Dog Breed Classifier using CNN Domain

Background

The Dog breed classifier is an interesting problem in ML. The problematic is to identify a breed when a dog image is given in input. If supplied an image of a human, the algorithm will identify the r=closest resembling dog breed. The concept is based on a pipeline that can process real world user supplied images and identify an estimate of the canine's breed. This is a multi-class classification problem where supervised machine learning is used. After completing this model, I am planning to build machine learning models that use CNN to automatically detect geological faults on subsurface image datasets. As a geoscientist, these concepts will allow me to bring innovative approaches to my daily professional workflows.

Problem Statement

The goal of the project is to build a machine learning model that can be used within web app to process real-world, user-supplied images. The algorithm must perform two tasks:

- *Dog face detector*: Define an algorithm that defines a dog's breed based on a given picture of a dog.

- *Human face detector*: If supplied an image of a human, the code will associate the most resembling dog breed.

Datasets and Inputs

For this project, the input format is an image type, because we want to input an image and identify the breed of the dog. The dataset for this project is provided by Udacity. The dataset has pictures of dogs and humans.

Dog images dataset: The downloaded dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. The data is not balanced because we have 1 image for some people and many images for some.

Solution Statement

To perform this multiclass classification, we can use Convolutional Neural Network to solve the problem. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. Before using any of the face detectors, it is standard procedure to convert the images to grayscale. The `detectMultiScale` function executes the classifier stored in `face_cascade` and takes the grayscale image as a parameter. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV's implementation of Haar feature based cascade classifiers. Second, to detect dog-images we will use a pretrained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN which will process the image and predict the breed that matches the best out of 133 breeds.

Benchmark Model

✓ The CNN model created from scratch must have accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

✓ The CNN model created using transfer learning must have accuracy of 60% and above.

Evaluation Metrics

For this multi class classification, Multi class log loss will be used to evaluate the model. Because of the imbalance in the dataset, accuracy is a not a good indicator here to measure the performance. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

Project Design

Step 1: Import the necessary dataset and libraries, Pre-process the data and create train, test and validation dataset. Perform Image augmentation on training data.

Step 2: Detect human faces using OpenCV's implementation of Haar feature based cascade classifiers.
Step 3: Create dog detector using pretrained VGG16 model.

Step 4: Create a CNN to classify dog breeds from scratch, train, validate and test the model.

Step 5: Create a CNN to Classify Dog Breeds using Transfer Learning with resnet101 architecture. Train and test the model.

Step 6: Write an algorithm to combine Dog detector and human detector.

✓ If dog is detected in the image, return the predicted breed.

✓ If human is detected in the image, return the resembling dog breed.

✓ If neither is detected, provide output that indicates the error.

References

1. Original repo for Project - GitHub: https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/dog_app.ipynb

2. OpenCV: Haar feature-based cascade classifiers

3. Resnet101: https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101

4. Imagenet training in Pytorch: https://github.com/pytorch/examples/tree/master/imagenet

5. Pytorch Documentation: https://pytorch.org/docs/master/