Dog Breed Classifier Project Proposal

Capstone project proposal
Machine Learning Engineer Nanodegree @ Udacity
Suzanne ten Hage
March 2022

Background (½ page)

Humans have been keeping dogs as pets for a long time, and they provide a lot of happiness and companionship. Furthermore, pets are a lovely target for taking pictures, and many pictures of pets, and more specifically dogs, exist on the internet. There are many different dog breeds and it has become difficult to recognize them all instantly based on their looks. The American Kennel Club recognizes 197 dog breeds globally, of which the most recent addition is the Biewer Terrier in 2021 [1]. Our dataset contains 133 of these dog breeds. In order to be able to tell the breed of any dog, in this project a new application will be developed. The users of the application will be able to take pictures of their dogs, their friend's dogs, or dogs they meet on the street and use this application to classify which breed of dog it is. To make the application even more fun to use, users would be able to know which breed they resemble most closely.

Problem Statement

The aim of this project is to create an application that takes dog images as input and outputs the predicted dog breed. In the application, it is also possible to input a human face, after which the application returns the dog breed that has the closest resemblance with the human face.

Datasets and inputs (¼ page)

Both dog and human images are provided by Udacity and can be downloaded in the dog_app notebook. The zip files are not provided in this submission, as asked in the project description. There are 13233 images of humans and 8351 images of dogs available for this project. The dog images are split into a train, test, and validation set of 6680, 836, and 835 images, respectively. The human images are not split into separate sets, but they also don’t have to be since the algorithm won’t be trained on the human images at all.

Solution statement (3 sentences)

The application will run on a convolutional neural network that is able to take an image as input and output a dog breed classification. The model is trained using transfer learning, with weights from the vvg16 model [2]. Only the final linear layer will be adapted to represent the 133 dog breeds.

Benchmark Model (¼ page)

As a benchmark model, we use a small convolutional neural network built from scratch, using PyTorch. The model will contain three convolutional layers, each followed by ReLu as nonlinearity and a pooling step. This will be followed by two linear layers. Even though the model is small, we will use dropout to prevent overfitting the training set. For both the benchmark and actual model, we will use cross-entropy loss and momentum optimization.

Evaluation Metrics (¼ page)

Two different evaluation metrics are used, one for the detection of humans and dogs, and the other to test the accuracy of the convolutional neural network. For the detection of humans and dogs, we have labeling of the images available, and we can use the human faces as negative examples for dogs and vice versa. The performance is evaluated by calculating the percentage of false positives and false negatives. For the evaluation of our convolutional neural network, we use the test accuracy to determine performance. The test accuracy of the dog breed classification is calculated as follows; 100 * correct / total.

Outline of project design (½ page)

The project will contain the following steps:

1) Dog detection: For this task, we will use the VGG16 model to identify whether or not an image contains a dog.
2) Human face detection: For this task, we will use the Haar feature-based cascade classifier by OpenCV [3].
3) Build the benchmark model: This will be the small convolutional neural network built and trained from scratch.
4) Build a more accurate model: For this, we will use transfer learning to adapt the VGG16 model to classify dog breeds.
5) Build the app infrastructure.

References

[1] https://www.akc.org/press-center/articles-resources/facts-and-stats/breeds-year-recognized/ (01/31/2022)

[2] https://pytorch.org/vision/stable/models.html (01/03/2022)

[3] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (11/03/2020