HUMPBACK WHALE IDENTIFICATION USING MACHINE LEARNING ALGORITHMS FOR COMPUTER VISION

Amit Asghar, Sophie Lee, Julia Mitroi,
and Varun Muriyanat
York University

March 15, 2019

Abstract

Individual identification of humpback whales is of great interest to scientists and marine biologists because identification plays a key role in their long-term studies of the population and behavioural patterns of the whales, and aids conservation efforts. Humpback whales are identified by the underside and trailing edge of their tail flukes; each one is different, just like a fingerprint. Machine learning methods can be used to automate photo identification of whales, by building algorithms that process patterns in images, such as Convolutional Neural Networks (CNNs), in conjunction with a whale photo-identification system. This paper describes our solution to the Kaggle Humpback Whale Identification challenge, a computer vision competition that involved identifying individual whales by a picture of their tails. The data that we analyzed is from the Happywhale database of over 25000 images, gathered from research institutions and public contributors. To improve model performance, we preprocessed the images to reduce their size through Python's Keras library. We compared the performances of a Siamese Network Architecture (a CNN variant tailored to Few-Shot Learning tasks, where we must correctly make predictions given only few examples of each class) and a regular CNN, and found that the Siamese performed better than the regular CNN.

1. Introduction

Identifying whale species and individual whales is a key step in accessing information related to their ecosystem, and is important in efforts to avoid extinction and to help whale species recover to sustainability. A history of whaling has made recovery difficult for whale populations, and this difficulty has been compounded by warming oceans and the struggle to compete daily with the industrial fishing industry for food. Identifying whales from shipborne, drone, or helicopter imagery can be a tedious task, and scientists, in their whale conservation efforts, use photo surveillance systems to monitor ocean activity and identify humpback and other types of whales. A main approach is to use the shape and colour of whales' tails, and unique markings found in footage, to identify what species of whale are being analyzed. For example, humpback whales' tails have white spots or undersides and ridges on the ends.

In the 1970s, scientists discovered that individuals of many species could be recognized by their natural markings [1]. Using natural markings to identify individual animals over time is usually known as photo-identification (or photo-id). This research method is used on many species of marine mammals. For humpback whales, the underside of their tail's flukes has a unique pattern, like a fingerprint. In the context of the photo-id technique, to identify individual humpback whales, marine biologists have photographed the flukes (two sides of the whale's tail), made notes on the location where the whale was spotted, and associations with other whales. The photograph was then "matched" with other photos of a particular whale, and filed in a catalogue. Each photograph within these catalogues documents the sighting of a whale at a specific time and place; often, the same whale seen in places thousands of miles apart at different times of the year [2]. As the photographic collections of whales grew, so did the need for more efficient retrieval methods that would allow a researcher to quickly compare and match new photographs against a database of registered individual whales. In order to improve the monitoring of humpback whales, leave less data untapped, and make the process scalable to handle larger amounts of data, an automated image identification system was needed. The Kaggle competitions on this topic contribute to research efforts to help automate individual humpback whale identification through computer vision techniques.

Our work, undertaken as part of the Humpback Whale Identification challenge on Kaggle, presents and compares several machine learning models, which take whale tail images as input, extract features, and identify the individual whales in the images. In all cases, the classification of the whale (model output) is based on Convolutional neural networks (CNNs) and their adaptations.

A challenge with the dataset that we used was dealing with a skewed class distribution. While a few classes have many samples, the majority of the classes contain less than three examples, making the problem subject to 'Few-Shot Learning' [3]. In the case of few-shot learning, a classifier must learn to recognize new classes given only few examples from each. Our approach to Few-Shot learning was to first determine similarities between images based on their lower dimensional embeddings, and then make the classification, which is specific to a Siamese Networks Architecture (Koch et al., 2015) [4].

2. Related Work

The ability to recognize individual humpback whales from photographs of their tail flukes' pigmentation and scarring patterns was first achieved by researchers in the 1970s [5]. Since then, the method of photo-identification has been widely used on humpbacks and other whale populations around the world to identify and monitor individual whales and gather valuable information about population sizes, migration, health, sexual maturity and behaviour patterns. Most of the photo-identification approaches documented in the literature are manual computer-based approaches.

Mizroch et al. (1990) [6] developed a framework of manually-generated code based on a set of thirty-eight generic fluke patterns, which considers the shape of the central notch and the location of blotches/scars (relative to a specific area of a standardized fluke). After the code is run, the resulting descriptors are subsequently used for matching. Araabi (2000) [7] extended a curve-matching technique, originally developed for the identification of bottlenose dolphins, for the encoding of the fluke's trailing edge of humpback whales. Ranguelova et al. (2004) [8] introduced an identification framework based on light and dark pigmentation patches on the whale flukes. They developed an interface that assists the user in segmenting the whale's tail from the sea, and fitting an affine invariant coordinate grid to it; a numerical feature vector capturing the patch distribution relative to the grid is then automatically extracted and used to match the individual photo against a database of similarly processed images.

Kniest & Burns (2010) [9] proposed a software application to manually query images and compare them to an existing catalogue of humpback images. This comparison to the catalogue is made by using fixed features, such as black and white patches, or certain shapes and edges, as well as user defined features which describe spots, areas, scars, or other image features. WhaleNet (2015) [10] is a graphical user interface (GUI) which allows the user to narrow down the search for matches by visually selecting one of eighteen fluke types.

The above-noted approaches all have in common that features are hand-crafted, not learned by an algorithm. Until recently, none of the systems used for computer-aided identification employed the wide range of features and unique properties of humpback whale flukes that can be utilized to more efficiently identify individual humpbacks. This led to a need for improvement of identification methods through the automatic extraction of features, as well as an increased scalability by automatically identifying the whales. Attempts to develop a computerized automatic whale image identification system based on machine learning algorithms are still in their infancy, and several Kaggle competitions have facilitated collaboration and access to whale photographs for algorithm development, both for humpbacks and other types of whales.

The Right Whale Recognition competition on Kaggle involved the classification of individual right whales (Eubalaena glacialis), given an image with an aerial shot from a right whale. The leading competitors used a pipeline of regular CNNs, as well as several CNN-based pretrained models (e.g., AlexNet, DeepSenseNet) to first extract the region of interest – a characteristic pattern on the whales' heads – and to then classify based on those extracted regions. The class distribution of the underlying data set of this competition was less imbalanced than in the humpback whale competitions, the ratio of classes with only a few images compared to the total number of classes being rather low. As such, the proposed algorithms for right whale recognition are not applicable to Few-Shot Learning, but are useful examples of automatic identification of individual whales from photos.

As stated, the Happywhale data associated with the Kaggle Humpback Whale Identification challenge is of the Few-Shot Learning type, with few samples for many classes. A framework tailored to an automatic feature extraction, scalability, and to the issues brought about by Few-Shot Learning was developed by Schroff et al. (2015) [11] for human face recognition for YouTube users. Their method consisted of a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer in the network, as in previously-developed deep learning approaches. Their compact embedding lent itself to be used to cluster YouTube users' personal photos into groups of people with the same identity.

In their solution to an earlier version of the Humpback Whale Identification Kaggle competition, Botler et al. (2018) [12] transferred Schroff et al.'s idea of clustering and recognizing faces to the clustering and recognition of unique whale flukes. They preprocessed the whale tail images by automatically cropping the areas of interest, developed two machine learning algorithms, a regular CNN and a Siamese Network Architecture, and found that the CNN slightly outperformed the Siamese Network.

3. Methodology

3.1. Data Understanding

3.1.1 Data Collection

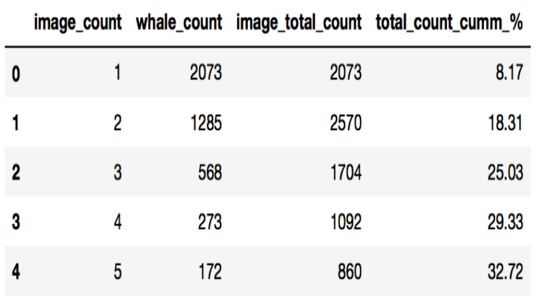
The data used for this project was sourced from a public dataset from the Humpback Whale Identification Kaggle competition's repository at https://www.kaggle.com/c/humpback-whale- identification. This training data contains thousands of images of humpback whale flukes. Through the Happywhale platform noted in the Introduction, individual whales were identified by researchers and given an ID. The challenge (goal of this project) was to predict the whale ID of the images in the test set.

3.1.2 Dataset Description

The Humpback Whale Identification dataset consists of three subsets: training images, test images to predict the whale ID, and a set of whaled IDs that maps the training image to its whale ID. Whales that did not have a label identified in the training data were labeled as "new_whale".

3.1.3 Data Preprocessing

To improve model performance, we preprocessed the training images to change their size to 100x100x3 pixels (retaining the RGB channels) using Python's keras.preprocessing.image.load_img() function, and convert them into an array.



	image_count	whale_count	image_total_count	total_count_cumm_%
0	1	2073	2073	8.17
1	2	1285	2570	18.31
2	3	568	1704	25.03
3	4	273	1092	29.33
4	5	172	860	32.72

Figure 1: *Five Highest Whale ID Counts*

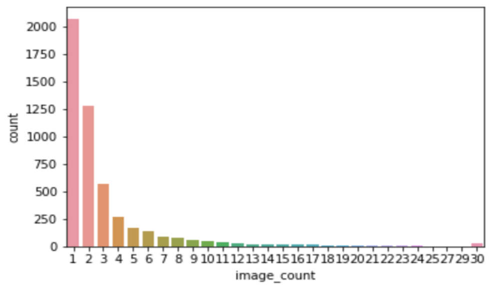


Figure 2: *Whale ID Distribution*

3.2. Exploratory Data Analysis

The training data consists of 9850 images while the test set consists of 15610 images. We performed summary statistics, and charted patterns and distributions of the IDs from the Happywhale dataset that we downloaded from the Kaggle competition's website.

As expected based on the description on Kaggle, the distribution of images per whale is highly skewed. More than 2000 whales have just one image, and the single whale with most images has 73 images. From the cumulative proportions calculations in Python, we determined that 41.4% (2073) of the total unique IDs came from the 'new whale' group (no ID), almost 30 percent (29.33%) came from whales with 4 or less images, and the rest of 30 percent, comes from whales with 5 to 73 images. The table in Figure 1 shows the counts and cumulative counts for the 5 highest whale ID counts (whale count column), and Figure 2 depicts this distribution graphically.

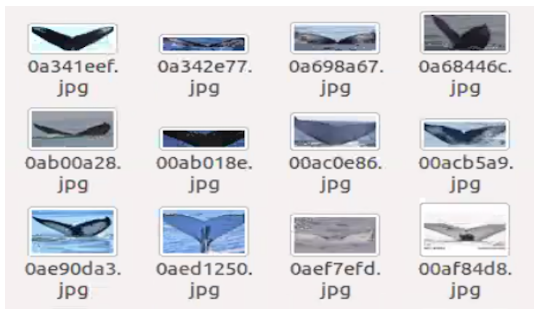


Figure 3: *Whale Fluke Samples*

Regarding the quality of the images, we noticed that the pixel resolution of the images differed between the whale images. The rectangular shape of the images suggests that most images have been cropped to center the fluke better. Figure 3 shows a sample of the whale images in the training set.

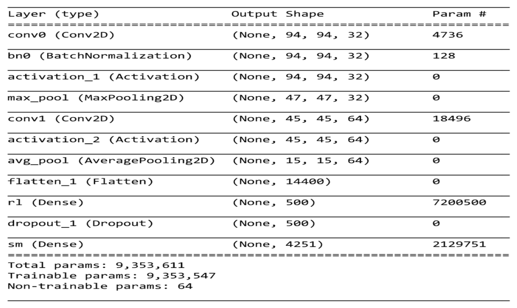
3.3. Classifiers

We used the Keras deep learning package to build our CNN and Siamese Network. Keras is a minimalist Python library for deep learning that can run on top of TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

3.4. Modeling

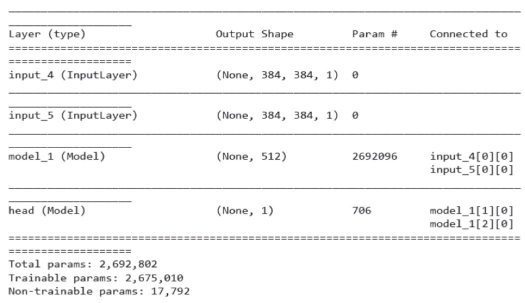
3.4.1 Regular Convolutional Neural Net

Our main solution to the for the Humpback Whale Identification challenge is based on a CNN Architecture using Keras. Wresized the images using Keras, as noted above, and prepared the labels by converting them to one-hot vectors. We did not split the training set to create a test set, because many of the whale classes have only one picture; randomly selecting a test set would have been detrimental for training. The table in Figure 4 shows the structure of our regular CNN.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv0 (Conv2D)               (None, 94, 94, 32)        4736
_________________________________________________________________
bn0 (BatchNormalization)     (None, 94, 94, 32)        128
_________________________________________________________________
activation_1 (Activation)    (None, 94, 94, 32)        0
_________________________________________________________________
max_pool (MaxPooling2D)      (None, 47, 47, 32)        0
_________________________________________________________________
conv1 (Conv2D)               (None, 45, 45, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 45, 45, 64)        0
_________________________________________________________________
avg_pool (AveragePooling2D)  (None, 15, 15, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 14400)             0
_________________________________________________________________
rl (Dense)                   (None, 500)               7200500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
sm (Dense)                   (None, 4251)              2129751
=================================================================
Total params: 9,353,611
Trainable params: 9,353,547
Non-trainable params: 64
_________________________________________________________________
```

Figure 4: *CNN Network Structure*



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 384, 384, 1)  0
__________________________________________________________________________________________________
input_5 (InputLayer)            (None, 384, 384, 1)  0
__________________________________________________________________________________________________
model_1 (Model)                 (None, 512)          2692096     input_4[0][0]
                                                                 input_5[0][0]
__________________________________________________________________________________________________
head (Model)                    (None, 1)            706         model_1[1][0]
                                                                 model_1[2][0]
==================================================================================================
Total params: 2,692,802
Trainable params: 2,675,010
Non-trainable params: 17,792
```

Figure 5: *Siamese Network Structure*

3.4.2 Siamese Network Architecture

As an alternative to the regular CNN, we also used a Siamese Network model, that is tailored for Few-Shot learning problems, adapted from the notebook of this Kaggle competition's winner (Martin Piotte). As noted in a previous section, the Siamese model has been used successfully for tasks similar to whale identification, such as the identification of individual faces by a small amount of training samples for each face. Also, the architecture allows for an arbitrary number of classes, and an implicit detection of duplicates. The table in Figure 5 shows the structure of our Siamese network.

4. Results

For our regular CNN, we ran the Python code for the CNN on a GPU-optimized * GB RAM Windows machine and obtained an accuracy of 89.9% after training the CNN over 100 epochs (Figure 6).

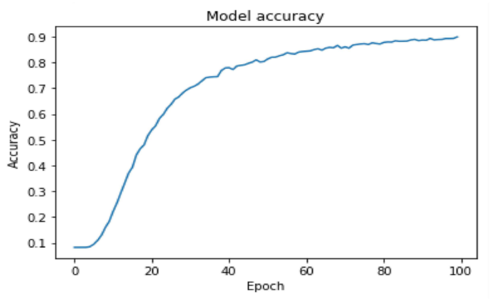


Figure 6: *CNN Model Accuracy*

Finally, we prepared the images using the same class applied to the training data, and obtained the predictions based on the CNN model. The code chose the predictions with the highest probability (up to five options, according to the challenge's rules) and used a threshold to choose how many predictions to make for each image. The one-hot vectors corresponding to the chosen predictions were transformed back to their corresponding names and printed in the submission file, which we submitted at the Kaggle website

The accuracy we obtained with the Siamese model was higher than the accuracy with the classical CNN, namely 97% versus 89.9%. This confirms the suitability of Siamese Networks for Few-Shot learning data structures.

5. Discussion

Comparing the two approaches to modeling that we used, the classical CNN architecture has the advantage of its simplicity compared to the Siamese architecture. For example, its architecture can be optimized in a relatively easy way by choosing a suitable pre-trained model and varying the number of layers. In addition, the validation process with CNN is much leaner, as we just need to feed the validation data to the network. On the other hand, the Siamese network requires creating and storing all training embeddings first, in order to compare the validation embeddings against them afterwards.

For the application of classical CNNs to our project, there are significant drawbacks however. One main issue is that there is no trivial way to add new classes to the data set. Adding new whales in the future would require changing the layout of the network given that the last dense layer would require additional neurons. Another drawback specific to this dataset is that due to the forgetting nature of CNNs, it is not guaranteed that the network will classify duplicate images correctly, while the Siamese Network allows for an implicit detection of duplicates. Thus, for these reasons, a classic CNN Architecture is less suited for Few-Shot learning tasks.

Another key takeaway is the importance of manually examining the data in depth, which allows for detection of parts of the data set which are not useful for the training process: Preparing and cleaning the data is in fact at least equally important as choosing a suitable network architecture and optimizing the parameters.

References

[1] Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.-P., Planque, R., Rauber, A., Fisher, R., Müller, H.: LifeCLEF 2014: multimedia life species identification challenges. In: Kanoulas, E., Lupu, M., Clough, P., Sanderson, M., Hall, M., Hanbury, A., Toms, E. (eds.) CLEF 2014. LNCS, vol. 8685, pp. 229–249. Springer, Heidelberg (2014).

[2] Fox, D. (2016). National Geographic: Who's That Whale? Your Photo Could Help I.D. a Humpback. Retrieved from: https://www.nationalgeographic.com/adventure/adventure-blog/2016/05/04/whos-that-whale-your-photo-could-help-i-d-a-humpback/

[3] Deng, B., Liu, Q., Qiao, S., & Yuille, A.L. (2018). Few-shot Learning by Exploiting Visual Concepts within CNNs. Submitted to ICLR 2018.

[4] Koch, G., Zemel, R., & Salakhutdinov R. (2015. Siamese Neural Networks for One-shot Image Recognition. Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[5] Katona, S. K., B. Baxter, O. Brazier, S. Kraus, J. Perkins and H. Whitehead. 1979. Identification of humpback whales by fluke photographs. Pages 33–44 in H. E. Winn and B. L. Olla, eds. Behavior of marine animals—current perspectives in research. Volume 3. Cetaceans. Plenum Press, New York, NY, U.S.A

[6] Mizroch et al. (1990). Computer Assisted Photo-Identification of Humpback Whales, Individual Recognition of Cetaceans, International Whaling Commission, Cambridge.

[7] Araabi, B. Kehtarnavaz, N., Mckinney, T., Hillman, G. & Wusig, B. 2000. A string-matching computer assisted system for dolphin photoidentification. Annals of Biomedical Engineering 28:1269–1279.

[8] Ranguelova, E., Huiskes, M., & Pauwels, E. J. Towards computer-assisted photo-identification of humpback whales. In Image Processing, 2004. ICIP '04. 2004 International Conference on, volume 3, pp. 1727–1730, Vol. 3, Oct 2004. doi: 10.1109/ICIP.2004.1421406.

[9] Kniest, E. & Burns, D. Fluke matcher: A computer-aided matching system for humpback whale (megaptera novaeangliae) flukes, 2010. URL http://old.whaleresearch.org/articles/flukeMatcher.pdf

[10] WhaleNet. (2015). http:/whale.wheelock.edu.

[11] Schroff, F., Kalenichenko, D., & Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823, 2015.

[12] Botler, M., Bundea, A., Peters, C., & Schidlak, M. (2018). Open Kaggle: Humpback Whale Identification Challenge. Retrieved from: https://github.com/marschi/ kaggle_whale_challenge