Iceberg detection from satellite images

Capstone Project | Machine Learning Engineer Nanodegree
Cornelius Krull | July 5th, 2018 | cornelius.krull@mailbox.org

1. Definition

1.1. Project Overview

Drifting icebergs pose a threat to navigation and shipping activities in remote areas e.g. the east coast of Canada. Currently, aerial reconnaissance and shore-based support are used to assess the risks from icebergs. However, due to bad weather conditions, these methods are not feasible, and satellite based monitoring is the only viable option[1], commonly used today[2]. One such system is the Sentinel-1 satellite constellation. It uses a Synthetic-aperture radar (SAR), operating in the C-Band providing data in any weather condition during day and night[1], [3] Satellite radar bounces an electromagnetic signal off an object and records the echo (also called backscatter), which is then 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. When the radar detects an object, it is not immediately clear what caused the signal: an iceberg, a ship or any other solid object. The backscatter image needs to be analysed for certain characteristics - shape, size, brightness,etc – to determine the type of the detected object. Humans can visually classify these type of images, but satellite data typically contain hundreds of objects making this a time consuming and inefficient process.

Automated iceberg detection from SARs satellite data is a problem that has attracted academic interest, and over the past 20 years several methods have been employed and proposed: These range from simple threshold based methods looking at individual pixels, which are affected by weather conditions, to sophisticated models that cluster neighbouring pixels into objects[2]. Recent (2017) scientific publications on SARs imaging report a 93.2% accuracy of iceberg vs. background detection, and a 90% classification accuracy for maritime objects (ship, icebergs, sea ice) in general[4].

1.2. Problem Statement

This project aims to use Machine Learning approaches to address the problem of ship-iceberg discrimination, based on the SAR data set provided by the Statoil/C-CORE Iceberg classifier challenge[1]. The objective is to classify image data by assigning a probability (between 0 and 1) if an iceberg is present or not.

1.3. Metrics

The logistic regression cost function *log loss* is the main metric for the model. It is defined with the predictions $\hat{p}_{(i)}$ and the ground-truth $y_{(i)}$ for all *m* data point as a bench-marking measure:

$$J(\Theta) = \frac{-1}{m}\sum_{i=1}^{m}\left[y_{(i)}\log\left(\hat{p}_{(i)}\right) + \left(1-y_{(i)}\right)\log\left(1-\hat{p}_{(i)}\right)\right]$$

In addition to the *log loss* function, other metrics common for binary classifiers were used to select a good model. Namely, the *accuracy* and the *receiver operating characteristic (ROC)*. *Specificity* and *Recall* are derived from the confusion matrix and are defined as follows:

$$Recall\,(true\ positive\ rate) = \frac{true\ positives}{(true\ positives + false\ negatives)}$$

$$Specificity\,(true\ neagtive\ rate) = \frac{true\ negatives}{(false\ positives + true\ negatives)}$$

The *ROC* curve plots *Recall* against the false positive rate(=1 − *Specificity*), by varying the threshold which determines which label is assigned to a given instance. The area under the *ROC* curve (*AUC*), then is a measure of the quality of the model. A perfect model would have a *AUC ROC* score of 1 and a purely random one 0.5. Importantly, this metric shows that there is a trade-off between a high Recall and low false positive rate and how strong it is for a given model.

Accuracy is defined as follows:

$$Accuracy = \frac{correctly\ labeled\ instances}{total\ number\ of\ data\ points}$$

It can be a problematic metric if the label distribution is heavily skewed towards one label. For instance if 90% of a dataset have the same label, then always guessing that label will results in an *Accuracy* of 90% without any model. This dataset has balanced labels (see next section), hence *Accuracy* is a good (and intuitive) measure of the performance of the model.

2. Analysis

2.1. Data Exploration

The dataset is provided by the Kaggle competition. It consists of 75x75 SAR image data with two channels produced from different polarisation of the analyser during the recording of the image: HH (transmit and receive horizontally) and HV (transmit horizontally and receive vertically). The different channels play an important role to determining the object characteristics, as different objects tend to affect the polarisation of the reflected energy differently (See Figure 1)[1]. Several factors affect the backscatter of the ocean or background area, for instance high or low winds will generate a brighter or darker background, respectively. The Sentinel-1 satellite is a side looking radar. This means it sees the image area at an incidence angle, affecting the ocean background which will be darker at a higher incidence angle[1]. The incidence angle is provided for most of the training data, and for all of the testing data. The instances are labelled by human experts with geographic knowledge of the target. The data is presented in json format. The files consist of a list of images, and each image has the following fields: `id, band_1 (the HH data), band_2 (the HV data), inc_angle, is_iceberg.`

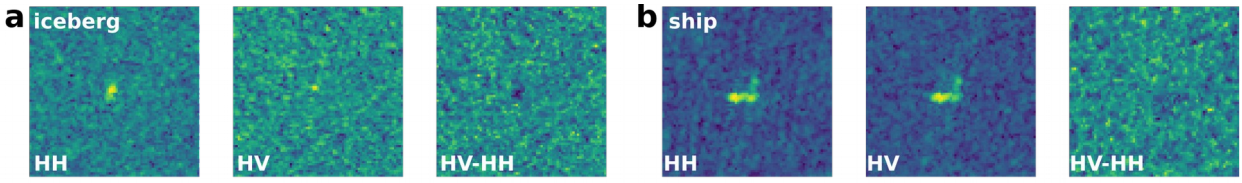

Figure 1: *Image representations of two data instances representing an iceberg* ***a*** *and a ship* ***b****. The two channels are plotted as HH and HV. The phase channel (HV-HH) is typically used in SARs image analysis.[5]*

Importantly, the band_1,band_2 values are not non-negative integers used in images but instead reflect the physical loudness of the received signal for each pixel, which is measured in decibel [dB] and represented as a float. The competition provides a small but balanced training data set (1604 labelled training instances consisting of 46.9% iceberg objects) and a larger testing set. The testing set is partially extended based on augmented data instances (see below) to avoid hand labelling the instances in the competition[1].

Set	Size	Band 1 (HH)	Band 2 (HV)	Incidence angle	is iceberg
training	1604	75x75 image (numeric)	75x75 image (numeric)	1 numeric (8.29% missing)	46.9% icebergs 53.1% ships
testing	8424	75x75 image (numeric)	75x75 image (numeric)	1 numeric (0% missing)	-

Table 1: *The composition of the datasets available from the Kaggle competition.*

2.2. Exploratory Visualisation

The dataset consists of mainly image data, that means each pixel is a feature that can be used to predict the label. Figure 2 show the importance of the individual pixel as determined by fitting a Random Forest classifier. The data set is well centred and the most important pixel/features lie in the middle of the image. This means data augmentation, which involves operations that potentially crop or otherwise affect the border (see Data pre-processing below), can be applied. The incidence angle feature becomes important after applying a noise reduction filtering on the image data, its importance then lies above those of most important individual pixels (0.0055).

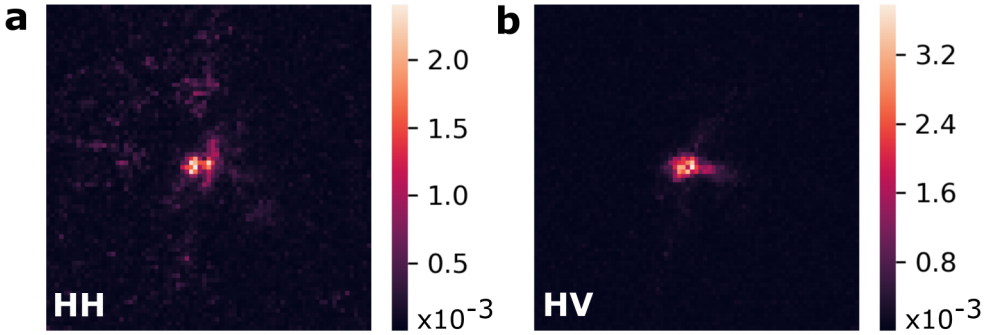

Figure 2: Importance of individual pixel for the two bands. Determined by fitting a Random Forest (sklearn,N=1000).

Additionally, we calculated various statistical moments (mean, variance, skewness, kurtosis) to explore correlation between image data and target labels. Table 2 shows the correlation coefficients for these measures. Several features are correlated significantly, the band 2 kurtosis reaches -0.45. The data can be thus used to train Machine Learning algorithms, to predict the target label. Theses correlation are visualised in Figure 3, where it can be seen that certain values of the statistical moments can only be found for one type of label (e.g. high values for band 2 kurtosis and skewness only occur for ship instances). We also observe that the missing data for the incidence angle feature is exclusively found for iceberg instances.

Incidence angle	Band 1 mean	Band 1 variance	Band 1 skewness	Band 1 kurtosis	Band 2 mean	Band 2 variance	Band 2 skewness	Band 2 kurtosis
0.27	-0.17	-0.21	-0.22	-0.3	-0.12	-0.31	-0.38	-0.45

Table 2: *Pearson correlation coefficient between the target label and various statistical moments of the image data, and the numeric feature incidence angle.*

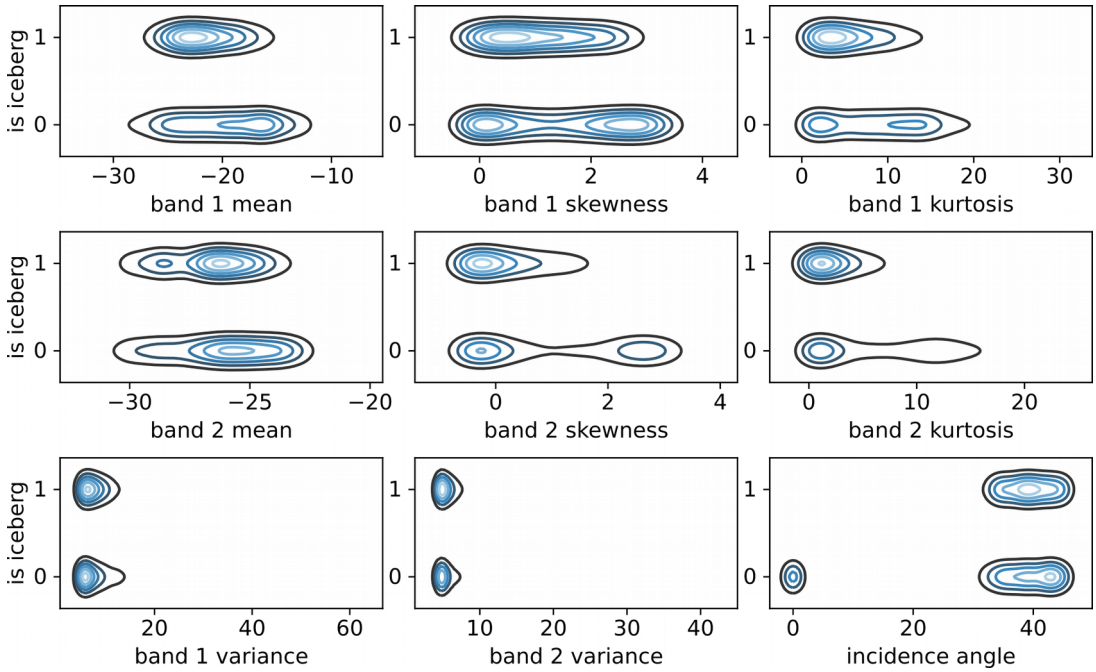

Figure 3: *Exploration of the correlation between the incidence angle feature, various statistical moments of the image data with the target labels. Note that the missing incidence angle values were set to 0, only instances with the label iceberg had missing data.*

2.3. Algorithms and Techniques

A variety of different techniques were used to solve the SAR data Iceberg classification task. In the following we will briefly outline the concepts and the rationale for their use:

Convolutional Neural Networks: The problem requires the classification of images, a task for which convolutional neural networks (CNNs) have become a very powerful approach in recent years. Especially the increase in computational resources (Graphic Processing Units (GPUs)), and advanced training methods for deep neural networks have firmly established CNNs for a variety of different tasks[6], in particular complex visual classification tasks like the ImageNet challenge[7] or the present project. Current deep learning frameworks such as Keras[8], allow for a straightforward implementation of CNN architectures. These networks can be trained to output the probability that a given input image belongs to a certain class (In this case this would mean if the image contains an iceberg.) CNNs form the base of this project.

Transfer learning: One of the main challenges of this project is the small amount of training data available. This makes training complex and extended neural networks difficult. A transfer learning strategy uses pretrained CNNs, for instance networks trained on the ImageNet data set (VGG16/19[7]available from Keras[8]), only changing the top fully connected layers to adapt the output to the given task. This is a very powerful approach that has been shown to work well even for images/data and classes not present in the original training data, for instance visual skin cancer classification based on ImageNet CNNs[9]. In this project two transfer learning approaches were applied[10]: (i) training a densely connected network with the output of the pretrained convolutional part applied to the training data, the *bottleneck features.* And (ii), by retraining the last convolutional block (which is initialised with the pretrained weights from step (i)) together with the last densely connected layer, while all other layers are kept constant. Note that it is crucial to use a trained densely connected model at the top of the network to avoid destroying the pretrained weights in the convolutional block with large gradients created by a random initialisation.

Data Augmentation: Another method of training models when only limited amounts data is available, is to augment the existing data to create more instances. The principle is to modify (rotate, shear, flip, shift,etc) the image data in such a way that it still shows realistic depictions of the original class. For in-

stance a flipped image of a cat still shows a realistic image of a cat. This is a regularisation technique that will prevent overfitting -especially on small datasets- and lead to more generalising models . Many frameworks provide classes that will perform image augmentation, this project used the Keras implementation.

Ensemble learning: A common technique to improve the performance of a model, is to combine the output of several other models.[6] For classification tasks, this can be done by majority voting using the predictions of all ensemble members (hard voting). Alternatively, by averaging the probabilities of the predicted class slightly better results can be achieved (soft-voting). We employed soft voting in this project.

K-fold cross-validation: This is a method used to evaluate the performance of a model without the need the create a separate validation datasets. The training data is split into *k* batches, and *k-1* of these batches are used to train the model, the last batch is used to validate/score the model. This is repeated *k*-times each time with a different validation set. The final score of the model is the average of the *k* scores obtained in the loop. This method is computationally expensive but makes better use of the available data. Using *k*-fold cross validation ensures that model and hyperparameters are not selected on just one validation set. Because it uses the average error rate over the *k* validation sets, it selects those parameters/model that best generalise. Typically this approach is not applied in Deep Learning Models, as it would be very computationally expensive with the large dataset that are usually used in such cases. In this project however, owing to the small datasets, it is not only viable but also helps to determine to model that best generalises.

Early stopping: An effective way to avoid overfitting is to stop training a model once its performance on the validation set decreases again. This is called early stopping, and is used extensively throughout this project.

3. Benchmark

The Kaggle competition provides an extensive testing set (8425 data points) to evaluate participants´ models. Candidates can submit label predictions (that is the probability for each image to contain an iceberg) from their models Θ. And the Kaggle website calculates the *log loss* (See above) of these predictions, and a placement in the scoreboard (based on approximately 80% of the test data). This value allows determining the performance of the model. The competition finished recently (6 month ago) and had over 3000 contestants, the winning entries thus represent an fine-tuned state-of-the-art solution for this particular problem[1]a solid placement in the middle field would already indicate a viable model (the 25,50,75 percentile of the *log-loss* score on the leaderboard are 0.167, 0.204,0.28). Note that, the competition experienced a data leak [11], where it become known that instances with the same incidence angle have the same label in both the test and the train sets. This knowledge enables a dramatic improve on the competition score, which however would not be applicable in a real world setting. It can be assumed that most entries in the competition made use of this leak. To gauge its impact we also created a fine tuned solution. (see `6_Soft_voting_ensemble.ipynb`)

4. Methodology

4.1. Data pre-processing

The datasets needed several pre-processing steps:

- **Noise Reduction:** Usually SARs image data contains speckle noise, this is detrimental to the performance of the evaluated CNNs. Hence the image data was smoothed with either a modern edge preserving smoothing filter: guided filter[12] or a filter, typically used in SAR-based remote sensing applications called Lee filter[13] (see Figure 3). Note that in addition to the two filters above, several other filtering approaches were explored (see `1_Data_Exploration.ipynb`), these included simple Gaussian smoothing, and advanced filters from the scikit-image library.

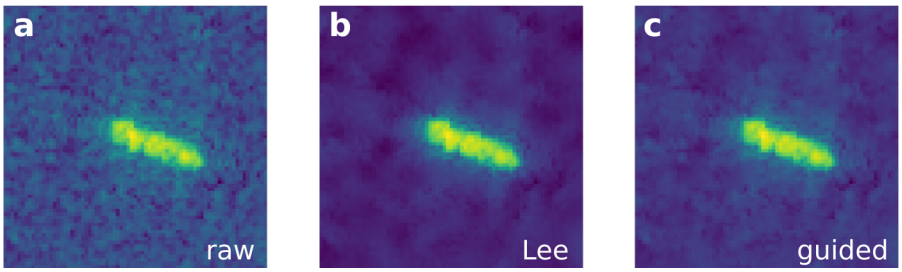

Figure 3: *Image smoothing to remove speckle noise:* ***a*** *raw HH data,* ***b*** *Lee filtered, (window size 7) this type of filter is typically used in SAR-based remote sensing applications*[13]*,* ***c*** *guided filter a modern edge preserving image smoothing technique*[12].

- **Scaling the data:** Rescaling the data to values between 0 and 1 either on a per batch and channel or per individual image range was also implemented.
- **Creating a third channel:** CNNs trained on the ImageNet dataset, expect RGB type images (that is with 3 channels), however the dataset has only 2 channels per instance. To address this we added a third derived channel consisting of the HV-HH signal. This phase channel is typically used in SARs data analysis[5].
- **Treat instances with missing data:** The training set had 133 missing data points for the incidence angle feature. For models that considered this feature this data instances were removed frin the training set. Models that only considered the image data (all CNNs), were used the entire training set with the incidence angle feature removed.

4.2. Implementation

The Iceberg classifier was implemented in Python:

- **Data loading and pre-processing filters**: The loading formatting of the dataset relies on standard functionalities from the Pandas and Numpy Libraries. The Guided Filter, used a implementation from github[14], and the Lee Filter is based on the original paper [13] and a stackoverflow post [15]. These implementations can be found in `helper.py`.
- **The Models**: The Keras library (with a Tensorflow backend), was used for the implementation of CNNs with varying degrees of complexity:
  - simple CNNs based on increasing number of convolutional filters per layer (16,32,64), and final fully connected layer. The precise implementation allows for fine tuning these parameters as well as adding batch-normalisation layers[16] (see `statoil_models.py`)
  - Keras was also used to implement transfer learning CNNs, based on several models (VGG16/19 [7], Xception[17]) from the ImageNet competition. (see `statoil_models.py`)
  - Other machine learning based models (e.g. Random Forest, K-nearest neighbors) were implemented using Scikit-Learn[18] (see `4_Training_Scikitlearn_models.ipynb`)
- Since **k-fold cross validation** is not typically used for deep learning models, we had to implement a version creating a stratified split of the training data (based on the sci-kit learn function `StratifiedKFold`). This also allowed the incorporation of an early stopping regularisation approach into the cross validation. (see `kfold_keras.py`)
- The **soft voting** approach used to combine the best performing model was straightforward to implement by just averaging the predicted probabilities of the involved models (see `6_Soft_voting_ensemble.ipynb`).

- Additionally, several other small functions needed to be implemented/imported: The Savisky Golay smoothing is based on the Scipy-Cookbook [19] (`helper.py`).

4.3. Refinement

To optimise the performance of the model, a range of hyperparameters needed to the tuned or selected. Initially, and typically for deep learning, only an *early stopping* method, which used *one* randomly selected validation set was used to do this. However, due to the small size of the dataset, this resulted in models that did not sufficiently generalise. To address this issue, we used a selection process based on *k-fold-crossvalidation* (k=5), which mitigated the impact of small validation set by averaging over *k* sets. Based on this validation method, a gridsearch was used to evaluate many possible parameter combinations, and determine the best model. The metrics used were *Accuracy*, *AUC ROC*, and the *log loss* function. The following parameters were considered:

- **Preprocessing**: The pre-processing parameter considered were the type of speckle filtering(Lee filter or Guided filter), and whether or not the image data was scaled prior to traning the networks. (see `2_Training_simple_CNN-k-fold-CV.ipynb`)
- **Model architecture smaller CNN from scratch:** Model width and depth were varied, that is the number of nodes per layer and the total number of fully connected layers. Additionally batch normalisation was considered as on option. Finding the right optimiser and learning rate was not straightforward, initially too high training rate lead to models not training well or not at all. Finally an Adam optimiser with a learning rate of 0.0005 was used (see `statoil_models.py` and `2_Training_simple_CNN-k-fold-CV.ipynb`).
- **Early stopping patience:** Finding an adequate value for the number of episodes to wait before declaring a model as fully trained was not unambiguous, too low settings let the training stopping already after a few(3-5) episodes.
- **Model architecture transfer learning:** The network architecture from the ImageNet competitions was selected from VGG16/19 and Xception (see `3_Keras_transfer_learning.ipynb`)
- **Data Augmentation:** To improve the performance and create a more generalising model, the best models were also trained with augmented data. To find the best augmentation parameters we performed a grid search with the best models (see `5_Data_Augmentation.ipynb`)
- **Final training for CNNs**: To obtain a trained model after it was selected using *k-fold cross-validation* and *early stopping,* we kept the *k* models trained during the cross-validation and created a soft-voting ensemble, as the final model. Due to the small dataset, this approach should yield more generalising models, compared to just retraining the model with a single validation set and early stopping (see `kfold_keras.py`).

5. Results

5.1. Model

The final model was build by combing the best models from the previous sections through a soft voting ensemble, this is the probability that an instance is an iceberg is calculated by taking the mean of the predictions of all participating models (See `6_Soft_voting_ensemble.ipynb`). The contributing models were:

- a small CNN consisting of 3 convolutional layers (3x3 kernels), and a 256 node fully connected layer with batch normalisation.
- a VGG16 based CNN pretrained on ImageNet dataset, where the last convolutional block and a fully connected layer (256 nodes) were retrained.
- A random forest classifier (3000 estimators, max depth 20), using the incidence angle and extracted features from the image data like mean, variance, skewness and kurtosis

ID	Model	Accuracy	AUC ROC	Validation loss
A	small CNN	86%	0.95	0.308
A_{aug}	(augmented)	86.5%	0.949	0.309
B	VGG16	90.8%	0.969	0.234
B_{aug}	(augmented)	90.9%	0.969	0.234
C	Random Forest (image data)	79.4%	0.877	0.480
D	Random Forest (extracted features + inc.angle)	91.1%	0.954	0.282

Table 3: *Performance of the best models, using various metrics, obtained using a 5-fold crossvalidation, with early stopping.*

5.2. Data

Best performance was achieved using the guided filter, without any scaling of the original data. The CNN based classifiers use 3 channel input data (HH,HV,HV-HH), while the random forest classifier uses only extracted features. The performance of the CNN does not depend significantly on the type of filter used for the pre-processing. The Guided filter, the Lee filter, and unfiltered data perform similar. Scaling the data however leads to a slight loss in performance. Equally the impact of data augmentation is minor (see `2_Training_simple_CNN-k-fold-CV.ipynb`). Both these observations indicate that the models are robust against small changes in the dataset, and generalise well.

5.3. Final Evaluation

The resulting soft-voting model was tested against the extensive test dataset from the Kaggle competitions and archived a metric of 0.266 which corresponds to a position of 2486 (72 percentile) (see `6_Soft_voting_ensemble.ipynb`) The competition suffered from a data leak, showing a strong correlation between the inc_angle feature and the is_iceberg label, instances with the same feature valued tend to have the same label. To gauge the impact of the data leak we also tried to improve the prediction of the final model. This did indeed lead to an improvement of the score to 0.247 (placement 2338, that is an increase of 148 positions, 67 percentile). If one assumes that most entries employed the knowledge of the data leak than this is the score to be used to benchmark the model.

This placement in the lower mid field indicates that the model elaborated in this project works and predicts reasonably well the presence of an iceberg in a given data instance, compared to the top contenders in this competition.

6. Conclusion

6.1. Free-Form Visualisation

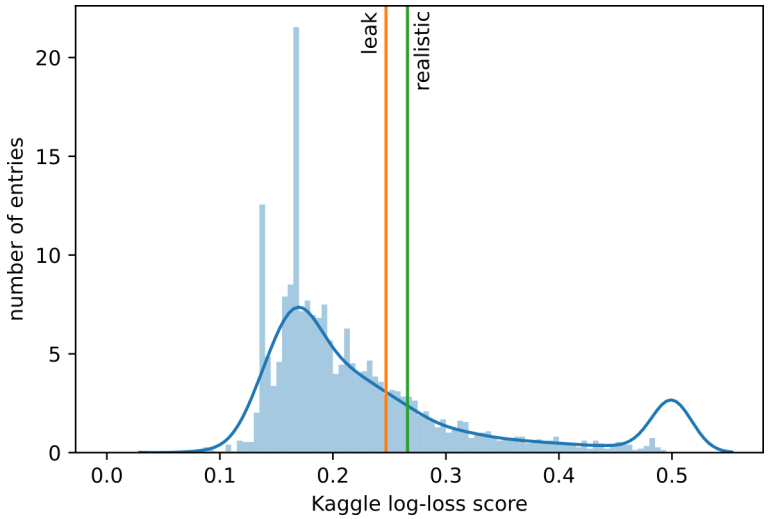

Figure 4: *Placement of the final soft voting ensemble in the Kaggle competition. A histogram of the log-loss scores of contestants is shown. The scoring of the best models of the project is indicated, for a realistic treatment of the data, and for a treatment that includes the data leak Scores above 0.5 are grouped together.*

6.2. Reflection

Here we presented an approach to use machine learning to detect icebergs from satellite images. We use a small dataset from a Kaggle competition as a basis. The labels within this set are balanced, thus avoiding problems that skewed data would create. On the other hand, the small number of instance amplifies the risk of creating models that do not generalise well, i.e. that overfit on the training data. The labels are provided by human experts and the raw data consists of dual channel SARs data (HH,HV polarisation), and the incidence angle at which the radar image was recorded. After a noise reduction pre-processing of the images this data was used to train several different classification algorithms. To address the small size of the dataset we used a k-fold crossvalidation approach, and typical classification metrics (*Accuracy*, *log loss* and *AUC ROC*) to evaluate their performance. The best algorithms are based on partially retraining complex publicly available Convolutional Neural Networks used to classify photographs of a range of different object (ImageNet). Smaller CNNs designed from scratch performed only showed a slightly reduced performance, followed by Random Forest classifiers working only on the incidence angle and statistical moments(mean,variance, skewness, kurtosis) extracted from the image data. In a final step these trained models were combined by averaging their output probabilities (soft-voting). The ensemble classifier was then bench-marked against the testing dataset from the Kaggle competition obtaining a position in the lower mid field of contenders.

6.3. Improvement

Two main points could be addressed for further improvement: (i)The transfer learning approach did not perform well with the Xception architecture, and equally many other ImageNet models were not used due to input data shape constrains. This does however not mean it is impossible to use these model in for the Iceberg detection task. On one hand the images could be upscaled so that other ImageNet models could be used (e.g. ResNet), and probably the the Xception training might be fixed by fine tuning both transfer training steps.

(ii) The target label in this dataset was created by human experts, with knowledge of the subject and the

area, this does however not mean that it contains no mistakes, as some of the data point might be ambiguous, and the labelling might be subjective. Due to the small training dataset these noisy labels could severely impact the performance of any classifier. Recent literature outlines a method to address this, by augmenting the prediction objective with a notion of consistency. This means that data instances that have similar network responses should have similar label[20]. Such an approach might have further improved the performance of the models employed for the Iceberg classification, especially in light of the large distance of this models benchmark to the best solution submitted to the Kaggle competition (score: include).

---

[1] "Statoil/C-CORE Iceberg Classifier Challenge." [Online]. Available: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge. [Accessed: 09-Jun-2018].

[2] A. K. Mazur, A. K. Wåhlin, and A. Krężel, "An object-based SAR image iceberg detection algorithm applied to the Amundsen Sea," *Remote Sens. Environ.*, vol. 189, pp. 67–83, Feb. 2017.

[3] "Sentinel-1," *Wikipedia*, 30-Apr-2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sentinel-1&oldid=839052136. [Accessed: 09-Jun-2018].

[4] P. Heiselberg and H. Heiselberg, "Ship-Iceberg Discrimination in Sentinel-2 Multispectral Imagery by Supervised Classification," *Remote Sens.*, vol. 9, no. 11, p. 1156, Nov. 2017.

[5] P. Ferrazzoli, L. Guerriero, and G. Schiavon, "Experimental and model investigation on radar classification capability," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 2, pp. 960–968, Mar. 1999.

[6] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1 edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly Media, 2017.

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv14091556 Cs*, Sep. 2014.

[8] "Keras Documentation." [Online]. Available: https://keras.io/. [Accessed: 09-Jul-2018].

[9] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, Feb. 2017.

[10] F. Chollet, "Building powerful image classification models using very little data." [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html. [Accessed: 09-Jul-2018].

[11] P. Pleskov, "Try to improve your solution with the leak." .

[12] K. He, J. Sun, and X. Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[13] J.-S. Lee, "Speckle analysis and smoothing of synthetic aperture radar images," *Comput. Graph. Image Process.*, vol. 17, no. 1, pp. 24–32, Sep. 1981.

[14] swehrwein, *python-guided-filter: Numpy/Scipy implementation of the (fast) Guided Filter*. 2017.

[15] "Speckle (Lee Filter) in Python," *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/39785970/speckle-lee-filter-in-python. [Accessed: 17-Jul-2018].

[16] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," p. 11.

[17] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *ArXiv161002357 Cs*, Oct. 2016.

[18] "scikit-learn: machine learning in Python — scikit-learn 0.19.2 documentation." [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 17-Jul-2018].

[19] "SciPy Cookbook — SciPy Cookbook documentation." [Online]. Available: http://scipy-cookbook.readthedocs.io/. [Accessed: 17-Jul-2018].

[20] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training Deep Neural Networks on Noisy Labels with Bootstrapping," Dec. 2014.