Machine Learning Engineer Nanodegree

Proposal for Capstone Project: Dog Breed Classifier with CNN

Nishanth Kumar Gunda
October 6, 2023

Domain Background

Dog breed classifiers can estimate the breed of a dog from an image. This can be helpful for dog owners who are unsure of their dog's breed and want to confirm it with someone or a web app by simply showing a picture. Dog breed classifiers are built using image classification techniques, which is a research area in computer vision.

Computer vision is a core technology that enables the digital world to interact with the physical world. It powers self-driving cars to understand their surroundings and allows us to unlock our phones with our faces. Most computer vision algorithms use a type of neural network called a convolutional neural network (CNN). CNNs are different from regular neural networks in that they can extract features, such as texture and edges, from spatial data (images). This makes them well-suited for image classification tasks. Deep CNN models can achieve high accuracy on image classification tasks, such as identifying objects in images, given a large amount of training data, such as the ImageNet dataset.

Problem Statement

In this project, we will build a pipeline to process real-world images and identify the breed of a dog in the image, if there is one. The algorithm should achieve an accuracy of at least 60%. If the image contains a human, the algorithm should identify a resembling dog breed. If the image contains neither a dog nor a human, the algorithm should indicate this.

Datasets and Inputs

Udacity provides two datasets for this project: 8,351 dog images and 13,233 human images. Since the datasets are already stored in the **`/data`** folder of the Udacity workspace, I do not need to re-download them.

The human dataset can be used for detecting human faces and for testing the performance of the dog breed classifier. The dog dataset will be used for training the classifier. The dog images are organized into three categories under the **`/data/dog_images`** directory

- 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 dataset contains 133 dog breeds, from Affenpinscher to Yorkshire Terrier. Even if the classes were perfectly balanced, a random guess would only be correct about 1 in 133 times, which corresponds to an accuracy of less than 1%. This makes the task of building an accurate dog breed classifier very challenging.

Solution Statement

To classify dog breeds in images, we will design a convolutional neural network (CNN) model. However, our dog dataset is too small to train a deep CNN model from scratch. Therefore, we will use transfer learning to load a pre-trained CNN model, such as VGG-16 or ResNet-50, and only update the last fully connected layer, which is tailored to our specific task.

For example, we can replace the last fully connected layer of ResNet-50 (2048, 1000) with a new one (2048, 133), set random initial weights, and train only this layer. This will improve the performance of our dog breed classifier.

Additionally, we need to create a human face detector and a dog detector to distinguish between dogs and humans in images before applying the dog breed classifier.

Benchmark Model

In this project, I will build a CNN model from scratch as a benchmark to compare the performance of transfer learning. The benchmark model will be relatively simple, with 4 convolutional layers and 2 fully connected layers, due to the small size of the training dataset. I expect the test accuracy of this benchmark model to be at least 10%.

Evaluation Metrics

I will use accuracy as the evaluation metric for both the scratch-built and transfer learning models. To calculate the accuracy, I will count the number of correct predictions for the 836 dog images in the test dataset.

Project Design

To distinguish between humans and dogs in images, I will design a human face detector and a dog detector. Then, I will use transfer learning to build a CNN model, which I will compare to a benchmark model built from scratch. Here are my design steps:

1. **Design a human face detector**
   I will use OpenCV's implementation of the Haar feature-based cascade classifier to design the human face detector.

2. **Design a dog detector**
   I will use the pre-trained VGG-16 model to design the dog detector. The VGG-16 model was trained on ImageNet, a very large and popular dataset used for image classification and other vision tasks. If the output of the VGG-16 model predicts a class index between 151 and 268, then a dog is detected.

3. **Create a CNN to classify dog breeds from scratch**
   I will design a relatively simple CNN architecture with 4 convolutional and pooling layers, followed by 2 fully connected layers.

4. **Create a CNN to classify dog breeds using transfer learning**
   I will use transfer learning to load the pre-trained ResNet-50 model, which has a good top-1 accuracy (76.130) and top-5 accuracy (92.862) on the ImageNet dataset [2]. I will replace the last fully connected layer of the ResNet-50 model with a new one with 133 output nodes, which corresponds to the number of dog breeds in our dataset.

5. **Put everything together for the dog breed classifier**
   Use the dog detector and the human face detector to determine whether the image contains a dog or a human. If it contains a dog, use the CNN model to predict the breed. This 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-west1.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