appendix A
Installing Keras and its dependencies on Ubuntu

The process of setting up a deep-learning workstation is fairly involved and consists of the following steps, which this appendix will cover in detail:

1. Install the Python scientific suite—Numpy and SciPy—and make sure you have a Basic Linear Algebra Subprogram (BLAS) library installed so your models run fast on CPU.
2. Install two extras packages that come in handy when using Keras: HDF5 (for saving large neural-network files) and Graphviz (for visualizing neural-network architectures).
3. Make sure your GPU can run deep-learning code, by installing CUDA drivers and cuDNN.
4. Install a backend for Keras: TensorFlow, CNTK, or Theano.
5. Install Keras.

It may seem like a daunting process. In fact, the only difficult part is setting up GPU support—otherwise, the entire process can be done with a few commands and takes only a couple of minutes.

We'll assume you have a fresh installation of Ubuntu, with an NVIDIA GPU available. Before you start, make sure you have `pip` installed and that your package manager is up to date:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install python-pip python-dev
```

Python 2 vs. Python 3

By default, Ubuntu uses Python 2 when it installs Python packages such as `python-pip`. If you wish to use Python 3 instead, you should use the `python3` prefix instead of `python`. For instance:

```
$ sudo apt-get install python3-pip python3-dev
```

When you're installing packages using `pip`, keep in mind that by default, it targets Python 2. To target Python 3, you should use `pip3`:

```
$ sudo pip3 install tensorflow-gpu
```

A.1 Installing the Python scientific suite

If you use a Mac, we recommend that you install the Python scientific suite via Anaconda, which you can get at www.continuum.io/downloads. Note that this won't include HDF5 and Graphviz, which you have to install manually. Following are the steps for a *manual* installation of the Python scientific suite on Ubuntu:

1. Install a BLAS library (OpenBLAS, in this case), to ensure that you can run fast tensor operations on your CPU:

```
$ sudo apt-get install build-essential cmake git unzip \
     pkg-config libopenblas-dev liblapack-dev
```

2. Install the Python scientific suite: Numpy, SciPy and Matplotlib. This is necessary in order to perform any kind of machine learning or scientific computing in Python, regardless of whether you're doing deep learning:

```
$ sudo apt-get install python-numpy python-scipy python- matplotlib
➥python-yaml
```

3. Install HDF5. This library, originally developed by NASA, stores large files of numeric data in an efficient binary format. It will allow you to save your Keras models to disk quickly and efficiently:

```
$ sudo apt-get install libhdf5-serial-dev python-h5py
```

4. Install Graphviz and pydot-ng, two packages that will let you visualize Keras models. They aren't necessary to run Keras, so you could skip this step and install these packages when you need them. Here are the commands:

```
$ sudo apt-get install graphviz
$ sudo pip install pydot-ng
```

5. Install additional packages that are used in some of our code examples:

```
$ sudo apt-get install python-opencv
```

A.2 Setting up GPU support

Using a GPU isn't strictly necessary, but it's strongly recommended. All the code examples found in this book can be run on a laptop CPU, but you may sometimes have to wait for several hours for a model to train, instead of mere minutes on a good GPU. If you don't have a modern NVIDIA GPU, you can skip this step and go directly to section A.3.

To use your NVIDIA GPU for deep learning, you need to install two things:

- *CUDA*—A set of drivers for your GPU that allows it to run a low-level programming language for parallel computing.
- *cuDNN*—A library of highly optimized primitives for deep learning. When using cuDNN and running on a GPU, you can typically increase the training speed of your models by 50% to 100%.

TensorFlow depends on particular versions of CUDA and the cuDNN library. At the time of writing, it uses CUDA version 8 and cuDNN version 6. Please consult the TensorFlow website for detailed instructions about which versions are currently recommended: www.tensorflow.org/install/install_linux.

Follow these steps:

1. Download CUDA. For Ubuntu (and other Linux flavors), NVIDIA provides a ready-to-use package that you can download from https://developer.nvidia.com/cuda-downloads:

```
$ wget http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/
➥x86_64/cuda-repo-ubuntu1604_9.0.176-1_amd64.deb
```

2. Install CUDA. The easiest way to do so is to use Ubuntu's `apt` on this package. This will allow you to easily install updates via `apt` as they become available:

```
$ sudo dpkg -i cuda-repo-ubuntu1604_9.0.176-1_amd64.deb
$ sudo apt-key adv --fetch-keys
➥http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/
➥x86_64/7fa2af80.pub
$ sudo apt-get update
$ sudo apt-get install cuda-8-0
```

3. Install cuDNN:
   a. Register for a free NVIDIA developer account (unfortunately, this is necessary in order to gain access to the cuDNN download), and download cuDNN at https://developer.NVIDIA.com/cudnn (select the version of cuDNN compatible with TensorFlow). Like CUDA, NVIDIA provides packages for different Linux flavors—we'll use the version for Ubuntu 16.04. Note that if you're working with an EC2 install, you won't be able to download the cuDNN archive directly to your instance; instead, download it to your local machine and then upload it to your EC2 instance (via `scp`).
   b. Install cuDNN:

```
$ sudo dpkg -i dpkg -i libcudnn6*.deb
```

4. Install TensorFlow:
   a. TensorFlow with or without GPU support can be installed from PyPI using Pip. Here's the command without GPU support:

      ```
      $ sudo pip install tensorflow
      ```

   b. Here's the command to install TensorFlow with GPU support:

      ```
      $ sudo pip install tensorflow-gpu
      ```

A.3 Installing Theano (optional)

Because you've already installed TensorFlow, you don't have to install Theano in order to run Keras code. But it can sometimes be useful to switch back and forth from TensorFlow to Theano when building Keras models.

Theano can also be installed from PyPI:

```
$ sudo pip install theano
```

If you're using a GPU, then you should configure Theano to use your GPU. You can create a Theano configuration file with this command:

```
nano ~/.theanorc
```

Then, fill in the file with the following configuration:

```
[global]
floatX = float32
device = gpu0

[nvcc]
fastmath = True
```

A.4 Installing Keras

You can install Keras from PyPI:

```
$ sudo pip install keras
```

Alternatively, you can install Keras from GitHub. Doing so will allow you to access the keras/examples folder, which contains many example scripts for you to learn from:

```
$ git clone https://github.com/fchollet/keras
$ cd keras
$ sudo python setup.py install
```

You can now try to run a Keras script, such as this MNIST example:

```
python examples/mnist_cnn.py
```

Note that running this example to completion may take a few minutes, so feel free to force-quit it (Ctrl-C) once you've verified that it's working normally.

After you've run Keras at least once, the Keras configuration file can be found at ~/.keras/keras.json. You can edit it to select the backend that Keras runs on: `tensorflow`, `theano`, or `cntk`. Your configuration file should like this:

```
{
    "image_data_format": "channels_last",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "tensorflow"
}
```

While the Keras script examples/mnist_cnn.py is running, you can monitor GPU utilization in a different shell window:

```
$ watch -n 5 NVIDIA-smi -a --display=utilization
```

You're all set! Congratulations—you can now begin building deep-learning applications.