Assignment 2: Recognizing Objects in Images with Deep Learning Features

Arash Vahdat

Fall 2015

Readings You are highly recommended to read the following chapters while doing this assignment:

- Caffe Tutorials at `http://caffe.berkeleyvision.org/tutorial`
- Spark Basics: chapter 1-4, “Learning Spark: Lightning-Fast Big Data Analysis", Karu et al.
- Spark Partitioning: chapter 6, “Learning Spark: Lightning-Fast Big Data Analysis", Karu et al.
- Spark Linear SVM: `https://spark.apache.org/docs/1.3.1/mllib-linear-methods.html`

Provided Data In this assignment, we will use 11,540 images from the Pascal VOC 2012 dataset[1]. All images and their annotations are available at the following location on the SFU RCG network:

`/cs/vml2/avahdat/CMPT733_Data_Sets/Assignment2`

1 Introduction

The deep learning package Caffe[2] is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. It is one of the popular packages that is widely used for developing deep convolutional neural network models for image and video processing applications. Its expression, speed, and modularity make this package ideal for our task on this assignment.

Training a deep learning model using Caffe on advanced GPUs typically takes a very long time ranging from a day to several weeks depending on the model and the dataset size. Caffe can also run on CPU, however, the CPU mode will run with several orders of magnitude slower. Even though big data has contributed dramatically to most of the computer vision success stories, there is no “common” open source distributed framework for training deep models.

In this assignment, we will use deep learning features to identify objects from a given category of objects on a partition of the Pascal VOC 2012 dataset. Instead of training a model from scratch on our 11K dataset, we will use a pre-trained model that has been trained on millions of images. By extracting image features using this model, we attempt to “transfer knowledge” from a model that has seen millions of images to our object

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html
[2] http://caffe.berkeleyvision.org/

recognition task. We will distribute the Caffe computation on CPUs available on our Spark framework to speed up the feature extraction process. We will also use Spark's machine learning library to train classifiers to recognize categories of objects in our images.

We will work with a partition of the Pascal VOC 2012 dataset which includes: 5,717 images for training, 2,911 images for validation and 2912 images for test. The dataset contains images from the following 20 categories: 'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'dining-table', 'dog', 'horse', 'motor-bike', 'person', 'potted-plant', 'sheep', 'sofa', 'train', 'tv-monitor'. You can find the dataset at the aforementioned location on the RCG file system. Here are the list of files/directories at this location:

- **`images/:`** A directory containing all 11,540 images used in train/val/test. Each JPEG image is named with a unique ID.
- **`train.txt, val.txt, and test.txt:`** Text files containing the list of images used for train/val/test.
- **`train/, val/:`** Annotation directories for train and validation images. In each directory, you will find 20 text files corresponding to 20 object categories with the list of images IDs and their labels. Labels are +1 for positive, -1 for negative and 0 for others. You can ignore images with label 0 or 'others'. Test labels are not provided.
- **`Caffe_Models:`** A directory with the Caffe models used in Sec.2.

Please note that our goal in this assignment is to train **binary** classifiers to recognize each object category from a large collection of images. This is a typical application in content-based image retrieval tasks. Therefore, a separate classifier should be learned for each class and its performance will be measured using an average precision criterion.

2 Feature Extraction

Question 1: (10 marks) During lectures, we saw an example of neural network where an image is fed to a network and activations in different levels of the network are visualized. You can find this example at the following url `https://github.com/BVLC/caffe/blob/master/examples/00-classification.ipynb`. Write a simple Python script that replicates the tutorial. Similar to the tutorial, your script should accept an image and output the top 5 categories with highest probability. Answer the following questions:

1. Try different images as input to your network and visualize the output for these images. How is the performance of the network on the images you chose?
2. List the layers forming the network and briefly explain how each layer is changing its input.
3. List the parameters of the network. Briefly explain what each layer corresponds to. For example, a convolutional layer may perform a convolution of size 5x5. In that case it will have 5*5=25 parameters corresponding to its convolution filter. What is the total number of parameters in this network?

In order to use Caffe in a **Python**[3] script you will need to load the Caffe module that sets the appropriate environment variables on the RCG network:

[3] Caffe in **Spark** is another story!

```
$ ssh -X your_id@linux.cs.sfu.ca
$ cp /cs/vml2/avahdat/software/vml ~/privatemodules/
$ module load natlang
$ module load NL/LANG/PYTHON/Anaconda-2.3.0
$ module load /cs/vml2/avahdat/software/modules/VML/TOOLS/CAFFE/20150618_CPU
```

NB: For doing this tutorial you need to have access to the following auxiliary files: The deploy proto text file, trained Caffe model, and synet words list. They can be found under the `Caffe_Models` directory in the data directory provided with this assignment. For the Caffe root location, you can use `/cs/vml2/avahdat/software/caffe_cpu_only/`.

Question 2: (10 marks) The tutorial page used in the previous question extracts all the activations in the network. Every layer of a convolutional neural network performs a non-linear transformation of its input. The output of each layer can be considered as a complex feature that describes the content of an input image capturing simple visual cues at initial layers to more elaborate and complex representations at deeper layers. The fc7 layer is a 4096 dimensional feature vector forming a rich representation of the input image. In this question, we will extract the output of the layer fc7 and use it to represent images.

Based on the script you wrote in Q1, write a new Python script to extract fc7 activations from all images provided with this assignment. Your script should utilize the Spark cluster to parallelize the process. In this scenario, the feature extraction can be considered as a map operation that accepts a string as an image filename/path and outputs the corresponding fc7 feature vector.

While writing your code you may notice that loading a Caffe model (net) in each map transformation can be very slow. A better solution is to use a mapPartition() instead of a simple map() operation. The mapPartition() will run on each partition of your RDD and will allow you to load a Caffe model (net) only once for each data partition.

Using Caffe in Spark: In order to be able to use Caffe in a Spark script, you should make sure that you are setting the Caffe python library and its paths correctly. For doing so follow the next three steps:

First, run the following lines to get list of locations for `$PATH` and `$LD_LIBRARY_PATH` environment variables:

```
$ module load natlang
$ module load NL/LANG/PYTHON/Anaconda-2.3.0
$ module load /cs/vml2/avahdat/software/modules/VML/TOOLS/CAFFE/20150618_CPU
$ echo $PATH
$ echo $LD_LIBRARY_PATH
```

Second, you should assign the content of `$PATH` and `$LD_LIBRARY_PATH` to two separate strings in Python and pass them to SparkContext inside your python script:

```
conf = SparkConf()
conf.setExecutorEnv('PATH', PATH)
conf.setExecutorEnv('LD_LIBRARY_PATH', LD_LIBRARY_PATH)
sc = SparkContext(conf=conf)
```

Third, you need to use the following commands to set the python path before running Spark on the Hadoop server:

```
setenv PYSPARK_PYTHON /cs/vml2/avahdat/software/Anaconda-2.2.0/release/bin/python
setenv HADOOP_CONF_DIR /usr/hdp/current/hadoop-client/etc/hadoop/
spark-submit --master yarn-client --executor-memory 1G --num-executors 5 your_file.py
```

3 Image Classification

In this section, we will use the previously extracted features to creaet a binary classifier for object recognition on the provided dataset.

Question 3: (10 marks) Using the Python Scikit-learn package, train a linear SVM classifier for each object category listed in section 1. For training use images from the training dataset. Reserve the validation set images for setting the parameters of your classifier. Evaluate your classifier's performance using the Average Precision (AP) computed for each category and the mean Average Precision (mAP) over all categories.

Tune your parameters on the validation dataset and report your best performance in terms of AP per category and mAP over all categories. Note that this question will allow you to test your feature extraction code and make sure the features you have extracted in section 2 are correct. If your features are extracted correctly and the parameters of your SVM classifier are validated carefully, you should be able to get an mAP value around 70%.

Question 4: (10 marks) Spark's Machine Learning Library (MLlib) provides a wide range of scalable machine learning models including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives. The main difference between Spark MLlib and other Python machine learning libraries such as Scikit-learn is its ability of parallelism over a commodity cluster. With Spark MLlib, we are not restricted to datasets that can fit into one machine's memory. Both, memory and computation power of a cluster of computers can be used to speed up the training process of a machine learning model using Spark MLlib.

Write a Python script that uses your extracted features and trains a linear SVM model using Spark's SVMWithSGD class. For training a SVM, you will need to validate the regularization parameter and training step size (See the footnote[4]). Report the best regularizer parameter and training step size and their corresponding mAP values.

Question 5: (5 marks) Assume instead of 10K images, you were provided with 10 million images. Could you still use use your Spark scripts that were developed for the feature extraction and classification parts. For example for 10 million images, all the feature vectors representing these images will take 10,000,000 * 4096 * 4 (for float32) = 163.84 GB of memory. Check your code and identify the parts that you are storing all the feature vectors together. Modify your code such that you do not hold all the data together.

[4] https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.classification.SVMWithSGD

Question 6: (10 marks) There are multiple ways to improve an image classification model's performance. The following is a list of potential approaches:

1. Data augmentation: there are multiple ways to augment your training dataset, for example, you could double the size of your training dataset by considering horizontally flipped (or mirrored) images.

2. Different features: in this assignment, we used 'fc7' as image features. You could also use other layers such as 'fc6' or 'fc8' or some concatenation of these features.

3. Different classifiers: different classifiers such as a kernelized SVM or a random forest classifier could potentially perform better.

4. Different CNN model: There are many advanced models that have been proposed for image classification. These models have some differences (sometimes subtle) that reflects in their classification performance. Many of the state-of-art image classification models are implemented within the caffe library. You could use one of these models for feature extraction: `https://github.com/BVLC/caffe/wiki/Model-Zoo`

Implement one or a combination of these approaches. Design experiments to validate these approaches on the validation dataset. Feel free to propose your own approaches to boost your classifier's performance.

4 Submission

Your assignment should be submitted online at `https://courses.cs.sfu.ca/`. You should submit two files for this assignment :

1. **Report:** Create your report in PDF format including your answers to all questions.

2. **Code:** Create an archive file containing all scripts you have designed for this assignment. You should create your scripts such that your results can be reproduced easily. Please include a small readme.txt file in your archive briefly explaining which script was used for each question.