Machine Learning

Pawel Wocjan

University of Central Florida

Spring 2019

Keras

- Keras is a deep-learning framework for Python that provides a convenient way to define and train almost any kind of deep-learning model.
- Keras has the following features:
  - It allows the same code to run seamlessly on CPU or GPU.
  - It has a user-friendly API that makes it easy to quickly prototype deep-learning models.
  - It has built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and for any combination of both.
  - It supports arbitrary network architectures such as multi-input or multi-output models, and layer sharing.
  - It is appropriate for building essentially any deep-learning model, from a generative adversarial network to a neural Turing machine.
- Keras documentation `https://keras.io/`

Keras, TensorFlow, Theano, and CNTK

- Keras is a model-level library, providing high-level building blocks for developing deep-learning models.
- It doesn't handle low-level operations such as tensor manipulation and differentiation.
- Instead, it relies on a specialized, well-optimized library to do so, serving as a backend of Keras.
- Rather, than choosing a single tensor library and tying the implementation of Keras to that library, Keras handles the problem in a modular way:

<table>
<tr><td colspan="2">Keras</td></tr>
<tr><td colspan="2">TensorFlow/Theano/CNTK/ ...</td></tr>
<tr><td>CUDA/cuDNN</td><td>Blas, Eigen</td></tr>
<tr><td>GPU</td><td>CPU</td></tr>
</table>

Keras, TensorFlow, Theano, and CNTK

- TensorFlow
  `https://www.tensorflow.org/`
- Theano
  `http://www.deeplearning.net/software/theano/`
- CNTK
  `https://www.microsoft.com/en-us/cognitive-toolkit/`
- CUDA
  `https://developer.nvidia.com/about-cuda`
- cuDNN
  `https://developer.nvidia.com/cudnn`
- Blas
  `http://www.netlib.org/blas/`
- Eigen
  `http://eigen.tuxfamily.org/index.php?title=Main_Page`

Developing with Keras: a quick overview

The typical Keras workflow looks like:

- Define your training data: input tensors and target tensors.
- Define a network of layers (or model) that maps your inputs to your targets.
- Configure the learning process by choosing a loss function, an optimizer, and some metrics to monitor.
- Iterate on your training data by calling the `fit()` function method of your model.