Humpback Whale Identification: Few Shot Learning with Siamese Network and Adversarial Training

Runzhe Wan
rwan@ncsu.edu

Yiming Wang
ywang225@ncsu.edu

Min Zhang
mzhang27@ncsu.edu

Abstract

The problem of whale identification based on tail images is crucial to whale monitoring and conservation. It is also proposed as a Kaggle competition[1]. The minor difference in whale images and the limited and long-tailed dataset challenge us to design a robust and sophisticated neural network. The method we implemented here is to use the Siamese Neural Networks (SNN). The SNN trains a ConvNets model to extract domain specific features from two input images, and use those extracted features to make decisions on whether the two images belong to the same category or different categories. We design and implement several different neural network algorithms and compare the results.

1 Introduction

Rapid climate change and over-exploitation put a severe threat of extinction on whale species. Therefore, the protection of whale stares us in the face. To aid whale conservation efforts, photographic system has been utilized by scientists. They use the shape of whales' tails and unique markings found in footage to identify what species of whale they're analyzing and document the residence times, spatial and temporal distribution and interaction with human. In the past, the majority of this work has been done by hand, which is very inefficient and not precise. With the rapid development of computer vision techniques, it gradually becomes feasible to make this identification and tracking task automatic. This task was formally proposed by Happywhale and later became a Kaggle competition[1].

Our goal in this project is to design and implement several computer vision algorithms to identify whale ID based on images and compare their results. The difficulty of this problem is that we may only observe a single or a few examples of each possible class before making a prediction about a test instance. This is called one-shot/few-shot classification. For this task, we adopt the Siamese Neural Network approach and designed several different architecture. Many other tricks learned in class, for example, transfer learning and adversarial training are also used in this project.

2 Dataset and Features

We obtain the image and label datasets from the Kaggle competition [1]. Due to our limited computational resource, we randomly choose 953 images of totally 199 whales from Kaggle and use $\frac{2}{3}$ of them as the training dataset with the remaining as the testing dataset. Each of the images has

Preprint. Under review.

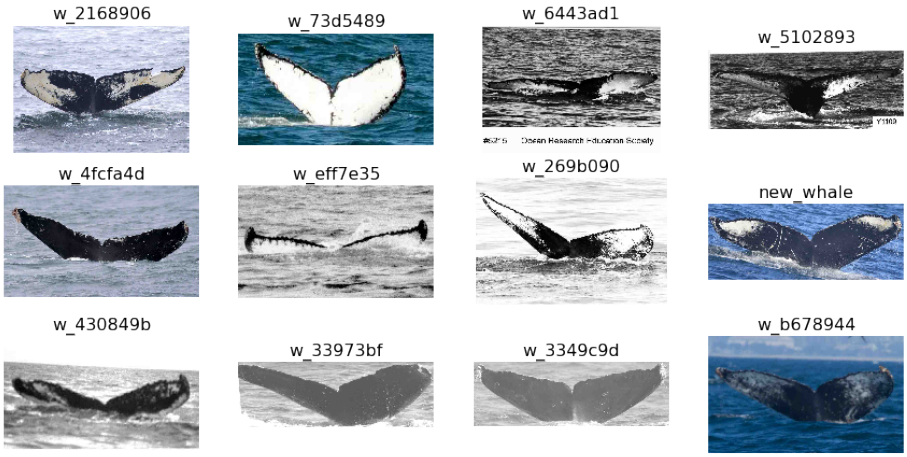

Figure 1: Sample images of whales

various and relatively high resolution. Figure 1 is an illustration of a random sample from them. For each image, we have its label as the identity of that whale.

It is noted that even images from different whale may still share great similarities. Another important feature of this task is shown by figure 2: the distribution of the number of images of each whale in our training set. It is noted that the number of images for each whale is unbalanced and for some whale ID, there is only 1 image.

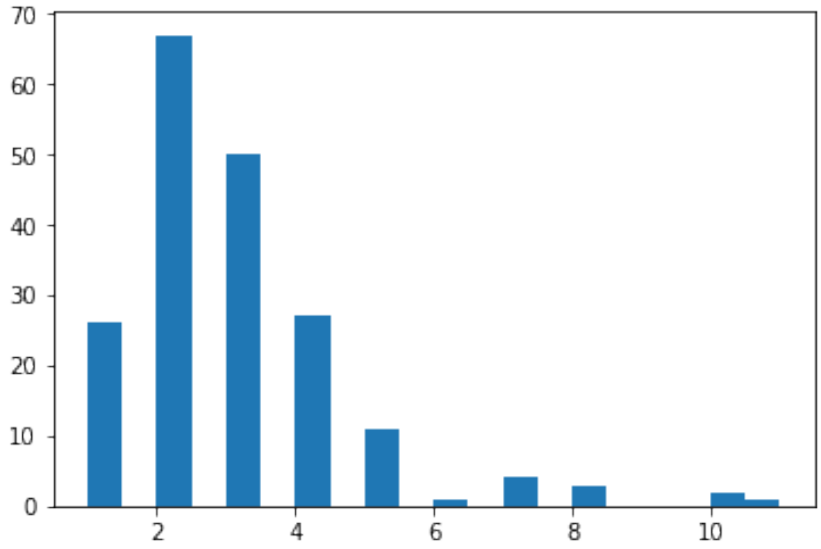

Figure 2: Class distribution

Clearly, it is different from the standard image classification task, like Imagenet, where we have a relatively large amount of images for each class. Thus, we need a different approach other than just simple CNN to tackle this problem. We will discuss about the Siamese network structure below.

3 Methods

3.1 Network Structures

One-Shot classification models is a particularly sub-problem of classification. The task is to make classifications under the restriction that we may only observe a single example of each possible

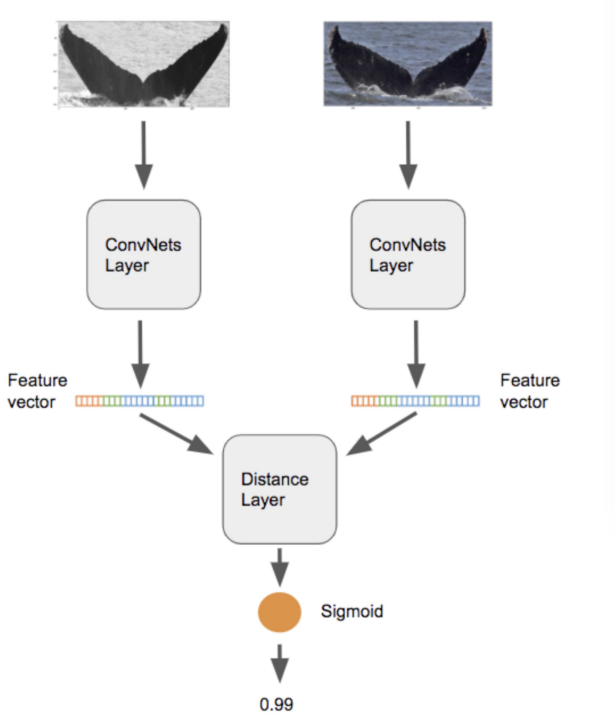

Figure 3: Illustration of the Siamese Neural Network architecture from [5])

class before making a prediction about a test instance. Siamese Neural Networks (SNN)[2] has been proved to be a good way to solve this problem.

The idea is to compare a pair of images and decide if the two images are taken from the same whale (positive pairs), or different whales (negative pairs). The entire SNN architecture contains a branch model (ConvNets Layer) and head model (Distance Layer). The Siamese Neural Network takes two distinct images as input, and trains a ConvNets model to extract domain specific features from the two input images. In the distance layer, the extracted features are used to make decision on whether the two images belong to the same class or different classes. Figure3 illustrates the structure of SNN.

With the Siamese network structure, we can turn the almost impossible classification problem with one or two images into a tractable classification problem where we only care about a binary output. The network will learn the commonality from all images and then focus on those minor difference.

For the Branch model, due to the limited computational resource, we tried two options: the first is to design a relative small CNN by ourself and train on our branch and head model simultaneously, which proved to be severely time-consuming. Another approach is to use pretrained neural network structure from Imagenet, for which we choose the Resnet50. With the technique of transfer learning, we freeze the branch model and only train on head model. This approach can spare us the burden of sophisticated annoying training process.

The Head model predicts the similarity between two feature vectors and decides if two images come from the same class. We adopted a neural network design as well as simpler distance metrics such as L1 or L2 distance, which are more popularly selected in models such as triplet model.

3.2 Adversarial training

With a large amount of pairs to choose from, how to decide the training pairs in each epoch is a vital problem in the Siamese Network. Some whale images, though come from different whales, appear to be hard to distinguish; where some other unmatched pairs have clear difference. With limited images per class, we need to make sure our network learn things from easy to hard. Thus, an adversarial training framework is required.

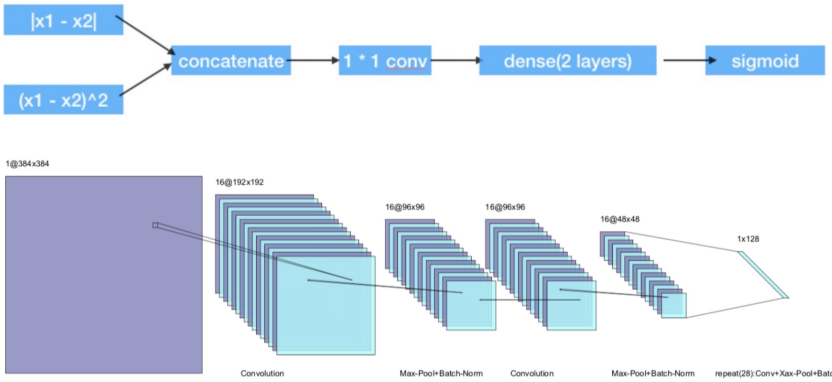

Figure 4: Self-defined Head model and Branch model

In each epoch, We choose N/4 unmatched pairs and N/4 matched pairs as the training data, and use them by turns. we first calculate the pairwise score of unmatched pairs with the current Siamese Network, and then choose r% unmatched pairs by greedily choosing the most hardest unmatched pairs, and the remaining 1- r % randomly. With epoch grows, we choose larger and larger r to make the distinguishing problem become harder and harder.

3.3 Image Preprocessing and Data Augmentation

For our training datasets, we first delete those duplicated images and images of "new_whales", which we do not find a good way to take advantage of. We use pre-trained bounding box results[6] to crop out the whale tail part, which shows a significant to help to the final performance. Due to the various color and size for those images, we choose to binarize the images and resize them all to 384×384.

For data augmentation, we conducted some commonly-used methods, like random zoom, shift, rotation and shear.

3.4 Other training settings

We use the binary cross entropy loss to predict whether or not the two images come from the same whale. And we select Adam optimizer to conduct stochastic gradient descent. We impose L2 penalty to avoid the over-fitting problem. The detailed schedules of the learning rate, penalty parameter and adversarial training parameter are chosen through a large amount of experiments and varies from different methods. They can be found in the attached code. We implemented the models using Keras with Tensorflow backend, and the pre-trained ResNet50 was also downloaded with Keras.

4 Experiments

4.1 Candidate methods

We report the performance of four attempts in our experiments:

1. CNN-NN: fully train the self-designed CNN branch model and the self-designed NN head model
2. CNN-L1: fully train the self-designed CNN branch model and L1-distance head model
3. Res50-NN: freeze the pre-trained ResNet50 as the branch model and only train the self-designed NN head model
4. Res50: directly run the 199-class-output ResNet50 to do classification.

4.2 Results

For each test image, we compare it with all training images and select the top 5 whales with the higher scores as candidates. Similar with the criteria in the competition, we evaluate the performance by the frequency that the true whale ID is among the predicted top 5.

Table 1: Results

Method	Random Guess	Res50	CNN,NN head	CNN,L1 head	Res50,NN head
Num of Epochs		100	100	20	200
Performance	2.5%	40.6%	13.4%	5.6%	4.6%

Due to the limited GPU resource, we do not have the opportunity to carefully tune our network, which may partly explain the unsatisfactory experiments results. However, the overall testing performance is significantly better than random guess, and are potential to be useful for real application with further tuning. Moreover, the overall pattern does show the better performance of fine-training the branch model than simply using ResNet50 because the difference of our task with Imagenet, and also the advantage of metric learning than simply using L1 distance.

4.3 Visualization

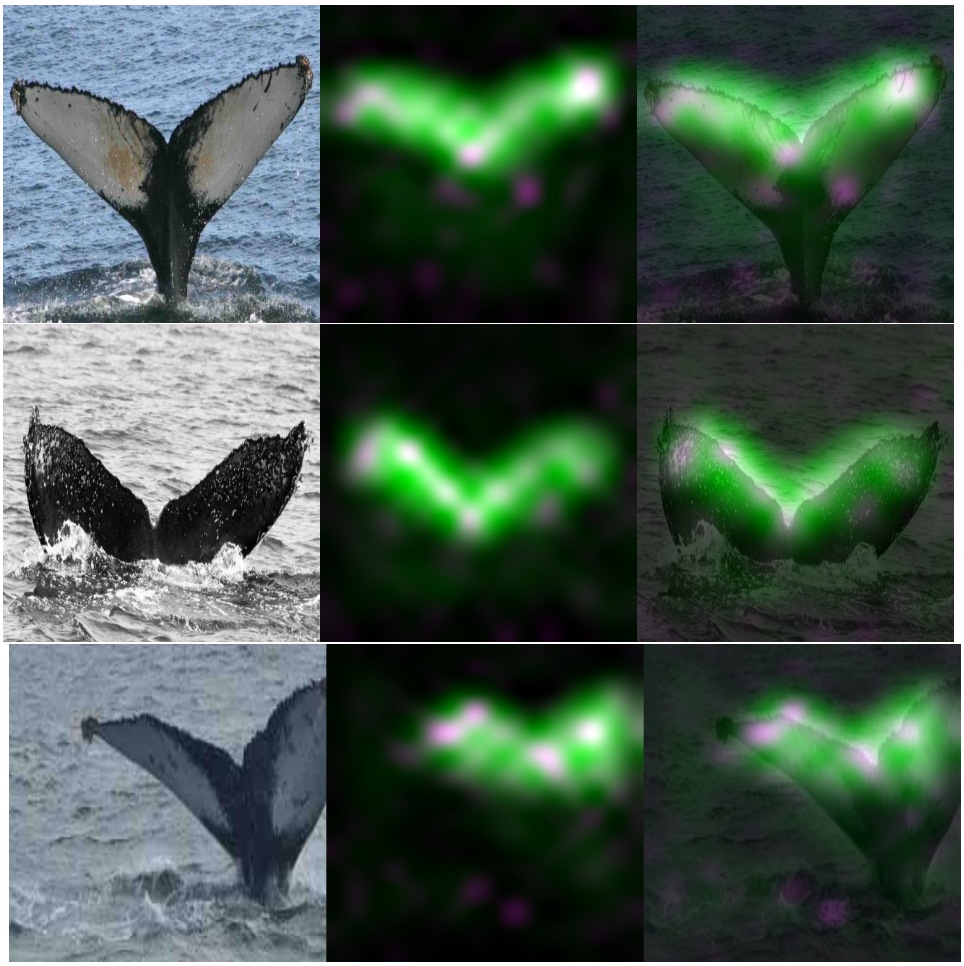

Figure 5: Some illustrations of what our Siamese Network is looking at

To understand what areas the Siamese network is actually focusing on when the input images are fed into the neural network, we use some visualization tools. We generated HeatMaps to emphasize what the neural networks recognized and focused on from whales.

The output of the last BatchNormalization layer for Branch model was used to create HeatMaps. We got tensor from last BatchNormalization layer with dimension (16, 16, 1024) or (16, 16, 1024) for single image in a batch. Here 16 $\times$16 is the size of the feature map in pixels. In this architecture, the total number of feature maps is 1024. This layer is followed by RELU activation, and then by GlobalMaxPooling layer, which selects the maximum pixel from each of the 1024 feature maps.

We can either count how many times each individual pixel takes the maximum value in the feature map, or measure how often the pixel value changes from one map to another. Here we adopt the second method and calculate the standard deviation.

Then, we normalize the obtained values in the interval from 0 to 255 and reshape 16 $\times$ 16 image to the size of the original image - 512 $\times$ 512.

In the HeatMap, areas with the maximum standard deviation are marked in green. Pink is used to mark areas that have the maximum value on feature maps most often and thus determine the vector that goes to head model. White areas have these two properties simultaneously. Through visualization, the model shows its ability to identify some whales visual features well, such as scars, birthmarks and the shape of the tail.

5 Conclusion and Future Work

Another intriguing network design is an extension of the Siamese Network, with triplet loss. It shows appearing performance in some experiments and can also be one choice. If more computational resource is available, we can also try other CNN branch model and head model design, and try more hyper-parameter tuning. Some other tricks like pseudo labeling, ensemble are also promising extensions of this work.

Acknowledgments

We used [3] as the base code for data preprocessing and [4] as the base code for Siamese Network visualization. We learned many tricks from the discussion board on Kaggle[1]. We are also very grateful to the generous help from Prof. Tianfu Wu and the teaching assistants of ECE763 during this semester.

References

1. https://www.kaggle.com/c/whale-categorization-playground
2. G Koch, R Zemel, and R Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning workshop, 2015*
3. https://www.kaggle.com/martinpiotte/whale-recognition-model-with-score-0-78563/output
4. https://www.kaggle.com/zfturbo/visualisation-of-siamese-net
5. https://weiminwang.blog/2019/03/01/whale-identification-5th-place-approach-using-siamese-networks-with-adversarial-training/
6. https://www.kaggle.com/martinpiotte/bounding-box-model