DEEP LEARNING TOOLS and FRAMEWORKS

HAMID PALANGI

DEEP LEARNING GROUP, MICROSOFT RESEARCH AI

REDMOND, WA, USA

November 16, 2017 at IEEE GlobalSIP, Montreal, Canada

Acknowledgements: Xiaodong He, Susan Dumais, Li Deng, Jianfeng Gao, Kenneth Tran, Yelong Shen, Xinying Song, Po-Sen Huang, Paul Smolensky, Z. Jane Wang, Rabab Ward
DEEP LEARNING (DL)

• Is it “always” good to use DL models for my task?
DEEP LEARNING (DL)

- Is it “always” good to use DL models for my task?

No!!!
DEEP LEARNING (DL): SOME EXAMPLES

• Feedforward NN
• Recurrent Neural Network (RNN)

\[
\begin{align*}
    \mathbf{x}(1) & \rightarrow h(1) \rightarrow y(1) \rightarrow U \\
    \mathbf{x}(2) & \rightarrow h(2) \rightarrow y(2) \rightarrow U \\
    & \vdots \ \vdots \ \vdots \\
    \mathbf{x}(t) & \rightarrow h(t) \rightarrow y(t) \rightarrow U \\
    & \vdots \ \vdots \ \vdots \\
    \mathbf{x}(T) & \rightarrow h(T) \rightarrow y(T) \rightarrow U
\end{align*}
\]
DEEP LEARNING (DL): SOME EXAMPLES

• Long Short-Term Memory
  • LSTM
DEEP LEARNING (DL): SOME EXAMPLES

LSTM-DSSM used in Information Retrieval
MANUAL GRADIENT CALCULATION

• Good idea to learn stuff. Bad idea to get the job done ASAP.
Manual gradient calculation & implementation is prone to bugs, make sure to perform the "gradient check"
CPUs and GPUs

Picture from https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/
CPUs and GPUs


2,000 CPUs (16,000 cores) – 600 kWatts - $5,000,000

From https://www.wired.com/2012/06/google-x-neural-network/
CPUs and GPUs

- [2013]

3 GPUs (18,432 cores) – 4 kWatts - $33,000

From https://www.wired.com/2013/06/andrew-ng/
CPUs and GPUs

- GPU acceleration

CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- CUDA is C with a few extensions
  - Use of function type qualifiers (__global__, __device__, __host__) to:
    - Determine if a function is executed on the host (CPU) or device (GPU)
    - Determine if a function is callable from the host or the device
  - Use of variable type qualifiers (__shared__, __device__) to:
    - Determine the memory location of a variable
  - Adding a new directive to:
    - Determine how a “kernel” is executed on the device from the host
  - Using 4 built in variables (gridDim, blockDim, blockIdx, threadIdx.) to:
    - Specify grid dimensions, block dimensions, block indices and thread indices
CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- A little more details

Blocks are independent, they must be able to execute in any order.

Threads in a block can synchronize execution.

Kernels are executed by threads. Each thread has its own ID. Usually many threads execute the same kernel.
**CPU**s and **GPU**s: (A SHORT FLAVOR OF) **CUDA**

- **A little more details**
  - Quite fast, R/W, Only accessed by 1 thread – This is thread space
  - Fast, R/W, Accessed by all threads in a block (16 KB) – This is for thread collaboration
  - Quite fast, R/W, Only accessed by 1 thread
  - Not as fast as local & shared memory, R/W, Accessed by all threads and CPU (4 GB) – Used for IO for Grid
  - R, Accessed by all threads and CPU
  - R, Accessed by all threads and CPU

Picture from [http://gecomin.es.edu/tesla/cuda_tutorial_mio/](http://gecomin.es.edu/tesla/cuda_tutorial_mio/)
CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Workflow [from http://geco.mines.edu/tesla/cuda_tutorial_mio/]

1. Initialize/acquire device (GPU)
2. Memory allocation on device
3. Memory copy: host → device
4. Kernel execution on device
5. Memory copy: device → host
6. Memory deallocation on device
7. Release device
**CPUs and GPUs: (A SHORT FLAVOR OF) CUDA**

- Simple example: adding 2 arrays

```c
// Device code
__global__ void VecAdd(float* A, float* B, float* C, int N)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < N)
        C[i] = A[i] + B[i];
}
```

CPU\textsubscript{s} and GPU\textsubscript{s}: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays

```c
// Host code
int main()
{
    int N = ...;
    size_t size = N * sizeof(float);

    // Allocate input vectors h_A and h_B in host memory
    float* h_A = (float*)malloc(size);
    float* h_B = (float*)malloc(size);

    // Initialize input vectors
    ...

    // Allocate vectors in device memory
    float* d_A;
    cudaMalloc(&d_A, size);
    float* d_B;
    cudaMalloc(&d_B, size);
    float* d_C;
    cudaMalloc(&d_C, size);
```
CPU and GPUs: (A SHORT FLAVOR OF) CUDA

• Simple example: adding 2 arrays

```c
// Copy vectors from host memory to device memory
cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

// Invoke kernel
```
**CPUs and GPUs: (A SHORT FLAVOR OF) CUDA**


```c
int threadsPerBlock = 256;
int blocksPerGrid =
    (N + threadsPerBlock - 1) / threadsPerBlock;
VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
```

```c
// Device code
__global__ void VecAdd(float* A, float* B, float* C, int N)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < N)
        C[i] = A[i] + B[i];
}
```

- Initialize/acquire device (GPU)
- Memory allocation on device
- Memory copy: host → device
- Kernel execution on device
- Memory copy: device → host
- Memory deallocation on device
- Release device
CPUs and GPUs: (A SHORT FLAVOR OF) CUDA


```c
// Copy result from device memory to host memory
// h_C contains the result in host memory
cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
```
CPU and GPUs: (A SHORT FLAVOR OF) CUDA

• Simple example: adding 2 arrays

```c

// Free device memory
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);

// Free host memory
...
```

DL FRAMEWORKS: MICROSOFT COGNITIVE TOOLKIT (CNTK)

• From Microsoft

• Supported interfaces: C#, Python, C++ and Command Line

• High scalability
  • Scales across GPUs & machines

• Very fast for sequential models
  • E.g., RNNs, LSTMs

• No commercial support
DL FRAMEWORKS: TENSORFLOW

• From Google
• Supported interfaces: Python, C++ (and experimental support for Java API: not quite stable yet)
• Capability to run on multiple CPUs / GPUs.
• Computation graph compiles faster than Theano (RIP)
• There is no commercial support
• Creates static graphs
• Not closely similar to numpy
DL FRAMEWORKS: TORCH & PYTORCH

• Torch is Maintained by Facebook/Twitter/Google (DeepMind)

• Supported interfaces for Torch: C, C++, Lua

• PyTorch (open sourced in Jan. 2017 by Facebook) is not a Python binding into a C++ framework, it is built to be integrated in Python. Can be used naturally like numpy, scipy, ...

• PyTorch Tensors can be used either on CPU or GPU, a replacement for numpy to use GPUs

• PyTorch builds NNs dynamically [computation graph built at run-time]:
  • TensorFlow, CNTK, Caffe and Theano (RIP): Build NN & reuse it, if you want to change NN architecture, you should build another NN from scratch [static: computation graph is first “compiled”, and run after that]
  • PyTorch: Uses Reverse-mode auto-differentiation that allows changing NN’s behavior with quite low overhead = high flexibility for research projects

• It is easy to write your own layer types

• There is no commercial support
DL FRAMEWORKS: CAFFE

- From Berkeley Vision and Learning Center (BVLC)
- Supported interfaces: Python, MATLAB, C++, C, Command line
- Quite useful when using CNNs
- Rich set of pre-trained models (Model Zoo)
- Initial focus: Computer Vision
- Drawbacks: Documentation, many dependencies, flexibility (need to code in C++ and cuda for significant modifications to the architecture, e.g., new layers to be run on GPU)
- Appropriate for computer vision production code (robust and fast)
- For initial experiments and exploration use a high level API (e.g., Keras), after that use Caffe for production
- Not appropriate for Recurrent Neural Networks (RNNs)
- No commercial support
DL FRAMEWORKS: CAFFE2

• From Facebook, built on the top of Caffe
• Supported interfaces: Python, C++
• High scaling properties
  • E.g., close to linear scaling with ResNet-50
  • Has made CNNs’ distributed training easier
  • Better memory optimization than Caffe
  • Capability for mixed precision computations
    • float, float16, int8, …
• No commercial support

ImageNet training using 64 NVIDIA Tesla P100 GPUs, 8 servers each one having 8 GPUs
(Figure from: https://devblogs.nvidia.com/parallelforall/caffe2-deep-learning-framework-facebook/ )
DL FRAMEWORKS: THEANO (RIP)

• One of the first DL libraries, from Yoshua Bengio’s lab (MILA)
• Supported interfaces: Python
• Not as scalable as other DL frameworks, e.g., lacks multi-GPU support
• A lot of low level coding should be done when using Theano (there are high level wrappers on the top of Theano though, e.g., Keras and Lasagne)
• Compile time of computation graph is too long sometimes
• On Sept. 28, 2017 MILA announced that it will stop developing Theano (RIP Theano) …
DL FRAMEWORKS: MXNET

• Supported interfaces: Python, C++, R, Julia
• Scalable, can run experiments on multiple GPUs and machines
• Amazon’s “DL framework of choice”
DL FRAMEWORKS: DEEPLEARNING4J (DL4J)

• Developed by Skymind (a San Francisco-based software firm)
• Supported interfaces: Java & Scala, compatible with JVM
• Can be implemented on the top of Big Data tools, e.g., Apache Hadoop and Apache Spark
• Good documentation
DL FRAMEWORKS: BIGDL

- From Intel
- Supported interfaces: Python and Scala
- Distributed DL library for Spark: Can run directly on the top of Apache Hadoop and Spark clusters.
- High scalability
- You can load pretrained Torch / Caffe models into Spark
DL FRAMEWORKS: HIGH LEVEL NN APIs

• An easier way to build your DL models:
  • Keras
    • Supported interface: Python
    • You can build a DL model in a few lines of code.
    • Can use Theano (RIP), TensorFlow, Microsoft Cognitive Toolkit (CNTK), MXNet or DL4j as backend
    • Good documentation

Picture from https://www.datasciencecentral.com/profiles/blogs/search-for-the-fastest-deep-learning-framework-supported-by-keras
DL FRAMEWORKS: HIGH LEVEL NN APIs

• An easier way to build your DL models:
  • Lasagne
    • Supported interface: Python
    • Can only use Theano (RIP) as backend
    • Not as good documentation as Keras

• Note: although these APIs make it easy to build DL models, they might not be as flexible as using the backend DL libraries directly.
DL FRAMEWORKS: A FEW MORE

• Chainer
  • From a Tokyo start up named Preferred Networks
  • Was the only framework for dynamic computation graphs before PyTorch

• DSSTNE
  • From Amazon, written mainly in C++
  • Amazon decided to support MXNet on AWS

• DyNet
  • From CMU, supports dynamic computation graphs
  • Community is not as large as other frameworks

• Gluon
  • From Microsoft & Amazon: High level API on the top of MXNet [Announced Oct. 2017]

• Paddle
  • DL framework from Baidu
DL FRAMEWORKS: BENCHMARKING EFFORTS

- CNTK vs TensorFlow vs Theano vs Torch vs Caffe

Picture from MSR blog, Dec. 7, 2015
DL FRAMEWORKS: BENCHMARKING EFFORTS


### Table 1: Benchmark Results

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Caffe (ms)</th>
<th>CNTK (ms)</th>
<th>MXNET (ms)</th>
<th>TensorFlow (ms)</th>
<th>Torch (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>13.6255</td>
<td>11.2158</td>
<td>11.6721</td>
<td>37.7609</td>
<td>14.2661</td>
</tr>
<tr>
<td>256</td>
<td>36.8153</td>
<td>27.2151</td>
<td>28.3944</td>
<td>103.9600</td>
<td>37.4622</td>
</tr>
<tr>
<td>512</td>
<td>71.1042</td>
<td>64.5481</td>
<td>64.8811</td>
<td>202.3420</td>
<td>72.8714</td>
</tr>
<tr>
<td>1024</td>
<td>140.0072</td>
<td>100.0693</td>
<td>105.9753</td>
<td>398.1143</td>
<td>140.6100</td>
</tr>
<tr>
<td>2048</td>
<td>277.6562</td>
<td>197.7561</td>
<td>212.7002</td>
<td>783.4888</td>
<td>285.1773</td>
</tr>
</tbody>
</table>

### Table 2: Benchmark Results

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Caffe (ms)</th>
<th>CNTK (ms)</th>
<th>MXNET (ms)</th>
<th>TensorFlow (ms)</th>
<th>Torch (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>101.3205</td>
<td>77.1435</td>
<td>50.5369</td>
<td>108.8271</td>
<td>47.5484</td>
</tr>
<tr>
<td>16</td>
<td>109.3049</td>
<td>64.4404</td>
<td>59.2891</td>
<td>123.4180</td>
<td>58.7782</td>
</tr>
<tr>
<td>32</td>
<td>143.9875</td>
<td>81.4070</td>
<td>84.5450</td>
<td>181.1040</td>
<td>90.9350</td>
</tr>
<tr>
<td>64</td>
<td>225.8536</td>
<td>181.3029</td>
<td>147.3745</td>
<td>301.4450</td>
<td>164.7160</td>
</tr>
<tr>
<td>128</td>
<td>382.2126</td>
<td>288.1778</td>
<td>288.3220</td>
<td>663.1900</td>
<td>307.3230</td>
</tr>
</tbody>
</table>
Almost all of them are equally good for feedforward NNs.
DL FRAMEWORKS: BENCHMARKING EFFORTS


Time per mini-batch. Table from Shaohuai et al, 2016.
DL FRAMEWORKS: BENCHMARKING EFFORTS

- TensorFlow vs MXNet on CIFAR-10, 8 GPUs [from https://medium.com/@julsimon/keras-shoot-out-tensorflow-vs-mxnet-51ae2b30a9c0]
• Benchmarking using Keras [graph from https://www.datasciencecentral.com/profiles/blogs/search-for-the-fastest-deep-learning-framework-supported-by-keras]
DL FRAMEWORKS: ON GITHUB

- GitHub star count! [from https://www.cio.com/article/3193689/artificial-intelligence/which-deep-learning-network-is-best-for-you.html]
DL FRAMEWORKS: PYTORCH VS TENSORFLOW

• Community of PyTorch is not as large as TensorFlow
• PyTorch does not have a visualization tool as powerful as Tensorboard in TensorFlow
• PyTorch is better for rapid prototyping for research. TensorFlow is better for large scale deployment
• PyTorch is easier to learn for beginners
• TensorFlow builds computation graphs “statically” but PyTorch does it “dynamically”

```python
for _ in range(N):
    y = torch.matmul(W, y) + bias
```
**DL FRAMEWORKS: PYTORCH VS TENSORFLOW**

- PyTorch code is easier to debug than TensorFlow
- PyTorch is new & does not cover all functionalities yet (e.g., there is no fft in PyTorch yet). Over time with more contributions to PyTorch this gap will be closed …
- TensorFlow is better for deployment
  - Using TensorFlow serialization the whole graph (including parameters and operations) can be saved, and then loaded for inference in other languages like C++ and Java. Quite useful when Python is not an option for deployment.
  - TensorFlow also works for mobile deployments (building mobile apps with TF [https://www.tensorflow.org/mobile/](https://www.tensorflow.org/mobile/)), you don’t need to code the DL architecture again for inference
- TensorFlow assumes you are using GPUs if any available. In PyTorch, everything should be explicitly moved to the device when using GPUs.
- PyTorch also have visualization tools like with similarities to TensorFlow’s Tensorboard
  - Tensorboard_logger: [https://github.com/TeamHG-Memex/tensorboard_logger](https://github.com/TeamHG-Memex/tensorboard_logger)
  - Crayon: [https://github.com/torrvision/crayon](https://github.com/torrvision/crayon)
DL FRAMEWORKS

DL FRAMEWORKS

- TensorFlow: [https://www.tensorflow.org/](https://www.tensorflow.org/)
- Torch: [http://torch.ch/](http://torch.ch/)
- Caffe2: [https://caffe2.ai/](https://caffe2.ai/)
- Theano (RIP): [http://deeplearning.net/software/theano/](http://deeplearning.net/software/theano/)
- Deeplearning4j: [https://deeplearning4j.org/](https://deeplearning4j.org/)
- High level NN APIs:
  - Keras (CNTK, TensorFlow, MXNet, DL4J and Theano backend): [https://keras.io/](https://keras.io/)
- Chainer: [https://chainer.org/](https://chainer.org/)
- DSSTNE: [https://github.com/amzn/amazon-dsstne](https://github.com/amzn/amazon-dsstne)
- DyNet: [https://github.com/clab/dynet](https://github.com/clab/dynet)
- Gluon: [https://github.com/gluon-api/gluon-api/](https://github.com/gluon-api/gluon-api/)
- Paddle: [https://github.com/PaddlePaddle/Paddle](https://github.com/PaddlePaddle/Paddle)
DL FRAMEWORKS

• TensorFlow: https://www.tensorflow.org/
• Torch: http://torch.ch/
• PyTorch: http://pytorch.org/
• Caffe: http://caffe.berkeleyvision.org/
• Caffe2: https://caffe2.ai/
• Theano (RIP): http://deeplearning.net/software/theano/
• MXNet: http://mxnet.incubator.apache.org/
• Deeplearning4j: https://deeplearning4j.org/
• BigDL: https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark
• High level NN APIs:
  • Keras (CNTK, TensorFlow, MXNet, DL4J and Theano backend): https://keras.io/
  • Lasagne (Theano backend) https://lasagne.readthedocs.io/en/latest/
• Chainer: https://chainer.org/
• DSSTNE: https://github.com/amzn/amazon-dsstne
• DyNet: https://github.com/clab/dynet
• Gluon: https://github.com/gluon-api/gluon-api/
• Paddle: https://github.com/PaddlePaddle/Paddle
DL FRAMEWORKS

No! Not Another Deep Learning Framework

Linh Nguyen*
 lvnguyen@umich.edu
 University of Michigan

Peifeng Yu*
 peifeng@umich.edu
 University of Michigan

Mosharaf Chowdhury
 mosharaf@umich.edu
 University of Michigan

DL FRAMEWORKS: WHO ARE YOU?!

- Are you industry?
  - Speed and scale
  - Stability

- DL4J
- Microsoft Cognitive Toolkit (CNTK)
- Caffe
- TensorFlow
- BigDL
- MXNet ?!, Caffe2 ?!, DSSTNE ?!
DL FRAMEWORKS: WHO ARE YOU?!

• Are you a research organization?
  • Flexibility
  • Easy debuggability

- PyTorch & Torch
- Theano (RIP)
- MXNet
- TensorFlow
- Microsoft Cognitive Toolkit (CNTK)
DL FRAMEWORKS: WHO ARE YOU?!

• Are you a DL beginner?
  • Do gradient calculation and backprop manually on paper once to fully understand it
  • Then start with a high level API to train your first DL model

Keras
Lasagne
DL FRAMEWORKS: WHO ARE YOU?!

• Are you a DL practitioner who wants to implement a model ASAP?
  • Use a high level API

Keras
Lasagne
DL FRAMEWORKS: WHO ARE YOU?!

• Are you a university prof planning to use a DL framework for your class?
  • Use an easy to learn framework with fast ramp-up time

  - PyTorch
  - MXNet
  - TensorFlow
  - Microsoft Cognitive Toolkit (CNTK)
DL FRAMEWORKS: WHO ARE YOU?!

• Are you an organization or company that needs commercial support?
DL FRAMEWORKS: WHO ARE YOU?!

• Are you doing computer vision?

- Caffe
- Caffe2
- MXNet
- Torch
- Microsoft Cognitive Toolkit (CNTK)
DL FRAMEWORKS: WHO ARE YOU?!

• Are you using RNNs (LSTMs, GRUs, …) and variable length sequences?

- Microsoft Cognitive Toolkit (CNTK)
- PyTorch
THANK YOU!