A Fluke in the Data: Humpback Whale Recognition

David Zweig, Marc Negre

March 23, 2018

1 The Problem

Commercial whaling, climate change, and industrial overfishing have devastated humpback whale populations. Marine biologists would like to track sightings of individual whales so that they can build an accurate picture of the movement of whale populations worldwide. To do so, scientists rely on the unique features of each whale's fluke. When photographed, these can be matched to other photos on record that are associated with a specific whale. This is no easy task. Photographs are taken at various angles and under different lighting conditions, and the characteristics that distinguish one whale's fluke from another's are quite subtle.

Currently, only a handful of scientists are able to manually label whale individuals based on pictures, but this process is time-consuming and error prone. We built a model to assist them by passing the whale fluke pictures into a convolutional neural net, and predicting which individual whale the tail fluke corresponds to. Of course, because we have so few images of each individual, this algorithm is not meant to replace human expertise but rather to drastically reduce the time spent on labeling the whales.

2 Dataset

2.1 The whale fluke dataset

The dataset comes from Kaggle's Humpback Whale Identification Challenge. The images in it are pooled from photographers around the world, and vary widely in their quality, color, and angle. There are 25,060 total images: 9,850 and 15,610 of these have been allocated to the training and test sets, respectively. We only have access to the labels of the training set, and the test set labels are not accessible to competitors but are used by Kaggle to evaluate the performance of the predictions. Several factors made this an especially challenging dataset:

- There are 4,251 unique classes (labels)
- One class, "new whale", aggregates images for all unidentified whales and constitutes 8% of the training dataset
- 2,200 classes ($> 50\%$) contain only one sample
- Images are a mix of 3-channel RGB and grayscale

Figure 1 shows the statistics of how many pictures are labeled as one class. For instance, we see that roughly 1000 classes only have 2 whale tail pictures. The scarcity of each individual in the dataset has quickly led us to augment our data in order to create several versions of each image, as we will present in a further part.

2.2 Dataset preparation

The dataset was prepared through several different steps:

- Reduce the diverse size of each images down to 64x64 each. While running preliminary tests on our local computers, we rapidly abandoned the idea of using 128x128 images which would have been too computationally intensive.

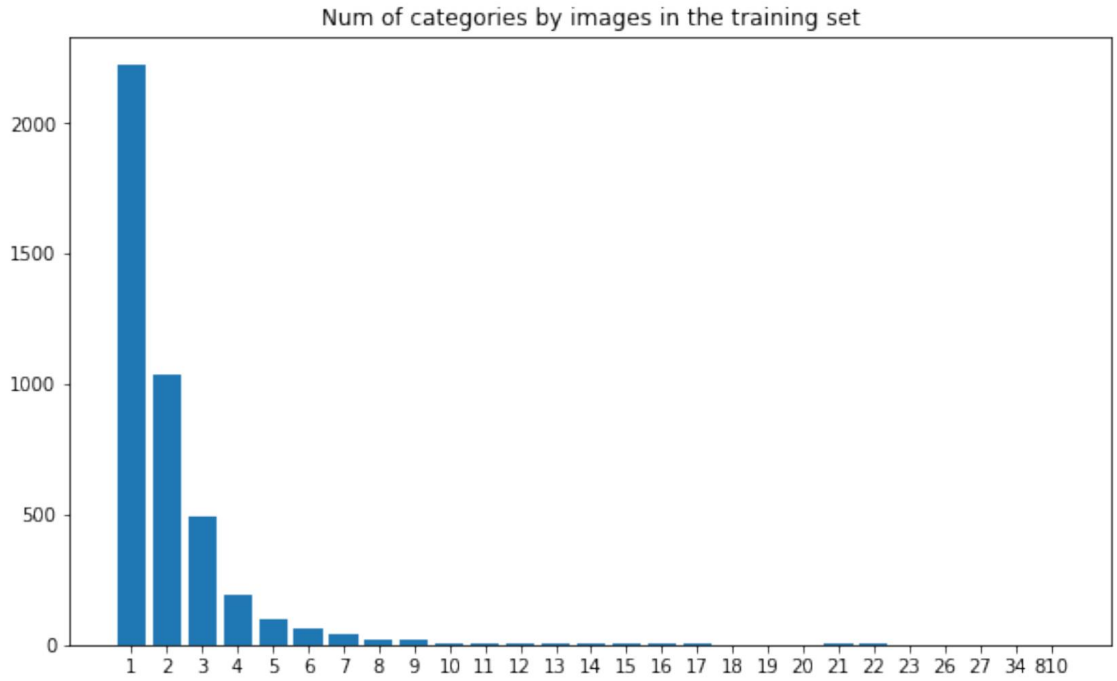

Figure 1: Number of samples per class[2]

- Split our original labeled dataset into train and development sets, containing respectively 80% and 20% of the total 9,850 images. This basic percentage choice was adapted to the small size of our dataset.
- Rename each whale image according to its label, to make the handling of further models more convenient. This basic sorting and renaming task is characteristic of all the things that we learned on the way of project completion, which challenged us at first.
- Remove the images labeled as "new whale" from the train and development sets. This class aggregates images for unidentified whales, which would have created a lot of noise for the algorithm during training. Because the Kaggle challenge authorized five predictions per test image, we instead chose to hard-code the label "new whale" for each test image, and compute only the four most probable labels that corresponded to known whale individuals.
- Augment the data. Having 2 or fewer images for the vast majority of classes naturally led us to consider the option of augmenting our data. Furthermore, our hope to implement a triplet loss function made it even more important, as we needed at least two images per class. We thus generated a total of five images per class, with two random shifts, two random rotations, and one random zoom.

3 Approaches

Relying heavily on the tensorflow model code provided by Olivier Mondrot and Guillaume Genthial, we fully implemented two softmax models utilizing different loss functions and compared the results. The first used cross-entropy loss; the second used triplet loss. We were interested in this approach because the whale flukes differed from each other in subtle ways similar to the differences between human faces, and the widely cited FaceNet model used triplet loss to great effect in facial recognition. Unfortunately, as discussed below, we did not find as much success in our implementation. We also started an attempt to use pretrained weights from the FaceNet model to do transfer learning, but we ran out of time to complete this task.

3.1 Cross-Entropy Loss

We implemented a 4,250-class softmax model using tensorflow's provided sparse softmax cross-entropy loss function. Because the Kaggle challenge allowed us to output 5 predictions per sample, we hard-coded "new whale" as an output and optimized for the other 4 predictions. The formula for cross-entropy loss is:

$$-\sum_{i=1}^{N} y^i \; log \; \hat{y}^i \tag{1}$$

where y^i takes on a value of 0 for all incorrect classes and 1 for the correct class of the i'th sample and $\hat{y}^i$ is the probability output for that class.

3.2 Triplet Loss

We also trained a softmax model utilizing triplet loss rather than cross-entropy loss. Triplet loss is expressed as follows, where a corresponds to an anchor image, p to a positive sample, and n to a negative sample [1]:

$$\sum_{i=1}^{N}[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i)||_2^2 + \alpha]_+ \tag{2}$$

where superscript "a" denotes an anchor image to which a positive "p" image of the same class and a negative "n" image of a different class are compared, and alpha is the margin. The aim of triplet loss is to train the weights to maximize the L2 distance between the anchor and negative inputs and minimize the L2 distance between the anchor and positive inputs. The margin ensures that the embeddings for each sample are sufficiently far apart.[1]

3.3 Transfer Learning

From lecture and from our own experience training our model using triplet loss, we were inspired to attempt a transfer learning experiment by using the pre-trained weights from the FaceNet model before retraining the last fully connected layer to optimize for whale flukes instead of faces. This approach would have allowed us to use weights developed for a much richer dataset and apply them to our own classification problem. We were successful in loading the model into AWS following the approach used in the "Happy House" exercise in Coursera Course 4, but because we didn't start this approach until close to the project deadline, we did not manage to retrain the output for our whale classification problem. As such, we won't discuss the transfer learning approach in our results and discussion.

4 Result and Discussion

Our softmax model using cross-entropy loss delivered superior results compared to our model trained with triplet loss. In fact, the softmax model utilizing triplet loss did not classify a single image correctly. The models trained with cross-entropy loss, both with and without augmented datasets, all scored around 0.30 on dev accuracy, and the model selected for the test set performed similarly with that data. The full results, with different learning rates, are reported in Table 1 below.

Table 1: Prediction Accuracy for Different Models

Experiment \Accuracy	Train	Dev	Test
Softmax, alpha=0.0001	0.296	0.285	N/A
Softmax, alpha=0.001	0.334	0.285	0.3262
Softmax, alpha=0.01	0.322	0.268	N/A
Softmax, alpha=0.001, Augmented Dataset	0.393	0.306	0.3263
Triplet Loss attempt	0	0	N/A

It is worth noting that the prediction accuracy in the test set for the models trained on augmented and on non-augmented data were nearly identical. Several factors can account for the lack of efficacy of our augmentation. First, the initial sparsity of the dataset meant that any diversity we were introducing through augmentation was still a manipulation of a very limited set of images (and in over half the classes, of only one image). Second, the allowed range for our random rotations of +/- 20° and the allowed range for our random shifts of 0.2 * the image size were likely too severe, changing our initial input so drastically that the resulting image was not useful for training.

We also must address the complete failure of the triplet loss model to improve in accuracy, even after 50 epochs. In retrospect, we should have anticipated this: triplet loss is meant to replace triplet loss by building encodings in differentiated embedding spaces for each class. We simply replaced the loss function in our standard softmax model without building the data flow around it to support the pairing of anchor images with positive and negative companions. Therefore, our backpropagation never provided our weights with meaningful information, and there was no way the model could improve.

5 Lessons learned and future paths

5.1 Approaches to Deep Learning

As two non-CS students with little coding experience, one of the biggest lessons from this project was a newfound understanding of the extensive backend work required to actually deploy machine learning models in the wild. We spent so much time building our workspaces and data pipelines that we were left with much less time to tweak our models than we would have liked. That said, we did come away with some solid insights from our models:

- Our error analysis found that 3 classes were present in >90% of our predictions. Our models ended up optimizing based on probability rather than distinguishing features.
- Our results were in the middle of the pack for participants in the Kaggle challenge
- Data augmentation cannot overcome extreme sparsity when training triplet loss

5.2 Data Augmentation

Because fewer than 5% of our classes had more than five samples, we knew that training would be a challenge even with data augmentation. Indeed, multiplying the training image set by 6x had negligible impact on performance. In future, we would explore the data augmentation process with a bit more nuance. Some approaches of interest are:

- Utilize Keras Image Preprocessing to implement real-time data augmentation, multiplying our effective dataset and ensuring the model can't overfit to our limited set of augmented data during training time.
- Evaluate performance with different allowed ranges for our random transformations.
- Utilize cross-validation to predict model performance before training the final model for deployment. This way, all training data is used to train the model. With our 80/20 test/dev split, 10% of the whale classes weren't even trained on.

6 Contributions

Marc handled the initial data renaming process to ensure our file labels were appropriate for our model inputs. He ran and analyzed the results from the cross-entropy softmax models and modified the code so that we could output five predictions per input image, as specified by the Kaggle rules. David handled the data augmentation pipeline and the triplet loss and transfer learning attempts, and set up our AWS environment.

7 Github Link

https://github.com/dmzweig/whale_flukes

References

[1] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society, 2015.

[2] Lex Toumbourou. Humpback whale id: Data & aug exploration. Available at https://www.kaggle.com/lextoumbourou/humpback-whale-id-data-and-aug-exploration/notebook(2018/03/21).