CNN Project: Dog Breed Classifier

Lingchen Zhu

July 23, 2020

1 Domain Background

The goal of this project is to learn how to build a pipeline to process real-world, user-supplied images. Particularly, This project is to classify the breed of dog using Convolutional Neural Network (CNN). Given an image of a dog, your algorithm will identify an estimate of the canine's breed. If supplied an image of a human, the code will identify the resembling dog breed. In order to train a CNN that can successfully classify dog breed, sophisticated CNN model architectures may be used, and transfer learning from some well-known CNN model architectures pre-trained on natural image dataset may also be applied.

2 Problem Statement

.

Describing the content of an image is a typical computer vision problem. In this project, we will build a two level image classifier. The first level, which is a rough classification object, is to tell whether an input image describes a human face or a dog. The second level, which is more detailed, is to tell which exactly the dog breed is if the image has a dog, or to tell which dog breed the human face mostly resembles if there is one in the image.

OpenCV provides human face detector but it does not provide any "dog detector". That's the reason we will need to build one with CNN in this project. In this project, we will first use OpenCV to detect human face using Haar cascades. Then, in order to classify dog breed, we will define our own CNN model architecture from scratch and use a well-known CNN model architecture, e.g., VGG16, ResNet50, etc., pre-trained on the ImageNet dataset and update its top (head) layers for transfer learning.

3 Datasets and Inputs

The dog dataset is provided by Udacity and can be downloaded from here. The human face dataset is also provided by Udacity and can be downloaded from here. Since we will train the CNN model on the dog dataset, we will focus on more it. After unzipping the dog dataset, we can see that the images are well organized in three subfolders: train, valid, and test. Under each of these three sub-folders, we see 133 sub-sub-folders and each one is named by a breed class index along with the its class name. Fig. 1 shows an example of the breed sub-sub-folders in the train sub-folder of the dog dataset.

We will create custom PyTorch Datasets for these folders as the input of our CNN model.

4 Solution Statement

We will go through the Jupyter Notebook of this project step and step.

- Step 0: Import Datasets
- Step 1: Detect Humans
- Step 2: Detect Dogs
- Step 3: Create a CNN to Classify Dog Breeds (from Scratch)
- Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)
- Step 5: Write your Algorithm
- Step 6: Test Your Algorithm

I will finish the coding assignments in the "TODO" parts of the notebook, and test the results to see if they satisfy my expectations. Also, I will answer all questions provided in the notebook.

5 Benchmark Model

OpenCV provides a human face detector but it cannot find the dog. Some well-known pre-trained CNN models such as VGG16, ResNet50 can be used to detect many different kinds of dogs, as defined in the ImageNet dataset upon which it has been trained, but it cannot exhaustively detect all possible 133 dog breeds in the dog dataset.

We will then define our own CNN model architecture from scratch and use a pre-trained CNN model architecture as a base model and update its top (head) layers, known as transfer learning, to accurately classify all 133 dog breeds.

6 Evaluation Metrics

Similar to other classification problems, the CrossEntropyLoss function provided by PyTorch will be used to evaluate the train, validation and test losses of the CNN model. We can also use a plain accuracy metric, i.e., the total number of occurrences when the predicted dog breed class label is equivalent to the reference dog breed class label divided by the total number of images, to evaluate the performance of the trained CNN model.

7 Project Design

The project will be carried out as described in Section 4. Please refer to this finished Jupyter Notebook for full detail.

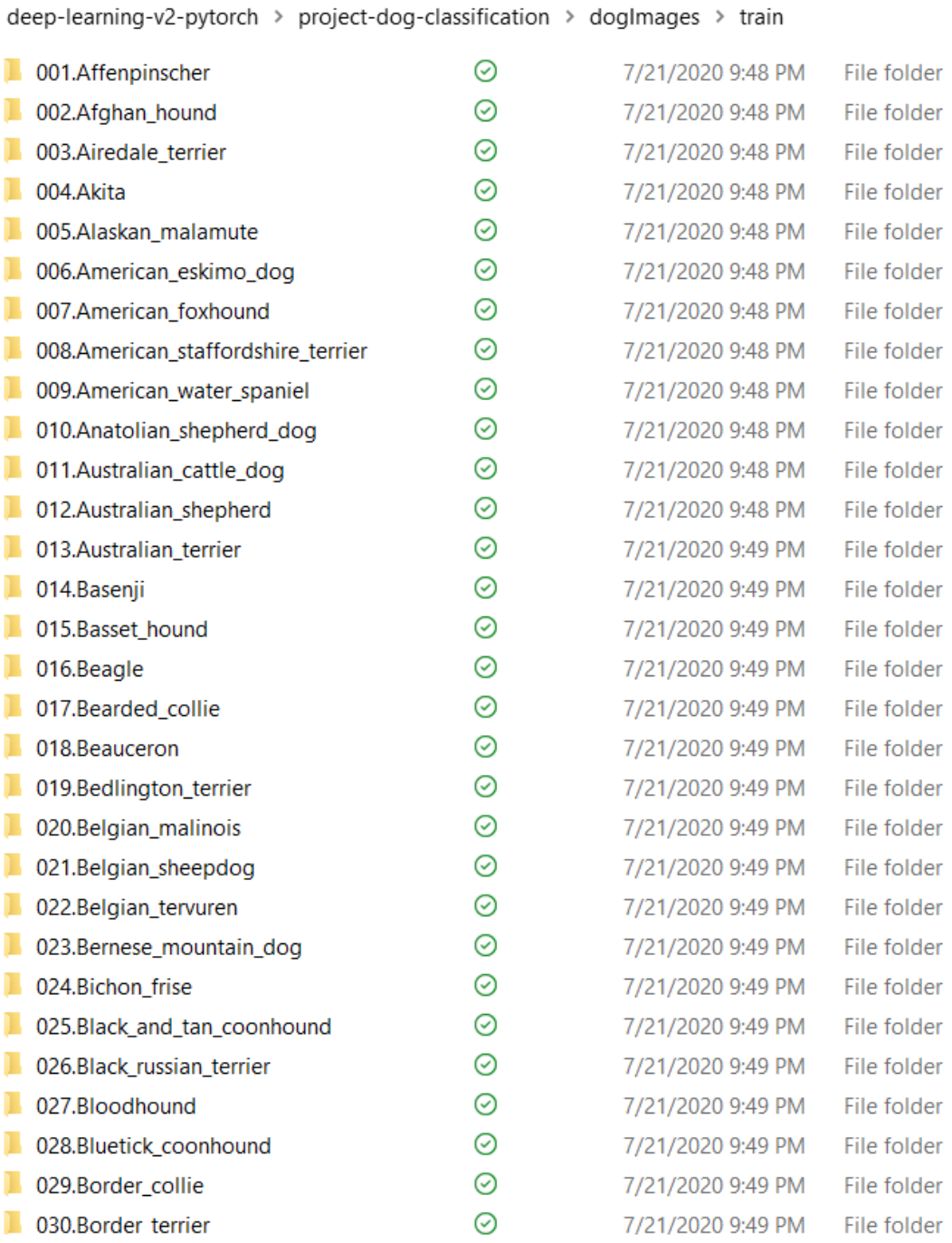

Figure 1: Folders of training images