# Capstone Project

Viswanath Ravindran

May 12th, 2018

## I. Definition

### Project Overview

Blue Whale tracking has been considered an exciting and dear problem for Happywhale as in link. They have quoted specifically tracking of the Humpback whale as a laborious process by scanning through several thousands of pictures posted online by whale enthusiasts/scientists/tourists/fisherman throughout the world.

It simply works by submitting photos of your marine mammal encounters and they would verify the whale species by looking for the unique marking on its fluke(whale's tail). By doing so Happywhale tracks the movement of whales across the world.

### Problem Statement

After centuries of intense whaling, recovering whale populations still have a hard time adapting to warming oceans and struggle to compete every day with the industrial fishing industry for food. To aid whale conservation efforts, scientists use photo surveillance systems to monitor ocean activity. They use the shape of whales' tails and unique markings found in footage to identify what species of whale they're analyzing and meticulously log whale pod dynamics and movements. For the past 40 years, most of this work has been done manually by individual scientists, leaving a huge trove of data untapped and underutilized. Happywhale has also reached out the the Kaggle community to solve this problem @ link

In this competition, I am challenged to build an algorithm to identifying whale species in images. I will analyze Happy Whale's database of over 25,000 images, gathered from research institutions and public contributors. By contributing, I will help to open rich fields of understanding for marine mammal population dynamics around the globe.

## Metrics

The evaluation metric for this challenge is called the Mean Average Precision score at K. The k selected here is 5, The output prediction should simple output the 5 most possible class labels of the several available class.
Its use is different in the field of Information Retrieval and popular among various Multi Label Classification problems setting. The mathematical representation is available as in kaggle link

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

where `U` is the number of images, `P(k)` is the precision at cutoff `k`, and `n` is the number predictions per image.

# II. Analysis

## Data Exploration

The Happy Whale dataset that has been provided to us has both the train and test data set. We can have a look at the dataset from the following link

The training dataset has about 9850 images after constructing a dataframe referring individual image to its ID it looks like below:

| Image | Id |
|---|---|
| ../input/train/00022e1a.jpg | w_e15442c |
| ../input/train/000466c4.jpg | w_1287fbc |
| ../input/train/00087b01.jpg | w_da2efe0 |
| ../input/train/001296d5.jpg | w_19e5482 |
| ../input/train/0014cfdf.jpg | w_f22f3e |

The test data which is the non-labelled images are about 15610 images. Some sample images from the train data with their labels are as below:

Another reference article online indicates the following information from link. There are five basic types of whale flukes: 1. Mostly white. 2. 25% white. 3. Half white, half black. 4. 25% black. 5. Mostly black. With some luck, one might only have to look into one type- this would usually only occur in mostly white or black.

Often one can eliminate three types, and look through only two types. Sometimes one has to look through three types, especially if not 100% of the fluke can be seen.

Another article online indicates the following properties above whale fluke at link

- The whales fluke has the following parts indicating its anatomy.

- The shape of the flukes can change to some degree depending on the angle to the surface of the water the instant the photograph was taken
- If your photo has flukes with a large area of white you may want to focus on a specific scratch or shape in the white area. If your photo has dark flukes with some white scratches, focus on finding a photo with similar scratches.
- One of the most important areas of the flukes to look at is the trailing edge. The bumps and nicks along this edge remain relatively stable throughout the whale's lifetime.
- Since the photograph of the whale are taken over a period of time there are possibilities that the fluke shape may be changed, the scratch pattern may be different now etc..
- Matching a young whale is significantly more difficult since the pigmentation on the flukes becomes more distinct as the whale gets older and then the pattern seldom changes.

To summarize the above findings the quality of picture plays an important role, pictures with a higher pixel exhibit more prominent features.

## Exploratory Visualization

We will try to understand the distribution of the labels that are available in the train dataset. Some of the important features of the train data are:

- There are a total of 4251 categories
- The number of image for the categories is very skewed. There are vast majority of classes that have only a single image in them. This makes the image problem harder.

- Interestingly the new whale category has the highest number of images however these are multiple whales which does not have a labelling in the existing database. The second whale with label w_1287fbc has about 34 images.

```
new_whale    810
w_1287fbc     34
w_98baff9     27
w_7554f44     26
w_1eafe46     23
```

- A sample of pictures of a particular class w_1287fbc is as below:

- Majority of the images on top are black and white which indicates we can have a basic image augmentation is to convert all images into black and white.
- The input images are of varied size and shapes, here is a plot of distribution of the different image sizes.

## Algorithms and Techniques

This specific problem can be approach in the following manner:

1. Input the training images and the labels for the training image is available in the necessary format in the necessary folder structure.
2. After understanding the distribution of the dataset across the various classes that we, I would then have to apply appropriate data split techniques to split between train and validation sets.
3. Identification of potential duplicate images has to be verified since it can skew the class distribution and potentially the model will be unable to learn new features in duplicate images.
4. We can create an initial model as a baseline model using external pre-trained models like ResNet, VGG16 etc.
5. Using the baseline model and applying various transformation to the image we then can train an inference model to help understand the differences in the predicted vs true label.
6. I would possibly using ImageDataGenerator for real-time data augmentation, layer freezing and model fine-tuning to improve my predictions.
7. We can then fine-tuning the top layers of a pre-trained network. Selecting an appropriate architecture such as the GoogleNet etc is essential for improved accuracy.
8. Choosing an appropriate solver algorithm for better convergence is necessary to be tested.

## Benchmark

The benchmark model that can be used to see how I perform could be used using the sample submission here. The details of the metric is in the next section. The performance is designed to indicate a better performance with a higher score. Screenshot attached shows the Benchmark score submitted using the sample by host organization of this problem listed in the leaderboard page for the competition @ link As of the time that I wrote this proposal the base submission is 0.32786. I would focus on improving the score better than this.

## III. Methodology

### Data Preprocessing

The python image PIL library was quite resourceful in handling and creating the necessary image processing before feeding them as input. Here are the list of processing that has been followed and its justifications:

- Resizing of the images into 224*224 as size since the final version of the model includes including a pre-trained VGG16 (using imagenet weights)
- Nearly half of the pictures in the train/test data were grayscale images and I had to convert them into dumb RGB since VGG16 accepts input in the form of 224*224*3
- Using the Image Data Generator function in Python after several attempts identified the best version of the arguments for the function

- There were version of a simple conversion of all images into grayscale images that was also tried which did not result in a good accuracy.

### Implementation

After trying several networks & techniques the best version that I ended up is the transfer learning methodology. The final model that is implemented is the base VGG16 model and the fully connected layer with a 2 depth neural network.

Some details on the VGG16 are:

- Convolutions layers (used only 3*3 size ) , Max pooling layers (used only 2*2 size) & Fully connected layers at end
- There is a total 16 layers
- The model is about 528 MB in size

The representation of VGG16 is as below:

The 16 layers that encompass are:

1. Convolution using 64 filters
2. Convolution using 64 filters + Max pooling
3. Convolution using 128 filters
4. Convolution using 128 filters + Max pooling
5. Convolution using 256 filters
6. Convolution using 256 filters
7. Convolution using 256 filters + Max pooling
8. Convolution using 512 filters
9. Convolution using 512 filters
10. Convolution using 512 filters + Max pooling
11. Convolution using 512 filters
12. Convolution using 512 filters
13. Convolution using 512 filters + Max pooling
14. Fully connected with 4096 nodes
15. Fully connected with 4096 nodes
16. Output layer with Softmax activation with 1000 nodes

Keras has the flexibility to assist in downloading the network using:

```
keras.applications.vgg16.VGG16(include_top=True, weights='imagenet',
input_tensor=None, input_shape=None, pooling=None, classes=1000)
```

Once called the model is downloaded and will reside in the local system, In my technique I would be removing the last layer of the network and replace to include a custom layer which used Softmax activation for the 4251 classes.

The full architecture of the network will be as follows:

## Refinement

The table below indicates a rough path that was taken towards the final model. The table above indicates the immediate increase in result after using a pre-trained model based complex architecture.

| Version | Implementation Architecture | Best result |
|---|---|---|
| 1 | A 3 layer convolutional model with 3 fully connected layers.<br>Total params: 622,555<br>Trainable params: 622,555<br>Whale_Model1.ipynb | 0.32700 |
| 2 | Applying pre-trained VGG16 with unchanged imagenet weights and 3 fully connected layers.<br>Total params: 18,969,939<br>Trainable params: 4,255,251<br>Whale_Model2.ipynb | 0.36376 |
| 3 | Applying pre-trained VGG16 with option to update weights and 3 fully connected layers + a custom label weight parameters<br>Total params: 18,210,651<br>Trainable params: 18,210,651<br>Final_Whale_Model.ipynb | 0.38369 |

## IV. Results

## Model Evaluation and Validation

The final model is able to beat the suggested benchmark that was initially set in the project proposal that was mentioned earlier.

## Justification

The result after implementing the above technique fetched me a score of 0.38369 which is much higher than the benchmark that I had suggested to be beaten 0.32786.

Here are the supporting screenshots for my most recent submission and my leaderboard stand.

My most recent submission is a modest enough solution that is beating the benchmark that was suggested. Indicating Map @ 5 accuracy can be used to replace traditional methods.

## V. Conclusion

### Free-Form Visualization

There were about 40 classes which has only 1 image for training and there was a sample image that I was able to observe which has only 1 tail. Using them in train or test dataset is quite tricky. As a next step I would be spending time to understand the number of classes with such images. Attached below is the example of classes with single images

### Reflection

The complete solution can be summarised of applying the transfer learning architecture at the base layers and retraining the model to update the weights across all layers. Some of the important aspect that is worth noting and were challenging are:

- For the large multi class classification problems it is important to apply a deeper architecture.

- Running models with large batches will exhaust the resources hence applying steps per epoch makes it more efficient.
- Since the VGG16 was trained on over million images but was limited to 1000 classes using the existing imagenet weights does not provide a good result. This is because our model has 4251 classes against the pertained 1000 classes on a completely incomparable dataset to what we have on hand
- Since there is a very large class of New whale in the dataset it was very challenging and removing this class in training dropped the score significantly
- Due to acute class imbalance using weights helps.

## Improvement

As it is reflected in the leaderboard result @ link my current submission scores at about 67th place, the 1st place submission has a score of 0.62722 against mine which is 0.38369.

Indicating the huge opportunities that are left for improvements. Some of the methodologies that can possibly improve the model are:

- Implementation of other pre-trained models apart from VGG16, or other pretrained models that is more specific to the data (whale trained model)
- Trying an ensemble version of different pre-trained models can also possible help
- Another idea that I could think of is to create a classifier to identify the black or white side of the whale fluke and the trained dedicated classifiers for both sides. This  can prove to be effective considering that we have images representing both the sides.

(Note  - All numbers presented in this section are true as of 12-May-2018)