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. Same well-known projects in the field of image classification include the MNIST project which tries to recognize handwritten digits in greyscale images, Fashion MNIST data which is used to classify types of apparels, ImageNet and CIFAR-10 which attempts to classify objects with around 1,000 types. Previous models developed by researchers and practitioners include different types of Convolutional Neural Network architectures such as ResNet, AlexNet and VGG16. In this project, I am actually going to apply transfer learning and take advantage of the pretrained VGG16 model to build my own classifier, for the specific purpose of classifying dog breeds.

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 8,351 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. The dataset consisted of 8,351 images of dogs from 133 different breeds. The images are of the RGB color model, meaning that each pixel has 3 channels. If an image is of 256 pixels by 256 pixels, it will be a 3-dimensional matrix of the size 256x256x3. Because of the large dimension of the data (each image can be regarded as a vector of 256x256*3 length, which is around 200,000), therefore Convolutional Neural Network will be the model of choice because of its ability to reduce the dimensionality drastically but at the same time learn both low level and high level features from the image. The images have been put in separate folders for the purpose of training, validation and testing. All I need to do for this project is to create 3 dataloaders for the images in Pytorch, so that I can send in the images by batch to the deep learning model.

Previous notable projects include ImageNet, CIFAR-10 and CIFAR-100, which are all image classification tasks. These image classification problems can help us automating the imaging tagging and classification tasks (which hopefully will no longer require manual labelling of the images), which has applications in autonomous driving, diagnosis with medical images etc.

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.