Machine Learning in practice: NOAA Fisheries Steller Sea Lion Population Count

Bauke Brenninkmeijer, Emma Gerritse, Démian Janssen,
Wietse Kuipers, Ties Robroek, Timo van Niedek

November 13, 2018

1 Introduction

Over the last 30 years, the steller sea lion count has decreased by 94 percent. The western population in the North Pacific is endangered. Annual counts are performed in an attempt to conserve these animals by keeping track of the population. So far, these surveys have been done by specially trained scientists at NOAA Fisheries, an Alaskan science center. They collect aerial images which can be used to estimate the sea lion population. This enables them to better understand contributing factors to the lack of recovery of sea lions in that area.

With the current system, via manual labor, these images take months to count. Automating this counting process would speed up this process significantly and free up critical resources for NOAA Fisheries.

2 Problem

The objective of this project was to recognize and classify sea lions from aerial images. The classification is done over a few appropriate classes; adult male, subadult male, adult female, juvenile and pup. The task seems perfectly tailored for machine learning. Analyzing thousands of pictures looking for the same creatures is theoretically the perfect job for an automated data extraction algorithm, and up to a certain point, it was. However, there were some difficulties. First, the provided pictures are aerial pictures and not specifically targeted on sea lions. This means that they also cover large amounts of bare rock and sea, as can also be seen in Figure 1. Second, especially on some pictures, the sea lions can blend in very well with the rock formations, making them very difficult to distinguish from these rocks. Lastly, the sea lions often group together very closely, which is causes problems with counting multiple sea lions as one.

3 Approach

Originally, we opted for a two-stage approach where the images are first segmented such that we could distinguish sea lions from background noise. The extracted sea lions would then be classified in the second stage. This would be accomplished by splitting in two teams, where one who would create the segmentation algorithm and the other the classification algorithm.

Figure 1: Example image of sea lions lying on the beach.

However, this approach was proven difficult, since programming a classification algorithm without input is quite a hard. Thus we decided to postpone the classification until the segmentation algorithm was finished. This meant the core mentalities of our approaches were similar, yet the execution was quite different.

3.1 Preprocessing

Since the training and test sets were extremely large, we created an efficient patch extraction system. We load images on-line, and cache them whenever we can extract multiple batches from them, which is the case for images where the amount of sea lions is larger than the batch size. By inspection of the images, we found that the sea lions cover at most an area of 80×80 pixels. Our patches are therefore 120×120 pixels large, to add some context. We create balanced batches by iterating over the sea lions (positive samples) and extracting an equal amount of patches not containing a sea lion (negative samples). The negative samples were sampled such that they are never further than 500 pixels away from a sea lion. This makes it so that most of the negative samples will be taken from land area, which allows the networks to distinguish sea lions from

rock formations more easily.

We augment the training data only by shifting the patches by a maximum of 12 pixels in both axes. Because of the immense size of the training set, we do not require sophisticated data augmentation techniques.

3.2 Segmentation

For the segmentation step, we built a convolutional neural network in Theano and Lasagne. This provided us with a well established GPU-enabled neural network. We extracted the segments real time, saved them for the classification, and analyzed them using the neural network. Additionally, we prevented an overfit of the neural network on the train data via flipping and shifting the images. The flipping and shifting forces the network to also be able to recognize upside down and shifted images. This should then result in an increased generalization performance.

3.3 Classification

Classifying the sea lions turned out to be more difficult than segmentation. We created a two different classifiers, with varying success. The first method involves a regression based on the area that was marked as containing a sea lion and prior knowledge about the distribution of sea lion types. The second method was a Convolutional Neural Network (CNN) that operated on the output of the segmentation stage.

3.3.1 Area-based Classification

After segmentation, a heatmap is generated containing the locations of the sea lions. We tried finding a relation between the area segmented and the amount of sea lions. This is because the sea lions take up certain space and their class distribution might be similar. Multiple methods of preprocessing on the heatmaps have been utilized to achieve this. We have selected different thresholds to cut off the heatmaps and we have used methods like eroding and dilating (using `cv2.erode` and `cv2.dilate` with a 4×6 kernel of ones and a threshold of 0.6) or peak local max (using `skimage.feature.peak_local_max` with a threshold of 0.95) to restrict the heatmaps as much as possible. After this, we have applied regression (1st order polynomial using `numpy.polyfit`) on the amount of sea lions in a class vs the segmented area.

3.3.2 Convolutional Neural Network

Another classification approach we explored was to use a Convolutional Neural Network (CNN) as described by LeCun et al. [2]. Our final setup had three convolutional layers and two fully connected layers, with ReLU activation applied everywhere [1]. Dropout is applied to the fully connected layers to prevent overfitting. A final softmax layer transforms the output of the CNN to class probabilities. We trained using a cross-entropy error, and stopped when the validation accuracy plateaued for two epochs. Because the batch generator yields half positive, half negative patches, we force the network to learn to distinguish between sea lion classes by scaling the losses for these classes by a factor of two.

4 Results

As we split approach in segmentation an classification, we will discuss their respective results separately as well. We will both describe how we developed our approaches as well as how our final approaches are structured.

4.1 Segmentation

Our segmentation algorithm worked quite well, and provided us with segments of almost exclusively sea lions for our classification. The segmentation algorithm managed a validation accuracy of 96.25%. An example segmentation can be seen in Figure 2.

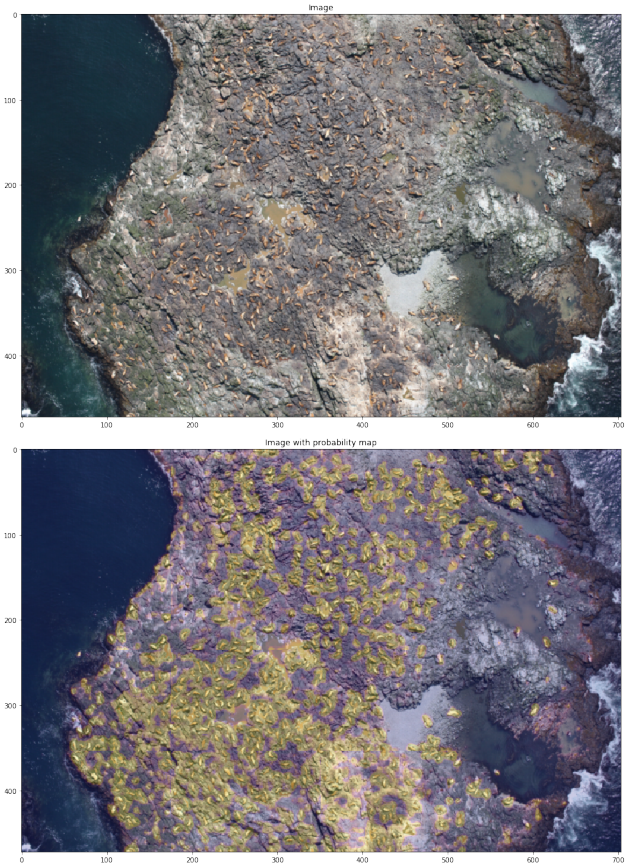

Figure 2: The upper image is the original, the lower image includes a probability map overlay. The marked spots (yellow) indicate the area which the network thinks are sea lions.

4.2 Classification

Our classification approaches were initially not very successful. We have attempted the following approaches.

4.2.1 Segmentation area based classifying

Validation results of the regression can be seen in Figure 3. We calculated the Pearson correlation coefficient (using `scipy.stats.stats.pearsonr`) between classified area and the amount of sea lions. For the total amount of sea lions per image, this got a correlation coefficient of 0.44; and the individual classes had a correlation coefficient between 0.26 and 0.39 so we were hopeful that there was some correlation.
By accident, we have also submitted a score which had constant values for each class (3 adult males, 3 sub adult males, 19 females, 12 juveniles, 3 pups). Surprisingly, this gave us our best score as of yet on the leaderboard. Many others on Kaggle have also submitted hardcoded results for each image, these people dominate much of the leaderboard even though they're not likely doing what the organizers of the competition are hoping for.

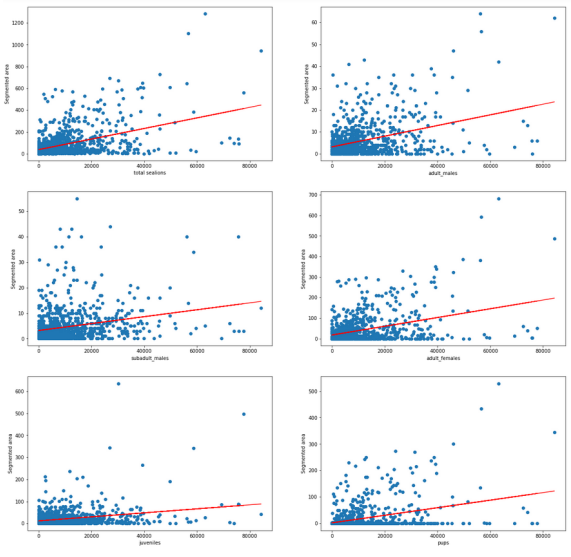

Figure 3: Scatter plots of segmented space vs the amount of sea lions in one class. The red line is the regression line.

Method	Score
Regression after erosion & dilation	27.69801
Regression after PLM	36.59243
Mistake with constant values	26.92200
Means (by Kaggler 'Benchiislett')	26.59766
All zeros (by Kaggle)	29.08704
CNN	30.09940

Table 1: Scores on the leaderboard with different methods

4.2.2 Convolutional Neural Network

In our experiments with the CNN classifier, we found that the validation accuracy stopped increasing after only 6 epochs. Unfortunately, due to time limitations, these are our best runs.

The maximum validation accuracy that we achieved was 80%. Because of the sheer size of the test set, we decided that it was not possible within the time we had left to classify the entire test set using the CNN. Instead, we classified 5000 test images, which is roughly 25% of the full test set. For the remainder of the test set, we entered the mean sea lion counts found by Kaggle user "benchiislett" – 5 adult males, 4 sub-adult females, 26 adult females, 15 juveniles and 11 pups per image.

We achieved a mean root mean squared error of 30.10 on the Kaggle leaderboard using this method. Unfortunately, the naive approach using the means achieved a score of 26.60, outperforming our CNN model.

5 Discussion

Even though the competition is actually about *counting* sea lions, we figured we had to classify them somehow in order to count them. Classifying sea lions proved to be quite difficult. There where several problems we encountered. First of all, sea lions tend to group together a lot and lie on top of each other, which made picking the individual sea lions out of the segmentation difficult. Second, the data set was very large. The train set consists of 948 images, 5.86 GB in total, and the test set consisted of a massive 18636 images, 86.2 GB in total. Even one forward pass through our segmentation network cost us a lot of time, simply because the amount of test data is overwhelming. Assuming an average of 4 seconds per image, it costs slightly less than 21 hours to segment the test set. This is one of the reasons why we tried regression on the segmentations, since it can be done relatively fast. As a third remark, some sea lion classes (like sub adult and juveniles) look very much alike (even to our human eyes they look very similar), so we understand why the network had difficulties distinguishing them. Lastly, we were a bit disappointed that both our classification methods performed worse than the means of the classes.

6 Who did what?

The division of labor was problematic due to the fact that some of our members, Ties and Bauke, did not follow earlier courses where neural networks were covered. They therefore could not contribute as much as the others to the deep learning tasks. We tried to resolve this by letting them do other tasks, but that did not amount to the same amount of work that was put

in the networks. Below we will list who has worked on what:

- Bauke: handled communication with coach, participated in the segmentation, uploaded all the data to Cartesius, set up and contributed to the report: specifically sections 1, 2, 3, 4.1 and 6.
- Ties: participated in the segmentation and contributed to the report.
- Démian: Big contributor to segmentation and classification, contributed to the report; specifically sections 3, 3.2, 3.3, 4 and 5.
- Emma: Big contributor to segmentation and area based classification, wrote sections 3.3.1, 4.2.1 and 5
- Timo: Created batch extractor and contributed to the second stage classifier. Wrote sections 3.1, 3.3, 4.2.2 and a part of 3.3.2.
- Wietse: Big contributor to classification, contributed to the report.

Any code we used can be observed at `https://github.com/Baukebrenninkmeijer/Kaggle-Challenge-NOAA-Fisheries-Steller-Sea-Lion-Population-Count`.

References

[1] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.