Capstone_Proposal

June 14, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Shehjar Kaul
Thursday 14, 2018

1.2 Proposal

Humpback Whale Identification Challenge (Kaggle Competition)

1.2.1 Domain Background

The Domain of the project proposed falls into the category of image classification. In general, to aid the whale conservation efforts, scientists use an image identification tool to get information about the whale species based on the image of it's fluke.

Image classification is one of the basic computer vision problems that can be solved via Deep Learning techniques and it first became a mainstream with its resounding success at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012 [1]. The algorithm then was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Given the success of deep learning in the field of computer vision, almost every following Image recognition/classification problem was dealt using this new technology.

Since then, there have been a plethora of neural network architectures developed for various applications. In my case, I find the idea of identifying image patterns really interesting and therefore I chose to develop a solution to an Image recognintion problem for my capstone project in this nanodegree. For the problem at hand, it is very challenging as one can see that the training data in this case is very skewed with respect to the classes and there are a lot of classes that would need to be trained using augmentation techniques due to unequal distribution of images per class. Challenges are typical in solving a real world problem and I intend to go through the "deeper layers" of the project to finally come up with an accurate classifier!

```
In [ ]: import math
        import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np
        import cv2
        from tqdm import tqdm
```

```
%matplotlib inline
```

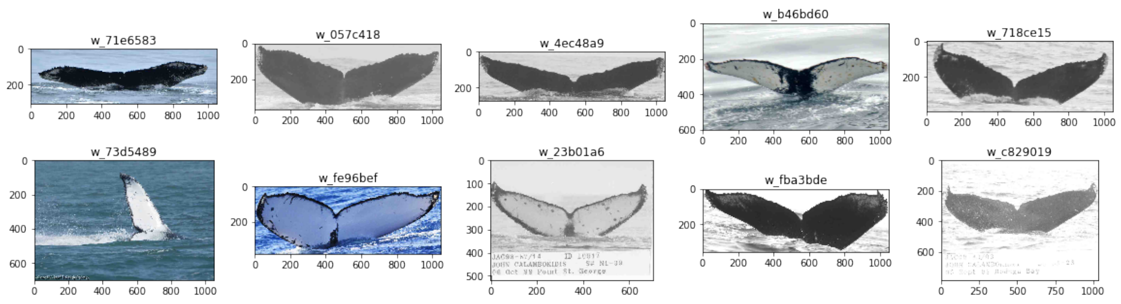
1.2.2 Problem Statement

Species of whales have to be classified as per the photo of their fluke. Since this problem falls in the category of object classification, I plan to use convolutional neural network architectures to classify images. The inputs to this classifer would be images (maybe Grayscale or Colored) and the output would be a list of probabilites for all classes. The class with the maximum probability shall be chosen as the class which the image belongs to. For this particular problem, this is an arduous task as the classifier will have to identify extremely minute and unique patterns on the image of the fluke to classify 4251 unique species of whales, which is quite a high number of classes.

Some of the examples of the fluke images from the training set is given below-

```
In [2]: # Plotting image samples
        def plot_train_imgs(img_data, fsize = (5, 5)):
            n_imgs = img_data.shape[0]
            img_names = img_data['Image'].tolist()
            y_label = img_data['Id'].tolist()
            cols = 5
            rows = int(n_imgs/cols) + 1
            rem = n_imgs%cols
            if rem == 0:
                rows -= 1
            fig = plt.figure(figsize=fsize)
            for i in range(n_imgs):
                img_path = 'data/train/'+img_names[i]
                img = cv2.imread(img_path)
                img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                ax = fig.add_subplot(rows, cols, i+1)
                ax.imshow(img_RGB)
                ax.set_title(y_label[i])
```

```
In [3]: # Get Data from the dataset
        train_df = pd.read_csv('data/train.csv')
        plotData = train_df.sample(n=10, random_state=99)
        # display(plotData.head())
        plot_train_imgs(plotData, (20,5))
```



```
In [4]: y_label = train_df["Id"]
        n_classes = len(y_label.unique())
        print('There are a total of '+str(n_classes)+' unique classes')
```

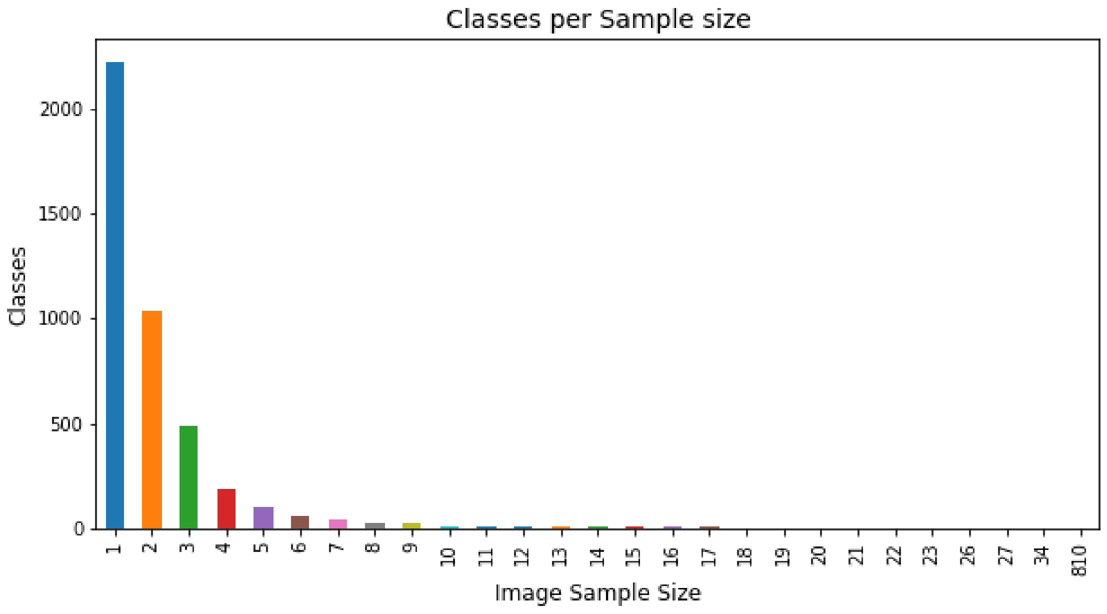
```
There are a total of 4251 unique classes
```

1.2.3 Datasets and Inputs

Since this problem is taken from the Kaggle competition of Humpback Whale Identification Challenge, the data involved in this competition is gathered from the same source as well. The dataset consists of training and testing set of images of fluke. The training set is matched with its respective whale species in `train.csv` file. We observe that the training images per class is very skewed, with most of them having just one example. This makes the data augmentation strategies very imperative. Some of the analysis can be seen below-

```
In [18]: # Checking the distribution of images with respect to the classes
         y_freq = y_label.value_counts()
         classes = y_freq.index.values
         freq = y_freq.values
         dfSamplesPerClass = y_freq.value_counts().sort_index()
         ax = dfSamplesPerClass.plot(kind='bar', figsize=(10,5))
         ax.set_title('Classes per Sample size', fontsize=14)
         ax.set_xlabel('Image Sample Size', fontsize=12)
         ax.set_ylabel('Classes', fontsize=12)
```

```
Out[18]: Text(0,0.5,'Classes')
```



1.2.4 Solution Statement

The solution to this problem is to implement a convolutional neural network model that accurately classifies the whale according to the photo of its fluke. This can be achieved developing neural architectures using tensorflow/Keras libraries, or doing a transfer learning on an already trained neural network architecture. Because of the skewed training set, there is a need for doing data augmentation too. The input dimensions to the network shall be made uniform and training images shall be duly cropped or resized. The training dataset shall be split into training and validation set, but this would be only after data augmentation (to increase the volume of images per class); otherwise the validation set with 1 image per class would definitely give a worse accuracy as it wasn't a part of the training dataset. Depending on the validation loss, an optimal classifer model shall be saved and evaluated for prediction.

1.2.5 Benchmark Model

Considering it is a skewed training set, my benchmark model would predict the most frequently occuring class in the dataset. It's accuracy and categorical cross entropy loss shall be calculated below.

```
In [33]: class BenchmarkModel():
             def __init__(self):
                 self.freq_class = 0

             def fit(self, train_X, train_y):
                 y_freq = train_y.value_counts().index.values
                 self.freq_class = y_freq[0]
                 print('The frequently used class is', self.freq_class)

             def predict(self, train_X):
                 return pd.Series(data=self.freq_class, index=train_X.index)

In [34]: model = BenchmarkModel()
         model.fit(train_df['Image'], train_df['Id'])
```

```
The frequently used class is new_whale
```

```
In [38]: #Calculating accuracy of the baseline model
         y_predict = model.predict(train_df['Image'])
         y_correct = y_predict == y_label
         accuracy = sum(y_correct)/y_correct.shape[0]
         print('The baseline model works with an accuracy of {0:.2f}%'.format(accuracy*100))
```

```
The baseline model works with an accuracy of 8.22%
```

Assuming that the probability of prediction is 1 in every case, the losses would go up to infinity, which obviously doesn't add any meaning to the given state of the Benchmark.

1.2.6 Evaluation Metrics

Being a multi-class classification problem, categorical cross entropy as loss metric seems like a logical metric to proceed with. Other than that, the accuracy should also be good.

In theory, if the number of cla-sses M > 2, then categorical cross entropy can be calculated as the negative sum of the product of correct classification label $y_{o,c}$ (with respect to observation o and actual class c) and logarithm of prediction probability $p_{o,c}$ as shown below-

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

1.2.7 Project Design

Considering the given problem of Whale Species classification, the following steps shall be undertaken to come up with a good model which can predict the species of whale with an input of a photo of its fluke-

- Preliminary analysis of the data (size of the picture, color scheme of the picture, what do the class labels really mean etc)
- Pre-processing the images, including the augmentation of the training dataset with respect to different augmentation schemes like flipping, rotating, random translating, zooming etc.
- Splitting the training dataset into training and validation dataset such that the dataset has a proportional number of images per class in both sets.
- Creating/loading a Convolutional neural model and setting the input and output pipelines including the metrics and optimizers
- Evaluating different network architectures by training with respect to the metrics and choosing the best architecture giving the best metric value
- Publishing the best architecture and a short study about the validation images that failed to be classified properly.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.