Deep Learning Frameworks

Nils Reimers

Course-Website: www.deeplearning4nlp.com

Please have your Python environment up and running

Requirements for a Framework

- To train Deep Neural Networks requires billions of operations
  - Google had trained some systems on up to 16.000 cores

- Performance in training time is **crucial**
  - More data = better results = your paper get published
  - Some times: Larger network = better results = your paper get published
  - Slow computation is not acceptable => Choose the framework that is the fastest

- Nearly all operations are matrix operations (multiplications, additions)
  - Easy syntax for matrices desired
  - Optimizing matrix multiplication for speed is hard, simple "two loop solution" is way too slow

- Must be runnable on a GPU

- Nice to have: Easy computation of gradients

Neural Network as One Long Function

- Neural Networks can be expressed as one long function of vector and matrix operations

$$output = \mathrm{softmax}(b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)))$$

$$E(x, W, b) = -\log(\mathrm{softmax}(b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)))_y)$$

Img-Source: http://ufldl.stanford.edu/wiki/
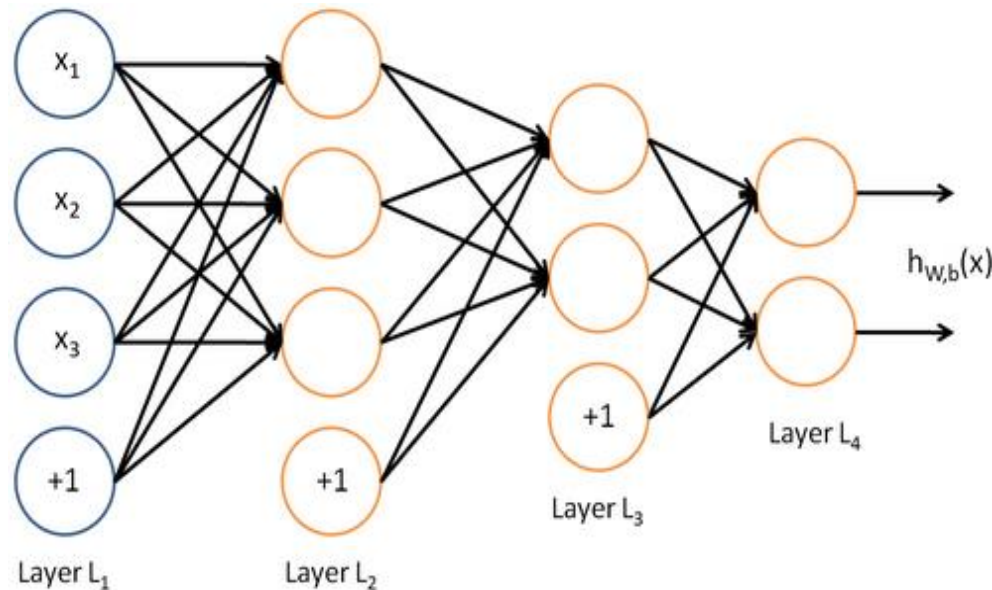
Common Frameworks

- Outdated: C/C++, Matlab, Caffe
- Torch (Lua)
- DeepLearning4j (Java)

Most used frameworks:
- Theano
  - One of the oldest deep learning frameworks
  - Python based framework
  - Largely used in the research community
- TensorFlow
  - Deep Learning framework by Google
  - Written in C/C++, wrapper exists for various languages
  - Most people use the Python wrapper for TensorFlow
  - My recommendation for starters in deep learning – learn TF

Introduction to Theano

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- TensorFlow uses similar concepts and porting code is not too difficult

UKP

Theano

Advantages

- Python library with tight integration of NumPy
  - Easy syntax for matrix operations ✓
- Transparent use of GPU (speed-up of up to 140x) ✓
- Efficient symbolic differentiation (Theano computes the gradient) ✓
- Speed and stability optimizations ✓
- Calculations are dynamically mapped to C code
  - We do our computations as fast as we would have written it in C
  - Great performance (>10 faster than Java in my experiments) ✓

Disadvantages

- Debugging is really hard

Some note on the installation of Theano

- Theano utilizes BLAS (Basic Linear Algebra Subprograms)
    - Building blocks for fast vector and matrix operations
    - Often written in Fortran, sometimes in Assembler

- For performance optimization install a BLAS package

- Benchmark different BLAS packages

- I use a manually compiled OpenBlas implementation

    - Installation notes: http://deeplearning.net/software/theano/install_ubuntu.html

Theano – Flow

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- The execution of a Theano script is a bit different

Python: Define a computation graph

↓

Python: Tell Theano to compile the graph

↓

Theano: Optimize the graph, generate C-Code and compile it

↓

Python: Pass input data to the compiled graph

↓

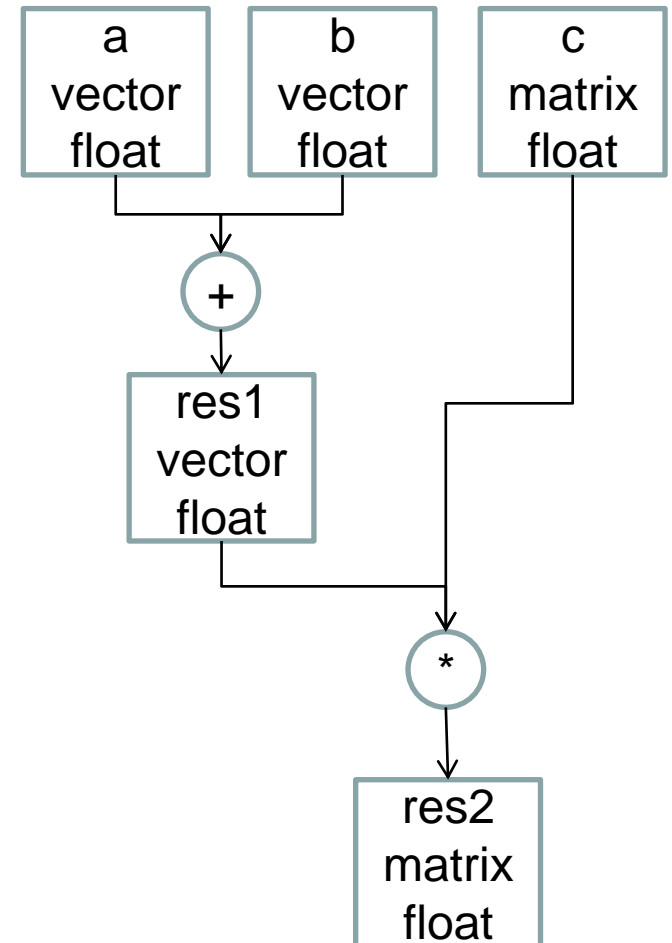| C: Compute the output, compute updates of weights, maybe run on GPU |
| :---: |

↓

Python: Get the final output

Theano – Computation Graph

- You write symbolic expression to define the computation graph

```
import theano
import theano.tensor as T


a = T.fvector()
b = T.fvector()
c = T.fmatrix()


res1 = a+b
res2 = T.dot(res1,c)
```

Introduction to Keras

One of the most popular Deep Learning Framewokrs

Keras

- Builds on top of Theano and TensorFlow
  - You can easily switch the backend without needing to change your code
  - As of Keras 1.1.1, TensorFlow the default backend

- Keras implements tons of useful Neuronal Network Moduls:
  - Feed Fordward Layers, Convolutional Layers, Recurrent Layers
  - Different optimization functions
  - Different loss functions
  - Storage and loading of models
  - ….

- Building your deep learning models becomes quite easy with Keras
  - … at least as long as it is already implemented in Keras