Machine Learning Engineer Nanodegree

Capstone Project

Abdalrhman Ahmed Ismail
October 11th, 2018

I. Definition

Project Overview

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. the challenge is to build an algorithm to identifying whale species in images. By analyzing Happy Whale's database of over 25,000 images, gathered from research institutions and public contributors. I will try to help to open rich fields of understanding for marine mammal population dynamics around the globe. Happy Whale is the provider of these data and problem. Happy Whale is a platform that uses image process algorithms to let anyone to submit their whale photo and have it automatically identified Papers related to this type

Problem Statement

The dataset of this problem contains thousands of images of humpback whale flukes. Individual whales have been identified by researchers and given an Id. the challenge is to build an algorithm to classifying whale species in images. What makes this such a challenge is that there are only a few examples for each of 3,000+ whale Ids.this problem was a competition on kaggle . for more details check this.
I attempt to solve this problem by train a convolution neural network to obtain a model to predict the whales' ids.

Metrics

- I used the Accuracy to test my model performance while training, to know if my model correctly predict whale's type or not.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

– The final solution can be measured by the Mean Average Precision @ 5 (MAP@5) which's provided by kaggle.

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

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

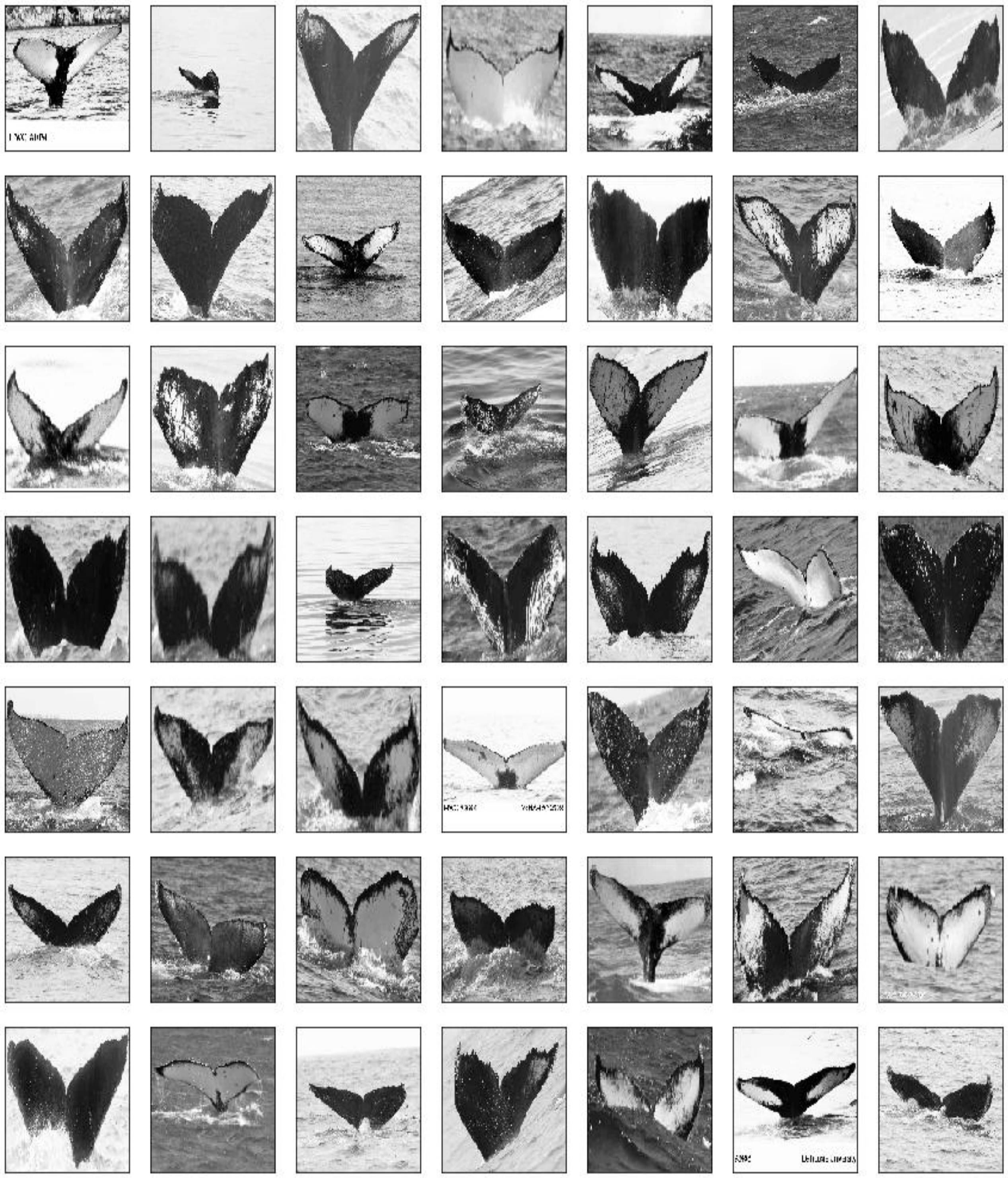
II. Analysis

Data Exploration

The dataset is provided at [Kaggle](), it contains the following files :

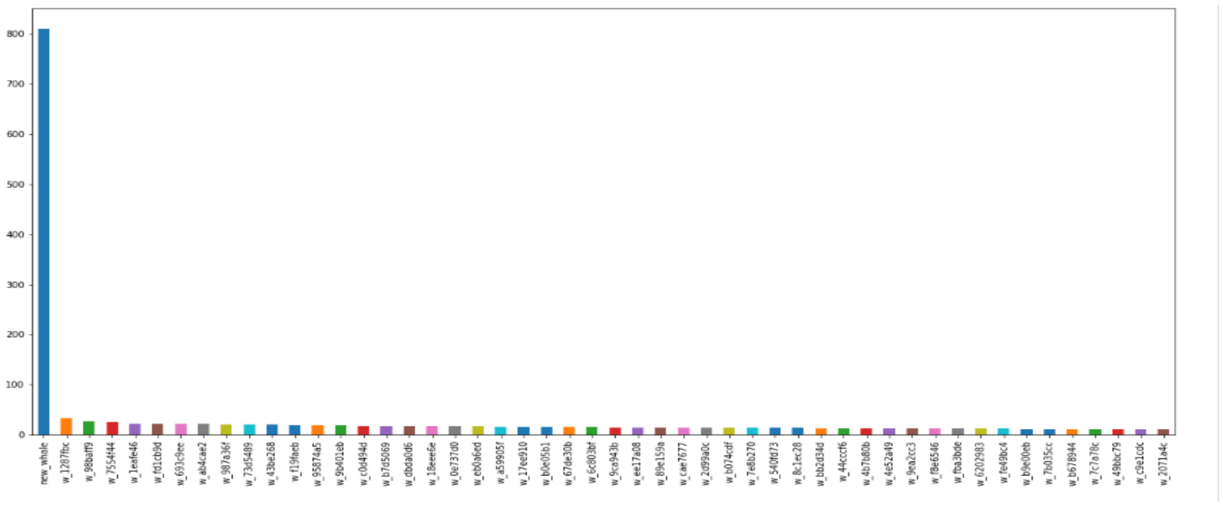
- *train folder: containing 9850 training images of humpback whale flukes.*
- *test folder: containing 15610 test images to predict the whale Id .*
- *train.csv: maps the training Image to the appropriate whale Id. Whales that are not predicted to have a label identified in the training data should be labeled as new_whale*
- *sample_submission.csv : a sample submission file in the correct format*

*The training images are colored and gray ,and its dimensions are not consistent, so I converted all images to gray with fixed size (128*128 dimension of each image).*



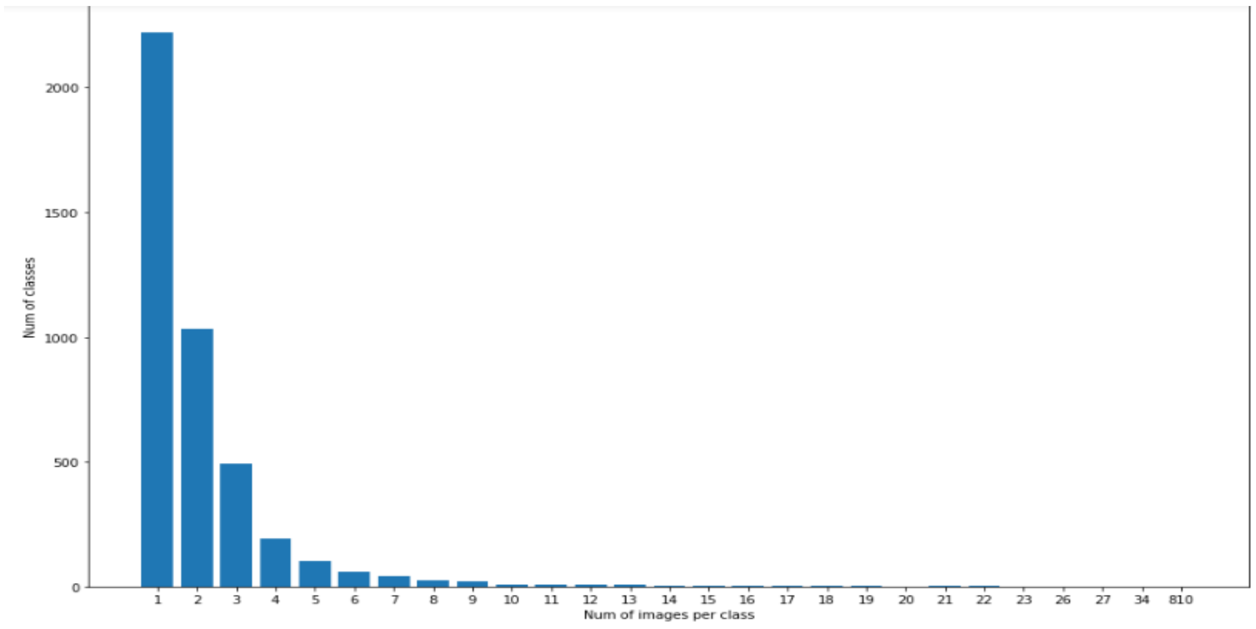
Samples of images after preprocessing

By visualizing the first 50 frequently classes, we see that dataset are not balanced, and the ‘new whale ’ class are most dominant.



Exploratory Visualization

there are 4251 different class in the dataset. By visualizing the number of categories by images in the training set, we see that there are a lot of classes have only one image in the training set , and that mentioned before in the competition details “What makes this such a challenge is that there are only a few examples for each of 3,000+ whale”. That’s make the classifier job harder .I tried to tackle this problem by using real time data augmentation.

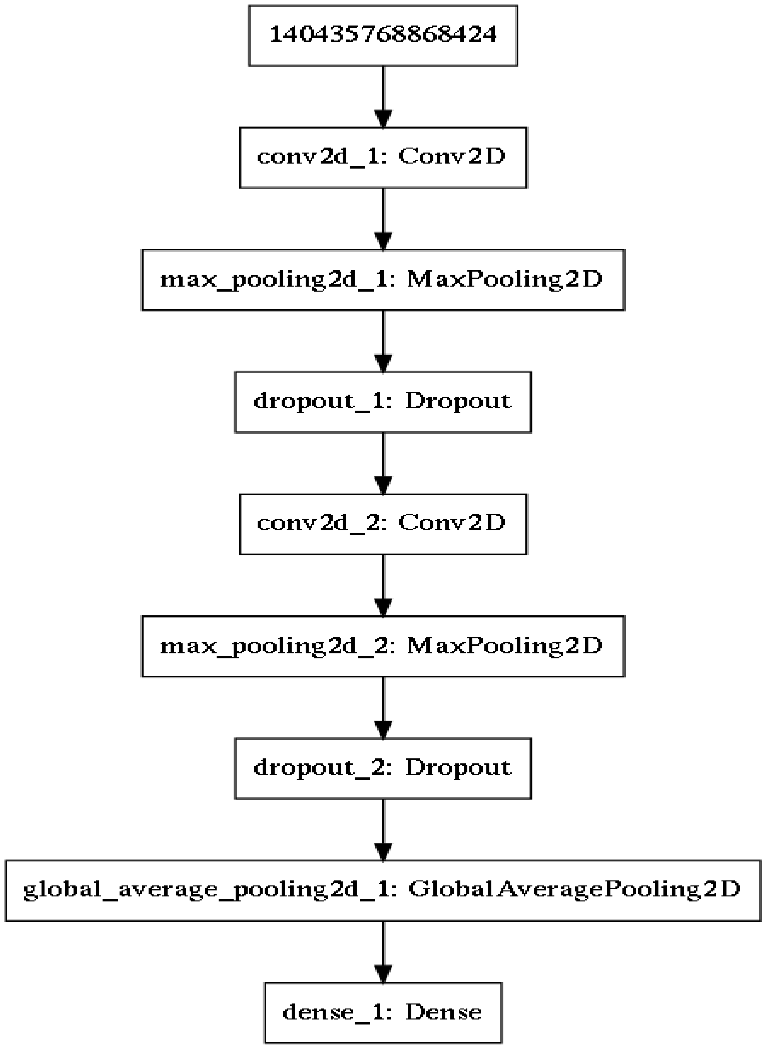


Algorithms and Techniques

I tried to solve this problem by training a CNN on the preprocessed training images to obtain a model to predict the whales' ids. I choose a CNN as it's more suitable for image classification and recognition

Benchmark

I started with simple CNN as a benchmark (2 convolution layers).
The architecture of benchmark CNN as follow:


140435768868424
conv2d_1: Conv2D
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout
conv2d_2: Conv2D
max_pooling2d_2: MaxPooling2D
dropout_2: Dropout
global_average_pooling2d_1: GlobalAveragePooling2D
dense_1: Dense

- Input
- Conv2d_1
- Max_pooling2d_1
- Droupout_1
- Conv2d_2
- Max_pooling2d_2
- Droupout_2
- Global_average_pooling2d_1
- Dense layer (output layer)

This model gets 0.0828 accuracy on the training set with loss 6.63 . and gets a 0.32665 score on the testing set (after uploading submission file on kaggle)

III. Methodology

Data Preprocessing

My preprocessing steps are:

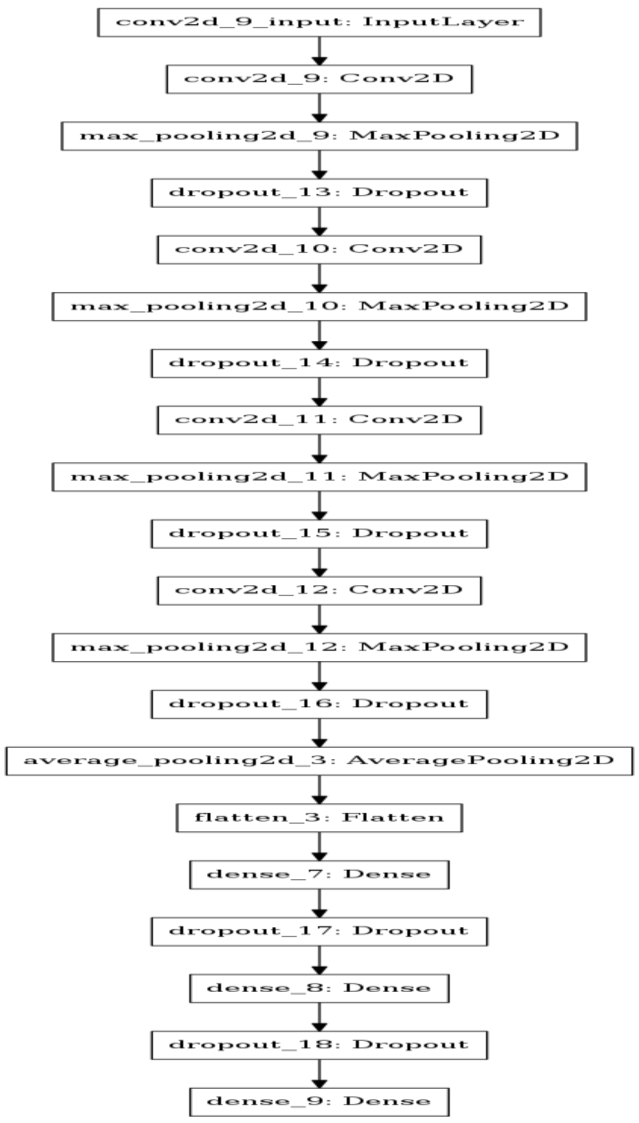
- Covert all images to gray.
- Resizing all images to be (128 * 128)
- Applying real time data augmentation by the following characteristics :

  - rescale the image by multiplying the data by the value : 1./255
  - rotation range : 20
  - zoom range 0.05
  -shear range : 0.02
  - shift range :0.02
  -width shift range 0.1
  -height shift range 0.1
  -applying horizontal flip on the image

Implementation

My solution workflow:

- *load all training images*
- *applying preprocessing step as described above*
- *define the network architecture*
- *define the loss function accuracy*
- *train the network and save the weights*



My_CNN Architecture

The network consist of 4 convolution layer each one followed by (Max pooling layer and dropout layer). And finally followed by 2 fully connected layers separated with dropout layers . As shown in the above figure.

Refinement

First I added 2 convolution layers to the benchmark architecture and train the model 200 epochs. The model gets an accuracy of 0.37 on the training set , but a score of 0.2714 on the testing set.

I decreased the number of epochs to 50 and gets 0.26 score on the testing set.

IV. Results

Model Evaluation and Validation

After Refinement step. The final model built by adding 2 fully connected layers at the end of the network and i decreases the training epochs to 20. The model gets 0.0836 accuracy on the training set with Loss 7.259 and score 0.32924 on the testing set

Justification

The results are shown in the following table:

The model	**Training accuracy**	**Training loss**	**Testing score**
benchmark	0.0828	6.63	0.32665
My_model	0.0836	7.259	0.32924

the results of my_model and the benchmark are quite similar, however the score of my model is a little bit larger.

In general I think this problem is a hard one , as a lot of whales' images are similar although it's image of different whales.

I want to use transfer learning, but I hardly test this model; as I test on my machine.

V. Conclusion

Free-Form Visualization

In classification problems it's common to visualize the final classifier output using confusion matrix, however that's pretty hard here as we have 4251 different class.

So, I consider the testing score of the model will be a good indicator instead (which is a 0.32924).

Reflection

In this project I tried to classify whale Id from humpback whale images. I downloaded the dataset from kaggle. Then I applying the pre-processing on the images before the training. I built the benchmark CNN model, and try to improve it.

Finally I used accuracy metric to test the both model while training process. And uploading the final results to get MAP@5 score on kaggle. The interesting parts I found that how to tackle the problem of a lot of classes only have few images in the training set. I tried later to tackle this issue by augmentation.

Improvement

I think the following will improve the solution:

- gathering more data (i.e. whale images) , as there are a lot of classes only have few images
- divide the data into groups according to number of images per class and apply augmentation only to classes have few images
- using transfer learning (like Resne50 , VGG16)