Title: How to train your own class with py-faster-rcnn.

Written by: ZongYuan Ge and Inkyu Sa
Version: 0.01
Note: Experimental and draft version

The following shows how to train a binary faster-rcnn.
First we need to setup the package:

1. Please download at: https://github.com/rbgirshick/py-faster-rcnn
2. Follow the instruction and install the pyfaster-rcnn
3. Download z_ge py-faster-rcnn package from the link
https://dl.dropboxusercontent.com/u/12446150/z_ge-py-faster-rcnn.tar
4. Go to z_ge py-faster-rcnn folder and copy "lib" and "tools" to the original pyfaster-rcnn folder
5. Copy dataset and annotation 'capcicum' folder to the py-faster-rcnn folder
5. Now you can delete z_ge py-faster-rcnn folder
6. Go to /data/scripts/ and run "fetch_faster_rcnn_models.sh" and "fetch_imagenet_models.sh"

Second, let's setup the training and testing path

1. First, let's change the images index file, open /lib/datasets/pascal_voc.py, in the line
"self._image_index = self._load_image_set_index('train_index.txt')"
you can swap "train_index.txt" with your own filename, this file should look like
"TRAIN/rgb_100_2015-03-19-17-37-11.png",
where "TRAIN" is the folder has all training images.

Next, in the
```
if __name__ == '__main__':
    d = datasets.pascal_voc('Train', '/home/ge9/Desktop/capcicum')
    res = d.roidb
    from IPython import embed; embed()
```

Change '/home/ge9/Desktop/capcicum' to you own path, something like
"/../py-faster-rcnn/capcicum"

2. Let's take a look of the ground-truth file "train.txt", in the first line:
TRAIN/rgb_100_2015-03-19-17-37-11.png 4 255 452 395 612 1149 482 1283 578 1181 548 1307 686 1613 526 1723 670
"4" indicates the number of foreground objects in the image, following is the upleft corner and bottom right corner annotation (x1,y1,x2,y2),
please make sure that x1<x2 and y1<y2. Even you forget, don't worry, I have added an assertion code to prevent this from happening.

3. Now open the factory.py and change
imageset = 'TRAIN'
devkit = '/home/ge9/Desktop/capcicum'
to your own path.

Ok, we are good to go. Let's start training,
you can run: ./tools/train_faster_rcnn_alt_opt.py --net ZF --weights data/imagenet_models/ZF.v2.caffemodel --imdb Train
--net ZF: Here I am using "ZF" net, if you want better performance, you can change to 'vgg16', however, it takes much longer time.
--weights: I suggest you use the pre-trained imagenet model, if you are very confident with the size of your own dataset, you can always
train from scratch.
--imdb Train: This is the protocol name we defined in the setup file.

You will expect an error saying number of class does not match, that's normal, because we still have not changed the prototxt file for the net.
Now, assume we are using ZF net for now, go to /models/ZF/faster_rcnn_alt_opt/, OPEN "stage1_fast_rcnn_train.pt",
change num of class from 21 to 2 (since we only have foreground and background). And number of outputs from 84 to 8 (num_class * 4).
Do the same thing for "stage1_rpn_train.pt","stage2_fast_rcnn_train.pt", "stage2_rpn_train.pt" and "faster_rcnn_test.pt".
(can't find 84 as number of outputs in stage1_rpn_train.pt)

Now we are good to go for the training, have fun. One more thing to mention, if you trained one model and want to train another one, please
remove the temporary cache file at: /data/cache/, otherwise it will always be the same model (this is the proposal file)

```
I1202 17:38:57.759721 28792 solver.cpp:571] Iteration 800, lr = 0.001
I1202 17:39:00.527992 28792 solver.cpp:242] Iteration 820, loss = 0.023413
I1202 17:39:00.528075 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.0100034 (* 1 = 0.0100034 loss)
I1202 17:39:00.528095 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0134096 (* 1 = 0.0134096 loss)
I1202 17:39:00.528110 28792 solver.cpp:571] Iteration 820, lr = 0.001
I1202 17:39:03.319857 28792 solver.cpp:242] Iteration 840, loss = 0.0670749
I1202 17:39:03.319941 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.0299509 (* 1 = 0.0299509 loss)
I1202 17:39:03.319959 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0371239 (* 1 = 0.0371239 loss)
I1202 17:39:03.319977 28792 solver.cpp:571] Iteration 840, lr = 0.001
I1202 17:39:06.099439 28792 solver.cpp:242] Iteration 860, loss = 0.028125
I1202 17:39:06.099520 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.0106684 (* 1 = 0.0106684 loss)
I1202 17:39:06.099539 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0174566 (* 1 = 0.0174566 loss)
I1202 17:39:06.099553 28792 solver.cpp:571] Iteration 860, lr = 0.001
I1202 17:39:08.895670 28792 solver.cpp:242] Iteration 880, loss = 0.0393452
I1202 17:39:08.895753 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.026238 (* 1 = 0.026238 loss)
I1202 17:39:08.895771 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0131072 (* 1 = 0.0131072 loss)
I1202 17:39:08.895787 28792 solver.cpp:571] Iteration 880, lr = 0.001
I1202 17:39:11.685619 28792 solver.cpp:242] Iteration 900, loss = 0.0242981
I1202 17:39:11.685701 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.017241 (* 1 = 0.017241 loss)
I1202 17:39:11.685720 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.00705709 (* 1 = 0.00705709 loss)
I1202 17:39:11.685734 28792 solver.cpp:571] Iteration 900, lr = 0.001
I1202 17:39:14.486081 28792 solver.cpp:242] Iteration 920, loss = 0.046215
I1202 17:39:14.486186 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.0186056 (* 1 = 0.0186056 loss)
I1202 17:39:14.486207 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0276094 (* 1 = 0.0276094 loss)
I1202 17:39:14.486220 28792 solver.cpp:571] Iteration 920, lr = 0.001
I1202 17:39:17.280892 28792 solver.cpp:242] Iteration 940, loss = 0.0956086
I1202 17:39:17.280974 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.0181764 (* 1 = 0.0181764 loss)
I1202 17:39:17.280997 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0774323 (* 1 = 0.0774323 loss)
I1202 17:39:17.281011 28792 solver.cpp:571] Iteration 940, lr = 0.001
I1202 17:39:20.073487 28792 solver.cpp:242] Iteration 960, loss = 0.0301026
I1202 17:39:20.073572 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.00746469 (* 1 = 0.00746469 loss)
I1202 17:39:20.073592 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0226379 (* 1 = 0.0226379 loss)
I1202 17:39:20.073606 28792 solver.cpp:571] Iteration 960, lr = 0.001
I1202 17:39:22.863840 28792 solver.cpp:242] Iteration 980, loss = 0.0237818
I1202 17:39:22.863924 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.00854201 (* 1 = 0.00854201 loss)
I1202 17:39:22.863942 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.0152398 (* 1 = 0.0152398 loss)
I1202 17:39:22.863956 28792 solver.cpp:571] Iteration 980, lr = 0.001
speed: 0.141s / iter
I1202 17:39:25.645427 28792 solver.cpp:242] Iteration 1000, loss = 0.0101864
I1202 17:39:25.645509 28792 solver.cpp:258]     Train net output #0: rpn_cls_loss = 0.00429316 (* 1 = 0.00429316 loss)
I1202 17:39:25.645527 28792 solver.cpp:258]     Train net output #1: rpn_loss_bbox = 0.00589324 (* 1 = 0.00589324 loss)
I1202 17:39:25.645541 28792 solver.cpp:571] Iteration 1000, lr = 0.001
```

(One everything is all setup, you can expect this)

Finally, we can start testing our trained model, you need to make a few modification in the /tools/demo.py

1. Copy the trained model from /py-faster-rcnn/output/default/.. to ../py-faster-rcnn/data/faster_rcnn_models/
2. in

im_names = ['rgb_11_2015-03-19-17-36-34.png','rgb_100_2015-03-19-17-37-11.png','rgb_105_2015-03-19-17-37-12.png'],

You can load your own test images, the test images are locate at ../py-faster-rcnn/data/demo/
You can do a few modifications in the demo file to output coordinates of the prediction.

PS: We have also made an annotaion tool which you can download at:
https://dl.dropboxusercontent.com/u/12446150/v2.4_full_screen_works.zip
This Matlab script can automatically generate the required format for this faster-rcnn training.

All the training process are designed for "selective-search" and "rpn" is not supported and hard-coded by himself. Basically you need to change all the flags and consider the boundary issue caused by different proposal method. There is tutorial online teach you how to re-train fast-rcnn, but it is never the same story for faster-rcnn.

If any issue you can not run the code, please feel free to contact us.

Possible workarounds for trouble shootings and some tips. (Hope this save some your time)

$PY_FSTER_RCNN_HOME=where you have installed py-faster-rcnn (e.g., my case $PY_FSTER_RCNN_HOME=/home/sa/deep_capsicum/py-faster-rcnn)

1. Before installation, load modules on HPC by (be aware of the cuda version compiled and loaded here)

```
module load python/2.7.5
module load cuda/6.5
module load opencv/2.4.10
module load caffe
module load gcc/4.8.1
module load matlab/2015a
```

2. Using the following command to get GPU node in an interactive session (which is recommanded). Note that after the command, you need to re-load modules. (see 1)

```
qsub -V -I -l ncpus=1 -l cputype=E5-2680v2 -l ngpus=1 -l mem=32gb -l walltime=2:00:00
```

3. Error message: “protoc --proto_path=src/caffe/proto --cpp_out=.build_release/src/caffe/proto src/caffe/proto/caffe.proto make: protoc: Command not found"
Workaround: `export PATH="/pkg/suse11/caffe/protobuf/2.5.0/bin:$PATH"`

4. Error message: ImportError: $PY_FSTER_RCNN_HOME/tools/../caffe-fast-rcnn/python/caffe/_caffe.so: undefined symbol: _ZN5caffe9SGDSolverIfE11ApplyUpdateEv
Workaround: This happens because of a different shared library has been linked. (i.e., it doesn’t have “caffeSGDSolverApplyUpdate” function in it). We have to link to libcaffe.so that we have created from $PY_FSTER_RCNN_HOME/caffe-fast-rcnn/build/lib/libcaffe.so

```
export LD_LIBRARY_PATH=$PY_FSTER_RCNN_HOME/caffe-fast-rcnn/build/lib:$LD_LIBRARY_PATH
```

ADDITIONAL COMMENTS: After aforementioned command, it is always good to check LD_LIBRARY_PATH by “echo $LD_LIBRARY_PATH” and make sure $PY_FSTER_RCNN_HOME/caffe-fast-rcnn/build/lib appears first from the environment value

since the linker looks for the first appearing shared library. (e.g. if you have two test.so files in /your_folder/lib1 and /your_folder/lib2. You add these to LD_LIBRARY_PATH in order of lib1 first and lib2 second to provide linking path. Then the linker always references lib1 and lib2 doesn't have any chance to be referenced.)

5. Error message: ImportError: $PY_FSTER_RCNN_HOME/tools/../lib/utils/cython_bbox.so: undefined symbol: PyUnicodeUCS4_DecodeUTF8
Workaround: You shouldn't face this issue with gcc/4.8.1 but if your gcc version is lower than 4.8.1, try to re-compile $PY_FSTER_RCNN_HOME/utils with loading gcc/4.8.1. Make sure you have cython_bbox.so under $PY_FSTER_RCNN_HOME/utils folder with the latest timestamp.

6. Error message: rcnn/tools/../lib/roi_data_layer/minibatch.py", line 139, in _get_image_blob
im = im[:, ::-1, :] TypeError: 'NoneType' object has no attribute '__getitem__'
Workaround: If you face this error that implies your opencv is not properly working. (cv2.imread loads no image then it complains there is no data ("NoneType"). Simply unload opencv/2.4.10 and load again with the following command.

```
module unload opencv/2.4.10
```

and then

```
module load opencv/2.4.10
```

Check your opencv is working properly with the following simple code. (in python console or you can create a python file then type "python test.py", see test2.png is created or not.)
test.py

```
import cv2
img = cv2.imread('./test.png',0)
cv2.imwrite('./test2.png',img)
```

7. Error message: ImportError: libcudart.so.7.0: cannot open shared object file: No such file or directory
Workaround: This happens due to $PY_FSTER_RCNN_HOME/lib built against libcudart.so.7.0 A possible workaround is rebuilding lib folder by touching files located lib folder. For example, just open a cpp file and add space and delete it. This updates the time filed modified and allows compiler and linker to recompile and to link. Shared libraries located in lib folder are needed to be recompiled against to your system.

8. Error message: ./include/caffe/util/math_functions.hpp:143:390: error: invalid qualifiers on non-member function type
Workaround: Check whether you typed "make -j8" and then simply do "make all" instead. It seems faster-rcnn-caffe/HPC doesn't like parallel compilation. Not sure but I could compile with "make all" though.