Train Py-Faster-RCNN on Another Dataset

This tutorial is a fine-tuned clone of zeyuanxy's one for the py-faster-rcnn code.

We will illustrate how to train Py-Faster-RCNN on another dataset in the following steps, and we will take **INRIA Person** as the example dataset.

Clone py-faster-rcnn repository

The current tutorial need you to have clone and tested the regular py-faster-rcnn repository from rbgirshick.

```
$ git clone https://github.com/rbgirshick/py-faster-rcnn
```

We will refer to the root directory with $PY_FASTER_RCNN.

You will also need to follow the installation steps from the original py-faster-rcnn readme

Build the train-set

Get the Dataset

When you download and extract the Inria Person dataset you obtain this architecture:

```
|-- INRIAPerson/
    |-- 70X134H96/
    |-- 96X160H96/
    |-- Test/
    |-- test_64x128_H96/
    |-- Train/
    |-- train_64x128_H96/
```

Format the Dataset

But we will use this common architecture for every dataset in $PY_FASTER_RCNN/data

```
INRIA_Person_devkit/
|-- data/
    |-- Annotations/
         |-- *.txt (Annotation files)
    |-- Images/
         |-- *.png (Image files)
    |-- ImageSets/
         |-- train.txt
```

A simple way to achieve it is to use symbolic links: (this is only an example for training, some refactoring will be needing in order to use the testset properly)

```
$ cd $PY_FASTER_RCNN/data
$ mkdir INRIA_Person_devkit/
$ mkdir INRIA_Person_devkit/data/
$ ln -s <INRIAPerson>/Train/annotations/ INRIA_Person_devkit/data/Annotations
$ ln -s <INRIAPerson>/Train/pos/ INRIA_Person_devkit/data/Images
```

Now we need to write `train.txt` that contains all the names(without extensions) of images files that will be used for training. Basically with the following:

```
$ cd $PY_FASTER_RCNN/data/INRIA_Person_devkit/data/
$ mkdir ImageSets
$ ls Annotations/ -m | sed s/\\s/\\n/g | sed s/.txt//g | sed s/,//g > ImageSets/trai
```

Add lib/datasets/yourdatabase.py

You need to add a new python file describing the dataset we will use to the directory `$PY_FASTER_RCNN/lib/datasets` , see inria.py. Then the following steps should be taken.

- Modify `self._classes` in the constructor function to fit your dataset.
- Be careful with the extensions of your image files.
  See `image_path_from_index` in `inria.py` .
- Write the function for parsing annotations. See `_load_inria_annotation` in `inria.py` .
- Do not forget to add `import` syntaxes in your own python file and other python files in the same directory.

Update lib/datasets/factory.py

Then you should modify the factory.py in the same directory. For example, to add **INRIA Person**, we should add

```
from datasets.inria import inria
inria_devkit_path = '$PY_FASTER_RCNN/data/INRIA_Person_devkit'
for split in ['train', 'test']:
    name = '{}_{}'.format('inria', split)
    __sets[name] = (lambda split=split: inria(split, inria_devkit_path))
```

NB : $PY_FASTER_RCNN must be replaced by its actual value !

Adapt the network model

For example, if you want to use the model **VGG_CNN_M_1024** with alternated optimizations, then you should adapt the solvers
in `$PY_FASTER_RCNN/models/VGG_CNN_M_1024/faster_rcnn_alt_opt/`

```
$ cd $PY_FASTER_RCNN/models/
$ mkdir INRIA_Person/
$ cp -r pascal_voc/VGG_CNN_M_1024/faster_rcnn_alt_opt/ INRIA_Person/
```

It mainly concerns with the number of classes you want to train. Let's assume that the number of classes is C (do not forget to count the `background` class). Then you should

- Modify `num_classes` to `C` ;
- Modify `num_output` in the `cls_score` layer to `C`
- Modify `num_output` in the `bbox_pred` layer to `4 * C`

Basically for our binary classifier (Person vs Background) C=2 and you have: - 7 lines to be modified from 21 to 2 - 3 lines to be modified from 84 to 8

```
$ grep 21 VGG_CNN_M_1024/faster_rcnn_alt_opt/*.pt
   INRIA_Person/faster_rcnn_alt_opt/faster_rcnn_test.pt:    num_output: 21
   INRIA_Person/faster_rcnn_alt_opt/stage1_fast_rcnn_train.pt:    param_str: "'num_c
   INRIA_Person/faster_rcnn_alt_opt/stage1_fast_rcnn_train.pt:    num_output: 21
   INRIA_Person/faster_rcnn_alt_opt/stage1_rpn_train.pt:    param_str: "'num_classes
   INRIA_Person/faster_rcnn_alt_opt/stage2_fast_rcnn_train.pt:    param_str: "'num_c
   INRIA_Person/faster_rcnn_alt_opt/stage2_fast_rcnn_train.pt:    num_output: 21
   INRIA_Person/faster_rcnn_alt_opt/stage2_rpn_train.pt:    param_str: "'num_classes
$ grep 84 VGG_CNN_M_1024/faster_rcnn_alt_opt/*.pt
   INRIA_Person/faster_rcnn_alt_opt/faster_rcnn_test.pt:    num_output: 84
   INRIA_Person/faster_rcnn_alt_opt/stage1_fast_rcnn_train.pt:    num_output: 84
   INRIA_Person/faster_rcnn_alt_opt/stage2_fast_rcnn_train.pt:    num_output: 84
```

Build config file

The $PY_FASTER_RCNN/models folder must be specified by a config file as in faster_rcnn_alt_opt.yml

```
$ echo 'MODELS_DIR: "$PY_FASTER_RCNN/models"' >> config.yml
```

NB : $PY_FASTER_RCNN must be replaced by its actual value !

Launch the training

In the directory $PY_FASTER_RCNN, run the following command in the shell.

```
$ ./tools/train_faster_rcnn_alt_opt.py --gpu 0 --net_name INRIA_Person --weights dat
```

Where:

> --net_name is the folder name in $PY_FASTER_RCNN/models
> (nb: the train_faster_rcnn_alt_opt.py script will automatically look into the

> /faster_rcnn_alt_opt/ subfolder for the .pt files)
> --weights is the optional location of pretrained weights in .caffemodel
> --imdb is the full name of the database as specified in the lib/datasets/factory.py file
> (nb: dont forget to add the test/train suffix !)

Or the following connection-proof version if you're afraid of ctrl+C or using your hardware remotely like me :)

```
$ nohup ./tools/train_faster_rcnn_alt_opt.py --gpu 0 --net_name INRIA_Person --weigh
$ tail nohup.out
```

(Just needs you to launch a "$ killall python" to interrupt training. Yes...)