Machine Learning Engineer Nanodegree Capstone Proposal

Kunal Gusain

May 1, 2017

Cats vs Dogs Image Classification

Domain Background

This problem belongs to the computer vision branch of the machine learning. This project or competition was first introduced on Kaggle in 2013 [1]. This is one of the most popular competitions ever organized on Kaggle. This is a type of binary image classification problem where we have to predict the class of an input image. The classes, in this case, are "cats" and "dogs", i.e the input image will be of either dog or cat. I personally want to do this project because this project can be considered as a good start to the venture in the field of computer vision or image classification problems. As I am new to Deep Learning and CNN [6], this project will help me get a better grasp of the algorithms in deep learning and its intricacies.

Problem Statement

The problem statement is to design an algorithm which is capable of finding out the correct animal (cat or dog) in new unseen images. For each image in the test set, we have to predict the class or label to which the new image belongs (cat or dog).

Datasets and Inputs

Asirra [3] (Animal Species Image Recognition for Restricting Access) is a Human Interactive Proof that works by asking users to identify photographs of cats and dogs. This task is difficult for computers, but studies have shown that people can accomplish it quickly and accurately. Many even think it's fun!

Asirra is unique because of its partnership with Petfinder.com, the world's largest site devoted to finding homes for homeless pets. They've provided Microsoft Research with over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States. Kaggle is fortunate to offer a subset of this data for fun and research.

The dataset contains thousands of images of cats and dogs. There are about 25,000 training images of dogs and cats. Each image in this folder has the label as part of the filename. The test folder contains 12,500 images, named according to a numeric id.

Solution Statement

While random guessing is the easiest form of prediction, various forms of image recognition can allow a person to make guesses that are better than random. There is enormous diversity in the photo database (a wide variety of backgrounds, angles, poses, lighting, etc.), making accurate automatic classification difficult. In an informal poll conducted many years ago, computer vision experts posited that a classifier with better than 60% accuracy would be difficult without a major advance in the state of the art. For reference, a 60% classifier improves the guessing probability of a 12-image Human Interactive Proof from 1/4096 to 1/459.

The benchmark model uses deep convolutional neural networks [3]. As deep learning algorithms like Convolutional Neural Networks (CNN's) have been very successful in the field of computer vision, I also plan to use CNN for the binary classification of this task.

The training images are of various resolutions, during the preprocessig step I will resize all the images to a common size. During the preprocessing I might also convert the image to grayscale (I can analyse results later on whether converting to grayscale helped in accuracy or not.). I will then normalize the input image before feeding it into the CNN model.

I will be using Keras python library for the CNN implementation.

Benchmark Model

The best model for the given problem statement has an accuracy of ~98.9% which was given by Pierre Sermanet on Kaggle competition. Pierre Sermanet [2] used the convolutional neural network to achieve this accuracy. His system was pretrained on Imagenet which was subsequently refined for dog vs cat classification. According to him, he used Overleaf library for implementing deep learning algorithm. The model given by him had just ~1.09% error compared to the error of ~1.69% for the person coming at second place in the competition. The benchmark model which I will use is a shallow CNN (2 conv layers).

Evaluation Metrics

The evaluation metrics which I plan to use are general accuracy score and log loss metric. The evaluation metric which was used in the Kaggle competition was log loss metric but the best model given by Pierre Sermanet [2] uses accuracy score as the evaluation metric.I log loss metric and later on use accuracy score for the summary of the model.

Log-loss, or logarithmic loss, gets into the finer details of a classifier. In particular, if the raw output of the classifier is a numeric probability instead of a class label, then log-loss can be used. The probability essentially serves as a gauge of confidence. If the true label is "0" but the classifier thinks it belongs to class "1" with probability 0.51, then the classifier would be making a mistake. But it’s a near miss because the probability is very close to the decision boundary of 0.5. Log-loss is a “soft” measurement of accuracy that incorporates this idea of probabilistic confidence.

Mathematically, log-loss for a binary classifier looks like this:

$$logloss = \sum_{i=1}^{N} y_i log(p_i) + (1 - y_i) log(1 - p_i)$$

Here, p_i is the probability that the i-th data point belongs to class "1", as judged by the classifier. y_i is the true label and is either "0" or "1". The beautiful thing about this definition is that it is intimately tied to information theory: Intuitively, log-loss measures the unpredictability of the “extra noise” that comes from using a predictor as opposed to the true labels. By minimizing the cross entropy, we maximize the accuracy of the classifier.

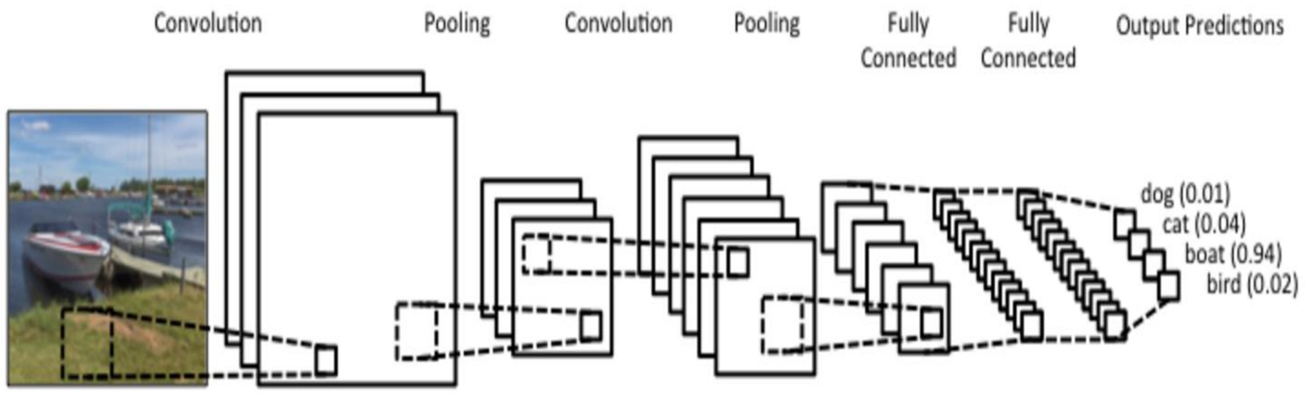
Project Design

Programming language: Python 3

Libraries: Keras, Scikit-learn, Matplotlib

Work flow:

- ➢ Loading the data for preprocessing. First step in preprocessing is to resize all the images this can be done using PIL library of python. Next step is to convert the image to grayscale (have not decided about it yet.). After this normalize the input image to feed in the CNN.
- ➢ Training a shallow convolutional neural network on the data for the benchmark model and analyzing the results.
- ➢ Train a SVM [5] with linear kernel and compare its result with the deep learning model.
- ➢ Training a deeper CNN on the data for improving the result. The architecture which I have in mind is Alexnet [6]. I might reduce the number of convolution layers depending on the capacity of my computer.
- ➢ I will use Adam or SGD as my optimizer function in the CNN.



- ➢ Compare the results of Alexnet with other models like shallow convolutional neural network and SVM with linear Kernel.
- ➢ Optionally trying to visualize the output from each layer of convolutional neural network for a better understanding of the model.

Refrences:

[1] https://www.kaggle.com/c/dogs-vs-cats

[2] http://cs.nyu.edu/~sermanet/

[3] http://research.microsoft.com/en-us/um/redmond/projects/asirra

[4] https://en.wikipedia.org/wiki/Convolutional_neural_network

[5] https://en.wikipedia.org/wiki/Support_vector_machine

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.