Dog Identification App

Project Overview/Domain Background

The task of assigning breed to dogs from images is considered exceptionally challenging. E*ven* human would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel. In this project, I created a dog identification application capable of classifying more than 100 different kinds of dog breeds.

Problem Statement

To build an app that takes user-supplied images as input and provides an estimate of the canine's breed of dog is detected in the image. If human is detected in the image, it will provide an estimate of the dog breed that is most resembling.

Datasets and Inputs

Dog dataset and human datasets are taken from the Udacity workspace.

Data Preprocessing

Train data:

1. Random resized crop of 224x244 is chosen.
2. Horizontal flip is done.
3. Images are normalized with mean and standard deviation values used for pretrained models as mentioned in PyTorch documentation.

Valid & Test data:

1. Images are resized to 256x256.
2. Center crop of 224x224 is taken.
3. Images are normalized with mean and standard deviation values used for pretrained models as mentioned in PyTorch documentation.

Solution Statement

1. CNN have outperformed all classic computer vision techniques and have become one of the widely used methods to solve computer vision problems especially classification, object detection and segmentation tasks.

2. Created a custom CNN with 4 conv layers and 2 fully connected layers to classify dog breeds.
    a. Obtained test accuracy of 17%.
3. Used transfer learning technique and trained a resnet50 model.
    a. Since resnet50 is trained on ImageNet dataset that has many different dog images, we can be confident that the parameters learned for resnet on ImageNet dataset would generalize well for our dataset too.
    b. Obtained test accuracy of 83%.

Benchmark Model

Pretrained Vgg16(model that has been trained on Imagenet, a very large and popular dataset used for image classification) Is used as benchmark model.

Evaluation Metrics

Accuracy is one of the common metrics for classifiers. The model is evaluated based on accuracy as described below.

Accuracy = (TN + TP)/(TN+TP+FN+FP) = (Number of correct assessments)/Number of all assessments)

Where

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

Project Design

The typical workflow starts with analysis of dataset (like visualizing few images in the dataset etc). Then we get start with getting some metrics using some benchmark models (like pretrained models) to get a rough idea of how well the existing models can map to our problem. If the existing models performs well enough (meets the metrics that are required for our application) we can use the existing models instead of designing models from scratch. But say for example, the model is big and you think it might be an overly complex architecture for our problem or the accuracy is not good enough, we can start with few small custom architectures and proceed from there.