Capstone Proposal - Iceberg detection

Cornelius Krull

Domain Background

Drifting icebergs pose a threat to navigation and shipping activities in remote areas of the planet i.e. the east coast of Canada. Currently, aerial reconnaissance and shore-based support are used to assess the risks from icebergs. However, in remote areas 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 a signal off an object and records the echo* (or backscatter)*, 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. When the radar detects an object, it can't tell an iceberg from a ship or any other solid object. The object needs to be analysed for certain characteristics - shape, size and brightness - to find that out.* [1] These objects can be visually classified by humans, however satellite images typically contain hundreds of objects making this a time consuming 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/proposed: These range from simple threshold based methods looking at individual pixels, which are affected by weather conditions, to sophisticated models that cluster neighboring 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]

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.

Datasets and Inputs

The dataset is provided by the Kaggle competition. It consists of 75x75 SAR image data with two channels produced from different polarizations: 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 reflect 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 data is labelled provided by human experts and geographic knowledge off the target, however some of the training data might be mislabelled. 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, band_2, inc_angle, is_iceberg.`

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, this 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 47% iceberg objects).

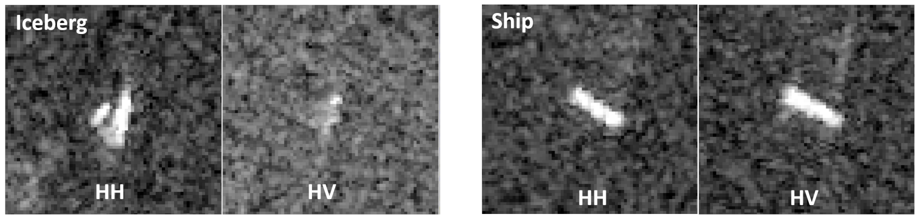

Figure 1: Image representations of two data instances representing an iceberg and a ship. The two channels are plotted as HH and HV.

Solution Statement

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[5] have firmly established CNNs for a variety of different tasks, in particular complex visual ones like this project. Current deep learning frameworks such as `Keras`, allow for a straightforward implementation of CNN architectures that have successful in image classification tasks. 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.)

Benchmark Model and Evaluation Metrics

The Kaggle competition provides an extensive testing set (8425 data points) to evaluate participants´ models. Candidates can submit a label prediction (that is the probability for each image to contain an iceberg) from their models Θ. Kaggle then provides the logistic regression cost function *log loss* (See Equation 1) between the predictions $\hat{p}^{(i)}$ and the ground-truth $y^{(i)}$ for all m data point as a benchmarking measure.

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

This value can be used to compare the proposed CNN model in this project with the scoreboard. Since the competition finished recently (5 month ago) and had over 3000 contestants, it is safe to assume the winning entries represent the extremely fine-tuned state-of-the-art for this particular problem.[1] A solid placement in the middle field would already indicate a viable model.

In addition to the log loss function, other metrics common for binary classifiers will be used. Namely *precision/recall* and the receiver operating characteristic (ROC). These values are derived from the confusion matrix, that is the number of true and false positives and negatives, and are defined as follows:

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

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

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 curve (AUC), then is a measure of the quality of the model. A perfect model would have a ROC AUC score of 1 and a purely random one 0.5.

Project Design (approx. 1 page)

The project workflow will follow this outline:

a) Explore the data:

In this first step I will explore the data, identifying missing data, and statistical distributions. Here I will visualize the images and, determine noise in the image data. If possible I will explore different type of pretreatment filters, based on the SAR literature[2], and on general image treatment (Gaussian smoothing, edge detection, etc).

b) Prepare the data

The main objective in this step is to implement a preprocessing pipeline, preparing the data for training. It includes all necessary steps (including any promising pretreatment steps identified in step a) :

i. Critically, the image data for this project is not normalized, I will identify a suitable normalization technique, to make data instances comparable (e.g. Scikit standard scaler).

ii. Additionally the missing values (incidence angle) will need be handled that is dropped or filled. (Note as there is a correlation between the background brightness and the incident angle this feature might also be extracted.)

iii. Due to the relatively small data set Data augmentation of the training images will be beneficial, specifically each data instance is duplicated by creating an image that is shifted, zoomed in/out, rotated, flipped, distorted, or changed in overall intensity level. Both image and duplicate are then used to train in the model.

c) Train Models

In this step, I will train CNNs with the training data. A validation set of the training data will be set apart, to evaluate performance during training and to avoid overfitting (using an Early Stopping approach, this means training will be stopped once performance on the validation set decreases with additional training). I aim to train three types of CNNs:

i. Using existing CNN architectures such as ResNet50 and VGG16/19 which are already implemented in the Keras library.
ii. A transfer learning approach[6] that uses CNNs pretrained on the ImageNet data set (also available from Keras), by slightly adapting the input, and output layers to the requirements of the SARs data these networks could yield high performance improvements.
iii. Experimenting with custom CNNs architectures, I will explore batch normalization and dropout layers.

d) Fine Tune/Ensemble

Once I have identified several promising models, I will try to combine the best models into an ensemble. Once I am confident about the model I will evaluate its performance on the testing set.

e) Additional Considerations

If the performance of the CNNs is initially not satisfactory, this could be due to noisy training label. In that case, I will explore noisy label treatment reported in literature: bootstrapping based on updated target labels[7] and/or adding additional noise layers.[8] These techniques have been reported to significantly improve performance when training Deep neural network with noisy target labels.

1. Statoil/C-CORE Iceberg Classifier Challenge. Available at: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge. (Accessed: 9th June 2018)
2. Mazur, A. K., Wåhlin, A. K. & Krężel, A. An object-based SAR image iceberg detection algorithm applied to the Amundsen Sea. *Remote Sens. Environ.* **189,** 67–83 (2017).
3. Sentinel-1. *Wikipedia* (2018). Available at: https://en.wikipedia.org/w/index.php?title=Sentinel-1&oldid=839052136. (Accessed: 9th June 2018)
4. Heiselberg, P. & Heiselberg, H. Ship-Iceberg Discrimination in Sentinel-2 Multispectral Imagery by Supervised Classification. *Remote Sens.* **9,** 1156 (2017).
5. Géron, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. (O'Reilly Media, 2017).
6. Esteva, A. *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542,** 115 (2017).
7. Reed, S. *et al.* Training Deep Neural Networks on Noisy Labels with Bootstrapping. (2014).
8. Goldberger, J. & Ben-Reuven, E. TRAINING DEEP NEURAL-NETWORKS USING A NOISE ADAPTATION LAYER. 9 (2017).