Deep Learning Software Packages\textsuperscript{1,2,3,4}

Tensorflow, Torch, Theano and Caffe

Amal Agarwal

\textsuperscript{1}Software Packages; Fei-Fei Li, Andrej Karpathy and Justin Johnson
\textsuperscript{2}TensorFlow Tutorial (Sherry Moore, Google Brain)
\textsuperscript{3}Torch Tutorial (Alex Wiltschko, Twitter)
\textsuperscript{4}Theano Tutorial (Pascal Lamblin, MILA)
Outline

1. Tensorflow
2. Torch
3. Theano
Outline

1. Tensorflow
2. Torch
3. Theano
Tensorflow Architecture

- Google infrastructure
- Research → Prototyping → Production
- Tensors: Data structured as multi-dimensional arrays.
- Data (Tensors) flow through a graph.
Data Flow Graph

[Diagram of a data flow graph showing layers such as SGD Trainer, Cross Entropy, Softmax, Logit Layer, ReLU Layer, and Reshape.]
Pros and Cons

- (+) Python + numpy
- (+) Computational graph abstraction, like Theano; great for RNNs
- (+) Much faster compile times than Theano
- (+) Slightly more convenient than raw Theano?
- (+) TensorBoard for visualization
- (+) Data AND model parallelism; best of all frameworks
- (+/-) Distributed models, but not open-source yet
- (-) Slower than other frameworks right now
- (-) Much “fatter” than Torch; more magic
- (-) Not many pretrained models
Outline

1. Tensorflow
2. Torch
3. Theano
Torch Architecture

- Based on Lua
- Efficient Tensor library
- Efficient CUDA backend; GPU is really fast for big networks
- Neural Networks package → build arbitrary acyclic computation graphs with automatic differentiation
Lua Basics

- Lua is pretty close to javascript.
  - Variables are global by default, unless local keyword is used
  - Only has one data structure built-in, a table: .
  - 1-based indexing.

- Lua glues C/C++ libraries together
  - Develop fast (scripting language), runs fast (minor overhead, C backend)

- The basic brick is the Tensor object
  - n-dimensional array
  - used to store any kind of data

- The torch package provides tensors... hundreds of packages are built upon it.
nn Containers

- nn.Sequential
- nn.Concat
- nn.Parallel
Sequential Convolution neural network
In Torch, loss functions are implemented just like neural network modules, and have automatic differentiation. We have two functions:

- forward(input, target)
- backward(input, target)

Common criterion:

- nn.CrossEntropyCriterion()
Training the network

- Do your own training loop
  - Pro: easy customization for complicated network
  - Con: code duplication
- Use existing packages
  - optim
  - nn.StochasticGradient
Pros and Cons

- (-) Lua
- (-) Less plug-and-play than Caffe
  - You usually write your own training code
- (+) Lots of modular pieces that are easy to combine
- (+) Easy to write your own layer types and run on GPU
- (+) Most of the library code is in Lua, easy to read
- (+) Lots of pretrained models!
- (-) Not great for RNNs
Outline

1. Tensorflow
2. Torch
3. Theano
Theano Architecture

Mathematical syntax expression compiler

- Easy to define expressions.
- Possible to manipulate those expressions.
- Fast to compute values for those expressions.
- Tools to check for correctness and debugging are very easy (debugprint).
Computational Graph

Graph representation with nodes labeled x, y, z, a, b, and c connected by arrows indicating the flow of computation.
Workflow

- Symbolic expressions
  - Declaring inputs
  - Defining expressions
  - Deriving gradients

- Function Compilation
  - Graph optimizations
  - Graph visualizations

- Optimized execution
  - Code generation and execution GPU
Different goals of optimization

- Merge equivalent computations
- Simplify expressions: $x/x$ becomes 1
- Numerical stability
- Destructive operations
- Transfer to GPU
Pros and Cons

- (+) Python + numpy
- (+) Computational graph is nice abstraction
- (+) RNNs fit nicely in computational graph
- (-) Raw Theano is somewhat low-level
- (+) High level wrappers (Keras, Lasagne) ease the pain
- (-) Error messages can be unhelpful
- (-) Large models can have long compile times
- (-) Much “fatter” than Torch; more magic
- (-) Patchy support for pretrained models