UDACITY MACHINE LEARNING ENGINEER NANODEGREE

Capstone Proposal

~

DOG BREED CLASSIFIER

Akash Kumar
September 13th, 2020

DOMAIN BACKGROUND

Humans interact with the world in several ways. One of the most important of these interactions is through the visual medium. Humans can easily interpret the three-dimensional world, and analyze and identify the objects seen. The modern computers today are capable of working with captured images, and the booming internet usage over the past couple of decades has provided us with gigantic amounts of image data. As a result, a lot of development has been going on to get a machine to extract meaning out of images. Some real-world examples include Optical Character Recognition, medical imaging, self-driving cars, face recognition, object classification, etc.[1]

With the advancement in computer hardware, we've been able to perform more and more computationally expensive tasks, which is one of the reasons that deep learning took off. One breakthrough application of Deep Neural networks surfaced from the 2012 ImageNet challenge, was the use of "Convolutional Neural Networks", to win the ImageNet competition. In this project, I aim to exemplify the use of convolutional neural networks by demonstrating a "Dog Breed Classifier".

PROBLEM STATEMENT

The objective of this project is to build a "Dog breed classifier", which will be able to take a user-supplied image as input, and classify it as per the following conditions:

1. Given an image of a dog, the algorithm should be able to identify an estimate of the canine's breed.
2. If supplied with an image of a human, the code will identify the resembling dog breed.

DATASETS AND INPUTS

As mentioned earlier, the classifier will accept images as an input. The dataset to be used in this project is provided by Udacity, which contains images of humans as well as dogs. The data will prove useful for the given problem as it contains plenty of pre-sorted images of both humans and dogs.

The dataset contains 13233 images of humans and 8351 images of dogs in separate folders. The dog images are divided into folders of train, test, and validation data. Each of the train, test, and validation folders contain images for 133 breeds of dogs, separated into folders with the breed name, and for each breed, we have a varying number of images. There are 6680 dog images in the training set, 836 images in the test set, and 835 images in the validation set, which means 80% of the total data has been kept for training and 10% each for validation and testing.

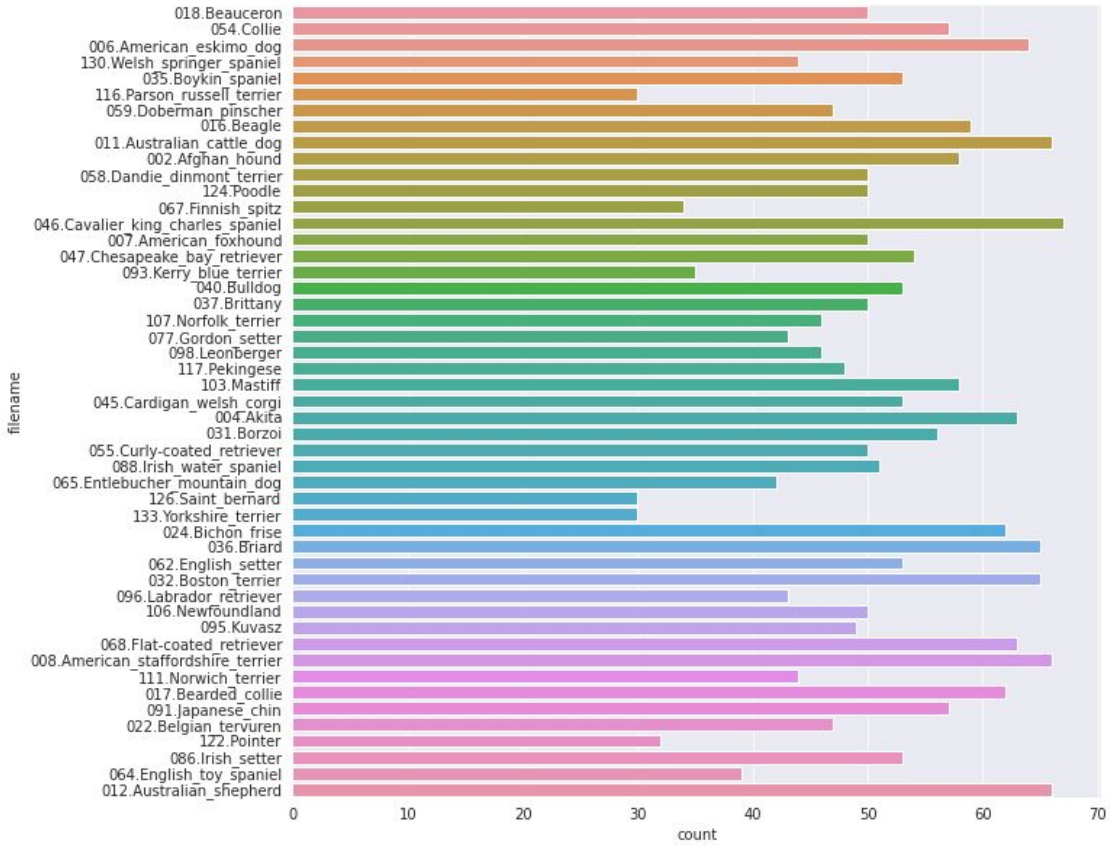

fig 1. a plot showing the number of images for 50 breeds in the training set

Figure 1 shows the number of images for 50 breeds of dogs present in the training dataset. From the figure we can infer that the dataset is imbalance, i.e., it does not have a proportionate amount of data for each of the classes. The number of images in each class in the training set ranges from 26 images to 77 images, with an average of 50 images. Such an imbalance can also be seen in the testing and validation sets.

The human image data is yet another highly imbalanced dataset. The images have been categorized by folders, where each folder contains images of a real person. 4069 people have only one image. 779 people have 2 images. The number of people keeps decreasing as the number of increases. 5 people have more than 100 images. We have the highest number of images for George W Bush, which is 530.

SOLUTION STATEMENT

The problem we are trying to solve is taking an input image and classifying the breed of the dog which is in the picture. Also, if the picture is of a human, we want to predict a closest matching dog breed to the human. Thus, to be able to produce an appropriate output in our app, we first need to classify whether our input image is a dog or a human. To classify whether the image is of a human, a fairly reliable approach would be to use Haar Cascades[2], which comes with OpenCV. To classify if the image is of a dog, we note that the ImageNet dataset has a number of labeled dog images. Thus, it would be a good approach to directly use one of the pre-trained models from the ImageNet competition[3] to find if it is a dog. Lastly, to predict the exact breed of the dog, we may again use one of the models from the ImageNet competition. This time, we will only use the pre-trained weights from the convolutional layer, and retrain the fully connected layers for our use case of 133 output dog-breed classes.

BENCHMARK MODEL

We choose our benchmark model to have the following properties:

- The Convolutional Neural Network that we make from scratch should have at least an accuracy of 10%. This is chosen keeping in mind that the task of differentiating between dog breeds is difficult even for humans themselves, and thus achieving a high accuracy using bare-CNN is not a trivial task.
- The pre-trained model should have an accuracy of at least 60% so that it can be successfully used in a dog breed classifier app.

EVALUATION METRIC

In our problem of classifying the correct dog breed, the percentage of accuracy would be an appropriate metric to judge how well our model is performing, since it provides a solid measure of how well our model is performing, bound between 0 and 100. The accuracy can simply be calculated using the following expression:

$$accuracy\ \% = \frac{number\ of\ correctly\ predicted\ images}{total\ number\ of\ images} \times 100$$

PROJECT DESIGN

This section provides a theoretical workflow for approaching a solution given the problem. These have been laid down in points below for a clearer understanding:

1. Importing Datasets

   In this section, the datasets, as provided by Udacity will be obtained and analyzed for our use case. Once confirmed that there is enough data for our

problem, the structure, and format of the data, as well as the structure in which it is sorted will be observed.

2. Detecting Humans

    Once the data has been obtained successfully, it can be used to detect human faces. As said earlier, we'll use OpenCV's implementation of Haar feature-based cascade classifiers. This involves converting the image to grayscale and then using the pre-trained face detector to extract the number of faces found. If this number turns out to be greater than zero, we can assume that our image has human faces in it.

3. Detecting Dogs

    As mentioned in the section "Solution Statement", we'll use one of the winning pre-trained models from the ImageNet competition, specifically, the VGG-16[6] Model. The model has been trained on a very large dataset, with 1000 categories. If we look at the indices of the categories, we'll notice that an uninterrupted sequence from 151 to 268 is all dog classes. Thus, checking if the prediction falls in one of these classes should provide us with the answer to whether the image has a dog in it.

4. Classifying the dog breed

    Once we successfully detect if our image has a dog in it, we can proceed with the task of building a classifier. This is not a trivial job, as even humans would have trouble in differentiating between some breeds, which look utterly similar. Thus, we approach this problem by using pre-trained weights from a ResNet-152[7] model. This has been discussed below:

    a. We will start by preprocessing our image data. This includes flipping the images, rotating them, resizing them, and cropping them. Doing these operations not only provides us with a greater variety of data but also prevents our model from overfitting. For example, an upside-down dog is still a dog, and hence we don't want our model to learn that dogs are always standing up.

b. After the preprocessing step, we will prepare our model architecture to be able to train it using these images. The ResNet-152 model was trained on a huge dataset, and hence its convolutional layers would already have learned useful information. We use the weights learned by these layers to our use, and then train only the fully connected layers to construct our dog breed classifier.
c. In training the model, we will use Cross Entropy Loss as our loss function. The model will be trained on a suitable number of epochs until the loss stops decreasing any further.

5. Lastly, these pieces will be integrated to build a working demo of our project. Separate messages will be printed for human faces, dog faces, and also images that contain neither of these.

References

[1] Szeliski, R. 2020. Computer Vision: Algorithms and Applications, 2nd ed. Springer. 1206 pp. [http://szeliski.org/Book/]. Accessed September 11, 2020.

[2] OpenCV. Face Detection using Haar Cascades. [https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html]. Accessed September 11 2020.

[3] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015. *arXiv:1409.0575, 2015.*

[4] Green, C. October 20th, 2016. Cross Entropy. [https://heliosphan.org/cross-entropy.html]. Accessed September 11, 2020.

[5] Torch Contributors. 2019. PyTorch 1.6.0 documentation.

[https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html]. Accessed September 11, 2020.

[6] Simonyan K, Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*.

[7] He K, Zhang X, Ren S, Sun, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385, 2015*.