896
batch_normalization_1 (Batch	(None, 75, 75, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 37, 37, 32)	0
zero_padding2d_2 (ZeroPaddin	(None, 39, 39, 32)	0
conv2d_2 (Conv2D)	(None, 37, 37, 64)	18496
batch_normalization_2 (Batch	(None, 37, 37, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 18, 18, 64)	0
zero_padding2d_3 (ZeroPaddin	(None, 20, 20, 64)	0
conv2d_3 (Conv2D)	(None, 18, 18, 128)	73856
batch_normalization_3 (Batch	(None, 18, 18, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 9, 9, 128)	0
zero_padding2d_4 (ZeroPaddin	(None, 11, 11, 128)	0
conv2d_4 (Conv2D)	(None, 9, 9, 128)	147584
batch_normalization_4 (Batch	(None, 9, 9, 128)	512
max_pooling2d_4 (MaxPooling2	(None, 4, 4, 128)	0
zero_padding2d_5 (ZeroPaddin	(None, 6, 6, 128)	0
conv2d_5 (Conv2D)	(None, 4, 4, 2)	2306
global_average_pooling2d_1 (	(None, 2)	0
activation_1 (Activation)	(None, 2)	0

Total params: 244,546
Trainable params: 243,842
Non-trainable params: 704