# Capstone Project

Machine Learning Engineering - Nanodegree
Saurabh Agrawal
March 2, 2018

# Dog Breed Identification

## Definition

### Project Overview

This project is based on a kaggle competition. The objective of the project is to train a model which can predict the breed of a dog based on the image.

### Problem Statement

There are more than 360 breeds of dogs as recognised by FCI (Fédération Cynologique Internationale) - https://en.wikipedia.org/wiki/List_of_dog_breeds_recognized_by_the_FCI.

This is one of those problems where if trained well, a model can easily outperform humans. A regular person can mostly identify the animal type (cat vs dog vs monkey) and differentiate between 4-5 popular breeds. Even for an expert, it might be difficult to differentiate between all the 360 breeds, let alone

remember details of all those. This could be a common problem if one considers the breeds of other similar animals like cats, monkeys, horses, cows, pigs etc.

## Metrics

The model for this problem will define probability of an input for each class. For this problem, a multiclass log loss function will be ideal. Kaggle has also chosen the same for scoring the project. For test data Kaggle will not provide us the labels but the directly the log loss score.

# Analysis

## Data Exploration

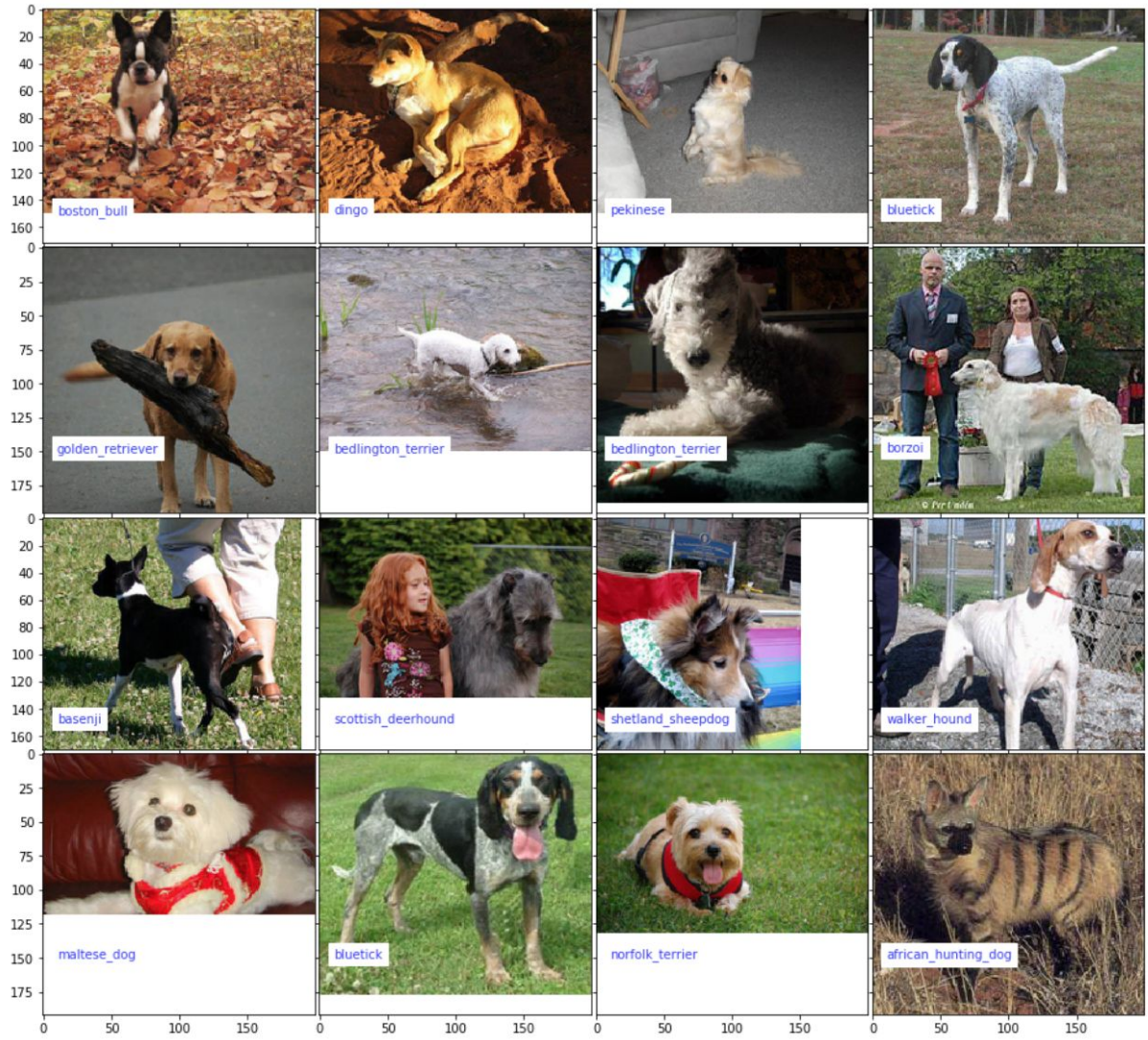
There are total 10222 images provided in training set. The training set images belongs to overall 120 classes (breed of dog). Average number of images per class is 85 while lowest number of images for a class is 66.

There is also a separate test set of images. There are total 10357 images in the test set. I do not have the labels for them as these set of images will be used by kaggle to calculate the score of the model.

Log Loss vs Predicted Probability Graph

## Exploratory Visualization

Below are some images taken randomly from the training set to demonstrate the nature of image




The resolution of the images is not same. Aspect ratio of images is also different. Dogs are not always in the center of image so we can not crop the images. It will be difficult to train a CNN with varying number of inputs. To avoid this, we will resize all the images to one common size before feeding to the neural network.

Log Loss vs Predicted Probability Graph

## Algorithms

Convolutional Neural Net is a popular model for image processing and recognition problems. There are many architecture popular for CNN. Some of the most popular architecture are ImageNet, AlexNet, Inception, Xception. There are also pretrained models available on these architecture which are able to provide error rates as low as 5%. Inception and Xception has had most success in image recognition tasks. There pretrained models are trained on large image data set (~1.2 million images) already. A common approach in CNN is to use these pretrained models, and then take 1 or 2 top layers off, add a fully connected layer and a softmax layer and retrain it for our image set.

Each layer of CNN identifies more and more complex patterns in the images. Eg. First couple of layers identifies mostly edges, then later layer starts identifying shapes and so on. Hence training of initial layers is going to be similar in image recognition models and can be reused across other image recognition models. It is the last couple of layers mainly which are getting trained to differentiate between the labels for a particular problem.

Performance of Xception (eXtreme inception) has been slightly better than inception model. We plan to make use of pre trained Xception using keras library and then retrain its final layers to our image classification problem.

## Benchmark

The target is to achieve a loss of around 0.3 and to have the score better than atleast 50% of kaggle leaderboard.

# Methodology

## Data Preprocessing

The resolution of images is not standardized. All the images have different images. Moreover standard input size for Xception is 299x299x3. All our images

have resolution in format of AxBx3. So we need to resize the images before we start feeding it to our model. I am using preprocess_input method of keras.applications.xception class.

I also split data into 80:20 ratio using 80% of the data for training and 20% for validation. For testing there is a separate set of images provided by Kaggle. Testing set has 10357 images. I do not have the labels for the test set and for test set I upload the probability results to kaggle and get the score.

## Implementation

Using Keras library, I instantiate a model of Xception without the top layer. This model would be used to get predictions for both training data and validation data. Then we will train another logistic regression model which will be fit on the training data prediction and use validation data prediction for validating the logistic regression model predictions. Thus Logistic regression model will act as the top layer of Xception model.

It would not be possible to load all the images and feed to CNN at the same time as system will run out of memory. To avoid this, I am loading only a smaller set of images, feeding it to CNN and then saving CNN prediction of it. The prediction object out of CNN is quite small. In the second step, I am able to feed entire prediction set out of CNN to my logistics regression model. In case, we have a very large number of images, then we can use Stochastic Gradient Descent in place of logistic regression for second step.

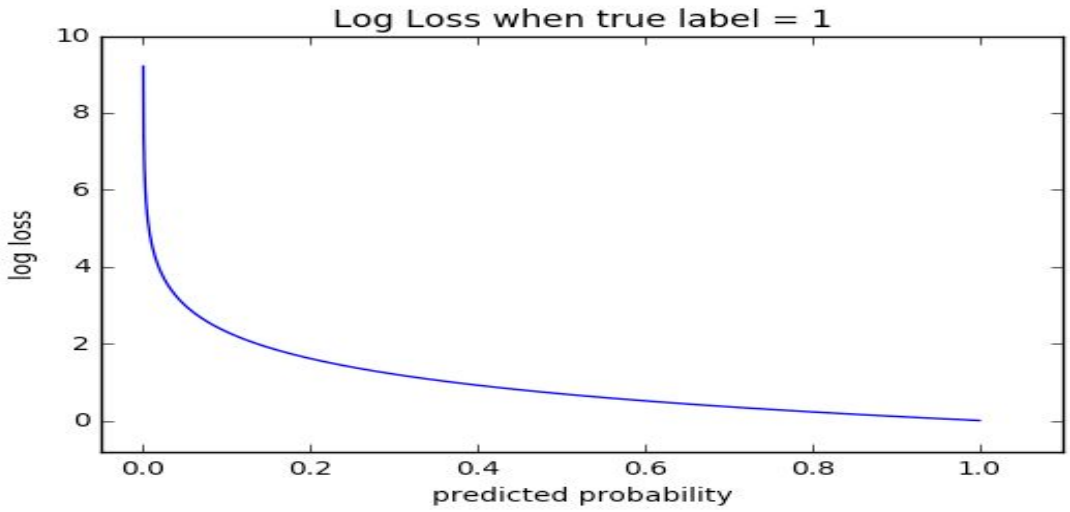
To get the final score from kaggle on the test set, I get the predictions on the test set using my model and store them in a separate csv file in the format defined by kaggle. Then I upload this file to kaggle to get the score on test set.

Log Loss vs Predicted Probability Graph

# Results

## Model Evaluation and Validation

My model was able to achieve a loss score of 0.3348 and accuracy of 89.9% . For test data, kaggle only provided the loss score which was 0.34205. Considering the loss score of my validation set and test set is very close, I can say that my model is trained well and performing well on unseen data as well. Below image provides the log loss vs predicted probability graph. A log loss score of 0.34 means a predicted probability of around 0.8 which is good for our algorithm.




There could be ways to improve this model further. The training dataset for training this model was really small. We had on average 85 images per class. If we could get more images to train the system, then its accuracy should definitely improve. Another approach could be to use data augmentation where new images will be generated from the existing set of images itself by randomly flipping them at different angles.

# References


https://www.tensorflow.org/tutorials/image_recognition


Log Loss vs Predicted Probability Graph

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

Log Loss vs Predicted Probability Graph