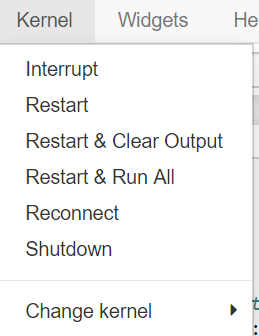
1. Open IPython notebook called HomeWork2\_RCNN.ipynb
2. Press “Kernel” and “Restart & Run All” in your notebook



1. This script will install all libs which are needed for starting homework: also such specific one as “protobuf”
2. After you start the script you’ll get the error “*ModuleNotFoundError: No module named 'model.utils.cython\_bbox'*”

Full log for this error is in file “error 1.txt”

The solution is in file “solution error 1.txt”

1. Let’s repeat: Press “Kernel” and “Restart & Run All” in your notebook
2. After you start the script you’ll get the errors with “gcc”

Full log for this error is in file “errors make.sh + solutions .txt”

The solution is in file “errors make.sh + solutions .txt”

Files to upload are in dir “for make.sh”

1. Let’s repeat: Press “Kernel” and “Restart & Run All” in your notebook
2. After you start the script you’ll get the error “*FileNotFoundError: [Errno 2] No such file or directory: 'data/pretrained\_model/resnet101\_caffe.pth''*”

Full log for this error is in file “error 2.txt”

The solution is in file “solution error 2.txt”

1. If we choose “solution 1“ (download pre-trained model), we won’t download file in case of problems with upload button or with problems with proxy
2. If we choose “alternative solution”: Let’s repeat: Press “Kernel” and “Restart & Run All” in your notebook
3. The network will train from scratch. It will finish the train. Let’s start our test mode
4. Error 2: no file named “test.txt”. So, we needed to create it ourselves

Solution Error 2: (also already done in my python notebook)



1. Only then we can test our network
2. Use the follow command: !python test\_net.py --dataset pascal\_voc --checkepoch 20 --checkpoint 10021 --net res50 –cuda

Also, if we will train the network from scratch, we’ll get the follow results (spoiler: awful results):

VOC07 metric? Yes   
AP for aeroplane = 0.0003   
AP for bicycle = 0.0003   
AP for bird = 0.0004   
AP for boat = 0.0000   
AP for bottle = 0.0000   
AP for bus = 0.0031   
AP for car = 0.0029   
AP for cat = 0.0073   
AP for chair = 0.0002   
AP for cow = 0.0002   
AP for diningtable = 0.0001   
AP for dog = 0.0013   
AP for horse = 0.0013   
AP for motorbike = 0.0018   
AP for person = 0.0106   
AP for pottedplant = 0.0001   
AP for sheep = 0.0000   
AP for sofa = 0.0002   
AP for train = 0.0010   
AP for tvmonitor = 0.0001   
Mean AP = 0.0016   
~~~~~~~~   
Results:   
0.000   
0.000   
0.000   
0.000   
0.000   
0.003   
0.003   
0.007   
0.000   
0.000   
0.000   
0.001   
0.001   
0.002   
0.011   
0.000   
0.000   
0.000   
0.001   
0.000   
0.002   
~~~~~~~~

Some info from readme.md of the given codebase:

### prerequisites

\* Python 2.7 or 3.6

\* Pytorch 0.4.0 (\*\*now it does not support 0.4.1 or higher\*\*)

\* CUDA 8.0 or higher

### Data Preparation

\* \*\*PASCAL\_VOC 07+12\*\*: Please follow the instructions in [py-faster-rcnn](https://github.com/rbgirshick/py-faster-rcnn#beyond-the-demo-installation-for-training-and-testing-models) to prepare VOC datasets. Actually, you can refer to any others. After downloading the data, create softlinks in the folder data/.

### Pretrained Model

We used two pretrained models in our experiments, VGG and ResNet101. You can download these two models from:

\* VGG16: [Dropbox](https://www.dropbox.com/s/s3brpk0bdq60nyb/vgg16\_caffe.pth?dl=0), [VT Server](https://filebox.ece.vt.edu/~jw2yang/faster-rcnn/pretrained-base-models/vgg16\_caffe.pth)

\* ResNet101: [Dropbox](https://www.dropbox.com/s/iev3tkbz5wyyuz9/resnet101\_caffe.pth?dl=0), [VT Server](https://filebox.ece.vt.edu/~jw2yang/faster-rcnn/pretrained-base-models/resnet101\_caffe.pth)

Download them and put them into the data/pretrained\_model/.

\*\*NOTE\*\*. We compare the pretrained models from Pytorch and Caffe, and surprisingly find Caffe pretrained models have slightly better performance than Pytorch pretrained. We would suggest to use Caffe pretrained models from the above link to reproduce our results.

\*\*If you want to use pytorch pre-trained models, please remember to transpose images from BGR to RGB, and also use the same data transformer (minus mean and normalize) as used in pretrained model.\*\*

1. If we have the pre-trained model – be happy, everything will work.