Final Project Report - Humpback Whale Identification

Martin Verstraete and Hend Zouari

January 2019

1 The competition

1.1 Introduction

This is a Kaggle competition about whale recgonition. Basically, we have photos of humpback whale flukes and we want to be able to identify all the whales.To aid whale conservation efforts, scientists use photo surveillance systems to monitor ocean activity. They use the shape of whales' tails and unique markings found in footage to identify what species of whale they're analyzing and meticulously log whale pod dynamics and movements.Thus, we're challenged to build an algorithm to identify individual whales in images.

1.2 The dataset

You can find both datasets on this link.
The training dataset contains 25000+ images of humpback whale flukes. Individual whales have been identified by researchers and given an Id, but one third of the pictures depict unidentified whales and are labeled as "new whale".New whale is the biggest category with 9664 samples over 25361 datapoints. New whale is any whale that isn't in scientist's database. The testing dataset contains 7000+ images of humpback whale flukes.

The critical aspect of the challenge is the uneven distribution of the different classes, for example more than half classes of whales contain less than 5 examples compared to classes with 9000 examples, which makes learning really difficult.

Here we present in the graph the number of classes by the number of images they contain, and we can observe the first peak, where we have a huge number of classes with just one sample (over 2000 classes among 5004).

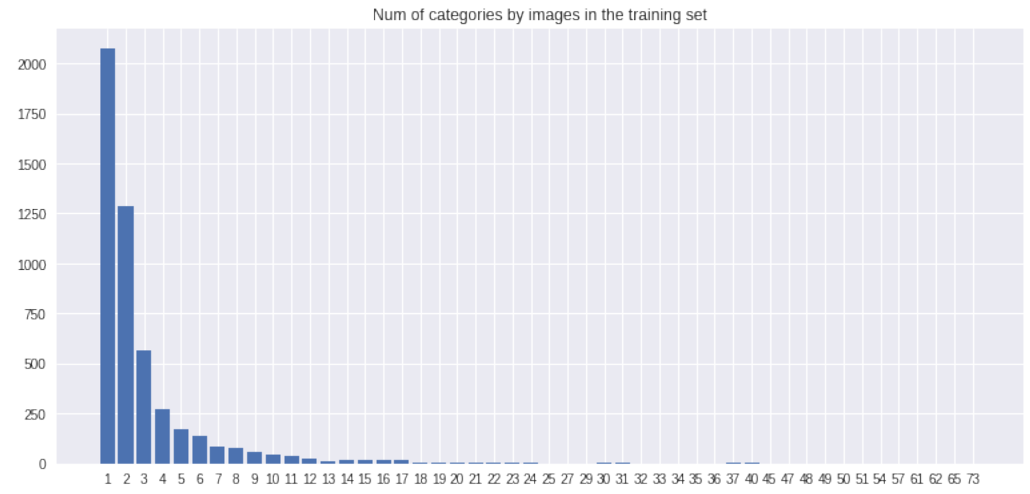

Figure 1: Number of classes by number of images contained in them

1.3 Evaluation

Submissions are evaluated according to the Mean Average Precision @ 5 (MAP@5): : $MAP@5 = \frac{1}{U} \sum_{u=1}^{U} \sum_{k=1}^{min(n,5)}$ where U is the number of images, $P(k)$ is the precision at cutoff k, n is the number predictions per image, and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant (correct) label, zero otherwise.

2 Our method

2.1 Coding environment

We first wanted to use a Jupyter notebook as a programming environment but the huge number of pictures made the computer run out of memory, and this was not really possible to work together on the Jupyter without dealing with conflicts on the code. We needed the computation power of a GPU. Consequently, we decided to code on a Google Colab which enabled us to use a GPU and the possibility to share instantly the different modifications.

We uploaded all the training and test data on a Google Drive and then unzipped them on the Colab in order to be able to work remotely.

2.2 Preprocessing of the images

We simply used the built-in function *preprocess* from Keras that preprocesses the image to the right format needed by the type of network (vgg, resnet..) as well as resizing the images to 120x120 and converting them to arrays.

2.3 The model

We worked with Keras and decided to use Transfer Learning. We chose the VGG16 model - a CNN- that was pre-trained on the ImageNet dataset. We removed the classifer on top and loaded the weights of all the layers except the last 3 of the feature extractor. Thus, the training phases consisted in fitting the weights of the last layers of the feature extractor and of the classifier (a fully connected layer with dropout and a softmax layer). Thus the number of nodes in the final layer was determined by the number of the different classes, while the structure of the other layers is a hyperparameter that we tune to get a better performance as well as a smaller number of parameters to train.The softmax layer generates probabilities for all the classes that were encoded with a one-hot encoder.

The optimizer used was Adadelta Optimizer, a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. As recommended in the description, we left the parameters of this optimizer at their default values.

3 Improvements to achieve a better score

As we were working on the competition, we encountered some issues and had to provide some improvements to face them.

3.1 Unbalanced classes

When looking at the number of images per classes, we realised that one third of the images were labeled as ”new whale”. That made the learning very biased and the model was more likely to identify a picture as a new whale. So we had to remove all the images labeled as ”new whale”. Still the problem of unbalanced class was persisting, within the remaining classes the densities of the different classes range from 1 to 73.

On the other hand, many classes contained only a few examples so the model had some difficulties to learn features for these classes. So we decided to do some Data Augmentation on the training dataset. We did feature standardization, flip, zoom and rotation on the images to generate more data. Doing this, we went from 15000 images (without the ”new whale” ones) to 22000.

3.2 Image size and memory exhaustion

When first loaded the entire images on the notebook, we encountered a ”Memory Error”, even if we were working on Colab. So we decided to load the images to a target size of 100x100 pixels. But even after doing that, we faced similar issues as the resulting array took a lot of memory resources. To counter this issue, we saved the output of the predictions by the CNN in order not to lose all our work and not to have to run every single cell again.

Similarly, we could try to save all the images converted to Numpy arrays in a text file, so that we find the best model without having to compute all the images when we open the Colab for the first time. But as long as we do not have a solid Data Augmentation it is useless to bother with saving the images.

3.3 Predicting new whales

As new whales are really present in the training dataset, we supposed it was also quite present in the test dataset, so our first strategy was to predict ”new whale” systematically and we got a score of 0.3. Which corresponds to the proportion of new ”whale” labels in the training dataset.
We now need to predict ”new whale” only when it is relevant. So we thought that we could predict ”new whale” only when the standard deviation of the probabilities vector was below a given threshold. That is to say, when the model was not so sure about the class of the picture. But this strategy gave us a score of 0.15 which is awful so we need to find something else. Our supposition is that our model is simply not good enough

3.4 Advanced strategies

3.4.1 Siamese Network

When searching about the different problems dealing with classification where classes can contain only one example, we realised the problem is considered as one-shot learning and Siamese Networks give a better performance.In a one-shot learning setting where we must correctly make predictions given only a single example of each new class Siamese neural networks employ a unique structure to naturally rank similarity between inputs. Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions. Using a convolutional architecture, this method is able to achieve state-of-the-art performance on one-shot classification tasks.

3.4.2 Bounding Box

Furthermore, a good move could be to crop the pictures and keep only a rectangular box around the fluke, this is called a bounding box.
That takes away all the irrelevant information contained in the images: the water. This could be an easy task done with opencv where we use *boundingRect*, however this is a needy task where every image needs a special preprocessing. We could though construct a convolutional neural network (CNN) capable of estimating the whale bounding box. Whale pictures can be cropped automatically to a more uniform appearance. This facilitates training of classification models, and improves the test accuracy. This method is described by one of the competitors in the competition https://www.kaggle.com/martinpiotte/bounding-box-model and is one of the interesting fields that may improve our model’s performance.

3.4.3 Targeted Data Augmentation

We used data augmentation for the whole dataset (except for the ’new whale’ examples. However, this was not enough to get the classes evenly distributed. Some ’poor classes’ are still poor with only one sample, while some relatively rich classes got richer. We restructured the dataset in a way to enrich every class to the maximum number of samples (73). This step is demanding in terms of access to the poor classes and caused several crashes of the notebook.