Dog breed classifier

Domain Background

This project is an exercise to build a deep learning model in order to classify dog breeds with Pytorch. Neural network models have regained tremendous popularity recently thanks to the development of computational power.

In this project, I will try to

1. Get familiar with the syntax of Pytorch, especially when doing preprocessing work.
2. Build a Neural Network model that works.
3. Apply transfer learning by using pretrained neural network architectures and use it on my own project.

problem statement

Use 8000+ dog images of 133 different breeds to train a classifier that can predict the breed of a dog, given a new image. Also, if a non-dog image is classified, it can also show that the image contains human faces or the image is neither a human nor a dog.

datasets and inputs

The datasets and the starting code are provided by Udacity.

solution statement

1. Build a Convolutional Neural Network from scratch and try to achieve test accuracy of 10%
2. Apply transfer learning, using pre-trained VGG16 model. Try to achieve test accuracy of at least 60%.

benchmark model

The model from scratch as described in the ‘solution statement’ section.

evaluation metrics

- The accuracy on the unseen test dataset
- In model training, I am planning to use cross entropy as the loss, which is the standard practice for multi-class classification

project design

1. Import the images and show the images in Jupyter Notebook
2. Apply Haar feature-based cascade classifier as described in https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html to serve as an attempt to detect non-dog images.
3. Write a dog detector from scratch in Pytorch as an attempt to get familiar with Pytorch
4. Evaluate the performance of the previous detector
5. Use VGG16 as the pretrained model of transfer learning, freeze most of the layer weights and add a fully connected layer to predict the dog breed.
6. Evaluate the performance of the previous detector. Minimum goal is 60% accuracy in test dataset.
7. Apply the algorithm on my own images.