# Group 18 Project Report: Happywhale, Whale and Dolphin Identification

**Seenan Iftekhar**
**Jessica Zhen Li**
**Jad Ayman Alriyabi**
**Sunayani Sarkar**
**Amirhossein Etaati**

## Abstract

In this paper, the Kaggle's competition entry on whale and dolphin identification is examined. The convolutional neural network (CNN) algorithm is used to identify whales and dolphins in the images given their individual ids. CNN algorithm is chosen because, compared to its predecessors, CNN has the great advantage of detecting the important features without any human supervision. Therefore, given many pictures of dolphins and whales, the algorithm can learn distinctive features for each class by itself.

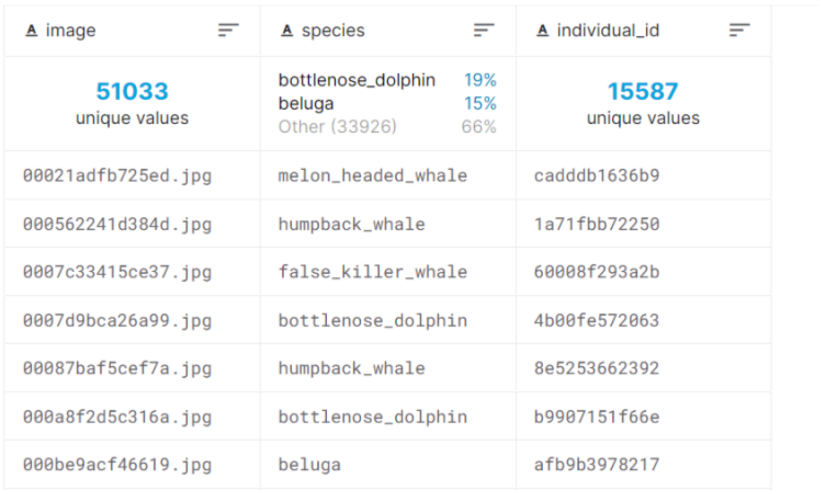
## 1 Introduction

The Happywhale project aims to identify different whales and dolphins by using different characteristics of their bodies, such as shape and markings on their tails, dorsal fins, heads and other body parts. Individual whales and dolphins must be identified by unique—but often subtle—characteristics of their natural markings, which are used for image analysis and classification [1].
Our team has aimed to develop a model using the CNN algorithm to match the individual ids, assigned manually by researchers, to dolphins and whales in the hundreds of thousands of pictures. Therefore, our final model is designed to identify an individual whale or dolphin given an image and individual id.
This identification task will be very helpful for marine scientists as it will not only save time but provides accuracy in identifying whales and dolphins in the ocean [2].

## 2 Data

The forms, characteristics, and markings (some natural, some acquired) of dorsal fins, backs, heads, and flanks can be used to identify whales and dolphins in this dataset. The data includes photos of approximately 15,000 distinct individual marine animals from 30 different species, acquired from 28 different research institutions. Marine researchers have manually identified and assigned them an `individual_id`, and our objective is to accurately identify these individuals in photos.

We are unable to add the data files into our .zip file containing the code because the total size is around 60 GB due to there being so many images in it. Zipping the file is not sufficient. The data files can be accessed through the Kaggle website [1].



| image | species | individual_id |
|---|---|---|
| 51033 unique values | bottlenose_dolphin 19%<br>beluga 15%<br>Other (33926) 66% | 15587 unique values |
| 00021adfb725ed.jpg | melon_headed_whale | cadddb1636b9 |
| 000562241d384d.jpg | humpback_whale | 1a71fbb72250 |
| 0007c33415ce37.jpg | false_killer_whale | 60008f293a2b |
| 0007d9bca26a99.jpg | bottlenose_dolphin | 4b00fe572063 |
| 00087baf5cef7a.jpg | humpback_whale | 8e5253662392 |
| 000a8f2d5c316a.jpg | bottlenose_dolphin | b9907151f66e |
| 000be9acf46619.jpg | beluga | afb9b3978217 |

Kaggle’s sample train data from train.csv [1].

Images provided in the dataset as such:

## 3 Logic and algorithm

CNN is a Deep Learning algorithm that can take in an input image, assign importance by using learnable weights and biases to various aspects of the image and eventually be able to differentiate one from the other [3].

The number of parameters in a neural network increases dramatically as the number of layers increases. This can make the training process for the model computationally intensive and sometimes not feasible [4]. Tuning so many settings may be a massive undertaking. CNNs reduce the amount of time it takes to fine-tune these parameters [4]. Furthermore, CNNs are feed-forward neural networks that are completely linked. The algorithm is exceptionally good at lowering the number of parameters without sacrificing the quality of the model [4]. The dimensionality of images is high. Multiple convolutional filters are active in all the layers of a CNN [4]. Scanning the whole feature matrix and doing dimensionality reduction enables CNN to be a particularly well-suited network, for image classifications and processing [4].

CNN has a series of layers, each of which uses a differentiable function to change one volume into another [5]. Layers are classified as follows:

In a picture with the dimensions W x H x D [5].

| | |
|---|---|
| Input Layer: | This layer contains the image's raw input, which has a width of W, a height of H, and a depth of D. |
| Convolution Layer: | The output volume is calculated using the dot product of all filters and image patches in this layer. For this layer, we'll receive an output volume of W x H x N for a total of N filters [5]. |
| Activation Function Layer: | This layer will apply an element-by-element activation function to the convolution layer's output. Because the volume remains constant, the output volume will be W x H x N. |
| Pool Layer: | This layer is added to a CNN on a regular basis, and its major goal is to lower the size of the volume, which speeds up computation, saves memory, and avoids overfitting [5].<br>Max pooling and average pooling are two typical types of pooling layers [5]. The final volume will have dimensions W/M x H/M x N if we utilize a max pool with M x M filters and stride M. |

## 4 Code Explanation

Our module used multiple libraries to train our model, although the most critical libraries were *torch*, *pandas*, and *sklearn*. *Torch* is an open-source machine learning library that provides a wide range of algorithms for deep learning. We implemented the CNN algorithm and its optimizer for training purposes in our model. Panda was used to analyze data and import our data, and sklearn contains tools for classification and data preprocessing; in our model, this was used to cluster and reduce the dimensions of the dataset.

We configure our model to batch 16 images at a time to be tested and modeled, then set the model to train using the GPU; if not available, the CPU can be used to train the model. Then a seed is established to ensure that the same training and validation data is attained every time the data is shuffled when loading the training model.

The CSV files for all the testing and the training dataset are loaded using Panda to preload the data. Then ground-truth labels are loaded to the datasets, which are 1 to 1 transformations of id's to number id labels. Ground-truth label transformation is the process in which raw data such as images, text files, videos, etc. It can be identified to provide the context it allows to add one or more meaningful and informative labels so that the model can learn something. A Dataset class was created to pass the CSV files and the image transformation method. A "getitem" method is used to retrieve one data sample at a time, and this method loads the image from the path saved in the CSV file and a ground truth label. Finally, the method applies the transformation on the image and then returns the image, path and its ground label. The resnet18 method is called from the torch library in the Classifier Model, which makes 1000 predictions per image. However, we need another layer implemented to accompany the +15000 images that are required to train this model.

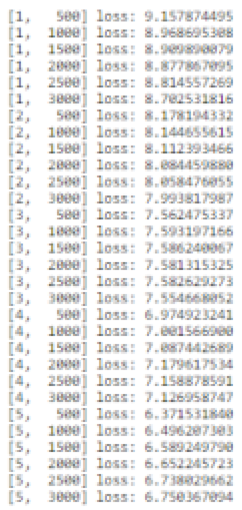
Loss value implies how poorly or well a model behaves after each optimization iteration. The lower the loss, the better a model and its interpretation is how well the model is calculated on training and testing datasets. # Unlike accuracy, the loss is not a percentage, and it is a summation of the errors made for each example in training or validation sets. Our model implemented entropy loss since it is a standard loss method for image classification. Also, the RGB values were modified to a normal distribution to allow the model to train more efficiently.

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i$$

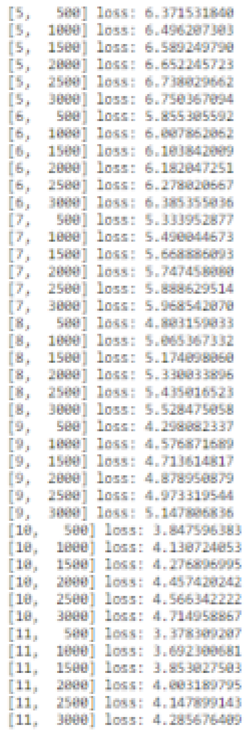
Finally, the model begins training by looping through each epoch of the dataset and training mini-batches from the data. During each epoch, the parameter gradients of the model are zeroed, then the data is forwarded, backward propagates the data, then the optimizer optimizes the weights, then prints the statistics for every 2000 mini-batches of the data. Finally, the testing of the model includes looping through the testing dataset and the top 4 predicted probabilities that are then saved into a CSV file.

## 5 Results

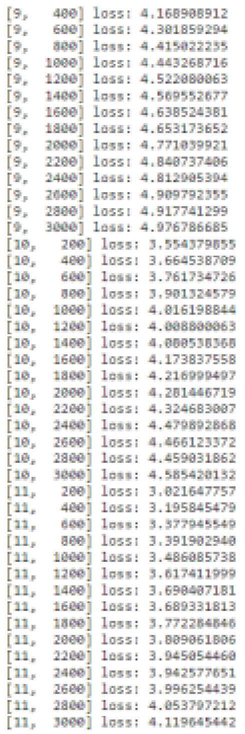
Initially, we trained the model to print the cross-entropy loss value for every batch of data, resulting in an initial value of ~9.15. We reached a loss value of ~3.1 with only 11 epochs of training. The results show our model is successful in its predictions, and we believe if we had the capabilities, we could even result in a better loss value. The loss value after around five epochs begins to fluctuate slightly, but the value still decreases, as shown in the figure below. The fluctuations can be due to the size of the validation set being too small; small changes in the output cause significant fluctuations in the validation error, although this did not happen in our model. During our training, we output what epoch the model is at, the label and the entropy loss value calculated as shown below:



```
[1,   500] loss: 9.157874495
[1,  1000] loss: 8.968695308
[1,  1500] loss: 8.909890079
[1,  2000] loss: 8.877867095
[1,  2500] loss: 8.814557269
[1,  3000] loss: 8.702531816
[2,   500] loss: 8.178194332
[2,  1000] loss: 8.144655615
[2,  1500] loss: 8.112393466
[2,  2000] loss: 8.084459880
[2,  2500] loss: 8.058476055
[2,  3000] loss: 7.993817987
[3,   500] loss: 7.562475337
[3,  1000] loss: 7.593197166
[3,  1500] loss: 7.586240067
[3,  2000] loss: 7.581315325
[3,  2500] loss: 7.582629273
[3,  3000] loss: 7.554668052
[4,   500] loss: 6.974923241
[4,  1000] loss: 7.001566900
[4,  1500] loss: 7.087442689
[4,  2000] loss: 7.179617534
[4,  2500] loss: 7.158878591
[4,  3000] loss: 7.126958747
[5,   500] loss: 6.371531840
[5,  1000] loss: 6.496207303
[5,  1500] loss: 6.589249790
[5,  2000] loss: 6.652245723
[5,  2500] loss: 6.738029662
[5,  3000] loss: 6.750367094
```



```
[5,   500] loss: 6.371531840
[5,  1000] loss: 6.496207303
[5,  1500] loss: 6.589249790
[5,  2000] loss: 6.652245723
[5,  2500] loss: 6.738029662
[5,  3000] loss: 6.750367094
[6,   500] loss: 5.855305592
[6,  1000] loss: 6.007862062
[6,  1500] loss: 6.103842009
[6,  2000] loss: 6.182047251
[6,  2500] loss: 6.278020667
[6,  3000] loss: 6.385355036
[7,   500] loss: 5.333952877
[7,  1000] loss: 5.490044673
[7,  1500] loss: 5.668886093
[7,  2000] loss: 5.747458000
[7,  2500] loss: 5.888629514
[7,  3000] loss: 5.968542070
[8,   500] loss: 4.803159033
[8,  1000] loss: 5.065367332
[8,  1500] loss: 5.174098060
[8,  2000] loss: 5.330033896
[8,  2500] loss: 5.435016523
[8,  3000] loss: 5.528475058
[9,   500] loss: 4.298082337
[9,  1000] loss: 4.576871689
[9,  1500] loss: 4.713614817
[9,  2000] loss: 4.878950879
[9,  2500] loss: 4.973319544
[9,  3000] loss: 5.147806836
[10,   500] loss: 3.847596383
[10,  1000] loss: 4.130724053
[10,  1500] loss: 4.276896995
[10,  2000] loss: 4.457420242
[10,  2500] loss: 4.566342222
[10,  3000] loss: 4.714958867
[11,   500] loss: 3.378309207
[11,  1000] loss: 3.692300681
[11,  1500] loss: 3.853027503
[11,  2000] loss: 4.003189795
[11,  2500] loss: 4.147899143
[11,  3000] loss: 4.285676409
```



```
[9,   400] loss: 4.168908912
[9,   600] loss: 4.301859294
[9,   800] loss: 4.415022235
[9,  1000] loss: 4.443268716
[9,  1200] loss: 4.522080063
[9,  1400] loss: 4.569552677
[9,  1600] loss: 4.638524381
[9,  1800] loss: 4.653173652
[9,  2000] loss: 4.771039921
[9,  2200] loss: 4.840737406
[9,  2400] loss: 4.812905394
[9,  2600] loss: 4.909792355
[9,  2800] loss: 4.917741299
[9,  3000] loss: 4.976786685
[10,   200] loss: 3.554379855
[10,   400] loss: 3.664538709
[10,   600] loss: 3.761734726
[10,   800] loss: 3.901324579
[10,  1000] loss: 4.016198844
[10,  1200] loss: 4.008800063
[10,  1400] loss: 4.080538368
[10,  1600] loss: 4.173837558
[10,  1800] loss: 4.216999497
[10,  2000] loss: 4.281446719
[10,  2200] loss: 4.324683007
[10,  2400] loss: 4.479892868
[10,  2600] loss: 4.466123372
[10,  2800] loss: 4.459031862
[10,  3000] loss: 4.585420132
[11,   200] loss: 3.021647757
[11,   400] loss: 3.195845479
[11,   600] loss: 3.377945549
[11,   800] loss: 3.391902940
[11,  1000] loss: 3.486085738
[11,  1200] loss: 3.617411999
[11,  1400] loss: 3.690407181
[11,  1600] loss: 3.689331813
[11,  1800] loss: 3.772284846
[11,  2000] loss: 3.809061806
[11,  2200] loss: 3.945054460
[11,  2400] loss: 3.942577651
[11,  2600] loss: 3.996254439
[11,  2800] loss: 4.053797212
[11,  3000] loss: 4.119645442
```

## 6 Conclusions

With a low entropy loss value, we can conclude that our model was successful. This means that the classifications we have made are sufficiently close to the true classification of whales and dolphins. Aside from training more epochs, there are many things we can do to further improve this error in our results, given more time and skill. For example, we could perhaps explore different algorithms, libraries, or models to use. Amongst the many solution writeups on Kaggle, it seemed that ArcFace and GeMPooling were models and layers that were used often. Some other interesting algorithms to consider are KNN, Random Forest, and Yolov5 Detection. However, in the end, we believe that CNN was the best algorithm for us to use at our scope with the time we had, as it was exactly what we needed, and we were going to learn about neural networks in class. Furthermore, we wrote our code in Kaggle, which had its own restrictions when it came to training. Kaggle limits the number of hours it allows you to train your model per week, so after we hit the limit, we had to wait a week to further train our model. This directly affects our results, as we had issues with the build crashing, thus forcing us to wait another week. Overall, this project, as well as the CNN algorithm, required great time and effort to thoroughly understand, but after some research and overcoming our first hurdle in the code, we found that the only direction to go was forwards. By persevering, we managed to reach a low loss value. Through this project, we realized that this is only the beginning of our journey in artificial intelligence. From here on, we will learn and improve on our machine learning skills, with our capabilities allowing us to reach new limits, so if we want to participate in next year's competition, we can get a loss of 0.

## 7 Contributions

| Names | Contributions |
|---|---|
| Seenan Iftekhar | - Research regarding algorithm<br>- Participated in coding the machine learning model<br>- Presented the project in class |
| Jessica Zhen Li | - Research regarding algorithm<br>- Participated in coding the machine learning model<br>- Created and wrote the structure and outline detailing what would be talked about in the video presentation<br>- Wrote the conclusion of the report |
| Jad Ayman Alriyabi | - Research regarding algorithm<br>- Participated in coding the machine learning model<br>- Presented the project in class<br>- Responsible for training and testing the model<br>- Wrote the code explanation part in the report |
| Sunayani Sarkar | - Research regarding algorithm<br>- Created the PowerPoint presentation.<br>- Created, edited and submitted the video<br>- Wrote the Logic and Algorithm part of the report.<br>- Formatted the report into NeurIPS style file format.<br>- Proofreading and editing the report |
| Amirhossein Etaati | - Research regarding algorithm<br>- Participated in coding the machine learning model<br>- Presented the project in class<br>- Wrote the abstract and introduction part of the report<br>- Organized and formatted the report according to NeurIPS style files |

## References

[1] *Happywhale - Whale and dolphin identification*. Kaggle. (2022). Retrieved April 21, 2022, from https://www.kaggle.com/competitions/happy-whale-and-dolphin/code

[2] Aalderink, E. van. (2022, March 2). *Photo-ID is the heart of whale research*. Whale Scientists. Retrieved April 21, 2022, from https://whalescientists.com/photo-id/

[3] Saha, S. (2018, December 17). *A comprehensive guide to convolutional neural networks - the eli5 way*. Medium. Retrieved April 21, 2022, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[4] Mishra, P. (2019, July 20). *Why are convolutional neural networks good for image classification?* Medium. Retrieved April 21, 2022, from https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8

[5] Mishra, A. P. (2021, November 25). *Introduction to convolution neural networks*. GeeksforGeeks. Retrieved April 21, 2022, from https://www.geeksforgeeks.org/introduction-convolution-neural-network/

https://stackoverflow.com/questions/34518656/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model

https://stackoverflow.com/questions/56023786/intuition-behind-fluctuating-training-loss

Intuition behind fluctuating training loss - Stack Overflow