HappyWhale - Whale and Dolphin Identification

Yucong Mo
Utah State University
Logan, Utah, USA
yucong.mo@usu.edu

ABSTRACT

In the field of marine mammal science, the photographs of the animals' body parts are important factors to identify individual animals. However, currently most research institutions rely on human eye matching for the process. It is time consuming and the results are sometimes inaccurate, so a new way to automate the identification of marine life is needed. This paper explores the application of machine learning and facial recognition methods on a dataset of whale and dolphin images provided during a Kaggle research competition. An approach using convolutional neural network models to match individual whales and dolphins by unique characteristics of their natural markings is presented. Our results during the competition as well as the methods and results of the winning solutions are discussed.

KEYWORDS

datasets, neural networks, deep learning, computer vision, pattern recognition

1 INTRODUCTION

Tracking marine wildlife over time can help researchers understand migration patterns, population status, and better evaluate the human impact on marine wildlife. The photographs of shape and markings on animals' tails, dorsal fins, heads and other body parts are important features to detect specific individual animals. The current method used to identify animals is done manually by researchers and is very time intensive. To reduce this overhead, the HappyWhale foundation hosted a whale and dolphin identification Kaggle competition [1]. The result of this project will a more efficient way of identification that will enable studies previously unaffordable or impossible.

This paper analyzes the image sets from a multi-species dataset built by 28 research institutions and applies the model trained on the image set to the test image sets. To perform the animal identification automatically, researchers have turned to variety of methods, borrowing from the related research of facial recognition. Recently, implementations of a loss method known as ArcFace has become been widely used in the fields of computer vision and facial recognition[2]. The methods we use to complete this task are discussed with their accompanying results. The methods used by winners of this competition are briefly discussed, and our conclusions are offered.

2 DATA EXPLORATION

The first task in this analysis is data exploration. The training image dataset consists of over 51k images. There are 15,587 individual animals in the training set that span 26 different whale and dolphin species. Figure 1 shows four sample test images. The competition hosts have mentioned that the dorsal fins and lateral body views are important features in individuals recognition because unique things such as fin shape and body scars are very useful for identification.

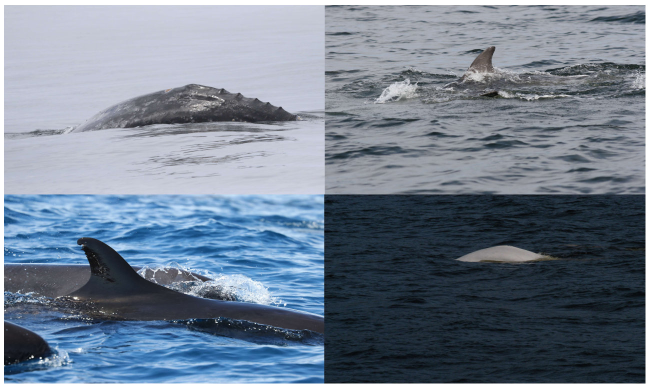

Figure 1: Sample Test Images

Figure 2 contains the distribution of images over the different species.

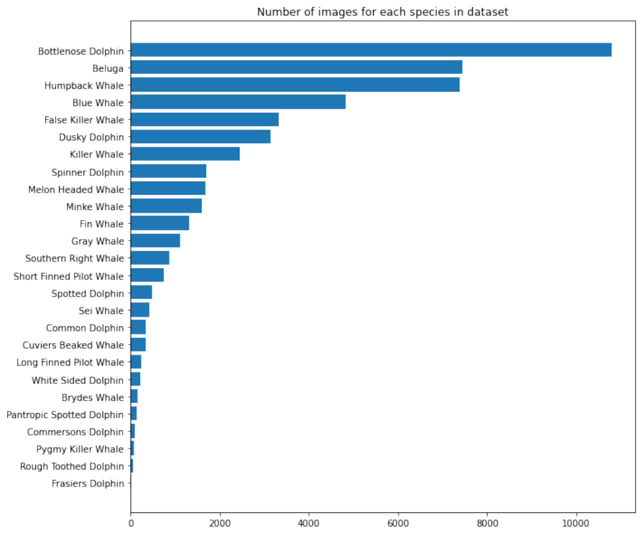

Figure 2: Species Distribution of the Training Datatset

The testing set consists of 28k images and contains images of animals both in and outside of the training dataset. One challenge associated with this project was determining when a new individual was being analyzed. The total size of the image dataset is 62GB.

The large size of the dataset was one challenge of this competition. The images in the dataset are typically fairly large and range from 650x800 to 3500x3500px and are all color images. To work with this large dataset, other Kaggle competitors created a uniform size dataset of cropped images [3] This new dataset, is a much more reasonable size of 7GB and all images were a uniform size of 512x512px.

For computational resources, even with the smaller dataset, Kaggle notebooks were the most convenient due to the dataset being ready-to-use directly from the notebook. However, this presented a challenge of GPU resources being limited to roughly 36 hours per week.

3 GENERAL FACE RECOGNITION PROCESS

A typical Convolutional Neural Network for classification consists of feature extraction and classification layers. During training, the model learns the unique facial features and produces feature embeddings in the feature extraction process, once the training is complete, we can skip the classification part and produce feature embedding for each image, which can be taken as digital "fingerprint".

For measuring the similarity of two embeddings extracted from image of the animal body parts, we use cosine distance. It is a metric that ranges from 0 to 1. If the cosine distance of two vectors is near 0, then the vectors have similar orientation and close to each other. If it is almost 1, the vectors are seen to be orthogonal. Figure 3 shows the cosine similarity of sample training images.

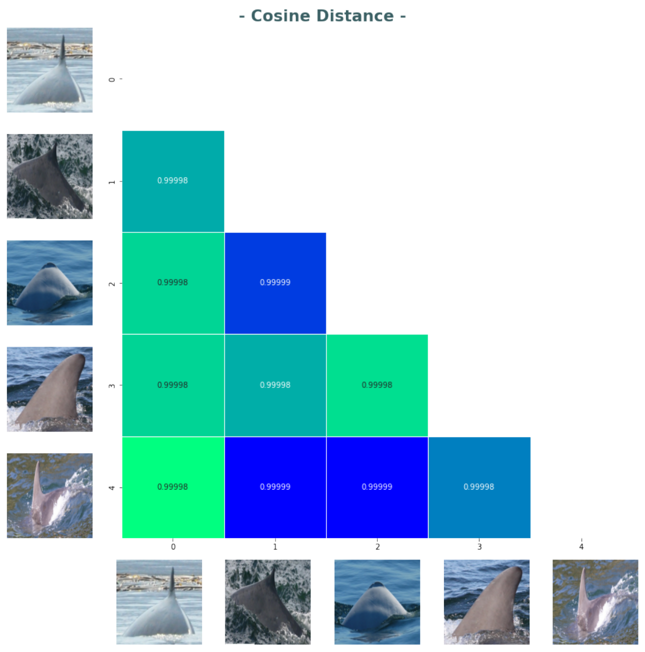

Figure 3: Computing Cosine Similarity[4]

3.1 ArcFace

In a typical classification network, SoftMax[5] is most widely used in the final stage of the network. It transforms the elements of the input vector into probabilities and ensures the sum of the components of the output vector is 1. The loss function of Softmax is defined as follows:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}, \tag{1}$$

Where x_i denotes the deep feature of the $i-th$ sample, belonging to the $yi-th$ class. W_j^T denotes the $j-th$ column of the weight W and b_j is the bias term. N and n are the batch size and class number respectively.

The drawback with this method is that it can produce blurry borders between classes after the classification. This has been one of the major challenges in the field of facial recognition.

In late 2018, ArcFace was introduced to to help maximize the spread of different class images by using the embedding layer and plotting them with a angular margin on a hypersphere. The loss function is show in in equation (2).

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}. \tag{2}$$

Here,θ_j is the angle between the weight W_j.The feature x_i. s denotes the hypershpere radius and m shows the angular margin penalty.

Instead of calculating the the cosine distance of embedding vectors in the euclidean space, Arcface proposes to find the geodesic distance on the hypersphere, where all distances are measured by rails and the shortest distance between two points is denoted by tracks. In this way, Arcface is able to directly produce an angular margin between two classes, as shown in figure 4.

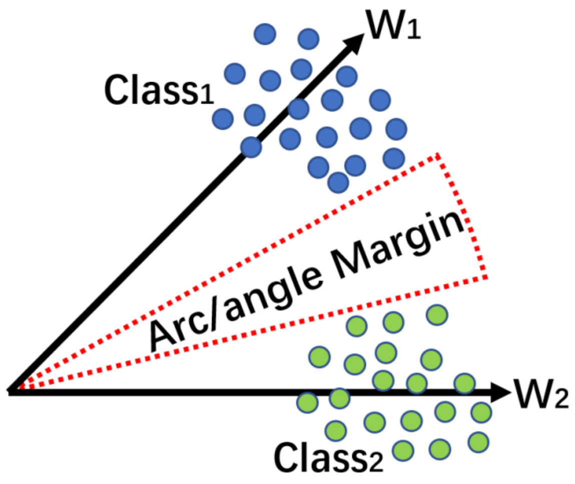

Figure 4: Angular Margin[2]

4 EVALUATION

The evaluation metric used for this competition was Mean Average Precision (MAP) with a cutoff of 5 (MAP@5). This means each test image could have 5 predictions. The order of the predictions matter, in that a higher score is attained by having the correct predictions first in the list. The equation used for submission evaluation is shown in equation (3).

$$MAP@5 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,5)} P(k) * rel(k) \tag{3}$$

In equation 1, U is the number of images in the test dataset. $P(k)$ is the precision score of a correct identification at location k. $rel(k)$ is an indicator function that is equal to 1 if the identification is correct and 0 otherwise. Once the correct label is obtained for an image, the rest of the predictions are skipped in the calculation.

Precision is a measure of quality between the prediction and the other predictions for the image. This is a better metric for this competition than raw accuracy because it gives incentive to have the most likely prediction higher up in the list. General precision is calculated as shown in equation (4).

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{4}$$

Therefore, the score is slightly decreased as each new false positive is introduced.

5 IMPLEMENTATION

For our implementation, we started from scratch using the PyTorch framework and the Kaggle implementation of Jupyter Notebooks. The first task is to convert the different individuals labels into a two-way mapping. This was done by assigning each individual id an index from 0 - 15,587. Then, a one-hot encoded array can be used as the label during classification. The two-mapping is required so that during inference time, the individuals that are identified as being the most likely can be easily mapped from max index back into the individual id string.

To load the images into memory, the Pillow library provides an easy-to-use open function that we integrated into our WhaleDolphinDataset class. Due to high memory requirements, batch loading is used and carefully tuned to not exceed the capacity of the RAM or the GPU memory. As stated in the Data exploration section, the cropped dataset consisted of 512x512 color images. To reduce the memory requirements and speed up training, we resized the images with sizes ranging from 128-384 square images. We observed that model training time and accuracy was sensitive to the chosen image size with increased results from bigger images. Additionally, in our implementation we needed to account for model over fitting. Even though there were 51 thousand images in the dataset, there were also nearly 16 thousand classes. This means there can only be, on average, just over 3 images of each individual. This lends to being prone to overfitting. One method we combat this is by using various image transformation techniques such as image rotation and color jitter. Some competitors even flipped the images horizontally and then labeled the flipped images as a different individual during training. At inference time, the different categories could be combined back together.

Various models were chosen from the image classification domain for this task. We experimented with pretrained models from the ResNet framework [6], EfficientNet series [7], and ConvNeXt models [8]. These models are frequently chosen for transfer learning. A discussion of each of their design is outside the scope of this current project, and we refer you to their respective papers. For our implementation, we decided to use the 1000 output classes of the model as the embedding features for ArcFace.

The most important implementation feature for obtaining good results with this dataset was the addition of the ArcFace layer as discussed in the previous section. We experimented both with and without this layer. The differences of results will be discussed in the next section. With and without the addition of the ArcFace loss layer we used PyTorch's built in BCEWithLogitsLoss as the differential training loss function and the Adaptive movement Adam as the optimizer. We trained with varying learning rates from 0.001 to 0.0001. The number of epochs also varied from 10 to 30.

One implementation we experimented with was to try to break the problem into an easier task first, and then work our way back to the classification task. We decided an easier task would be to classify the images into their respective species. Our thinking was that this would be easier to identify individuals within the context of their species rather than the context of the entire dataset. For this implementation we added a fully connected layer to a ResNet50 model with an output of 26 that correspond to the species. This was trained for 10 epochs, then the output layer was added to classify each of the 15,587 individuals.

Another implementation that we used to improve our results was very similar to our base, from scratch implementation. This stemmed from a public notebook that used the convnext_tiny model and added things like an adaptive threshold and backfin checking to the inference. We copied this notebook with some minor changes. Some whale and dolphin species have dorsal fins that can be used to discern individual features. Some of the competitors labeled which species and individuals did not have back fins. Then, at inference time, special care can be made to make sure that the predictions have or do not have fins corresponding to the fin status of the test image.

The final implementation we used was to take the results of well performing public notebooks, and ensemble them into a single result. We implemented two forms of ensembling. The first and simpler method is to put each prediction from the different submissions for a single sample into a bag. Then, the labels with a higher frequency are put first in the list. Once a label with less than three supporting submissions are found, the guess of new_individual was inserted. The remaining locations, if any, were then remaining predictions with the most votes.

The second ensembling method was to give each of the submissions a weight based on their individual score. Then, the weights were multiplied by a ranking value they occurred in the list. The ranking equation is shown in equation (5).

$$ranking = wt * \frac{1}{1 + ind} \tag{5}$$

For each image the ranking of every prediction was computed, and then summed together. Finally, the five labels with the highest total ranking were selected as the predictions.

6 RESULTS

The description of our methods and implementation discussed in this section are as detailed in the implementation section. The summary of all methods are displayed in Table 1. All training was completed between 10 and 30 epochs. This led to some overfitting on some of the models including the ResNet brute force method. The results of the brute force method without adding any face recognition considerations were poor. Our method of trying to predict the species first had interesting results. Completing the training for the species was very promising. The accuracy for both the training and validation set scored in the 98% accuracte range. Looking at the confusion matrix shows how well the model performed. We then attempted to use the species as the embedding features. Which performed really poorly. Our conclusion is that having 30 embedding features was not representative enough, even though the model was discriminating enough to determine the species.

Table 1: Summary of Results

Method	Public LB	Private LB	Rank
ResNet50, Species Classification	0.0256	0.0284	1505/1588
ResNet50, Brute Force	0.1325	0.0887	1399/1588
Convnext_tiny, Arcface, IMG transforms	0.3981	0.3534	1179/1588
EfficientNet, ArcFace, Backfin LR Scheduler	0.6820	0.6257	956/1588
Frequency Based Ensemble	0.7792	0.7266	500/1588
Weighted Ensemble	0.7845	0.7422	283/1588

Adding the ArcFace layer without changing any other parameter in the model increased the results by nearly 20% on the test set. The reason that we and all of the other competitors used this tool instead of other popular tools is that the implementation is relatively simple and the added computational overhead is minimal [2]. With the small number of images per class, the image transformations and learning rate scheduler were important to keep from overfitting. That is one reason why the EfficientNet model was able to acheive results nearly 0.29 higher than our next best solution using the convnext_tiny model.

Without changing any hyperparameters, the different models only performed marginally different in our experiments. One benefit, then, of using the ResNet models or the EfficientNet architecture over the ConvNext models is the large difference in training time. The residual network designs help the model train faster and have similar results with fewer model parameters. Using the ConvNext model, we were only able to train the model with a very small batch size of 16 to not overflow the GPU memory.

Our largest limitations in our experimentation and results was the lack of personal GPUs. Each experiment we ran took between 4 and 12 hours to run using the Kaggle notebooks. With only being allowed around 36 hours of GPU time per week, this limited the notebooks we could run and avenues we could pursue. Despite our hardware limitations, we were able to improve results by ensembling others' work. The individual results we used in our ensemble method ranged in public leader board score from 0.68 – 0.72. Our public result of 0.7845 demonstrated that different models able to learn different, but distinctive features. The downside of this solution, however, is that it would be difficult to implement an inference solution to help the HappyWhale foundation classify new images in a production environment.

7 WINNING METHOD

The methods used by the winning solution was based on a model ensemble trained with a dynamic margin ArcFace layer [9]. They used many different bounding box methods as well as some of the datasets that had cropped images. They mainly used an image size of 1024x1024px. They used a variety of different EfficientNet models. The best single model they trained was an efficientnet_b7 model.

The ArcFace margins were very sensitive to the various hyperparameters, so they used the Optuna framework to optimize the parameters and a cloud computing provider to complete the computationally heavy training. During training they used many different enhancements including pseudo-labeling, performing a large variety of image transformations, and using a cosine annealing learning rate scheduler.

Finally, they used some post-processing techniques in combination with a K-nearest Neighbor classifier to find the most similar individuals to the test image. Their final, winning submission consisted of an ensemble of 50 different models.

Their final score was 0.8968 on the public leaderboard and 0.8758 on the private leaderboard. The next best had a score of 0.87262 on the private leaderboard. This is obviously a significant improvement over our method.

8 CONCLUSIONS

In this paper, we discussed our approaches and results on a face recognition task given in a Kaggle competition, which aims to automate the process of identifying whales and dolphins based on the image sets of their body parts. We applied various data processing methods including formalizing the the size of each image and clustering different image datasets into their respective species. We also demonstrated that adding ArcFace layer to neural network can effectively improve the accuracy of identification on these marine mammals.

Finally, we examined the winning method of this competition, which appears to be far more sophisticated compared to our model in terms of how it properly tuned the hyperparemeters of Arcface margin with Optuna framework, fully utilized data training enhancements on cloud services, and selected suitable classifier for the output.

9 FUTURE WORK

After the results of this competition came out, inspecting outstanding public notebooks points out several directions for improvements that can be made to our method.

In data pre-processing stage, in addition to cropping the image, we can also remove the background to improve the accuracy for identification. This is because with the background removed, ArcFace generally performs better in a high contrast image. Figure 5 shows the effect of background removal.

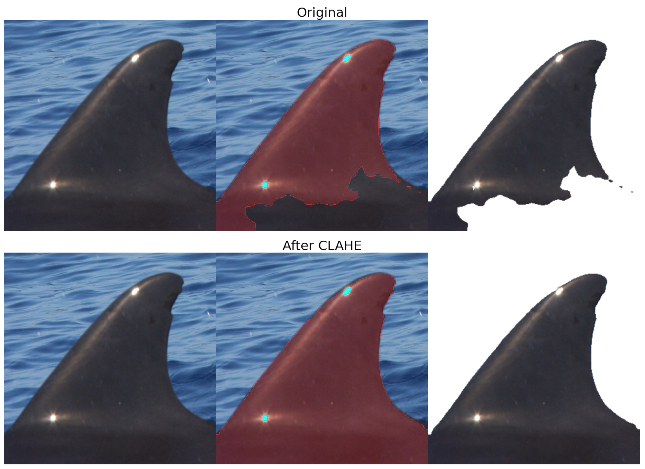

Figure 5: Background Removal[10]

In model training stage, we can apply various hyperparameter optimization framework including Optuna, Ray-Tune, Hyperopt etc. We have been mostly manually tuning the hyperparameters in our model and the result scores are not close enough to the top performers. Utilizing such frameworks can also direct the way to better fit our model to the dataset.

REFERENCES

[1] Research Prediction Competition, Happywhale - Whale and Dolphin Identification , “Identify whales and dolphins by unique characteristics,” Online Available: https://www.kaggle.com/competitions/happy-whale-and-dolphin/ 2022.

[2] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[3] R. Kinas, A. Olteanu, and J.M. Chen, “whale2-cropped-dataset,” Online Available: https://www.kaggle.com/datasets/phalanx/whale2-cropped-dataset 2022.

[4] Olteanu, A. (2022, March 12). whales & dolphins: EffNet Train & Rapids Clusters. Kaggle. Retrieved April 22, 2022, from https://www.kaggle.com/code/andradaolteanu/whales-dolphins-effnet-train-rapids-clusters#6.-Model-Embeddings

[5] Wikimedia Foundation. (2022, March 28). Softmax function. Wikipedia. Retrieved April 22, 2022, from https://en.wikipedia.org/wiki/Softmax_function

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv.org, [Online]. Available: https://arxiv.org/abs/1512.03385. 2015.

[7] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, CoRR, abs/1905.11946, 2019.

[8] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A ConvNet for the 2020s”. arXiv, 2022.

[9] knshnb, “1st Place Solution,” Kaggle. [Online]. Available: https://www.kaggle.com/competitions/happy-whale-and-dolphin/discussion/320192. [Accessed: 22-Apr-2022].

[10] CHEN, J. U. N.-M. I. N. G. (2022, April 4). background remove tutorial. Kaggle. Retrieved April 22, 2022, from https://www.kaggle.com/code/leoooo333/background-remove-tutorial.