Installing yolo v3-SPP

Step 1 - Requirements

- Anaconda – Python 3.6 (Win 10) https://www.anaconda.com/download/

- Conda Env – Yolo.yml
  https://github.com/reigngt09/yolov3workflow/tree/master/2_YoloV3_Execute
  ```
  cd C:\yolo
  conda env create -f yolo.yml
  ```
  If that does not work, try installing the dependencies with:
  ```
  pip install -r requirements.txt
  ```

- CUDA Toolkit - V9.0 (Nvidia GPU – GTX 1050 or higher) CUDA and CuDNN can now be installed via Anaconda, but if you choose to install them using the ol skool method then follow the links below.
  https://developer.nvidia.com/cuda-90-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal

- CuDNN - V 7.05
  https://developer.nvidia.com/rdp/cudnn-archive
  - Copy the following files into the CUDA Toolkit directory.
    Copy \cuda\bin\cudnn64_7.dll to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin.
    Copy \cuda\ include\cudnn.h to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\include.
    Copy \cuda\lib\x64\cudnn.lib to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64.
  - Add the following paths to Environmental Variables C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin
    C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\include
    C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64

  - Ensure that you have the latest nvidia graphics drivers install on your PC. You can do this from the nvidia website.

Step 2 - PyTorch Yolo v3

Change directory to a workplace where you want to download the repo

- Clone Yolo v3 Repo
  - `git clone https://github.com/ayooshkathuria/pytorch-yolo-v3.git`
- Download the Weights
  https://pjreddie.com/media/files/yolov3.weights

Step 3 - PyTorch Yolo v3

- Change Directory to cloned repo
  `cd C:\yolotorch`
- Download any test video (.mp4/.avi)
- Run demo on video
  `python video_demo.py --video video.mp4`
- Run demo on webcam
  `python cam_demo.py`

YoloV3 tiny model

For installing yolo v3 tiny model do the following steps:
1. Install Darknet
2. Download the weights
3. Run the detector using command `./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg`
4. If real time object detection is needed compile darknet with CUDA and OpenCv

Installing darknet:
Give commands:

```
git clone https://github.com/pjreddie/darknet.git
cd darknet
make
```

Compiling darknet with cuda and opencv

1. If you have CUDA 10.0, cuDNN 7.4 and OpenCV 3.x (with paths: `C:\opencv_3.0\opencv\build\include` & `C:\opencv_3.0\opencv\build\x64\vc14\lib`), then open `build\darknet\darknet.sln`, set x64 and Release https://hsto.org/webt/uh/fk/-e/uhfk-eb0q-hwd9hsxhrikbokd6u.jpeg and do the: Build -> Build darknet. Also add Windows system variable `CUDNN` with path to CUDNN: https://user-images.githubusercontent.com/4096485/53249764-019ef880-36ca-11e9-8ffe-d9cf47e7e462.jpg
   1.1. Find files `opencv_world320.dll` and `opencv_ffmpeg320_64.dll` (or `opencv_world340.dll` and `opencv_ffmpeg340_64.dll`) in `C:\opencv_3.0\opencv\build\x64\vc14\bin` and put it near with `darknet.exe`
   1.2 Check that there are `bin` and `include` folders in the `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0` if aren't, then copy them to this folder from the path where is CUDA installed
   1.3. To install CUDNN (speedup neural network), do the following:
   - download and install cuDNN v7.4.1 for CUDA 10.0: https://developer.nvidia.com/rdp/cudnn-archive
   - add Windows system variable `CUDNN` with path to CUDNN: https://user-images.githubusercontent.com/4096485/53249764-019ef880-36ca-11e9-8ffe-d9cf47e7e462.jpg
   - copy file `cudnn64_7.dll` to the folder `\build\darknet\x64` near with `darknet.exe`
2. 1.4. If you want to build without CUDNN then: open `\darknet.sln` -> (right click on project) -> properties -> C/C++ -> Preprocessor -> Preprocessor Definitions, and remove this: `CUDNN;`
3. If you have other version of CUDA (not 10.0) then open `build\darknet\darknet.vcxproj` by using Notepad, find 2 places with "CUDA 10.0" and change it to your CUDA-version. Then open `\darknet.sln` -> (right click on project) -> properties -> CUDA C/C++ -> Device and remove there `;compute_75,sm_75`. Then do step 1
4. If you don't have GPU, but have OpenCV 3.0 (with paths: `C:\opencv_3.0\opencv\build\include` & `C:\opencv_3.0\opencv\build\x64\vc14\lib`), then open `build\darknet\darknet_no_gpu.sln`, set x64 and Release, and do the: Build -> Build darknet_no_gpu

5. If you have OpenCV 2.4.13 instead of 3.0 then you should change paths after `\darknet.sln` is opened
   4.1 (right click on project) -> properties -> C/C++ -> General -> Additional Include Directories: `C:\opencv_2.4.13\opencv\build\include`
   4.2 (right click on project) -> properties -> Linker -> General -> Additional Library Directories: `C:\opencv_2.4.13\opencv\build\x64\vc14\lib`
6. If you have GPU with Tensor Cores (nVidia Titan V / Tesla V100 / DGX-2 and later) speedup Detection 3x, Training 2x: `\darknet.sln` -> (right click on project) -> properties -> C/C++ -> Preprocessor -> Preprocessor Definitions, and add here: `CUDNN_HALF;`
   Note: CUDA must be installed only after Visual Studio has been installed.