Statoil/C-CORE Iceberg Classifier Challenge

Tee Kah Hui, Kalkidan Fekadu, Samuel Pegg
Department of Computer Science and Technology
Tsinghua University
30 Shuangqing Rd, Haidian District, Beijing
jh-zheng20@mails.tsinghua.edu.cn, chefirakf10@mails.tsinghua.edu.cn,
peggsr10@mails.tsinghua.edu.cn

Abstract

This Kaggle Competition is a classification task. We seek to determine if a remotely sensed target is a ship or iceberg using machine learning techniques. We used ResNeXt and VGG + MobileNet models to achieve an log loss score of around 0.17. To further improve the performance, we used ensemble learning technique by stacking the best prediction result. The final log loss score of our model is 0.13661. Our result placed us in rank 599 out of 3341 teams. The source code can retrieved from https://github.com/KHTee/Kaggle-Statoil-Iceberg-Challenge

1 Introduction

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 from icebergs. However, in remote areas with particularly harsh weather, these methods are not feasible, and the only viable monitoring option is via satellite.

The data set of for competition is provided by Statoil, an international energy company operating worldwide, has worked closely 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 getting a fresh new perspective on how to use machine learning to more accurately detect and discriminate against threatening icebergs as early as possible.

In this competition, the challenge is to build a machine learning algorithm that automatically identifies if a remotely sensed target is a ship or iceberg. Improvements made will help drive the costs down for maintaining safe working conditions.

2 Data

The data (train.json, test.json) is presented in json format. The files consist of a list of images, and for each image, you can find 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, so the list has 5625 elements. Note that these values are not the normal non-negative integers in image files since they have physical meanings - these are float numbers with unit being dB. Band 1 and Band 2 are signals characterized by radar back scatter produced from different polarization at a particular incidence angle. The polarization correspond to HH (transmit/receive horizontally) and HV (transmit horizontally and receive vertically). More background on the satellite imagery can be found here.
- inc_angle - the incidence angle of which the image was taken. Note that this field has 133 missing data marked as "na", and those images with "na" incidence angles are all in the training data to prevent leakage.
- is_iceberg - binary target variable. 1 identifies an iceberg and 0 a ship.

3 Data Exploration

Firstly we take a look at the images in band_1 and band_2. Consider Figure 1. To the left is an iceberg, to the right is a ship. The first row is band_1, the second row is band_2 and the third row is an RGB image that uses band_1 as the first layer, band_2 as the second layer and the average of the two on the third layer. Note that in order to plot the third image, we had to convert the data to type uint.8 so that it could be displayed, but we used the original more detailed array in our model.

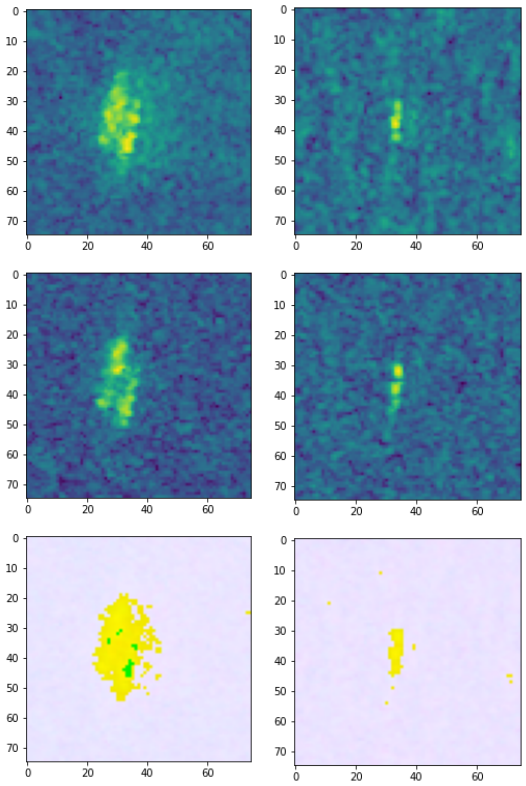

Figure 1: Raw Image Data

There are only two numeric variables, inc_angle and the target classification variable: is_iceberg. The data is quite evenly split between icebergs and ships, as we can see Figure 2.

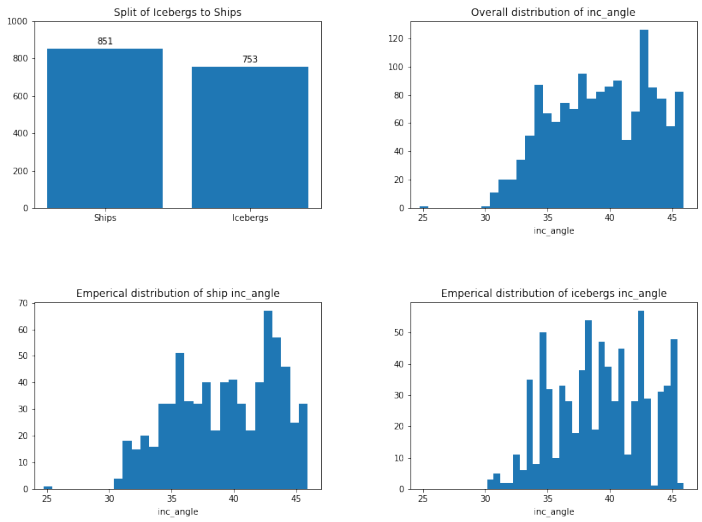

Figure 2: Simple Statistics

In the data, the number of unique inc_angles is 879, and the total number is 1604. We find that 98% of data with the same inc_angle have the same classification (only 6 out of 879 unique angles do not have the same classification).

4 Preliminary Results

Since the labels are provided in the training data, we embark on a supervised learning task. More specifically, a binary classification task as there are only two categories and we need to output the probability that a particular satellite image contains an iceberg. Some of the common methods in binary classifications are decision trees, random forests, Bayesian networks, Support Vector Machines (SVMs), Neural networks and Logistic regression.

The score for the competition is evaluated on the log loss between the predicted values and the ground truth. For any given problem, a lower log-loss value means better predictions.

4.1 Support Vector Machine (SVM)

Table 1: Scores for different parameters

Kernel	Gamma	Degree	Public Score	Private Score
Sigmoid	Auto	n/a	18.01806	18.19227
Sigmoid	Scale	n/a	9.53319	9.06462
Poly	Auto	2	8.68468	8.47043
Poly	Scale	2	8.68468	8.47043
RBF	Auto	n/a	18.01806	18.19227
RBF	Scale	n/a	6.58841	6.67523
Linear	Auto	n/a	18.01806	18.19227
Linear	Scale	n/a	18.01806	18.19227

We decided to start with SVM because it is one of the most robust prediction methods. As discussed in previous sections, the training data provided contains band_1, band_2 and inc_angle information of the satellite image.

Our first approach is to try with prediction with only band_1 and band_2 information. We aggregate the list of 5625 elements in each band for the images into a numpy array of shape (-1,11250). Next, we feed the input into a SVM model with linear kernel. With this model, we achieve scores for different parameters, as seen in Table 1.

We tested the model using train-test split ratio 0.75 on training set that consists of 75% train data to 25% test data. An interesting finding of using SVM is that the model achieve a high accuracy of 0.7780 and f1-score of 0.8017 for model with RBF kernel. However, the score for test data is bad. One of the reasons could be the amount test data is 8 times the amount of train data. Hence, there are a lot of unseen data points for the model and caused an inaccurate decision boundary.

4.2 Naive Bayes

Aside from SVM, we also tried the approach of Naive Bayes classifier. The reason behind NB classifier is that the train data provided is relatively small compare to test data. NB classifiers need only a small amount of train data to estimate the parameters needed for classification. Besides, in a supervised learning situation, Naive Bayes Classifiers can be trained very efficiently. Unfortunately, the results were very poor.

4.3 Logistic Regression

We also tried logistic regression. Here we use only the inc_angle data to train our model. As discovered in data visualization part, most of the images with same inc_angle tends to have similar label. Any data without inc_angle is dropped from the training set. We see an tremendous improvement in scores (private score: 0.69233, public score: 0.69269) compare to SVM and Naive Bayes. The reason for logistic regression model to outperform SVM and Naive Bayes model is because the output needs to be in the probability of the image contains iceberg. SVM and Naive Bayes model outputs the predicted class instead of probability of certain class.

4.4 Outcomes

While logistic regression produced the best results, our ranking on the leader board was still very low. It seems that simple models are not able to capture the complexity of the task at hand, and lack the ability to generalise well to the larger pool of testing images. We decided to move on to use neural networks, however, first we wanted to address our lack of training data.

5 Data Preprocessing and Augmentation

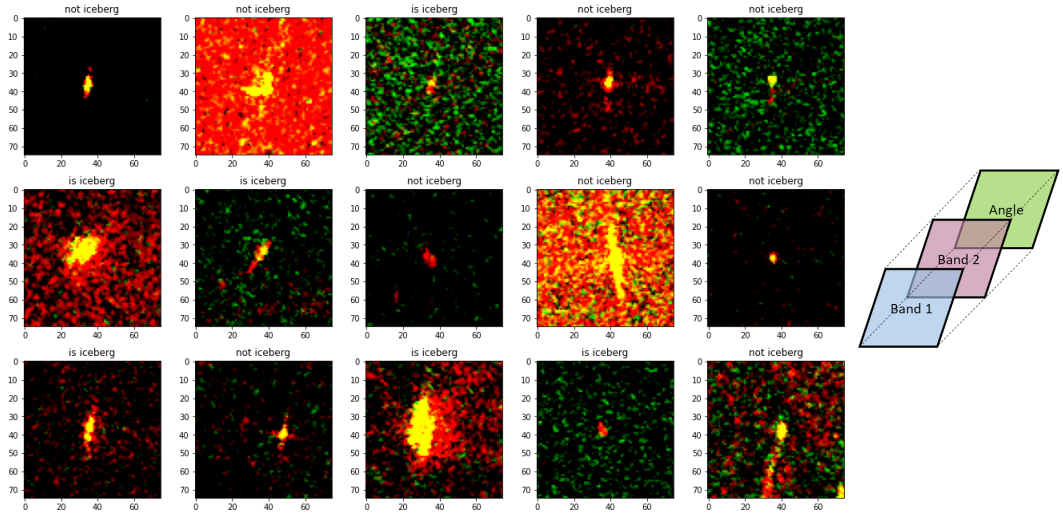

Figure 3: Transformed images and data structure

As discussed above, each observation in the data corresponds to two flattened images of the ship/iceberg using different light polarisations, and the incidence angle at which the photo was taken. We also get the binary ground truth - 1 representing an iceberg, and 0 a ship. We converted the training data into 75x75x3 matrices. The three layers of these matrices (channels) are band_1,

band_2 and inc_angle respectively. This structure can be seen in Figure 3. Also in Figure 3, we can see the transformed image data. Note that image data must be converted into integer format in order to be plotted, hence in the top left, the black pixels do not all have the same value. They are either over the maximum RGB value (255) or rounded to the same value. This means that while the images may not look that clear to the human eye, to the computer there are many fine details to be analysed.

5.1 Image Augmentation

The goal for adding data augmentation is to increase the generality of the model. Currently the training data only consists of 1604 samples whereas the test data contains 8424 samples. We use data augmentation techniques to avoid overfitting and to make up for the lack of data. The augmented data is obtained from the original satellite images by applying simple geometric transforms such as translation, rotation, shearing, horizontal or vertical flips etc. Though the new image produced has been transformed, the label of the image still stays the same. For our models, we did width and height shift and rotation. Below is an example of augmentation parameters we used.

- Rotation_range: 40
- Width_shift_range: 0.1
- Height_shift_range: 0.1
- Shear_range: 0
- Zoom_range: 0.1
- Horizontal_flip: True
- Vertical_flip: True

With image augmentation, we extend our training data by a factor of 32.

6 Final Solution

Our final solution utilises ensemble learning. We combine the prediction results from two state of the art models – ResNeXt-29 (1) and VGG-16 (2) – which we will discuss below.

6.1 ResNeXt-29 (1)

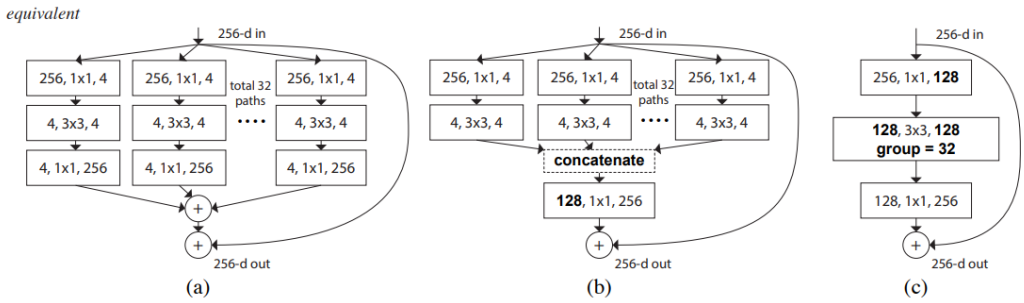

Figure 4: ResNeXt architecture

ResNeXt, created by Facebook, is a highly modularised architecture for image classification. We implemented ResNeXt-29 (1) using Keras on the iceberg data and managed to achieve a log loss of 0.16965. This managed us to get in the top 1000 submissions. The ResNeXt architecture is shown in Figure 4. Below is the configuration of our model.

- Depth: 29
- Cardinality: 4
- Number of filter: 8

- Regularization: None

As the labelled data is relatively small, we used 5-fold cross-validation to achieve better performance. For the training, we are using 1000 epochs and 10 steps per epoch. However too many epochs could lead to overfitting. We used callbacks API from Keras to perform early stopping when the valuation loss has stopped decreasing. Below is the parameters for callbacks.

- Minimum change: 0.001
- Patience: 45

6.2 VGG-16 (2)

This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. The architecture is shown in Figure 5. For this task, we used 16 layers VGGNet to perform the classification. We selected the pre-trained weight from ImageNet as our base model. Similar to ResNeXt, k-fold cross-validation and early stopping are used improve performance and prevent overfitting. In the final layer, we chose Sigmoid activation function and binary cross entropy loss is our metric. Besides we used Adam optimizer with learning rate of 1e-5. Our VGGNet model achieved a log loss of 0.17109.

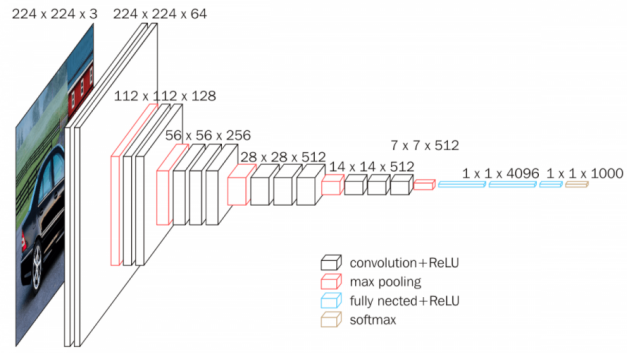

Figure 5: VGG-16 (2) Architecture

6.3 Ensemble Learning

Our final model used ensemble learning to increase performance further by combining the prediction results from ResNeXt-29 (1) and VGG-16 (2). The stacked model results are manually submitted to Kaggle. This approach decreased our log loss to 0.15152 and placed us in the 700th position out of 3300 teams.

To further improve the performance, we sourced public kernels from Kaggle. Some of the public kernels we used are (3), (4), (5), (6), (7), (8). In our ensemble algorithm, we generate the stacked submission files using best prediction (closer to 0 or 1), minimum, maximum, mean and median of all the public kernel submissions. Below are performances of each type of ensemble.

Table 2: Result of different ensemble techniques

Method	**Log Loss**
MinMax	0.13661
MinMax + BestBase	0.16468
MinMax + Mean	0.13972
MinMax + Median	0.13828

7 Conclusion

Table 3 shows our main models and their log loss.

Table 3: Final Results

Model Name	Log Loss
Logistic Regression	0.69233
ResNeXt-29	0.16965
VGG16 + MobileNet	0.17109
Ensemble Solution	**0.13661**

In this report we have described and given the results to our ensemble learning solution to the Statoil/C-Core Iceberg Classifier Challenge and showed how we ranked amongst the other competitors. This was a very interesting project. We learnt a lot about image classification, image processing, data manipulation and through implementing many different methods, a lot of machine learning theory. We believe this project will act as a strong basis for our future studies as it gave us tangible experience with machine learning and the associated Python libraries.

References

[1] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Solomonk, "Pytorch cnn densenet ensemble lb 0.1538," Nov 2017.

[4] wvadim, "Keras+tf lb 0.18," Nov 2017.

[5] devm2024, "Keras model for beginners (0.210 on lb)+eda+rd," Nov 2017.

[6] cttsai, "Ensembling gbms (lb .203 ," Nov 2017.

[7] bluevalhalla, "Fully convolutional network (lb 0.193)," Nov 2017.

[8] submarineering, "submission38 lb-0.1448," Nov 2017.