Capstone Project Proposal

Domain Background

My capstone project will be concentrated in the field of **computer vision and image recognition**. I've decided to focus on this because it's a primary subfield of artificial intelligence, one in which approaches involving Convolutional Neural Networks (CNNs) have achieved noteworthy performance gains over the past decade in particular. One of the main venues in which these gains have been recognized is the Imagenet Large Scale Visual Recognition Challenge, which since 2010 has consistently raised the standards for algorithmic approaches to object detection and image classification across a wide array of categories.[1] Deep CNNs, such as the VGG architectures from Oxford University[2] and Inception[3] and GoogLeNet[4] from Google, have performed very well in Imagenet challenges, but they are known to take exceedingly long to train. Luckily for the machine learning community, many of these state-of-the-art deep CNNs are available within popular libraries such as PyTorch.[5]

Not only did we not cover CNNs in Udacity's coursework, we also always approached deep learning problems by constructing an architecture and training from the *ground up*. As part of this project, therefore, I would like to experiment with **transfer learning**[6]: freezing the convolution layers of a deep CNN and then retraining the final, fully connected layers for a more highly specialized classification problem that focuses on a subset of images. Specifically, I would like to understand how much of a gain in accuracy we can get using transfer learning as opposed to starting from scratch like we were shown in our coursework.

Problem Statement

Given an image of a dog, can a machine learning algorithm tell me what breed of dog the image contains?

[1] http://www.image-net.org/challenges/LSVRC/
[2] https://arxiv.org/abs/1409.1556
[3] https://arxiv.org/abs/1512.00567
[4] https://arxiv.org/abs/1409.4842
[5] https://pytorch.org/docs/stable/torchvision/models.html
[6] https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce

Datasets and Inputs

For this project I plan to use one of the recommended data sets from our Udacity coursework: the Stanford Dogs data set[7], a canine-focused subset of the ImageNet database[8]. Although this data is available through the Kaggle contest, I will actually download it from the website of the Stanford researchers[9] who originally curated this data set—mostly because the Stanford link includes bounding boxes for each image to crop people and backgrounds out of the photos. As per this source, the dataset comprises **12,000 training images** and **8,580 testing images** that represent **120 different types of dog breeds**.

Note: The original statement of this problem as per the Kaggle contest is all about classifying each image into one of 120 possible breeds. I may opt to modify this problem to see whether I can classify the dogs into one of seven breed groups, as specified by the American Kennel Club.[10]

Solution Statement

The solution for this problem will almost certainly involve some kind of Convolutional Neural Network, as these have been shown to perform quite well at large-scale image recognition tasks over the past decade or so. I'd ideally like to compare how a simple CNN built in PyTorch trained from scratch, perhaps using an architecture similar to the one in this blog post about classifying Pokémon[11], performs versus a version of VGG-16 and *either* Inception *or* GoogLeNet modified by transfer learning to perform a 120-class (*breed level*) and/or 7-class (*group level*) classification problem.

Benchmark model

The simplest benchmark I can think of is to implement a probabilistic random guess classifier, based solely on the frequency of each of the different breeds across both the train and test sets. So, for example, if Chihuahua images comprised 5% of the overall data set, I would simply guess Chihuahua for any input image 5% of the time. This should be relatively easy to implement using numpy's random choice functionality with weights.[12]

---

[7] https://www.kaggle.com/c/dog-breed-identification/data
[8] http://image-net.org/about-overview
[9] http://vision.stanford.edu/aditya86/ImageNetDogs/
[10] https://www.akc.org/public-education/resources/general-tips-information/dog-breeds-sorted-groups/
[11] https://jgeekstudies.org/2017/03/12/who-is-that-neural-network/
[12] https://het.as.utexas.edu/HET/Software/Numpy/reference/generated/numpy.random.choice.html

Evaluation Metrics

For multiclass classification problems with relatively balanced frequencies across the classes, **Top-1 accuracy** and **Top-5 accuracy** seem to be relatively common metrics for evaluating performance on an independent test set. Each CNN can produce a probability of an image belonging to each class, so I can simply look at the *most probable* and *top five probable* classes and ask whether or not those subsets contain the true label for each image.

Note: If I opt to do breed group classification as well as breed classification, I may look at **Top-3 accuracy** instead of Top-5 accuracy, since there are only seven possible breed groups.

Project Design

I plan on going about this project in three steps that will be laid out much like the Plagiarism Detection project earlier on the course.

- First, I will have a notebook for **data exploration and preprocessing.** Tasks that will be handled here include:
    - Downloading the data files to a local directory
    - Understanding the size and shape of the data (frequency of each class, etc.)
    - Experimenting with resolving individual dog breeds to American Kennel Club breed groups
    - Testing out how to apply bounding boxes to the images to focus on the dogs that need to be classified
    - Implementing a benchmark RandomGuess model.
- Then, I will have a notebook for a **ground up convolutional neural network** that I will train myself. The steps handled here will include
    - Uploading the data files to S3 in a manner that PyTorch can handle
    - Working out a proper preprocessing pipeline, one that perhaps utilizes image augmentation techniques
    - Defining a *model.py, train.py, and predict.py* script for a custom model
    - Training the model
    - Standing up an inference endpoint
    - Generating Top-1 and Top-5 accuracy for predictions on the dog breed set
    - Deleting the endpoint
- Finally, I will have a notebook focused on **transfer learning**. The steps handled here will be similar to those in the previous notebook:
    - Defining a *model.py, train.py, and predict.py* script for a pre-trained architecture
    - Retraining the final layers of this architecture for a specialized breed prediction task
    - Standing up an inference endpoint
    - Generating Top-1 and Top-5 accuracy for predictions on the dog breed set
    - Deleting the endpoint