CECS 551: Advanced AI (Spring 2019)

Homework #4
Due 05/06/2019, 3 PM

For this assignment, we will use Keras for image classification. This assignment is designed to be able to be completed using only CPU, but if you have a GPU, you can also install Keras with GPU capabilities (e.g. CUDA for the backend).

Assignment Overview

Your goal for this assignment is to experiment with fine-tuning an existing ImageNet-trained model.

Please answer all questions (indicated with **Q#**: in bold on this webpage) in your readme, organized by question number.

Keras Documentation

You already used Keras in HW3. Keras has quite helpful documentation. For this homework, you may want to look at the core layers (Dense: a dense fully-connected layer, ...), the convolutional layers (Conv2D, ...), the model class (which has methods such as evaluate, which evaluates the loss and summary metrics for the model, and predict, which generates test time predictions from the model), and the sequential class (which has some additional methods such as predict_classes, which generates class predictions), andloading and saving of models. You may also find this example of prediction with ResNet-50 to be useful (see the heading in the previous link titled "Classify ImageNet classes with ResNet50", which includes code to resize the image before running prediction).

Training Times

On my laptop it took about ten to twenty minutes to train each model for a few epochs. However, slower computers may run training more slowly. So you may want to plan ahead especially if your computer is slow to avoid running into some training time bottleneck.

I also highly recommend that while debugging your program, you make the program run more quickly by lowering the number of training samples/epochs / and/or saving your model after training (followed by re-loading it on future runs of the program). This will avoid you having to wait for a long time while training the model, only to discover that you have some minor typo later in your program (which triggers e.g. a run-time exception in Python).

Fine-tuned Cats and Dogs (100 points)

In this assignment, you will experiment with fine-tuning an existing ImageNet-trained CNN for a problem of dog vs cat classification. This task is fairly simple for the CNN, since it has already been trained on different varieties of dogs and cats, and the CNN just has to find out how to combine this existing knowledge.

1. Download and extract the cat and dog images. We are going to apply a very simple fine

tuning of the VGG-16 model, which has been pretrained on ImageNet. In particular, we will remove the fully-connected layers from VGG-16, and add a single sigmoid neuron (so the output if closer to 0 will be cat, and if closer to 1 will be considered dog), then freeze the main VGG-16 network, and modify the weights only for the sigmoid neuron.

2. (40 points). You can start with this fine-tuning code, which was modified slightly from generic fine-tuning code from the Keras project. **Note**: check your Keras version and if it is 2.0, then make the modifications noted on the last lines of the above file. The changes you will need are to point it to your train and test data, and to enter the right number of layers to freeze for NUM_LAYERS. **Hint**: You can print the layers (model.layers) and figure out which layers should be frozen (or you can use pprint.pprint(), the Python "pretty printer," which prints this more nicely).

   Q#1: For debugging purposes, you can lower the number of training and validation samples by a factor of say 20, and run for 1 epoch (**Note**: if you make the number of training samples too small, it triggers some internal error in Keras). What accuracy do you obtain, and why?

3. (40 points). Modify the code to also save the model, and make a separate Python "classification testing utility" that loads in the model and tests it on an input image of your choice. Try it out on a few cat and dog images from the test set (or images of your own choice).

4. (20 points). Fine-tune on the original number of training and validation samples.

   Q#2: If you fine-tune for 1 or 2 epochs using the original number of training and validation samples, what accuracy do you obtain, and why? Does your saved model file now work better with your testing utility?

Submission

Feel free to collaborate on solving the problem but write your code individually.

Submission

Submit your assignment in a zip file named yourname_HW4.zip onto Dropbox. Please include your source code, and your image "classification testing utility". Please also include a readme with answers to the questions above.