Statistics 440: Case Study 3

Statoil/C-CORE Iceberg Classifier Challenge

Brad Smallwood, 301228034
In Collaboration with: Barinder Thind

December 7, 2017

Abstract. In this case study we aim to classify satellite images as either having a iceberg or ship by implementing a Convolutional Neural Network. Model performance was evaluated using the logarithmic loss metric in which the Convolutional Neural Network achieved .1918 error on the test set. Noise reduction methods were used prior to implementing the network to improve the quality of the image data.

Keywords: Convolutional Neural Network, Expectation Maximization, Mixture Models, Iceberg Classification, Predictive

1 Introduction

Icebergs pose a significant threat to ships as they navigate the waters off the eastern coast of Canada. A possible strategy to warn ships of icebergs in their vicinity is to perform aerial reconnaissance to identify these hazards, but due to harsh weather conditions and the remoteness of these locations, this strategy is not always possible. The alternative is to use a satellite to take images to identify these hazards, but the drawback to this strategy is that it can be difficult to differentiate between the ships and the icebergs. This report addresses the issue of differentiating icebergs and ships when utilizing satellite image data by performing image analysis using a Convolutional Neural Network (CNN).

The methodology section of this report will describe the features of the data as well as the steps taken to improve the quality of the data. Specifically, how a mixture model was implemented in order to separate out the object of interest (an iceberg or ship) from the background noise. Following this, a CNN will be presented as a model which is able to accurately classify the satellite images as either having a iceberg or ship in them.

2 Methodology

2.1 Data Features

The dataset used in this report was provided by Statoil and C-Core to be used for a kaggle competition. [1] The dataset is comprised of 1604 observations, each with two images (taken at differnet frequencies), and the angle at which the images were taken. Each image is 75x75 pixels, an example of which can be seen below in figure 1.

[1] https://www.kaggle.com/c/statoil-iceberg-classifier-challenge

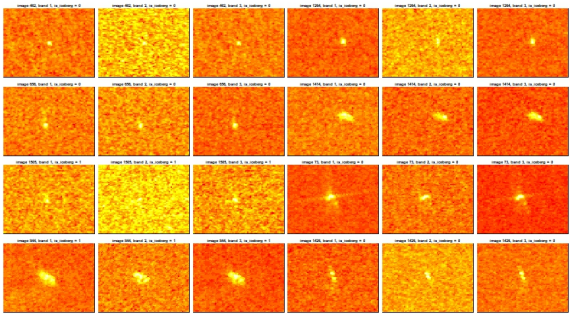

Figure 1: 8 Images from the Dataset.

In the figure above the images are in horizontal groups of three. The first two images are at different frequencies and third is the combined image (average of pixel values). From preliminary observations of these images, it can be difficult to determine in many of them whether the predominant mass is a iceberg or a ship. Additionally, in other images it can be difficult to distinguish the iceberg or ship from the background noise of the image. The initial step taken to classify these images was to improve the quality of the images by removing as much of the background noise as possible so that only the primary mass, which we are attempting to classify, would be considered.

2.2 Mixture Models and the Expectation Maximization Algorithm

The algorithm implemented is the same as the one presented by Dr. Davis in lecture. In figure 2 below, we are able to see the histogram generated by the pixels in a image. While it is difficult to distinguish, we are able to see that the distribution appears to have a few outliers far out into the right tail. These extreme pixel values belong to the object we would like to classify. By treating these as two different Gaussian distributions we are able to use a mixture model to distinguish between them. The mean and standard deviation of the two distribution are estimated using the Expectation Maximization (EM) algorithm which is a powerful method for finding the maximum likelihood solutions for models with latent variables. Through the estimates of μ and σ we are then able to find where the two Gaussian distributions intersect by using the optim() R function. We then keep the pixels that are greater than that intersection and the other pixels are reduced to a constant level. This process results in a reduction of the background noise around the iceberg or ship as seen in figure 3 below.

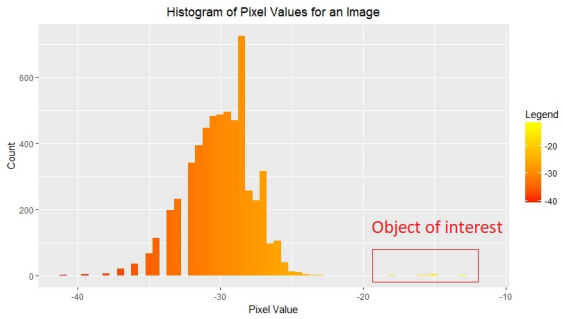

Figure 2: figure
Histogram of Pixel Values (Before Noise Reduction)

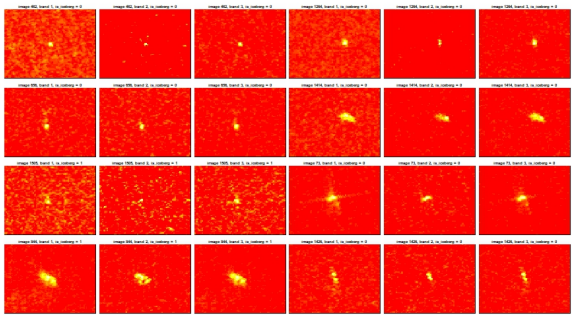

Figure 3: figure
Images after Noise Reduction

These images seen above are the same 8 images that were presented in the last section but this time we can clearly distinguish between the object of interest and the background.

2.3 Convolutional Neural Network

A Convolutional Neural Network is a popular deep learning technique that is commonly applied to analyzing images such as the ones in this case study. A regular neural net takes inputs and is able

to find intricate connections between them. The middle layers (in the diagram below the hidden layers), receive inputs (real numbers) from previous layers and takes a weighted sum of them. If the activation function threshold is reached (in class we've been showed ReLU activation), then that node sends its output to the nodes in the next layer. Through this process, connections are formed and the data is able to be classified. See the diagram below for a simplistic overview of how a basic neural network might operate on the dataset.

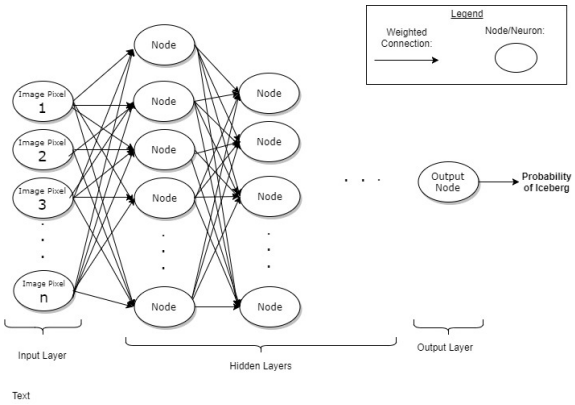

Figure 4: Neural Network

A Convolutional Neural Network is an extension of the ordinary Neural Network. It used other types of layers which are suited for image classification. Specifically, it uses filters that convolve (a.k.a slide) over the data, and through taking the dot product of the pixel values, it generates a new layer. After many of these layers they are initialized randomly and become the parameters and are used for future learning. Additionally, CNN does not only have the one type of layer, it also has pooled, and many more. Pooling layers operate by sampling from each of the filters to reduce the amount of parameters being created by the convolution layers. Many more types of layers exist and the process is much more complicated than as described here but these two layers are part of the key architecture of CNN models. The figure below shows the general process of how a CNN operates on a image in our dataset.

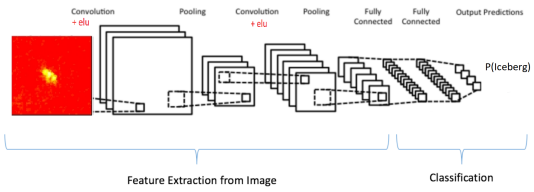

Figure 5: Convolutional Neural Network

Lastly, in order to improve the predictions made by CNN, a stacking approach was used. The CNN model was run with different specifications on the many parameters. Some of these models made better predictions on certain observations than others, so in other to find the best possible predictions, they were stacked together. The max, min, mean, and median were tabulated for each observation on the Kaggle test set. This produced new predictions that were a combination of the previous predictions. Of these stacked predictions, the best performing was where the minimum or maximum of the predictions was used for any predictions above (or below) a set threshold. The summary statistics used to improve the prediction accuracy can be seen in the figure below.

	id	CNN_Pred1	CNN_Pred2	CNN_Pred3	CNN_Pred4	CNN_Pred5	CNN_Pred6	CNN_Pred7	CNN_Pred8	stack_max	stack_min	stack_mean	stack_median
1	5941774d	0.844928920269012	0.195827901363373	0.008171051740646 36	0.349642624457677	0.572922468185425	0.343406400561333	0.077631764113903	0.151172727234642	0.844928920269012	0.008171051740646 36	0.317962982254724	0.269617150962353
2	4023181e	0.351700395345688	0.222424611449242	0.638347744941711	0.404157583912214	0.889885604381561	0.871613794088363	0.253133028745651	0.182670712471008	0.889885604381561	0.182670712471008	0.476741684441693	0.377928989628951
3	b20200e4	0.000311931711621583	3.578289761208e-05	0.008061107240867 86	0.002802940619100 51	0.761490464210 51	0.381498607783113	4.502552201302 02e-06	0.000427353341365233	0.761490464210 51	4.502552201302 02e-06	0.144329086295449	0.001615146980232 87
4	e7f018bb	0.984394252300262	0.971913397312164	0.999506115913391	0.985271255175272	0.983313381671906	0.985825557231903	0.990337967872 62	0.975865840911865	0.999506115913391	0.971913397312164	0.984553471048673	0.984832753737767
5	4371c8c3	0.051629215478897 1	0.151687458157539	0.774739325046539	0.326018666227658	0.811954557895 66	0.857980091094 97	0.071709327399730 7	0.072058081626892 1	0.857980091094 97	0.051629215478897 1	0.389722090365986	0.238853062192598
6	a8d9b1fd	0.508704483509064	0.118670634925365	0.001626615645363 93	0.209667244693264	0.573806583881378	0.412622920155525	0.035828806459903 7	0.082550205290317 5	0.573806583881378	0.001626615645363 93	0.242934686820023	0.164168939809315
7	29e7727e	0.015580158680677 4	0.065588235855102 5	0.057049203664064 4	0.046072532733281 4	0.149253502488136	0.152139521121978	0.222436442971 23	0.162143677473068	0.222436442971 23	0.015580158680677 4	0.108782909373442	0.107420869171619
8	92a51ffb	0.989555358886719	0.978758692741394	0.999976038932 8	0.989430030186971	0.967603743076324	0.978166181564331	0.990780889987946	0.981694400310516	0.999976038932 8	0.967603743076324	0.984495666960875	0.985562215248744

Figure 6: Dataset of CNN Predictions and Summary Statistics

3 Results and Discussion

The model is being evaluated based off the logarithmic loss metric, defined as,

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log\, p_i + (1-y_i)\log\,(1-p_i)].$$

This metric penalizes predictions that are incorrect more harshly when the model predicts with a higher degree of certainty (more extreme probability) the status of the response variable. At the time this report was written, the best CNN model produced a log loss of 0.1918 and an prediction accuracy of 0.9032 on the valuation set. A summary of the classification of the model can be seen in the confusion matrix below in Table 1, where the model was evaluated used a validation set (split from the Kaggle provided training data).

Table 1: Confusion Matrix

Observed/Predicted	Iceberg	Ship	Class Error
Iceberg	150	17	0.0548
Ship	13	130	0.0419

When implementing the additional stacking step, we were able to improve the log loss by nearly a full 0.02. The stacking method that was most successful was when we took the predictions from the best CNN model as the baseline and then changed all of the predicted probabilities over 0.75 to the maximum of all the predicted probabilities across all CNN models, for each image. Likewise, this same procedure was performed on the observations which had a probability less than 0.25 of being an iceberg, except for these observations the minimum was used.

This model placed us 482th in the Kaggle competition as seen below in Figure 6. By cleaning the images and removing the background noise, the model performed marginally worse than just using the regular images. A possible reason may be because the noise reduction methodology may have removed information that was important when classifying the images.

Figure 7: Kaggle Logloss and Rank

Other models that we considered included,

Model	Log Loss
Neural Net	0.4071
Random Forest	0.62910
XGBoost	0.38062
Gradient Boost	0.6288

These models were run on the noise reduced images and the regualar dataset. The best runs for each model were recorded. Of these models XGBoost produced the best results but even it was vastly inferior to a basic CNN.

4 Conclusion

The goal of this case study was to use satellite images and determine if the objects within those images were either icebergs or ships. CNN was able to classify the images with a high degree of accuracy. Moreover, CNN was able to fairly accurately assign a degree of certainty to each classification in that it did not assign a high likelihood of a ship being an iceberg (nor the converse). While the noise reduction methods implemented resulted in a decrease in accuracy, alternative methods such as smoothing, edging, or other noise reduction algorithms might perform better. Furthermore, the CNN models used in this case study were fairly simplistic compared to what other Kaggle competitors were using. Given more time, and a more powerful computer, a more sophisticated CNN could be implemented which would likely produced better results.

References

- James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017.
- Hastie, Trevor J., et al. The Elements of statistial learning: data mining, inference, and prediction. Springer, 2009
- Pokharna, Harsh. "The Best Explanation of Convolutional Neural Networks on the Internet!" Medium, TechnologyMadeEasy, 28 July 2016, medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8.
- Deshpande, Adit. "A Beginners Guide To Understanding Convolutional Neural Networks." A Beginners Guide To Understanding Convolutional Neural Networks – Adit Deshpande – CS Undergrad at UCLA (19), adeshpande3.github.io/adeshpande3.github.io/A-Beginners-Guide-To-Understanding-Convolutional-Neural-Networks/
- Deshpande, Adit. "A Beginner's Guide To Understanding Convolutional Neural Networks Part 2." A Beginner's Guide To Understanding Convolutional Neural Networks Part 2 – Adit Deshpande – CS Undergrad at UCLA ('19), adeshpande3.github.io/A-Beginner

Credit to the following authors and kernels. Their kernels were invaluable in creating the CNN model used in this report.

- DSEverything. "Explore Stacking (LB 0.1463)" kaggle, https://www.kaggle.com/dongxu027/explore-stacking-lb-0-1463.
- DimitriF. "keras with data augmentation (LB: 0.1826)" kaggle, https://www.kaggle.com/dimitrif/keras-with-data-augmentation-lb-0-1826.
- DeveshMaheshwari. "Keras Model for Beginners (0.210 on LB)+EDA+RD" kaggle, https://www.kaggle.com/devm2024/keras-model-for-beginners-0-210-on-lb-eda-r-d
- wvadim. "Keras+TF LB 0.18" kaggle, https://www.kaggle.com/wvadim/keras-tf-lb-0-18
- bgo. "CNN with Keras" kaggle, https://www.kaggle.com/bugraokcu/cnn-with-keras
- Yassine Ghouzam. "Introduction to CNN Keras - 0.997 (top 6%)" kaggle, https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6