Dog Breed Identification

Capstone Project Proposal

Grzegorz Lippe

February 24, 2021

Domain Background

The task of identifying dog breeds is a known topic amongst the community for machine learning. I've chosen the dog breed identifier amongst the proposed projects, because it attracted me the most.

The benefit of the dog breed problem is the high availability of labeled datasets which can be used for training and validation of the model. A quick search on kaggle and github provide lots of sources.

Also I must admit that it is the most supported task on Udacity's site.

Problem Statement

The goal of this project is to provide an app, that can take an arbitrary picture and return a statement of a most likely dog breed. If a picture of a cat or human is provided, the app should identify the human, but still provide a statement of a most similar looking dog breed.

Datasets and Inputs

The data consists of two different sets, provided by Udacity and are described as follows:

dog dataset The dog data set is already divided into three different datasets (folders) for test, training and validation. Each folder contains sub directories of the specific formating `DDD.breed_name`, where `DDD` represents a 3 digit number, followed by the breed name after a dot. Each subfolder contains a hand full of images of the specific breed.

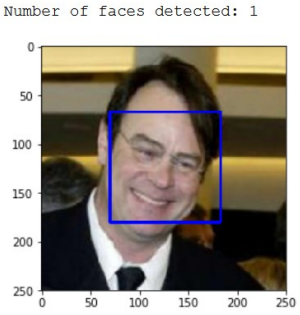
Overall there are 133 different dog breeds, and 835 images provided for validation, 6680 images for training and 836 images for testing. The images within the training dataset are unbalanced, the amount of images per breed varies from 26 for the Norwegian Buhund and the Xoloitzcuintli to 77 for the Alaskan Malamute. So some breeds are represented roughly 3 times more often.

The dog images contain a singular dog each, mostly of the whole dog, sometimes only of the snout, but they are not equally sized, they vary from 5 kB up to 5 MB. Their aspect ratios cover all ranges between portrait, landscape and quadratic.

human dataset The human dataset consists of 13233 images of 5749 persons, which are stored in a separate directory named after each celebrity. The images of the people are already cropped and centered around their face and all of the same size of 250x250, but the backgrounds vary. Sometimes there are additional people in the background.

Solution Statement

The human face detection will use an OpenCV model with a pre-trained haar cascade dataset of `haarcascade_frontalface_alt.xml` This algorithms provides bounding boxes for each face found in a given image, like shown in the figure below with an image of Dan Ackroyd:



For the dog breed detection a pre trained torch model `resnet50` is used within a transfer learning approach. The pre trained model will be trained on the described dog dataset until it reaches a suitable level of accuracy. The training image data will be transformed (random resized crop) to a documented [1] value of 224. Also in the footnote a horizontal flip is performed as an augmentation of the image. This may be tried as well.

Since the app shall provide some fun with human images the accuracy of the prediction may not be that high.

[1] github.com/pytorch/vision

Benchmark Model

As a benchmark I will use a naive predictor. The dataset is slightly unbalanced so assuming that there are 133 different dog breeds, some of the occurring up to three times more than others, the naive approach should guess the correct breed (b) with $p_b = \frac{k_b}{3 \cdot 133}$, $k_b = [1 \dots 3]$, or less than 1%. The implemented model shall get at least 10%.

Evaluation Metrics

For the evaluation the labelled data is used. For example the accuracy of the face detection algorithm can be measured by feeding it with (previously labelled) images of humans:

$$Score = \frac{\text{Humans detected}}{\text{Images presented}}$$

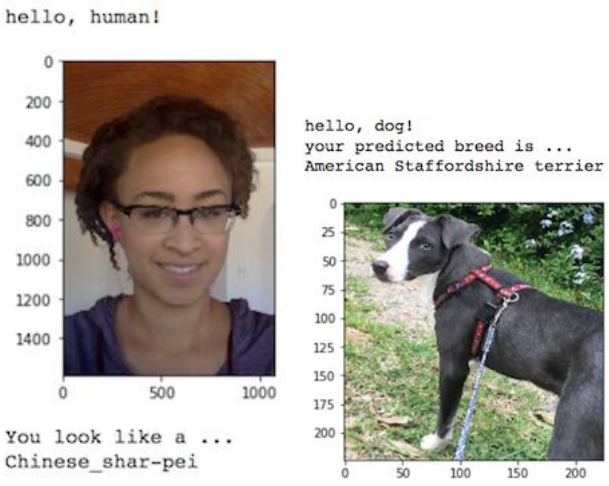
A slightly modified algorithm can be used to evaluate the dog detecting model. The dog detector provides a breed identification on top, so in the first step the upper term can be used to judge the accuracy of the dog detection. Additionally the quality of the breed identification can be measured as follows:

$$Score = \sum_{breeds} \frac{\text{correct breed identified}}{\text{breed provided}}$$

Project Design

Main App

In the initial state the app will consist of a python function which will provide the following informations: the provided image and a statement of whether a dog or a human was detected. Additionally a most likely dog breed is provided. The output may look as follows:



In the first iteration the app will fail to provide a reasonable output if some random "non-dog and non-human" image is provided.

Functions

The main app above needs two different predictor models to work, a `human_face_detector` and a `dog_breed_detector`. The two functions are outlined as follows:

For the `human_face_detector`:

- turn the image into grey scale.
- use the previously described OpenCV model to detect faces in the given image.
- if there are any, proved the bounding boxes for the detected faces.

For the `dog_breed_detector`:

- load the image into a pyTorch data set, resize (224, 224) and transform to Tensor.
- Get the predicted dog breed of the picture.
- There will always be a most likely breed.

For the `main` function:

1. Was there any human detected on the image?
2. If so, provide a statement of a similar looking dog breed.
3. Are there any dogs on the Image? If so, provide the most likely breed