Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

ronghanghu / rcnn

forked from rbgirshick/rcnn

Watch 1 Star 0 Fork 392

Code Pull requests 0 Pulse Graphs

R-CNN: Regions with Convolutional Neural Network Features

46 commits 8 branches 2 releases 2 contributors

Branch: master New pull request New file Find file HTTPS https://github.com/rongha Download ZIP

This branch is 3 commits behind rbgirshick:master. Pull request Compare

rbgirshick add notes about these functions which are not useful to the general p… … Latest commit 22f1605 on 15 Jul 2014

bbox_regression	load cached results if they exist	2 years ago
bin	add missing bin directory	2 years ago
cachedir	make cachedir setup same with a local override file	2 years ago
data	update data download READMEs and main README	2 years ago
datasets	initial checkin	2 years ago
examples	add threshold to demo	2 years ago
experiments	show how to use RCNN_CONFIG_OVERRIDE	2 years ago
external	cleanup installation instructions after a walkthrough; remove externa…	2 years ago
feat_cache	add feat_cache/README.md	2 years ago
finetuning	upgrade prototxt to caffe proto v1	2 years ago
imdb	nms thresh as argument	2 years ago
model-defs	improve demo; update models to caffe's v1 proto messages	2 years ago
nms	* fix caffe net paths	2 years ago
selective_search	add notes about these functions which are not useful to the general p…	2 years ago
utils	add one-off function for testing with the built-in softmax	2 years ago
vis	fix incorrect receptive field size	2 years ago
.gitignore	update data download READMEs and main README	2 years ago
LICENSE	Update LICENSE	2 years ago
README.md	fix git clone error	2 years ago
rcnn_build.m	initial checkin	2 years ago
rcnn_cache_pool5_features.m	sprinkle functions with basic documentation	2 years ago
rcnn_config.m	sprinkle functions with basic documentation	2 years ago
rcnn_config_local.example.m	make cachedir setup same with a local override file	2 years ago
rcnn_create_model.m	add copyright notices in files	2 years ago
rcnn_extract_regions.m	fix bug reported by Shubham Tulsiani	2 years ago
rcnn_feature_stats.m	add copyright notices in files	2 years ago
rcnn_features.m	allow for features from new and old matcaffe by canonicalizing them into	2 years ago
rcnn_im_crop.m	sprinkle functions with basic documentation	2 years ago
rcnn_load_cached_pool5_features.m	sprinkle functions with basic documentation	2 years ago
rcnn_load_model.m	sprinkle functions with basic documentation	2 years ago
rcnn_pool5_to_fcX.m	sprinkle functions with basic documentation	2 years ago
rcnn_scale_features.m	sprinkle functions with basic documentation	2 years ago

rcnn_test.m	sprinkle functions with basic documentation	2 years ago
rcnn_train.m	r2013a compatibility fix	2 years ago
startup.m	improve demo; update models to caffe's v1 proto messages	2 years ago

README.md

R-CNN: *Regions with Convolutional Neural Network Features*

Created by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik at UC Berkeley EECS.

Acknowledgements: a huge thanks to Yangqing Jia for creating Caffe and the BVLC team, with a special shoutout to Evan Shelhamer, for maintaining Caffe and helping to merge the R-CNN fine-tuning code into Caffe.

Introduction

R-CNN is a state-of-the-art visual object detection system that combines bottom-up region proposals with rich features computed by a convolutional neural network. At the time of its release, R-CNN improved the previous best detection performance on PASCAL VOC 2012 by 30% relative, going from 40.9% to 53.3% mean average precision. Unlike the previous best results, R-CNN achieves this performance without using contextual rescoring or an ensemble of feature types.

R-CNN was initially described in an arXiv tech report and will appear in a forthcoming CVPR 2014 paper.

Citing R-CNN

If you find R-CNN useful in your research, please consider citing:

```
@inproceedings{girshick14CVPR,
    Author = {Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra},
    Title = {Rich feature hierarchies for accurate object detection and semantic segmentation},
    Booktitle = {Computer Vision and Pattern Recognition},
    Year = {2014}
}
```

License

PASCAL VOC detection results

Method	VOC 2007 mAP	VOC 2010 mAP	VOC 2012 mAP
R-CNN	54.2%	50.2%	49.6%
R-CNN bbox reg	58.5%	53.7%	53.3%

- VOC 2007 per-class results are available in our CVPR14 paper
- VOC 2010 per-class results are available on the VOC 2010 leaderboard
- VOC 2012 per-class results are available on the VOC 2012 leaderboard
- These models are available in the model package (see below)

ImageNet 200-class detection results

Method	ILSVRC2013 test mAP
R-CNN bbox reg	31.4%

- For more details see the updated R-CNN tech report (Sections 2.5 and 4, in particular)
- This model is available in the model package (see below)
- The code that was used for training is in the `ilsvrc` branch (still needs some cleanup before merging into `master`)

Installing R-CNN

1. **Prerequisites**
   i. MATLAB (tested with 2012b on 64-bit Linux)
   ii. Caffe's prerequisites
2. **Install Caffe** (this is the most complicated part)
   i. R-CNN has been checked for compatability against Caffe release v0.999 (kona-snow), however it *should* also work with the current Caffe master
   ii. Download Caffe v0.999
   iii. Follow the Caffe installation instructions
   iv. Let's call the place where you installed caffe `$CAFFE_ROOT` (you can run `export CAFFE_ROOT=$(pwd)`)
   v. **Important:** Make sure to compile the Caffe MATLAB wrapper, which is not built by default: `make matcaffe`
   vi. **Important:** Make sure to run `cd $CAFFE_ROOT/data/ilsvrc12 && ./get_ilsvrc_aux.sh` to download the ImageNet image mean
3. **Install R-CNN**
   i. Get the R-CNN source code by cloning the repository: `git clone https://github.com/rbgirshick/rcnn.git`
   ii. Now change into the R-CNN source code directory: `cd rcnn`
   iii. R-CNN expects to find Caffe in `external/caffe` , so create a symlink: `ln -sf $CAFFE_ROOT external/caffe`
   iv. Start MATLAB (make sure you're still in the `rcnn` directory): `matlab`
   v. You'll be prompted to download the Selective Search code, which we cannot redistribute. Afterwards, you should see the message `R-CNN startup done` followed by the MATLAB prompt `>>` .
   vi. Run the build script: `>> rcnn_build()` (builds liblinear and Selective Search). Don't worry if you see compiler warnings while building liblinear, this is normal on my system.
   vii. Check that Caffe and MATLAB wrapper are set up correctly (this code should run without error): `>> key = caffe('get_init_key');` (expected output is key = -2)
   viii. Download the model package, which includes precompute models (see below).

Common issues: You may need to set an `LD_LIBRARY_PATH` before you start MATLAB. If you see a message like "Invalid MEX-file '/path/to/rcnn/external/caffe/matlab/caffe/caffe.mexa64': libmkl_rt.so: cannot open shared object file: No such file or directory" then make sure that CUDA and MKL are in your `LD_LIBRARY_PATH` . On my system, I use:

```
export LD_LIBRARY_PATH=/opt/intel/mkl/lib/intel64:/usr/local/cuda/lib64
```

Downloading pre-computed models (the model package)

The quickest way to get started is to download pre-computed R-CNN detectors. Currently we have detectors trained on PASCAL VOC 2007 train+val, 2012 train, and ILSVRC13 train+val. Unfortunately the download is large (1.5GB), so brew some coffee or take a walk while waiting.

From the `rcnn` folder, run the model fetch script: `./data/fetch_models.sh` .

This will populate the `rcnn/data` folder with `caffe_nets` and `rcnn_models` . See `rcnn/data/README.md` for details.

Pre-computed selective search boxes can also be downloaded for VOC2007, VOC2012, and ILSVRC13. From the `rcnn` folder, run the selective search data fetch script: `./data/fetch_selective_search_data.sh` .

This will populate the `rcnn/data` folder with `selective_selective_data` .

Caffe compatibility note: R-CNN has been updated to use the new Caffe proto messages that were rolled out in Caffe v0.999. The model package contains models in the up-to-date proto format. If, for some reason, you need to get the old (Caffe proto v0) models, they can still be downloaded: VOC models ILSVRC13 model.

Running an R-CNN detector on an image

Let's assume that you've downloaded the precomputed detectors. Now:

1. Change to where you installed R-CNN: `cd rcnn` .
2. Start MATLAB `matlab` .
   - **Important:** if you don't see the message `R-CNN startup done` when MATLAB starts, then you probably didn't start MATLAB in `rcnn` directory.
3. Run the demo: `>> rcnn_demo`

4. Enjoy the detected bicycle and person

Training your own R-CNN detector on PASCAL VOC

Let's use PASCAL VOC 2007 as an example. The basic pipeline is:

```
extract features to disk -> train SVMs -> test
```

You'll need about 200GB of disk space free for the feature cache (which is stored in `rcnn/feat_cache` by default; symlink `rcnn/feat_cache` elsewhere if needed). **It's best if the feature cache is on a fast, local disk.** Before running the pipeline, we first need to install the PASCAL VOC 2007 dataset.

Installing PASCAL VOC 2007

1. Download the training, validation, test data and VOCdevkit:

```
wget http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/VOCtrainval_06-Nov-2007.tar
wget http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/VOCtest_06-Nov-2007.tar
wget http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/VOCdevkit_08-Jun-2007.tar
```

2. Extract all of these tars into one directory, it's called `VOCdevkit`.

```
tar xvf VOCtrainval_06-Nov-2007.tar
tar xvf VOCtest_06-Nov-2007.tar
tar xvf VOCdevkit_08-Jun-2007.tar
```

3. It should have this basic structure:

```
VOCdevkit/                           % development kit
VOCdevkit/VOCcode/                   % VOC utility code
VOCdevkit/VOC2007                    % image sets, annotations, etc.
... and several other directories ...
```

4. I use a symlink to hook the R-CNN codebase to the PASCAL VOC dataset:

```
ln -sf /your/path/to/voc2007/VOCdevkit /path/to/rcnn/datasets/VOCdevkit2007
```

Extracting features

```
>> rcnn_exp_cache_features('train');   % chunk1
>> rcnn_exp_cache_features('val');     % chunk2
>> rcnn_exp_cache_features('test_1');  % chunk3
>> rcnn_exp_cache_features('test_2');  % chunk4
```

Pro tip: on a machine with one hefty GPU (e.g., k20, k40, titan) and a six-core processor, I run start two MATLAB sessions each with a three worker matlabpool. I then run chunk1 and chunk2 in parallel on that machine. In this setup, completing chunk1 and chunk2 takes about 8-9 hours (depending on your CPU/GPU combo and disk) on a single machine. Obviously, if you have more machines you can hack this function to split the workload.

Training R-CNN models and testing

Now to run the training and testing code, use the following experiments script:

```
>> test_results = rcnn_exp_train_and_test()
```

Note: The training and testing procedures save models and results under `rcnn/cachedir` by default. You can customize this by creating a local config file named `rcnn_config_local.m` and defining the experiment directory variable `EXP_DIR`. Look at `rcnn_config_local.example.m` for an example.

Training an R-CNN detector on another dataset

It should be easy to train an R-CNN detector using another detection dataset as long as that dataset has *complete* bounding box annotations (i.e., all instances of all classes are labeled).

To support a new dataset, you define three functions: (1) one that returns a structure that describes the class labels and list of images; (2) one that returns a region of interest (roi) structure that describes the bounding box annotations; and (3) one that provides an test evaluation function.

You can follow the PASCAL VOC implementation as your guide:

- `imdb/imdb_from_voc.m (list of images and classes)`
- `imdb/roidb_from_voc.m (region of interest database)`
- `imdb/imdb_eval_voc.m (evalutation)`

Fine-tuning a CNN for detection with Caffe

As an example, let's see how you would fine-tune a CNN for detection on PASCAL VOC 2012.

1. Create window files for VOC 2012 train and VOC 2012 val.
   i. Start MATLAB in the `rcnn` directory
   ii. Get the imdb for VOC 2012 train: `>> imdb_train = imdb_from_voc('datasets/VOCdevkit2012', 'train', '2012');`
   iii. Get the imdb for VOC 2012 val: `>> imdb_val = imdb_from_voc('datasets/VOCdevkit2012', 'val', '2012');`
   iv. Create the window file for VOC 2012 train: `>> rcnn_make_window_file(imdb_train, 'external/caffe/examples/pascal-finetuning');`
   v. Create the window file for VOC 2012 val: `>> rcnn_make_window_file(imdb_val, 'external/caffe/examples/pascal-finetuning');`
   vi. Exit MATLAB

2. Run fine-tuning with Caffe

   i. Copy the fine-tuning prototxt files: `cp finetuning/voc_2012_prototxt/pascal_finetune_* external/caffe/examples/pascal-finetuning/`
   ii. Change directories to `external/caffe/examples/pascal-finetuning`
   iii. Execute the fine-tuning code (make sure to replace `/path/to/rcnn` with the actual path to where R-CNN is installed):

```
GLOG_logtostderr=1 ../../build/tools/finetune_net.bin \
pascal_finetune_solver.prototxt \
/path/to/rcnn/data/caffe_nets/ilsvrc_2012_train_iter_310k 2>&1 | tee log.txt
```

Note: In my experiments, I've let fine-tuning run for 70k iterations, although with hindsight it appears that improvement in mAP saturates at around 40k iterations.