Tutorial: Subjective Scene Tagging

Summary

This tutorial goes over how to setup caffe in a Linux environment, coalescing labeled training and testing data, setting up and training with finetuning, and testing the resulting convolutional neural network.

Caffe Setup

Setup in Ubuntu VM

1. Use the preset image in the following website, https://atlas.hashicorp.com/malthejorgensen/boxes/caffe-deeplearning
2. The Vagrantfile example is included in caffe-deeplearning

Setup in Amazon AMI

1. Refer to the following website, https://github.com/BVLC/caffe/wiki/Ubuntu-14.04-VirtualBox-VM
2. Ignore the Virtual Box setup instructions, start the caffe setup at the Install Curl (for the Cuda download) command.
3. Follow the commands ending at the Compile Caffe section.
    - Running the make commands may result in a few errors
    - If errors occur during make all command, refer to the following website, https://github.com/BVLC/caffe/issues/2348, specifically the StevenLoL comment.
    - Ignore the following warning, libdc1394 error: Failed to initialize libdc1394, it is not consequential. Training will still work. Found a temporary fix that stops working when the virtual machine is restarted.

Data Setup

1. Label the training images first. Running the webapp, hotnot.py in /labelData, will pull up random images with checkboxes for labels. Saves the tags into a database, which can be viewed or saved to files in other pages. Edit the templates to change the labels.
2. Download the labeled images onto the machine. Refer to compile_train_data.py in /sceneTagging/finetune_setup/.
3. Edit the files containing the image ids and tags to contain the path to the images. This is necessary for training. Refer to ami_update.py in /sceneTagging/finetune_setup/.
4. Randomize the training and testing labeled files to improve training results. Refer to randomize.py in in /sceneTagging/finetune_setup/.

Finetuning

1. The finetune_flickr_style is a great example. It is located in caffe/examples/finetune_flickr_style once caffe is setup. The readme.md contains a review of finetuning. Other models are in the Model Zoo, https://github.com/BVLC/caffe/wiki/Model-Zoo
2. Refer to /sceneTagging/finetune_results/ for examples
3. Create the network you want to define in the deploy.prototxt and train_val.prototxt files. http://caffe.berkeleyvision.org/ contains information regarding different layers. Edit the path names in both of these files depending on where you store your data. **Note**: In order to reuse the weights of any trained model the layers must be identical including the name. For finetuning, refer to any network you want to use and adjust the last layer to your needs.
4. Create the solver file for the network, solver.prototxt. Here you can specify gpu or cpu usage, learning rates, snapshot capture, training iterations, and the meanfile you want to use. The meanfile is the average of all of the images used for standardizing during training. I used the imagenet one but as the Carmera dataset grows, it would be ideal to create a more specific one.
5. Go to the caffe root directory.
6. Begin training with the following command (-weights refers to the caffemodel where you want to import starting weights from, exclude –gpu all if you are planning on using the cpu)
   ./build/tools/caffe train -solver path/to/solver.prototxt -weights models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel –gpu all
7. Snapshots will be taken regularly, at an interval specified in solver.prototxt. You can stop training anytime with Ctrl-C, which will also save a snapshot.
8. Resume training with the following command
   ./build/tools/caffe train -solver path/to/solver.prototxt -snapshot path/to/desired/snapshot/snapshots_iter_current.solverstate -gpu all

Testing Trained Network

1. Initialize the network, refer to /sceneTagging/finetune_results/train02/safe_val.py or testim.py for an example.
2. Download random images through Granola API
3. Feed the images forward through the network
4. Analyze the output vector, use the label_names.txt files to match the vector output with an actual predetermined tag that the network was trained on. Compare with any verification data.
5. The following is a great tutorial, https://github.com/BVLC/caffe/blob/master/examples/00-classification.ipynb