Machine Learning Engineer Nanodegree

Capstone Project: Ship\Iceberg discrimination

March 26, 2018

I. Definition

Project Overview

The project presents a solution for the Statoil/C-CORE Iceberg classifier challenge hosted on Kaggle platform [1]. The competition sponsor suggests to address the problem of iceberg detection.
Drifting icebergs might be dangerous for naval activities, so people currently exploit different approaches to address this problem:

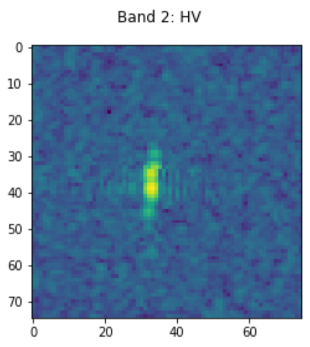
1. aerial reconnaissance
2. monitoring environment conditions from shore to assess risks
3. icebergs detection on images taken from satellites

The latest approach is usually the only one available in the remote areas or areas with severe weather conditions. Undoubtedly, images, in this case, should be processed automatically, using machine learning methods. The sponsor of the competition is already using one developed by C-CORE, but looking for improvements in the classification of objects in the water into two categories - ships and icebergs (binary classification).
The ship\iceberg discrimination problem was already addressed in multiple studies. Bentes et al. [2] compared several approaches to this problem and showed that convolutional neural networks outperformed others (F1 score 98% against 88% for SVM and 84% for SVN+ PCA). Howell et al. [3] suggest quadratic discriminant (QD) with feature selection based on the sequential forward selection. This approach was mentioned as the most advantageous (comparing to CNNs, unsupervised approaches like k-nearest neighbors and others) in C-Core report [4].

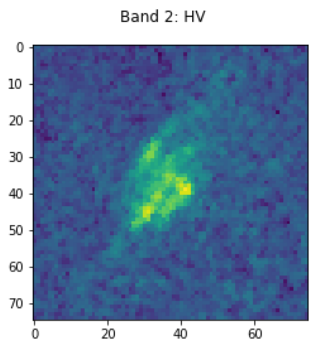
[1]. Statoil/C-CORE Iceberg classifier challenge hosted on Kaggle platform. https://www.kaggle.com/c/statoil-iceberg-classifier-challenge
[2]. Ship-Iceberg Discrimination with Convolutional Neural Networks in High Resolution SAR Images. http://ieeexplore.ieee.org/abstract/document/7559347
[3]. Dual Polarization Detection of Ships and Icebergs - Recent Results with ENVISAT ASAR and Data Simulations of RADARSAT-2. http://ieeexplore.ieee.org/document/4779319/
[4]. Summary of previous research in iceberg and ship detection and discrimination in SAR. http://cradpdf.drdc-rddc.gc.ca/PDFS/unc196/p800002_A1b.pdf

Problem Statement

The Statoil provides images of ships and icebergs made from the space by a satellite Sentinel-1. They are centered, cropped and manually labeled by human experts. The task is to develop a model that can classify these images and achieve a reasonable accuracy in the context of competition. As could be seen in the images below some of the objects can be easily classified without any expert knowledge. For example, image (a) is apparently a ship as it has very well-defined rectangular shape. On the other hand, image (b) is an iceberg, because its form would be very unusual for the artificially created naval object.

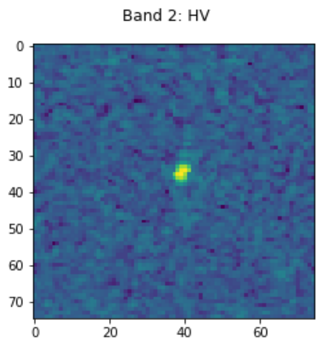


(a)

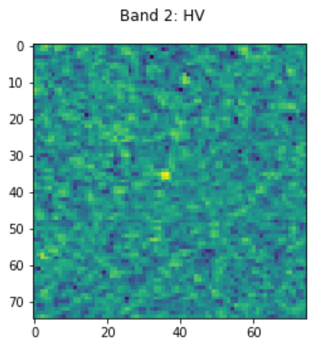


(b)

However, some objects are the challenge for a classifier. For example, on the image below objects are almost indistinguishable ((a) is a ship, (b) is an iceberg).



(a)



(b)

Metrics

In the project, the following metrics will be used:

- Log loss:

Represents a log-likelihood of the labels given their predicted probabilities. A convenient measure for classification problems, as it strongly penalizes misclassifications. As could be seen from the formula for binary classification:

$$loss = -\frac{1}{N}\sum_{i=1}^{N} y_i log\, p_i \;+\; (1-y_i)\, log\,(1-p_i)$$

it gets large values if the predicted probability for the correct label is close to zero. Moreover, log loss is usually used in neural networks in the context of classification problems, as it guarantees non-vanishing gradients during backpropagation process (if sigmoid activations are used). Log loss is also suggested by competition sponsor as metric to evaluate and compare predictions on the test dataset.

- Accuracy:

Represents percentage of correctly classified images. It is a more human-readable way to evaluate classifier, so it is often used as an auxiliary metric. As accuracy ignores any information about confidence (probabilities) of classification, it is much less expressive than log-loss.

- Confusion Matrix:

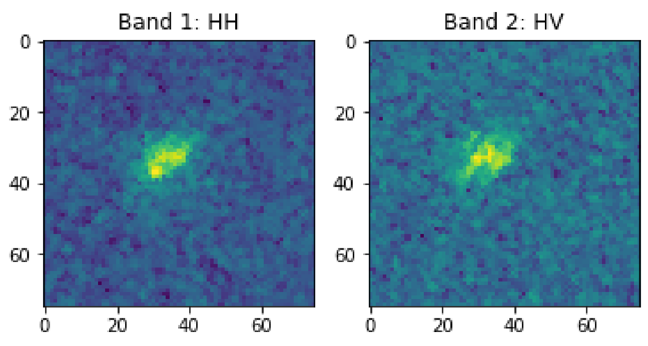
A matrix NxN, where N is a number of classes. Diagonal elements of this matrix show number or percentage of correctly classified examples of each class. Non-diagonal elements show statistics for misclassified examples. The confusion matrix is a convenient way to assess the quality of classifier as well as to analyze which classes are challenging for it.

II. Analysis

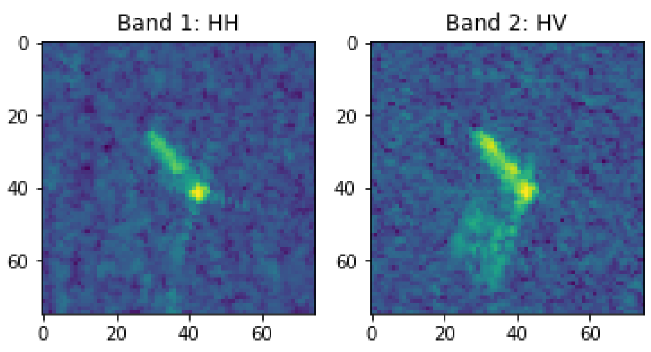
Data Exploration and Visualization

Images are obtained by satellite Sentinel-1 using C-Band radar [1]. Frequencies used by this radar allow making images of Earth surface even if it cannot be visually observed due to the weather conditions (clouds, fog, etc.). The radar emits the signal which reflects from objects and returns back (so-called backscatter), where these echoes can be recorded and translated to the image. Solid objects have stronger echoes comparing to water and appear as bright objects on the darker areas, representing water.

Sentinel-1 can emit and receive signals with horizontal and vertical polarizations, so data consists of dual-polarization images (images with two bands). HH band is obtained by recording signal which was transmitted and recorded horizontally. HV band is obtained from the signal which was transmitted horizontally but received vertically.



Example of HH and HV bands for an iceberg.



Example of HH and HV bands for a ship.

The dataset contains 1604 examples, and it is quite well-balanced: 753 examples of icebergs and 851 examples of ships. The dataset is organized into two files (training and testing data) in JSON format. Every example in the dataset has following fields:

1. **id** of the example
2. **band_1, band_2** - 75x75 1-channel images in flattened form, so each is a 1D vector with 5625 numbers. Images are represented not by intensities (integers), but by the strength of the backscatter signal (float numbers representing signals measured in dB units).
3. **is_iceberg** - target variable: 1 if iceberg and 0 if ship.
4. **inc_angle** - incidence angle. Means the angle, at which satellite sees the image area. This field might have “NA” values.

[1] SENTINEL-1 mission overview:
https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/overview

Algorithms and Techniques

The main problem with conventional supervised learning methods in the context of image classification is non-robustness to the translation and scale transformations. However, in the given dataset images a preprocessed and cropped so that objects are centered and have similar size. Thus, I decided to try a few supervised learning methods to address a problem:

1. Logistic Regression

A simple basic model which attempts to separate data with a linear (w.r.t. weights) decision boundary. In the project, it will be used as a benchmark model.

2. Support Vector Machines

A powerful technique that is effective in high-dimensional spaces, even if a number of features close to the size of the dataset (as in a given dataset)[2]. The main disadvantage of this approach is the necessity to fine-tune the model.

Also, I decided to try convolutional neural network (CNN), which is a common approach to the image classification problem and shows impressive results in this area. For example, deep CNN architecture VGG16 achieves 7.4% top-5 error on ILSVRC-2012-test [1]. The large advantage of this approach is an out of the box feature learning and robustness to the scale and translation transformations. However, neural networks are prone to overfitting and require large dataset and high computational resources. As given dataset is quite small, different techniques to prevent overfitting should be used (dropout, data augmentation, etc.). Another problem is two-channel images, while CNN expects 1 or 3 channels inputs. Incidence level feature can be fed as an additional input to the top fully connected layers of the CNN.

Finally, Principal Component Analysis (PCA) was performed in order to reduce dimensionality of data and, possibly, improve accuracy of both - CNN and SVM approaches.

[1]. K.Simonyan, A.Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/abs/1409.1556
[2]. Support Vector Machines http://scikit-learn.org/stable/modules/svm.html

[3]. Gradient Tree Boosting
http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting

Benchmark

The competition sponsor does not provide any benchmark results to compare with. Thus two following naive models were used:

- Random guess model: as shown above, the dataset is well balanced, so random guess provides mean accuracy 0.50 and mean log loss 17.17 over ten runs.
- Logistic regression: non-optimized default sci-kit model trained on averaged bands showed mean accuracy 0.67 and mean log loss 11.33 in 5-fold cross-validation runs. The same model trained on the concatenated bands showed mean accuracy 0.71 and mean log loss 10.06. In both cases, information about angles was ignored. I also trained Logistic Regression model in the pipeline with principal component analysis (with a grid search over hyperparameter space) and the best found model showed mean accuracy 0.73 and mean log loss 9.46.

	Random guess	Logistic Regression (trained on averaged bands)	Logistic Regression (trained on concatenated bands)	Logistic Regression (with PCA)
accuracy	0.5	0.67	0.71	0.73
log loss	17.17	11.33	10.06	9.46

III. Methodology

Data Preprocessing

The following data preprocessing was required:

- Reshaping for CNN

The examples in the given dataset contain two flattened images representing HH and HV bands. Thus, to be an input for CNN they should be reshaped to (75x75) matrices and stacked together as 2 channels of color image.The third channel was faked depending on the applied model: either with average values of two other channels or with incidence angle. Also, some models were trained om images with 2 channels only.

- Concatenation\averaging of bands

To feed both bands to the rest of the algorithms (logistic regression and SVM), bands should be either averaged or concatenated together. I tried to fit models with both approaches.

- “na” values in the incidence angle feature

Incidence angles are not provided for approximately 8% of the data (has “na” values). These values were set to 0.

- Rescaling

All values were rescaled to the range [0, 1].

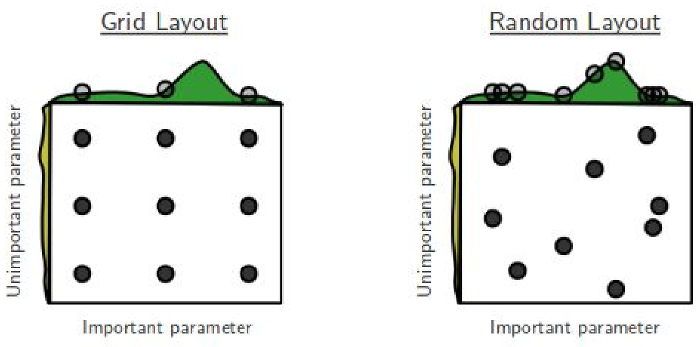
Implementation

Hyperparameter tuning

The 2 following approaches were used in this project:

- Grid Search: given predefined values along each dimension, it explores all possible combinations of them (points lying on the grid in hyperparameter space).
- Random Search: explores random points in hyperparameter space.

The grid search is a simple exhaustive technique, helpful in the low-dimensional spaces with a small number of values of interest along each dimension. However, as shown in the figure below, in the case of high effective dimensionality, grid search will explore only a few, predefined values along each dimension. On the contrary, random search tries new values every trial and has a higher probability to escape local extremum. In [1], J.Bergstra and Y.Bengio showed that random search technique is preferable, comparing to the grid search.



Grid and random search of nine trials. [1].

In this work, scikit versions of grid and random search were used: GridSearchCV and RandomizedSearchCV. The first one was applied to the logistic regression and PCA for exploring best regularization strength and number of components. The latter was applied to the SVM for optimizing C, gamma and class weight hyperparameters.

[1]. J.Bergstra, Y.Bengio. Random Search for Hyper-Parameter Optimization.

K-Fold Cross Validation

Both GridSearchCV and RandomizedSearchCV perform also a cross-validation. The K-fold cross validation is a validation technique, when data is split to the k sets called “folds” and model is trained k times with k-1 folds used as a training dataset and 1 used for validation. This approach is more computationally expensive comparing to classic validation using held out set. However, it allows to not waste so much data for validation and helps to avoid overfitting of hyperparameters (as more data on average is used for validation of model).

[1]. Cross validation:
http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Support Vector Machines

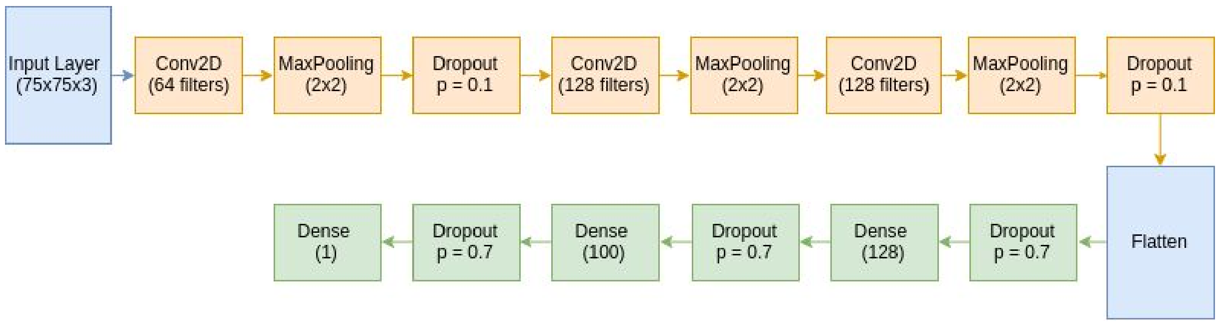
In the case of applying SVM to the high dimensional data, optimization of regularization term is essential. Thus, I tried to apply random search approach with a log-loss as a scoring function from a beginning. It was run in two steps:

1. with a wast dispersion of hyperparameters
2. a more fine-grained search around best parameters found on step 1.

The best model found on the first step showed mean validation accuracy 0.80 and mean log loss 6.85, which are better than values achieved by Logistic Regression models. On the second step, the random search found slightly different parameters, but the final mean validation log loss and accuracy remained the same.

Convolutional Neural Network

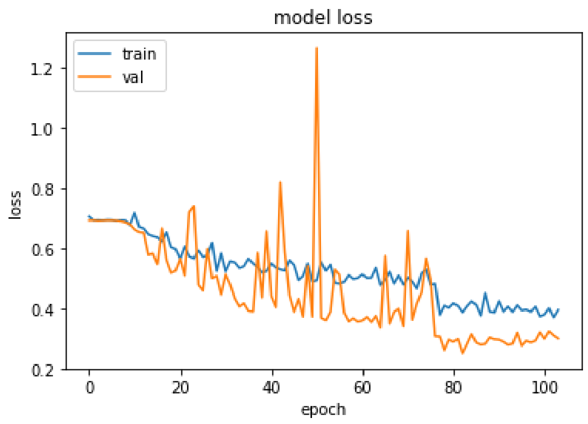
The naive implementation includes 3 convolutional layers with ReLU activation function alternated by max pooling and dropout layers (with low dropout probability). The classifying part consists of 3 dense layers (2 with ReLU and output one with a sigmoid activation) alternated with extensive dropout. The overall CNN architecture can be seen in the picture below.



The naive CNN architecture.

In this case, the images from the 2 bands were stacked together as 2 channels of a color image (RG). The third channel was faked and filled with corresponding incidence angle.

As soon as given dataset is relatively small, the image augmentation technique (ImageDataGenerator in Keras [1]) was applied to it with following image transformations: shift, shear, rotations, flips and zooming. The validation dataset remained in its original form. With a constant learning rate, log loss started to fluctuate significantly at the end of the training process when it is approaching its minimum. To fix this, I tried to reduce learning rate on plateaus by a factor of 0.1, using the ReduceLROnPlateau callback from Keras [2]. This technique helped to avoid these “jumps” and achieve better final log loss.



Log loss history for naive CNN model.

This model achieved test log loss of 0.2995 on Kaggle platform.

[1] Image Augmentation for Deep Learning With Keras:
https://machinelearningmastery.com/image-augmentation-deep-learning-keras/
[2] ReduceLROnPlateau: https://keras.io/callbacks/#reducelronplateau

Refinement

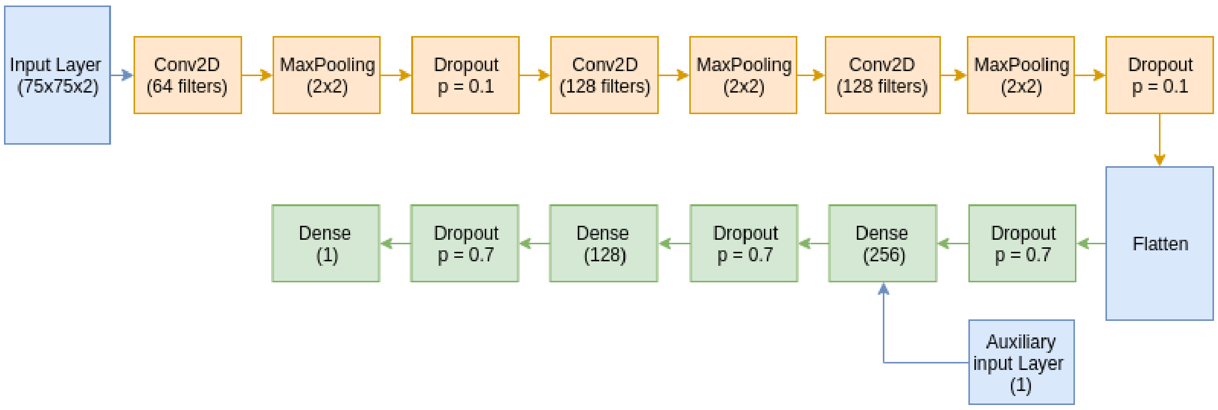
Support Vector Machines

The PCA transformation was applied in the pipeline in prior to SVM. This approach, however, has not demonstrated any improvements. The best-found models showed mean validation accuracy 0.61 and log_loss 13.41.

Convolutional Neural Network

Auxiliary input

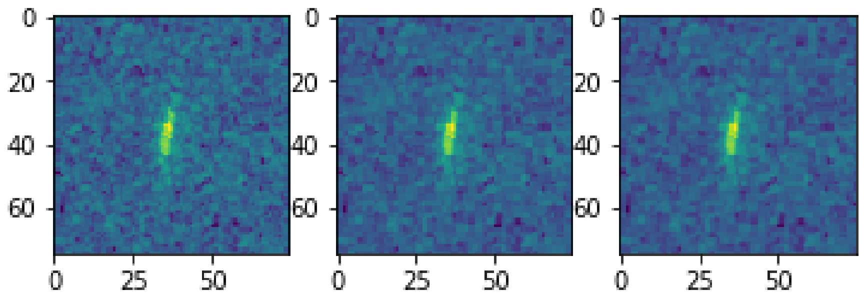
The incidence angle is not part of the image, but some extra information which might be used directly during classification, bypassing convolutional layers. Following architecture represents this idea (auxiliary input is an incidence angle, main input are 2 bands stacked as 2 channel image):



Besides this change, the same training techniques as for naive model were applied in this case. This architecture achieved test log loss 0.3073 on the Kaggle platform.

Additional dataset transformations

As soon as all CNN models achieved similar results, probably, the reason for the poor performance is in the dataset. Among possible issues might be the size of the dataset or the quality of images. In this project, I have tried to apply to images wavelet denoising filter with a successive gaussian smoothing (sigma=0.2).
Then, the naive CNN model was retrained on the preprocessed images. This model achieved test log loss 0.2947 on the Kaggle platform. Moreover, the naive model was retrained on combined (extra) dataset with original and filtered data (with different degrees of smoothing, sigma = 0.2 and sigma = 0.4) altogether. This model achieved test log loss 0.2821 on the Kaggle platform.
The applied transformations could be seen on the image below:



Transformations applied to the image (only one channel is shown). From left to right: original image, wavelet denoising + gaussian smoothing (sigma = 0.2), wavelet denoising + gaussian smoothing (sigma = 0.4)

Transfer learning

Transfer learning is a helpful technique in the case when a dataset is not large enough. However, the Statoil dataset is completely different from the ImageNet used for training VGG16 neural network (and other networks, available for loading in Keras). Thus, this weight transferring is not very beneficial for this challenge, especially if only the last few layers are retrained on the new dataset. Thus, I decided to retrain the full network in two steps:

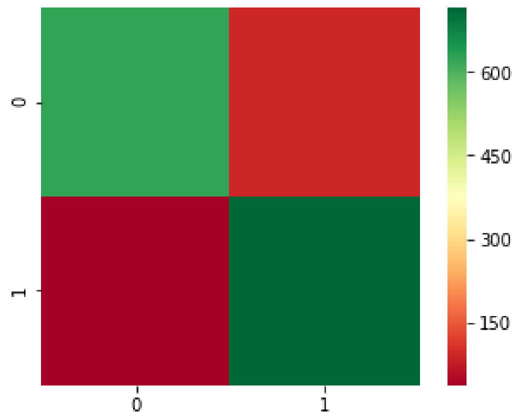
- Firstly, new fully connected layers were added to the network and retrained on the new dataset during 25 epochs
- Secondly, all layers were “defrozen” and retrained on the extra version of the dataset, which includes also the additionally preprocessed images.

The model with transferred weights achieved 0.2275 log loss on the test set.

[1]. K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition

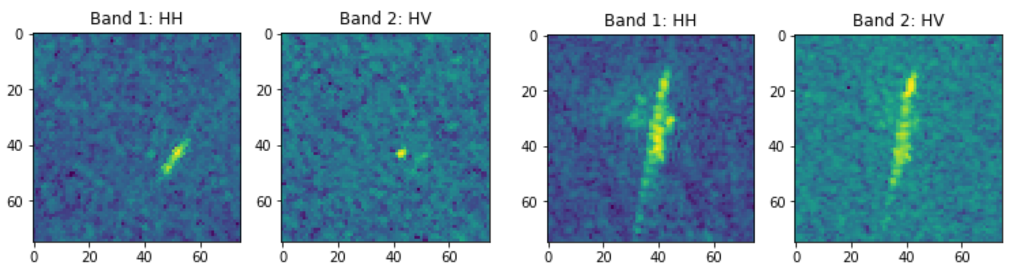
IV. Results

Considering all mentioned above, it could be seen that the best result was achieved by deep convolutional neural network with transferred (from VGG16) weights and trained on the extended dataset. This model showed log loss of 0.2275 on the Kaggle platform. On the validation set this model showed log loss 0.1854 and accuracy of 0.9202.
The confusion matrix built for this model tested on the whole dataset including validation and training set is shown below:

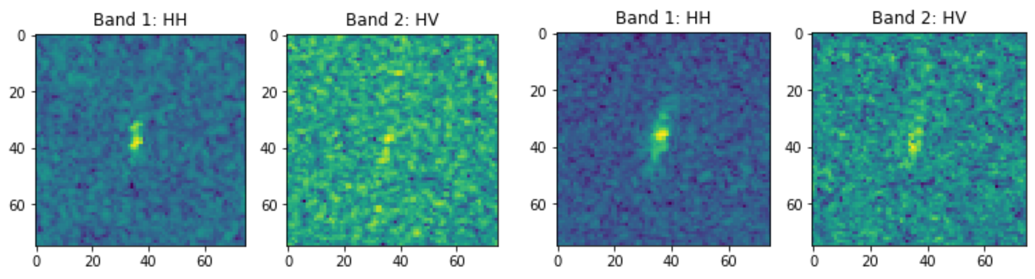


The confusion matrix: the expected value is on the x axis, the predicted value is on the y axis. 0 stand for ships, 1 for icebergs.

The examples of misclassified images (from the training and validation datasets) could be seen below:



Misclassified icebergs



Misclassified ships

Surely, the result shown by this CNN is significantly better, comparing to the results of baseline models, the best of which showed log loss 9.46 only (logistic regression with a preliminary Principal Component Analysis). However, the winner of the competition showed an impressive log loss of 0.0822. So, definitely the model shown here could be improved even more. For example, following improvements could be tried:

1. further adjustment of the dataset by applying of different denoising and smoothing filters like was done with a wavelet filter
2. an ensembling technique which can combine predictions made by several classifiers: for example, voting classifier that can learn a relative importance of individual models
3. another architecture: for example, it might advantageous to try model with three different inputs - one for each band and one for incidence angle.

V. Conclusion

Reflection

Within this project, different techniques were tried in order to address iceberg detection challenge suggested by Statoil on the Kaggle platform: logistic regression with and without PCA (results were used as a baseline for the further algorithms), support vector machines and convolutional neural network, which showed the best performance.
The main restriction of the project was the fact that given dataset is quite small so that even with an image augmentation techniques it was difficult to improve the predictive power of used CNN architectures. The additional adjustments to the dataset helped to solve this problem. The extended dataset in combination with transferring weights from the VGG16 network gave a promising result with log loss of 0.2275 on the Kaggle platform.
This result, however, is still could be largely improved, as the winner of the competition showed a result of 0.0822, most likely, using ensemble techniques.