Machine Learning Engineer Nanodegree

Proposal for Capstone Project: Dog Breed Classifier with CNN

Zhiguang Xue

October 1, 2021

Domain Background

Given an image of a dog, a dog breed classifier can give an estimate of the dog breed. It can be helpful in the scenario that a dog owner is unsure about the breed of his dog, and he wants to confirm it with someone or a classification web app by simply showing the picture. The work for building the dog breed classifier belongs to image classification, which is a research topic in computer vision.

Computer vision is one of the main technologies that enables the digital world to interact with the physical world, and it enables self-driving cars to make sense of their surroundings. Most computer vision algorithms use a convolution neural network, or CNN. A CNN is a model used in machine learning to extract features, like texture and edges, from spatial data, which is different from a normal feedforward neural network, where the input image would be flattened into a feature vector. A deep CNN model can achieve good prediction accuracy for image classification given a huge amount of training data, such as the ImageNet database.

Problem Statement

In this project, we will learn how to build a pipeline to process real-world, user-supplied images. Given an image of a dog, the designed algorithm/app will identify an estimate of the canine's breed. The prediction accuracy should not be too low, and we need to attain at least 60% accuracy. If supplied an image of a human, the algorithm will identify the resembling dog breed. If the image contains neither a dog nor a human, the algorithm should provide output indicating this.

Datasets and Inputs

The datasets are provided by Udacity, containing 8351 dog images and 13233 human images. Since I will be using Udacity workspace for this project, and the datasets are already stored in the /data folder, I do not need to re-download the datasets from amazon S3 bucket [1]. The human dataset can be used for detecting a human face and for testing the performance of the dog breed classifier. The dog dataset will be used for training the classifier. The dog images are under /data/dog_images with 3 categories:

- 6680 images under /data/dog_images/train for training
- 835 images under /data/dog_images/valid for validation
- 836 images under /data/dog_images/test for testing

The entire dataset contains 133 breed classes from Affenpinscher to Yorkshire Terrier. Setting aside the fact that the classes are slightly imbalanced, a random guess of the dog breed will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%. The task of building an accurate dog breed classifier is considered very challenging.

Solution Statement

The solution is to design a CNN model that will be able to give an estimate of the breed of the dog in a provided image. However, considering that our dog dataset is not very large, we could not train a deep CNN model from scratch. What we can do is to use transfer learning to load a pre-trained classic model like VGG-16 or ResNet-50, and then only update the last fully connected layer with the limited dataset. Taking the ResNet-50 as an example, we can replace the last fully connected layer (2048, 1000) with a new one (2048, 133), set random initial weights, and only train this layer. In this way, we can improve the performance of our dog breed classifier. Also, we need to create a human face detector and a dog detector, so that we can tell if the image contains a dog or a human before applying the dog breed classifier.

Benchmark Model

In this project, I will build a CNN model from scratch as the benchmark model, because I want to know how much the transfer learning can improve the prediction accuracy. Considering the small amount of the training dataset, the architecture of the CNN model from scratch will be relatively simple, consisting of 4 convolution layers and 2 fully connected layers. I expect that the test accuracy of this benchmark model is at least 10%.

Evaluation Metrics

I will use accuracy as the evaluation metric. We can get the accuracy of the models both from scratch and from transfer learning by counting the number of correct predictions for the 836 dog images in the test dataset.

Project Design

As described in the solution statement, I will design a human face detector and a dog detector to tell if an image contains a human or a dog. Then I will use transfer learning to design a CNN

model, which will be compared with the benchmark model that is built from scratch. Following are my design steps:

1. **Design a human face detector**
   The human face detector will be designed using OpenCV's implementation of Haar feature-based cascade classifier.

2. **Design a dog detector**
   The dog detector will be designed using pre-trained VGG-16 model, which was trained on ImageNet, a very large and popular dataset used for image classification and other vision tasks. If the predicted index of VGG-16 model is between 151 and 268, a dog is detected.

3. **Create a CNN to classify dog breeds from scratch**
   A relatively simple CNN architecture will be designed: 4 convolution + pooling layers, and 2 fully connected layers.

4. **Create a CNN to classify dog breeds using transfer learning**
   Pre-trained ResNet-50 model will be loaded for transfer learning for its good top-1 accuracy (76.130) and top-5 accuracy (92.862) [2]. Its last fully connected layer will be replaced with a new one having output dimension 133.

5. **Put everything together for the dog breed classifier**
   Use the dog detector and the human face detector to tell if the image contains a dog or a human. If yes, then we will use the CNN model to get a prediction. The algorithm will perform as expected.

6. **Test my algorithm**
   Test the algorithm with multiple images to see if the output is expected.

References

[1] Udacity provides the datasets. *The dog dataset*. URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip *The human dataset.* URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

[2] ResNet-50 model accuracy scores were checked on PyTorch website. URL: https://pytorch.org/vision/stable/models.html