CS 543 Final Project Report
Humpback Whale Identification

Ram Gupta
rgupta31@illinois.edu

Ashwin Ramesh
aramesh7@illinois.edu

Suyup Kim
skim361@illinois.edu

May 6th, 2019

1 Introduction

A very important task in the conservation of humpback whales is the ability to identify them in the wild. For years the task of uniquely identifying a whale based on just a glance at their tail fluke has been done manually by scientists. While difficult, this has generated enough data for us data scientists to be attempt to automate the process. This dataset has been made available as part of a kaggle competition. Given ∼20,000 images of the tail flukes of humpback whales, could we automate the identification of these whales? For our final project, we planned to take up this task. We tried several approaches using classical image processing techniques combined with machine learning models for classification. As expected, a comprehensive deep learning pipeline was necessary to make any kind of progress on this problem.

1.1 The Data and Initial Challenges

The dataset is the HappyWhale dataset of images of humpback whales' tails, and our job was to identify the whale to whom the tail belongs uniquely, or state that it was a new whale if we could not. The training set consists of 25,361 images of 5004 whales known whales, and some unidentified whales. The test set consists of 7,960 images.

Whale Id	**Count**
new_whale	9664
w_23a388d	73
w_9b5109b	65
w_9c506f6	62
w_0369a5c	61
...	...

Table 1: Class imbalances near the upper end

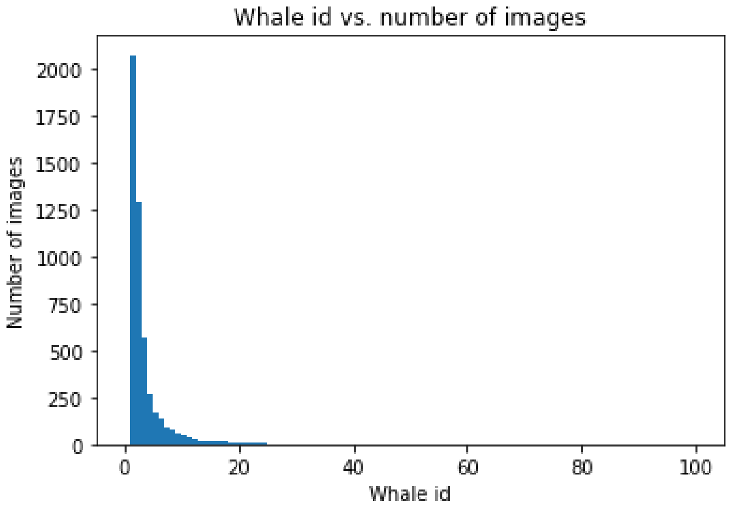


Figure 1: : Number of images per whale

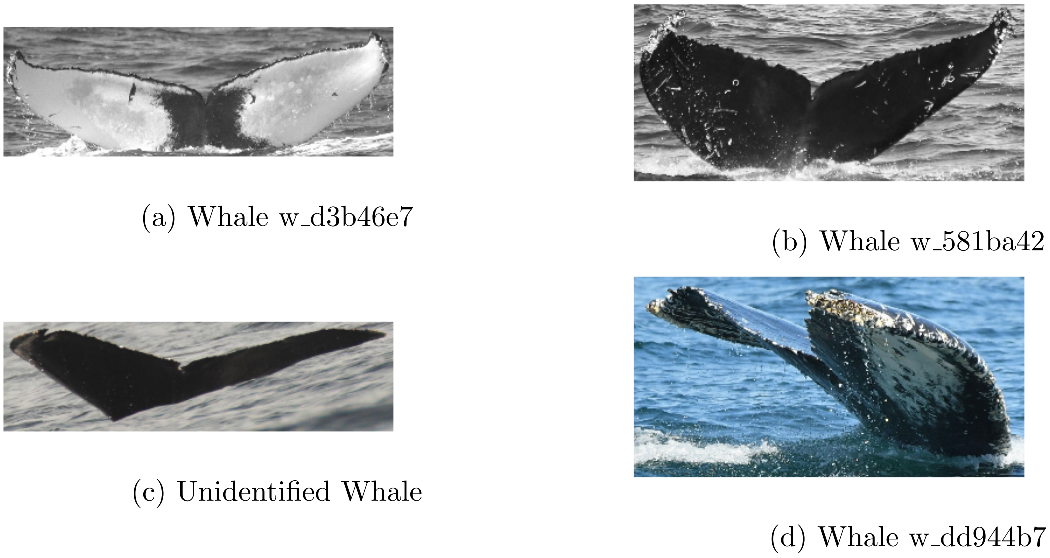
It is obvious that the problem is an instance of image classification with a large number of classes, however, when we took a look at the dataset, we discovered that there were large class imbalances in the dataset. For the remainder of this report when talking about “classes” we refer to individual whales. There were over 9,000 instances of unknown whales, and only 73 instances of the most frequent known whale. Table 1 shows the value counts of the first few most common whales in the training set. Here we see our first problem. The *new_whale* class is vastly over-represented. Figure 1 shows our second problem. The value counts of the whale classes falls off exponentially. In particular, more than *half* the classes have less than 5 examples, and for over 2,000 identified whales there is only one example image. This was a challenge we had to overcome with data augmentation, as described in later sections.

There were also some challenges that had to do with the images themselves, namely, they were not of uniform size, nor were they all color or grayscale. It is quite a heterogeneous dataset. Figure 2 shows four examples of images whose widths have been made uniform for display purposes. In reality, they all have very different dimensions, and some have fewer channels than others.

1.2 The Evaluation Criterion

The evaluation criterion for kaggle submissions in this competition was Mean Average Precision for upto 5 predictions. The submission format was 5 ranked labels for each of the test images. The submissions were scored using the following criterion.

$$MAP5 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{min(n,5)} P(j)$$



(a) Whale w_d3b46e7

(b) Whale w_581ba42

(c) Unidentified Whale

(d) Whale w_dd944b7

Figure 2: Variety in data

$P(j)$ is the precision (percent correct predictions) at cutoff k. This means that only the predictions from position 1 to j are considered. Here n is the number of submitted predictions for that image.

2 Methods

2.1 Overview of Methods

Our plan of attack for this task was to try a few different types of methods and see which worked the best. In our proposal, we proposed the following:

- Probabilistic models like Naive Bayes.
- Decision forests (Random Forest Classifiers)

However, we tried these approaches, and found that they performed very poorly on high dimensional, structured data like images. The assumptions on feature distributions in probabilistic models like Gaussian Naive Bayes do not hold at all here, and are therefore inappropriate. Models like RandomForest required extensive feature extraction, and did not give us correspondingly better results. Here we have shown a few methods that we thought were worth discussing. Our general pipeline consisted of 3 interconnected stages. The parts within a stage were options among many.

1. **Stage 1 : Preprocessing**
    - Data Preprocessing
        - Resizing
        - Grayscaling

    - Scale Normalization

- Data Augmentation: This consisted of primarily adding images to our dataset.
    - Randomly Flipped/Rotated/Cropped and Resized copies
    - Correcting class imbalances by oversampling under-represented classes (a lot of labeled whales had very few examples)
- Keypoint detection and Matching

2. **Stage 2: Training**
    - K-Nearest Neighbors
    - Pretrained ImageNet models (ResNet18, ResNet152 ...)
    - Bounding Boxes and ResNet50
3. **Stage 3: Post-processing and Testing**
    - Confidence Adjustment to account for unidentified whales
    - Classwise analysis

We'll describe each of the three methods we tried to implement and discuss their merits and shortcomings.

2.2 Keypoint Extraction, Matching and Classification

One of the first and most naive implementations we tried was identifying keypoints in the training images and trying to match them to keypoints in test images with a brute-force matcher using *opencv*. Of course, the largest barrier to this method was the high resolution of most of the images. We downsampled the images and resized them to a uniform size before commencing with this method.

2.2.1 Keypoint Extraction

We tried two main feature extraction methods namely, SIFT [3] and ORB [4]. Thankfully, there exists at least one version of opencv that provides functions for both detection and descriptor computation methods. Keypoints on whale tails are difficult to identify. In [5], Weideman et al. identified whales mainly by the contour of the edges of their tail flukes. Figure 3 shows examples of whales with SIFT key points on the left and ORB keypoints on the right.

A note on ORB vs SIFT descriptors: We found, as seen above, that ORB was far more robust to the background noise due to the ocean than SIFT. ORB is also much faster, which for as many training/testing combinations we had, added up. For these reasons we decided to use ORB keypoints and descriptors for our naive implementation.

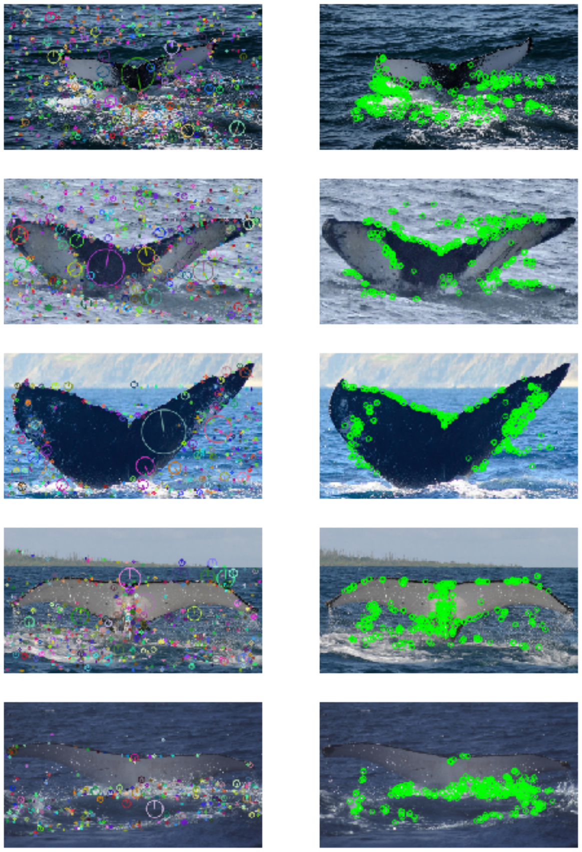


Figure 3: Left: SIFT keypoints. Right: ORB keypoints. Keypoints from the two methods drawn on top of some example whales

2.2.2 Keypoint Matching

We used a brute force matcher using *Hamming* distance as is normally done with ORB descriptors to calculate matches. Figure 4 shows an example match. This is how we were able to match keypoints between two images, but in order to score the matches, we had to compare them across all pairs of testing and training images. How do we score a pair of images? We decided to go with average match distance, since this was invariant to number of matches.

2.2.3 Classification

We decided to go with a K-Nearest Neighbor approach, as a first pass. In this approach, we took the average match distance matrix we had across all training and test pairs, and labeled each test image with the mode of the top K closest training images. Due to the low number of representatives for many classes it didn't make sense to have K¿1, so we just took the label of the closest training image. Our results are in the Results section.

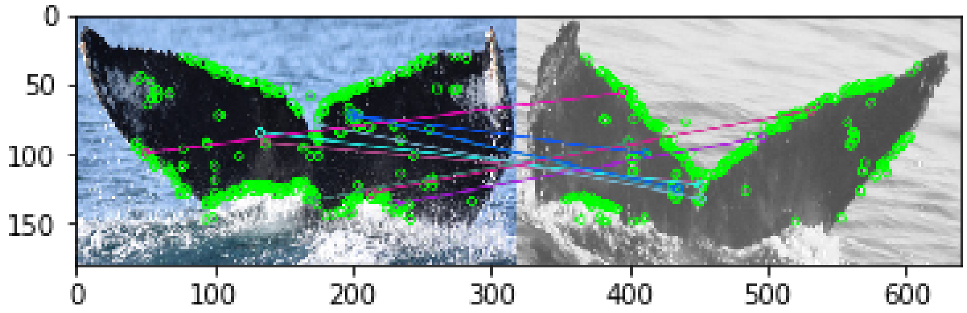


Figure 4: Matches between ORB keypoints between two tail flukes

2.3 Transfer learning using pretrained ResNet

One of our main avenues of attack in trying to perform the classification task was to use transfer learning using a proven pretrained model. For this, we tried various versions of ResNet [2] - ResNet18, ResNet50, Resnet152 with various layers in each of the models frozen.

2.3.1 Data Preprocessing and augmentation

The first challenge in processing the images was that they were all of different sizes, and some were colored and some were grayscale. As part of preprocessing, we reduced all of them to a fixed size (for example 224x224 pixels) when loading them into memory. This was necessary since otherwise the images didn't fit in memory, which made the training process go really slow. We tried learning models with both grayscale as well as colored images, and we discovered that ResNet already works with colored images; and performance with colored images being better, we converted the single-channel grayscale images to "colored" ones by copying the grayscale channel to two more channels to imitate RGB.

The next major challenge was that the frequency of occurrences of whales (number of classes) in the trainining dataset was highly imbalanced as discussed in section 1.1. To tackle this, we oversampled the underrepresented classes by adding copies of images to the training set to bring the frequency of all classes (whales) to 5 whales.

2.3.2 Models

We tried primarily to transfer learn using ResNet18 and Resnet152, and various configurations of frozen layers, learning rates, and preprocessing (without and with data augmentation). More information in the Results and Discussion sections.

2.4 Bounding Boxes and ResNet50

Naturally, our main method of noise reduction in the dataset (background ocean, small whale flukes) was to draw and crop to bounding boxes drawn around the animal in the image. Like many people in the original competition, our work was inspired and largely based on the initial work of the winner of the first Humpback Whale challenge, Martin Piotte [1], who trained a convolutional network model to estimate bounding boxes on the dataset.

Our architecture is loosely based on an adaptation of that network where we add 4 layers after a backbone of ResNet50. Table 2 shows a summary of the architecture we used.

Layer
ResNet50
BatchNorm1d
Dropout
Linear
LogSoftMax

Table 2: Bounding Box Model Architecture

In addition to bounding boxes, we tried to further make the data more uniform by using a reference image in the training dataset for image alignment. Such a technique is used in facial detection and recognition. We computed descriptors and matched keypoints to compute a homography for each training image with this reference image, however when we applied the homographies to the target images, they deformed in a way we could not debug.

3 Results

3.1 K-Nearest Neighbors

The first of our 3 methods performed quite poorly. Table 3 shows the test accuracy for our keypoint extracted nearest neighbor classifiers. We tried K values ranging from 1 to 100 but found as expected that the elbow point for this dataset was quite low. We concluded that the best value was actually 1 neighbor.

K value	**Test Accuracy**
1	0.0965
2	0.084
5	0.0733
10	0.04

Table 3: Shows the accuracy for various values of K in the ORB-Nearest Neighbors

3.2 Transfer learning

The table below highlights our performance with some specified values of the type of model, number of layers used, number of epochs. These results were obtained with a learning rate $lr = 0.01$.

We tried lower learning rates in the range [0.001, 0.01], but these learned too slowly to produce good results. For example, running ResNet18 with learning rate $lr = 0.001$ and

Model	Layers	# epochs	Train accuracy	Val accuracy	Test accuracy
ResNet18	Last only	50	54.33%	24%	3.61%
ResNet18	All	20	81.16%	58%	8.491%
ResNet152	All	15	84.19%	55%	-

Table 4: Results for transfer learning with various techniques

training only the fully-connected layer, after 200+ epochs our model only got to around 50% accuracy and 32% validation accuracy. That being said, we think that lower learning rates do overall produce better results. Unforunately, we were unable to record exact results of a lot of earlier runs with fewer preprocessing steps due to misconfigurations and time constraints. One of the higher test scores that we verifiably achieved was 15.716% on the public leaderboard, with an objectively weaker preprocessing pipeline. Also, further training of this same model reduced the accuracy, presumably because of overfitting.

3.3 Bounding Boxes and ResNet50

Here our model learned much more quickly, but it became tough for us to reduce overfitting on this model. Partially freezing some of the layers did not work as expected. However, despite all this we recieved a modest validation bump of about 3-5% on average from the Transfer learning models. Our test accuracy however, was not too much higher at 16%. This probably had to do with tuning hyperparameters. This approach requires a little more understanding on our part.

4 Discussion

We set out to make the leader board in this competition, but this turned out to be an ambitious goal. We settled with seeing how various vision methods perform on this task, and documenting our results. Here we discuss possible explanations and do a rough analysis of our results.

4.1 SIFT/ORB and Nearest Neighbors

This method suffered from several flaws. The poor performance can be attributed to many factors. Firstly, we lost a lot of data when we down-sampled the images to a size small enough to compute key points, matches, and distances in a feasible time period. We estimated that at full resolution, it would take more than 24 straight hours to just compute matches between all pairs of test and train images. Secondly, our distance metric, average Hamming distance across all matches, may be poor. Finally, nearest neighbor classifiers tend to underfit and suffer from the *curse of dimension* even with down-sampled images. This combination of requirements is yet another thing that points to the necessity for deep learning models. We didn't see any benefit by increasing the value of K, as for the non-augmented dataset, some training classes have only one example.

4.2 Transfer Learning

Transfer learning using even state-of-the-art pretrained networks has limitations; on the whale detection problem posed in this project, we achieved only mediocre success. The biggest reason for this is that ResNets which are trained on ImageNet, although well-trained in recognizing objects in images in general, are not trained for recognizing minute differences in the unique shapes of whale flukes, which is the main information in this dataset's images. Significant noise is caused by the waves in the water, similar to how it's picked up by the keypoint detectors discussed above. A lot of successful attempts at this whale classification task thus use bounding boxes to limit computational analysis of neural nets or other models to only the whale flukes.

The various preprocessing techniques we employed were educational in that they demonstrate additional preprocessing if not done correctly, or if not trained enough can actually result in worse results, as is evidenced by our high train and validation accuracies compared to the test accuracy in several configurations. The various half-frozen network configurations we tried yielded worse results than just training the last fully-connected (FC) layer, or fine-tuning the entire network.

4.3 Bounding Box Model

We were not able to get a significant increase in validation and testing accuracy with this model. We need to more accurately understand how to combine the bounding box learning model with other models in our pipeline. The standard deep learning interfaces such as what to do when our model is overfitting, and what to do when it is underfitting were not enough to properly apply this model.

4.4 Future Work

The key to this project, we found out closer to the deadline, was one-shot learning, and an effective representation of each whale. On representation, we needed to concentrate more on the pattern of contours on the edges of the whale tail flukes, rather than trying to automatically learn those features from scratch. A vector representation as in [5] is low dimensional enough to apply a strong learning technique for one-shot learning.

One such technique is Siamese Networks. Such a network consists of a head network and a branch network. One of the networks tries to minimize the distance between representations between examples of the same class and the other tries to maximize the distance between representations of other classes. We did not have the time nor the knowledge to complete this attempt for this project, but we will do so in the future.

5 Member Roles

We ended up pair programming most of the setting up the dev environment and collecting the dataset, etc. We initially tried using a VM, but then we settled on using Google Colab for the free GPU access it provides. We are used a shared Jupyter notebook on Google Colab. Broadly speaking, the work was split up in the following way:

- Ashwin Ramesh : Explore methods and parameter optimizations.
- Ram Gupta : Explore data transformations and structural optimizations
- Suyup Kim : Explore training/testing regimes and model selection.

References

[1] Bounding box model. `https://www.kaggle.com/martinpiotte/bounding-box-model`. Accessed: 2019-05-06.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. Orb: An efficient alternative to sift or surf. Citeseer.

[5] H. J. Weideman, Z. M. Jablons, J. Holmberg, K. Flynn, J. Calambokidis, R. B. Tyson, J. B. Allen, R. S. Wells, K. Hupman, K. Urian, et al. Integral curvature representation and matching algorithms for identification of dolphins and whales. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2831–2839, 2017.