Installing Keras RetinaNet

As you know from reading this book, we are using Python virtual environments to keep our development environments separate from each other, avoiding any package version conflict issues. I would therefore highly suggest that you create a new Python virtual environment (named `retinanet`) for the RetinaNet install:

Installing Keras RetinaNet — Shell

```
$ mkvirtualenv retinanet -p python3
```

You can *technically* use the existing `dl4cv` Python virtual environment as well, but I would not recommend it.

Regardless of whether you choose to use an *existing* or *new* Python virtual environment, the next step is to install the associated dependencies:

Installing Keras RetinaNet — Shell

```
$ pip install numpy scipy h5py
$ pip install scikit-learn Pillow imutils
$ pip install beautifulsoup4
$ pip install tensorflow-gpu
$ pip install keras
```

TensorFlow is a *requirement* when using RetinaNet. Here I am using the GPU version of TensorFlow `tensorflow-gpu` ; however, you could technically use the CPU version (`tensorflow`) as well, although networks would take an extremely long time to train). You may also need to change your TensorFlow version depending on which version of CUDA and cuDNN you are using. Be sure to refer to the TensorFlow install documentation for more details.

You should also note that the Keras version should be v2.2.2 or higher. Again, Keras is a *requirement* for RetinaNet.

We now need to sym-link our OpenCV `cv2` bindings into our Python virtual environment (provided you are using a brand new virtual environment versus an existing one):

```
Installing Keras RetinaNet                                                                  Shell
$ cd ~/.virtualenvs/retinanet/lib/python3.4/site-packages/
$ ln -s /usr/local/lib/python3.4/site-packages/cv2.cpython-35m-x86_64-linux-gnu.so cv2.so
```

Your exact path to the `cv2.so` bindings for your OpenCV install may be different than mine based on your system and Python version, so make sure you:

- Refer to the development environment configuration instructions on the companion website homepage.
- Verify your OpenCV install path before creating the sym-link (otherwise the sym-link will point to a nonexistent file and the import will fail).

If you are having problems with your OpenCV install you may want to try a simple pip install of the library:

```
Installing Keras Mask R-CNN                                                                 Shell
$ pip install opencv-contrib-python
```

I provide more information on installing OpenCV via pip in this blog post.

Take the time now to verify that your OpenCV install is working before continuing.

The RetinaNet implementation we will be using can be found on their GitHub project page. Use the following commands to clone down the Keras RetinaNet repository and install it:

```
Installing Keras RetinaNet                                                                  Shell
$ cd ~
$ git clone https://github.com/fizyr/keras-retinanet
$ cd keras-retinanet
$ git checkout 42068ef9e406602d92a1afe2ee7d470f7e9860df
$ python setup.py install
```

We'll be using the v0.4.1 release as our base; however, as I mentioned earlier in this chapter, the `keras-retinanet` repository is under active development and changes frequently. To ensure you can use the code out of the box with this chapter, ensure you checkout the `42068ef9e406602d92a1afe2ee7d470f7e9860df` commit via the commands listed above.

Provided the `setup.py` script exited without an error, you should now verify your install by opening a Python shell and trying to import `keras`, `cv2`, and `keras_retinanet`:

Installing Keras RetinaNet — Shell

```
$ python
>>> import keras
Using TensorFlow backend.
>>> import cv2
>>> import keras_retinanet
>>>
```

If all imports succeed, you are good to go!

I would also recommend double-checking that the `retinanet-train` command has been added to your system path during the install of Keras RetinaNet:

Installing Keras RetinaNet — Shell

```
$ retinanet-train
usage: retinanet-train
...
retinanet-train: error: the following arguments are required: dataset_type
```

If the command is not found, that's okay — there is a separate way to execute the script. First, change directory to `keras-retinanet` (i.e., the project repository that we cloned from GitHub) and then try to execute `keras_retinanet/bin/train.py`:

Installing Keras RetinaNet — Shell

```
$ cd ~/keras-retinanet
$ keras_retinanet/bin/train.py
...
train.py: error: the following arguments are required: dataset_type
```

The `retinanet-train` command is simply a shortcut to `train.py`, enabling you to execute the `train.py` script from *any* directory, saving you a *bunch* of keystrokes.

Finally, I recommend downloading the resnet50_coco_best_v2.1.0.h5 file from the official repo as well:

Installing Keras RetinaNet — Shell

```
$ wget https://github.com/fizyr/keras-retinanet/releases/download/0.5.0/resnet50_coco_best_v2.1.0.h5
```

This RetinaNet model with ResNet backbone has been pre-trained for us on the COCO dataset. We'll be taking this model and fine-tuning it for our own purposes. Keep a copy of this file on your local disk as you'll want to use it as a starting point for any object detectors you train.

Links

- Supplementary material
- Bug tracking and issues
- PyImageSearch contact form