Machine Learning Engineer Nanodegree

Capstone Proposal-Humpback Whale Identification

Ravi Mittal
May 11th, 2018

Proposal

Domain Background

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.

In this competition from Kaggle, the challenge is to build an algorithm to identify whale species in images. We'll analyze Happy Whale's database of over 25,000 images, gathered from research institutions and public contributors.

Happy Whale is a platform that uses image process algorithms to let anyone to submit their whale photo and have it automatically identified.

I chose this competition from Kaggle as I am inclined and interested in field of Computer Vision and Deep Learning.

Kaggle Competition path

https://www.kaggle.com/c/whale-categorization-playground

Problem Statement

The problem here is to predict the images in the test folder from model trained using train.csv file and images in train folder. After prediction we need to find upto 5 Ids from train.csv that maps to each predicted images of test folder.

We have been provided with training whale's fin images (in train folder) and testing whale's fin images (in test Folder). We have also been provided with a labelled csv file (train.csv) that maps (or labels) all training whale's fin images names (in train folder) with Id (or class) of the image.

One Id is mapped to multiple whale's fin images in train.csv, that means that there a multiple images of same Id (or class e.g. w_1287fbc) in training images (in train folder).

For each Image in the test set, we have to predict up to 5 labels for the whale Id. Whales that are not predicted to be one of the labels in the training data should be labeled as new_whale. The file should contain a header and have the following format:

```
Image,Id
00029b3a.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
0003c693.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
...
```

Datasets and Inputs

We have been provided with training whale's fin images (in train folder) and testing whale's fin images (in test folder). We have also been provided a labelled csv file (train.csv) that maps (or labels) all Training Whale's Fin images (in Train folder) with Id (or class) of the image. All datasets have been provided by Kaggle.

Below is the format of train.csv file

```
Image,Id
00029b3a.jpg,w_1287fbc
0003c693.jpg,w_1eafe46
...
```

Sample_submission.csv has been provided to show that the expected output is in below format

```
Image,Id
00029b3a.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
0003c693.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
...
```

Solution Statement

The problem here is to predict the images in the test folder from model trained using train.csv file and images in train folder. After prediction we need to find upto 5 Ids from train.csv that maps to each predicted images of test folder.

To achieve this, I would follow below steps

Step 1:

I am planning to use transfer learning using already trained models such as VGG16, VGG19 or Resnet50 for prediction of image data. I would use one or more layers on top of output of already trained model.

Step 2:

I would make predictions (using trained model in step 1) on both training and testing images (images in train and test folders).

Step 3:

To find upto 5 Ids from train.csv that maps to each of the images of test folder (as per expected output in sample_submission.csv), I would then train k-nearest neighbor algorithm using predictions on training images (in Step 2) as input and using k=5.

Step 4:

I would then use predicted data of test folder images (predicted using transfer learning in Step 2) to find 5 nearest neighbors (predict from trained model in Step 3 using k-NN) of each of the images in test folder. Since I have trained k-nearest neighbor using predictions of images in train folder, I would be able to map 5 nearest neighbor output on each image of test folder with image name in train.csv file using iterations of while loop etc. and then insert the output in a csv file which would be similar to sample_submission.csv.

Benchmark Model

We will first create a model without using transfer learning first and find precision based on that model which would be used as Benchmark.

The final model would be more sophisticated as it would use transfer learning of already tested and pre-trained model. Its precision would be compared with that of Benchmark model.

Evaluation Metrics

Evaluation would be made based on Mean Average Precision.

Mean Average Precision

Let's say that there are some users and some items, like movies, songs or jobs. Each user might be interested in some items. The client asks us to recommend a few items (the number is x) for each user. They will evaluate the results using mean average precision, or MAP, metric. Specifically MAP@x - this means they ask us to recommend x items for each user. So what is this MAP?

First, we will get M out of the way. MAP is just an average of APs, or average precision, for all users. In other words, we take the mean for Average Precision, hence Mean Average Precision. If we have 1000 users, we sum APs for each user and divide the sum by 1000. This is MAP.

So now, what is AP, or average precision? It may be that we don't really need to know. But we probably need to know this:

- we can recommend at most x items for each user
- it pays to submit all x recommendations, because we are not penalized for bad guesses
- order matters, so it's better to submit more certain recommendations first, followed by recommendations we are less sure about
  So basically we select x best candidates (in order) and that's it.

Source:

http://fastml.com/what-you-wanted-to-know-about-mean-average-precision/

Submissions are evaluated according to the Mean Average Precision @ 5 (MAP@5):

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

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

Submission File

For each `Image` in the test set, you may predict up to 5 labels for the whale `Id`. Whales that are not predicted to be one of the labels in the training data should be labeled as `new_whale`. The file should contain a header and have the following format:

```
Image,Id
00029b3a.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
0003c693.jpg,new_whale w_1287fbc w_98baff9 w_7554f44 w_1eafe46
...
```

Project Design

The problem here is to predict the images in the test folder from model trained using train.csv file and images in train folder. After prediction we need to find upto 5 Ids from train.csv that maps to each predicted images of test folder.

To achieve this, I would follow below steps

Step 1:

I am planning to use transfer learning using already trained models such as VGG16, VGG19 or Resnet50 for prediction of image data. I would use one or more layers on top of output of already trained model.

Pseudo Code

> Create a model using transfer learning from already trained model VGG19 or Resnet50 model and train that model using training data in train.csv and images in train folder.

Step 2:

I would make predictions (using trained model in step 1) on both training and testing images (images in train and test folders).

Pseudo Code

> Predict both training and testing images (images in train and test folders) and collect the prediction data of each image in respective lists called trainImages_Predictions and testImages_Predictions.

Step 3:

To find upto 5 Ids from train.csv that maps to each of the images of test folder (as per expected output in sample_submission.csv), I would then train k-nearest neighbor algorithm using predictions on training images (in Step 2) as input and using k=5.

Pseudo Code

> Use k-nearest neighbor algorithm with k=5, and train it using trainImages_Predictions.
> kNeighbors = NearestNeighbors(n_neighbors=5)
> kNeighbors.fit(trainImages_Predictions)

Step 4:

I would then use predicted data of test folder images (predicted using transfer learning in Step 2) to find 5 nearest neighbors (predict from trained model in Step 3 using k-NN) of each of the images in test folder. Since I have trained k-nearest neighbor using predictions of images in train folder, I would be able to map 5 nearest neighbor output on each image of test folder with image name in train.csv file using iterations of while loop etc. and then insert the output in a csv file which would be similar to sample_submission.csv.

Pseudo Code

distances_testImages, neighbors_testImages = neigh.kneighbors(testImages_Predictions)

for filepath, dist, kNeighbor in zip(test_file_names, distances_testImages, neighbors_testImages):

> For each of 5 neighbors found, find corresponding Ids from train.csv, and map all neighbors Ids with same filename in train.csv file and save it new output.csv file.