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

Microsoft Research AI

DEEP LEARNING (DL)

- Is it "**always**" good to use DL models for my task?

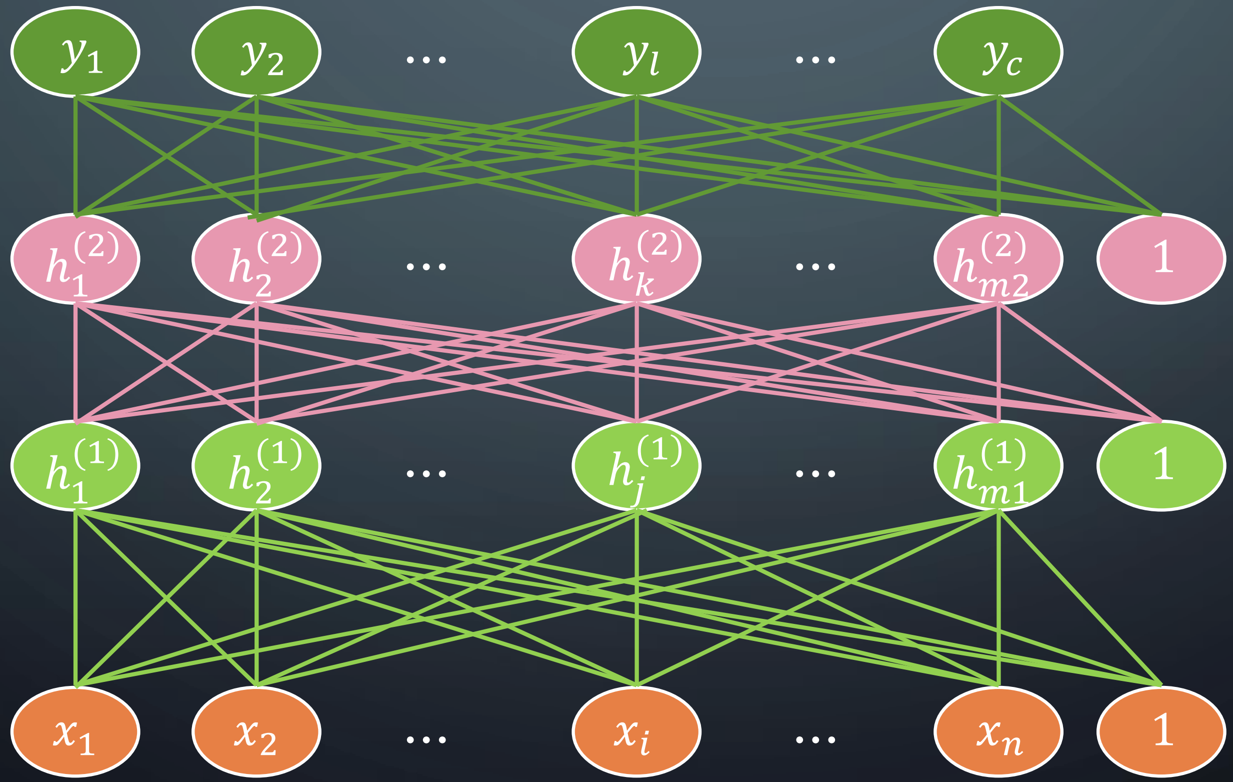
DEEP LEARNING (DL)

- Is it "**<u>always</u>**" good to use DL models for my task?

$\boxed{\text{No!!!}}$

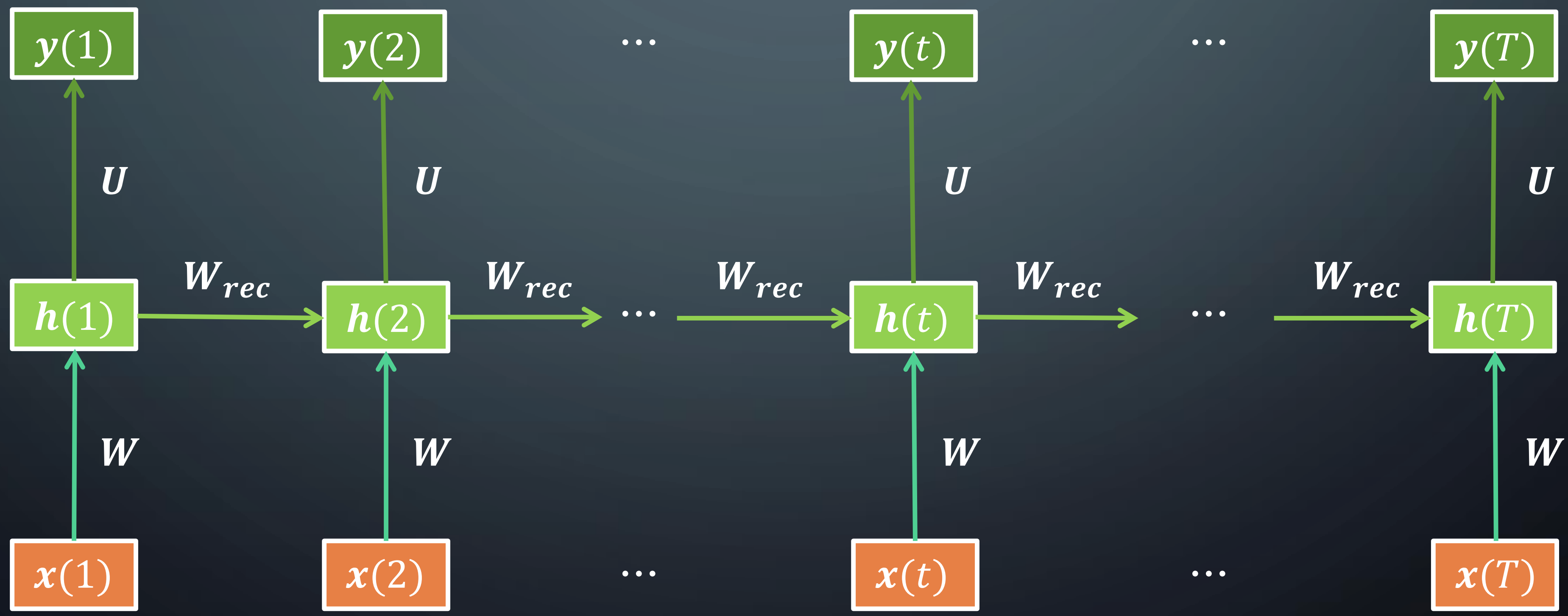
DEEP LEARNING (DL): SOME EXAMPLES

- Feedforward NN



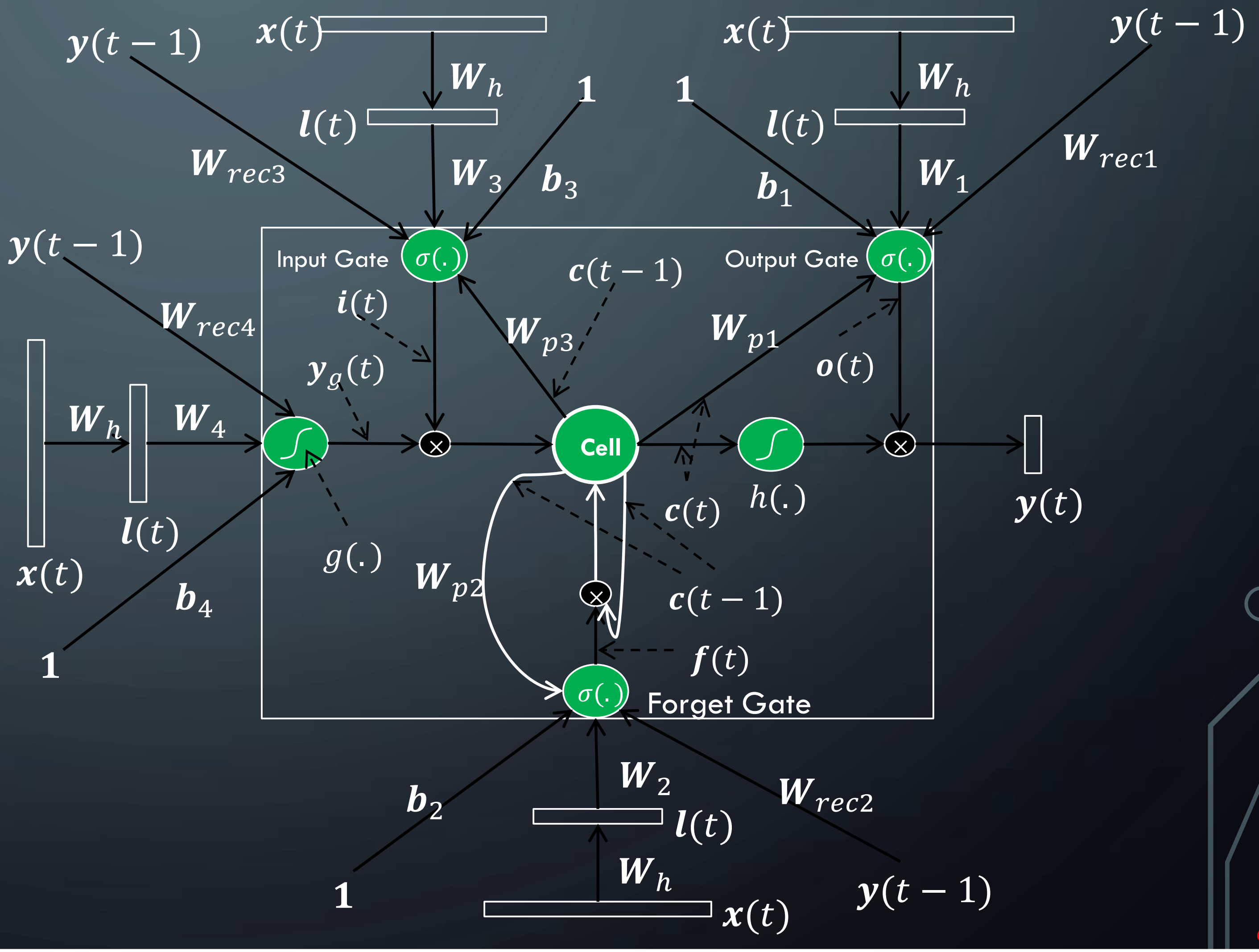
DEEP LEARNING (DL): SOME EXAMPLES

- Recurrent Neural Network (RNN)

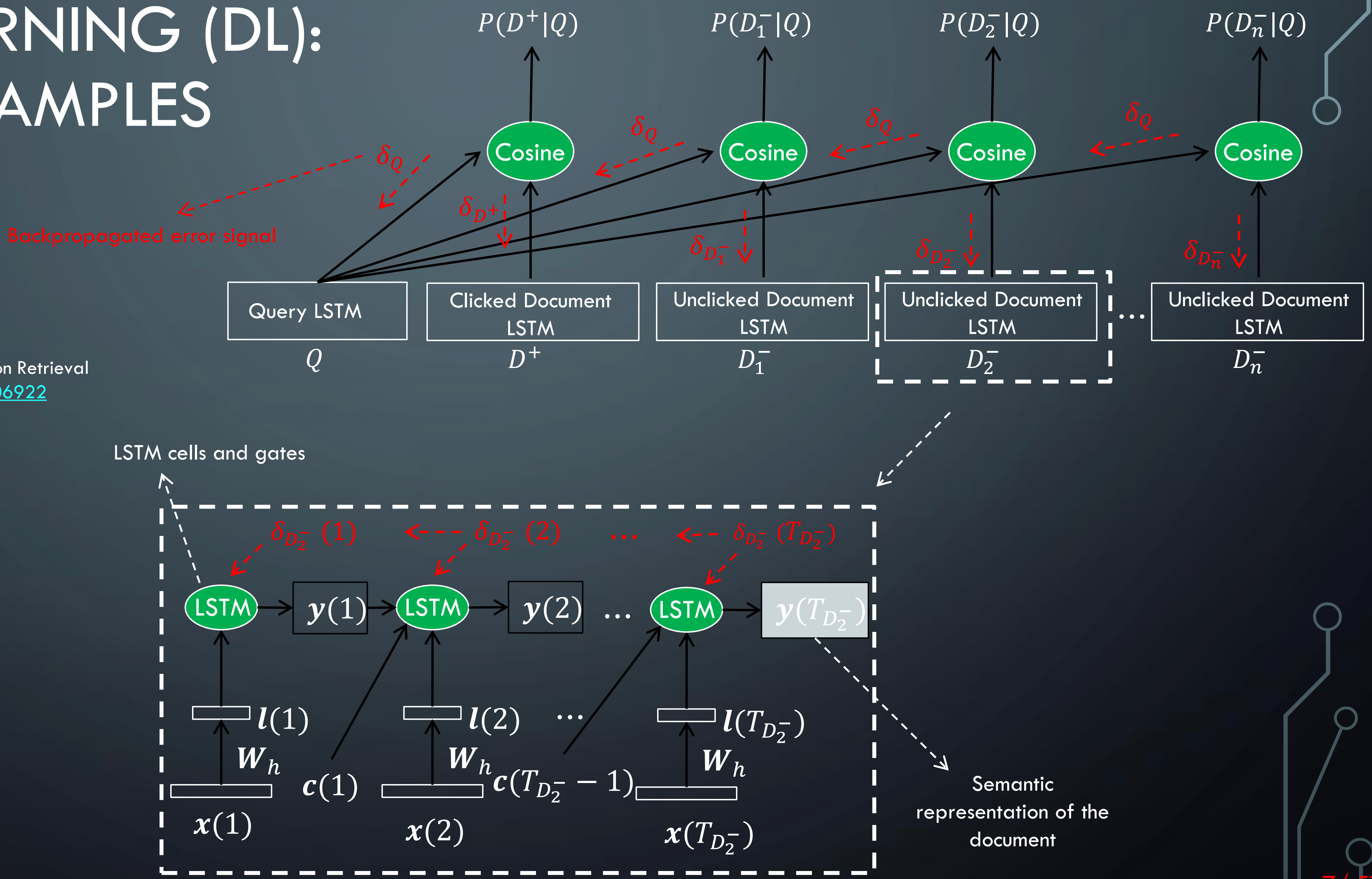


DEEP LEARNING (DL): SOME EXAMPLES

- Long Short-Term Memory
  - LSTM



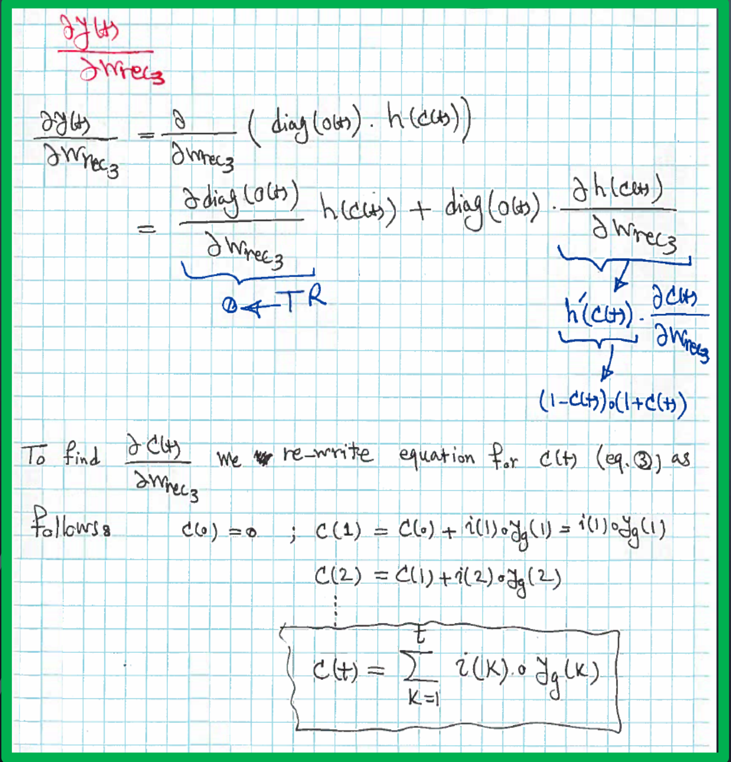
DEEP LEARNING (DL): SOME EXAMPLES



LSTM-DSSM used in Information Retrieval
https://arxiv.org/abs/1502.06922

MANUAL GRADIENT CALCULATION

- Good idea to learn stuff. Bad idea to get the job done ASAP.



$$\frac{\partial y(t)}{\partial W_{rec_3}}$$

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \frac{\partial}{\partial W_{rec_3}}\left(\mathrm{diag}(o(t)) \cdot h(c(t))\right)$$

$$= \underbrace{\frac{\partial\, \mathrm{diag}(o(t))}{\partial W_{rec_3}}}_{0 \leftarrow TR} h(c(t)) + \mathrm{diag}(o(t)) \cdot \underbrace{\frac{\partial h(c(t))}{\partial W_{rec_3}}}_{\underbrace{h'(c(t))}_{(1-c(t))\circ(1+c(t))} \cdot \frac{\partial c(t)}{\partial W_{rec_3}}}$$

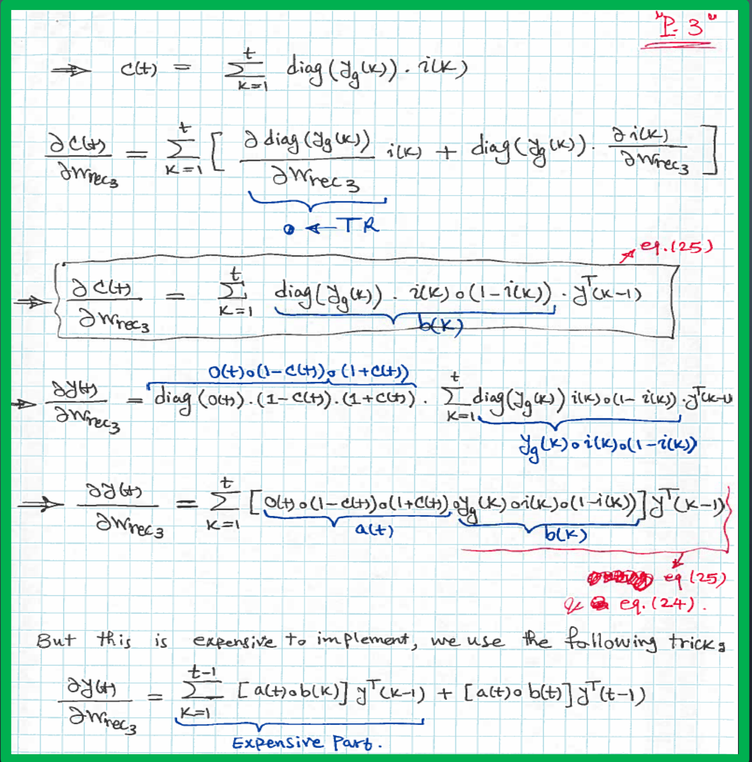
To find $\frac{\partial c(t)}{\partial W_{rec_3}}$ we re-write equation for $c(t)$ (eq. ③) as follows:

$c(0) = 0$; $c(1) = c(0) + i(1)\circ y_g(1) = i(1)\circ y_g(1)$

$c(2) = c(1) + i(2)\circ y_g(2)$

$\vdots$

$$\boxed{c(t) = \sum_{k=1}^{t} i(k)\circ y_g(k)}$$



"P. 3"

$$\Rightarrow c(t) = \sum_{k=1}^{t} \mathrm{diag}(y_g(k)) \cdot i(k)$$

$$\frac{\partial c(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t}\left[\underbrace{\frac{\partial\, \mathrm{diag}(y_g(k))}{\partial W_{rec_3}}}_{0 \leftarrow TR} i(k) + \mathrm{diag}(y_g(k)) \cdot \frac{\partial i(k)}{\partial W_{rec_3}}\right]$$

eq. (25)

$$\Rightarrow \boxed{\frac{\partial c(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t} \underbrace{\mathrm{diag}(y_g(k)) \cdot i(k)\circ(1-i(k))}_{b(k)} \cdot y^T(k-1)}$$

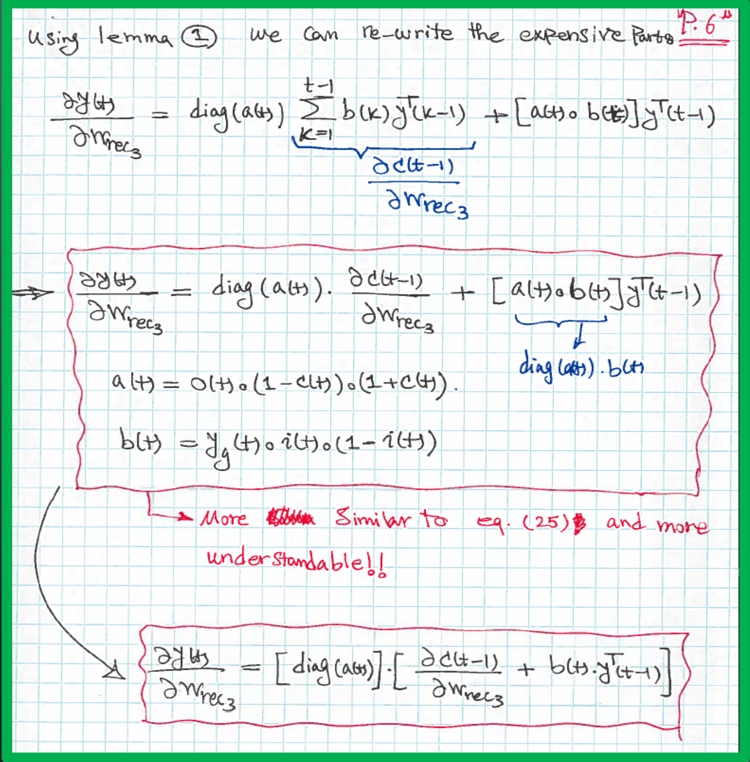
$$\Rightarrow \frac{\partial y(t)}{\partial W_{rec_3}} = \overbrace{\mathrm{diag}(o(t)).(1-c(t)).(1+c(t))}^{o(t)\circ(1-c(t))\circ(1+c(t))} \cdot \sum_{k=1}^{t} \underbrace{\mathrm{diag}(y_g(k))\, i(k)\circ(1-i(k))}_{y_g(k)\circ i(k)\circ(1-i(k))} \cdot y^T(k-1)$$

$$\Rightarrow \frac{\partial y(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t}\left[\underbrace{o(t)\circ(1-c(t))\circ(1+c(t))}_{a(t)} \circ \underbrace{y_g(k)\circ i(k)\circ(1-i(k))}_{b(k)}\right] y^T(k-1)$$

eq (25)
& eq. (24).

But this is expensive to implement, we use the following trick:

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \underbrace{\sum_{k=1}^{t-1} [a(t)\circ b(k)]\, y^T(k-1)}_{\text{Expensive Part.}} + [a(t)\circ b(t)]\, y^T(t-1)$$



"P. 6"

Using lemma ① we can re-write the expensive Part:

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \mathrm{diag}(a(t)) \underbrace{\sum_{k=1}^{t-1} b(k)\, y^T(k-1)}_{\frac{\partial c(t-1)}{\partial W_{rec_3}}} + [a(t)\circ b(t)]\, y^T(t-1)$$

$$\Rightarrow \frac{\partial y(t)}{\partial W_{rec_3}} = \mathrm{diag}(a(t)) \cdot \frac{\partial c(t-1)}{\partial W_{rec_3}} + [\underbrace{a(t)\circ b(t)}_{\mathrm{diag}(a(t)).b(t)}]\, y^T(t-1)$$

$a(t) = o(t)\circ(1-c(t))\circ(1+c(t))$.

$b(t) = y_g(t)\circ i(t)\circ(1-i(t))$

More Similar to eq. (25) and more understandable!!

$$\boxed{\frac{\partial y(t)}{\partial W_{rec_3}} = [\mathrm{diag}(a(t))] \cdot \left[\frac{\partial c(t-1)}{\partial W_{rec_3}} + b(t) \cdot y^T(t-1)\right]}$$

MANUAL GRADIENT CALCULATION

- Manual gradient calculation & implementation is prone to bugs, make sure to perform the "**gradient check**"

$$\frac{\partial y(t)}{\partial W_{rec_3}}$$

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \frac{\partial}{\partial W_{rec_3}}\left(\mathrm{diag}(o(t))\cdot h(c(t))\right)$$

$$= \underbrace{\frac{\partial\,\mathrm{diag}(o(t))}{\partial W_{rec_3}}}_{0 \leftarrow TR} h(c(t)) + \mathrm{diag}(o(t))\cdot \underbrace{\frac{\partial h(c(t))}{\partial W_{rec_3}}}_{\underbrace{h'(c(t))}_{(1-c(t))\circ(1+c(t))}\cdot\frac{\partial c(t)}{\partial W_{rec_3}}}$$

To find $\frac{\partial c(t)}{\partial W_{rec_3}}$ we re-write equation for $c(t)$ (eq. 3) as follows:

$c(0) = 0$; $c(1) = c(0) + i(1)\circ y_g(1) = i(1)\circ y_g(1)$

$c(2) = c(1) + i(2)\circ y_g(2)$

$\vdots$

$$\boxed{c(t) = \sum_{k=1}^{t} i(k)\circ y_g(k)}$$

"P. 3"

$$\Rightarrow c(t) = \sum_{k=1}^{t} \mathrm{diag}(y_g(k))\cdot i(k)$$

$$\frac{\partial c(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t}\left[\underbrace{\frac{\partial\,\mathrm{diag}(y_g(k))}{\partial W_{rec_3}}}_{0 \leftarrow TR} i(k) + \mathrm{diag}(y_g(k))\cdot\frac{\partial i(k)}{\partial W_{rec_3}}\right]$$

eq. (25)

$$\Rightarrow \boxed{\frac{\partial c(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t} \underbrace{\mathrm{diag}(y_g(k))\cdot i(k)\circ(1-i(k))}_{b(k)}\cdot y^T(k-1)}$$

$$\Rightarrow \frac{\partial y(t)}{\partial W_{rec_3}} = \overbrace{\mathrm{diag}(o(t))\cdot(1-c(t))\cdot(1+c(t))}^{o(t)\circ(1-c(t))\circ(1+c(t))}\cdot\sum_{k=1}^{t}\underbrace{\mathrm{diag}(y_g(k))\, i(k)\circ(1-i(k))}_{y_g(k)\circ i(k)\circ(1-i(k))}\cdot y^T(k-1)$$

$$\Rightarrow \frac{\partial y(t)}{\partial W_{rec_3}} = \sum_{k=1}^{t}\left[\underbrace{o(t)\circ(1-c(t))\circ(1+c(t))}_{a(t)}\circ\underbrace{y_g(k)\circ i(k)\circ(1-i(k))}_{b(k)}\right] y^T(k-1)$$

eq (25)
& eq. (24).

But this is expensive to implement, we use the following trick:

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \underbrace{\sum_{k=1}^{t-1}[a(t)\circ b(k)]\, y^T(k-1)}_{\text{Expensive Part.}} + [a(t)\circ b(t)]\, y^T(t-1)$$

Using lemma (1) we can re-write the expensive part: "P. 6"

$$\frac{\partial y(t)}{\partial W_{rec_3}} = \mathrm{diag}(a(t))\underbrace{\sum_{k=1}^{t-1} b(k)\, y^T(k-1)}_{\frac{\partial c(t-1)}{\partial W_{rec_3}}} + [a(t)\circ b(t)]\, y^T(t-1)$$

$$\Rightarrow \boxed{\frac{\partial y(t)}{\partial W_{rec_3}} = \mathrm{diag}(a(t))\cdot\frac{\partial c(t-1)}{\partial W_{rec_3}} + [\underbrace{a(t)\circ b(t)}_{\mathrm{diag}(a(t))\cdot b(t)}]\, y^T(t-1)}$$

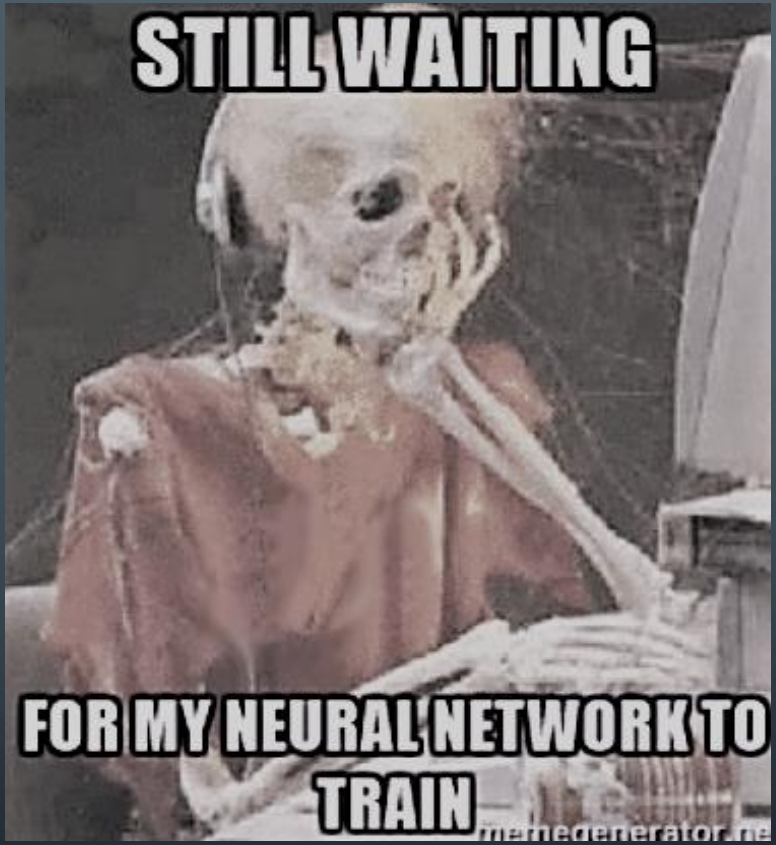
$a(t) = o(t)\circ(1-c(t))\circ(1+c(t))$.

$b(t) = y_g(t)\circ i(t)\circ(1-i(t))$

→ More similar to eq. (25) and more understandable!!

$$\boxed{\frac{\partial y(t)}{\partial W_{rec_3}} = [\mathrm{diag}(a(t))]\cdot\left[\frac{\partial c(t-1)}{\partial W_{rec_3}} + b(t)\cdot y^T(t-1)\right]}$$

CPUs and GPUs



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

CPUs and GPUs

- [2012]

2012

Building High-level Features Using Large Scale Unsupervised Learning

Quoc V. Le — QUOCLE@CS.STANFORD.EDU
Marc'Aurelio Ranzato — RANZATO@GOOGLE.COM
Rajat Monga — RAJATMONGA@GOOGLE.COM
Matthieu Devin — MDEVIN@GOOGLE.COM
Kai Chen — KAICHEN@GOOGLE.COM
Greg S. Corrado — GCORRADO@GOOGLE.COM
Jeff Dean — JEFF@GOOGLE.COM
Andrew Y. Ng — ANG@CS.STANFORD.EDU

From https://arxiv.org/abs/1112.6209

Google Brain

WIRED STAFF SCIENCE 06.26.12 11:15 AM

GOOGLE'S ARTIFICIAL BRAIN LEARNS TO FIND CAT VIDEOS

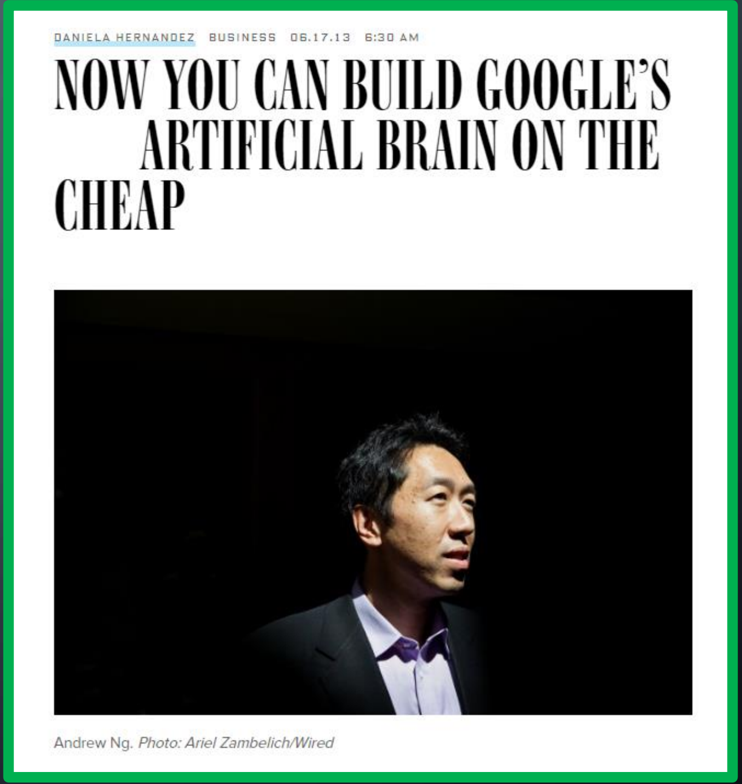
From https://www.wired.com/2012/06/google-x-neural-network/

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

CPUs and GPUs

- [2013]

Stanford AI Lab (Andrew Ng)


DANIELA HERNANDEZ BUSINESS 06.17.13 6:30 AM

NOW YOU CAN BUILD GOOGLE'S ARTIFICIAL BRAIN ON THE CHEAP

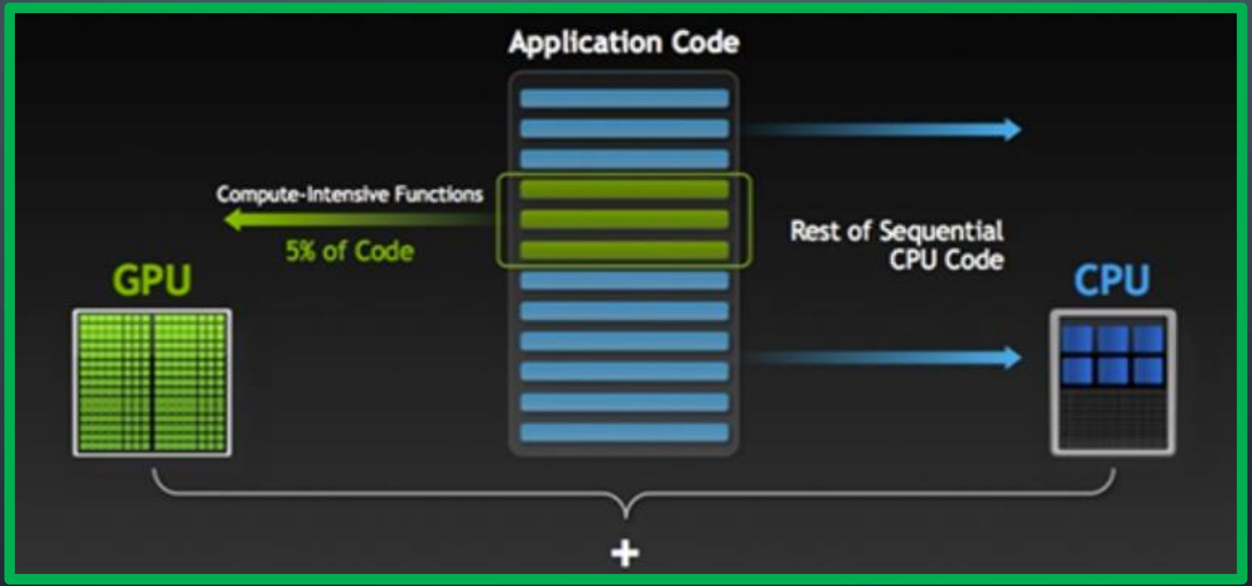
Andrew Ng. *Photo: Ariel Zambelich/Wired*

From https://www.wired.com/2013/06/andrew_ng/

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

CPUs and GPUs

- GPU acceleration



From http://www.nvidia.com/object/what-is-gpu-computing.html

CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

Called & executed by device

Called by host, executed by device

Called & executed by host

Variable in shared memory, has lifetime of block

A function that is executing portion of an application on the device (GPU)

Variable in global memory, has lifetime of the application

- 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

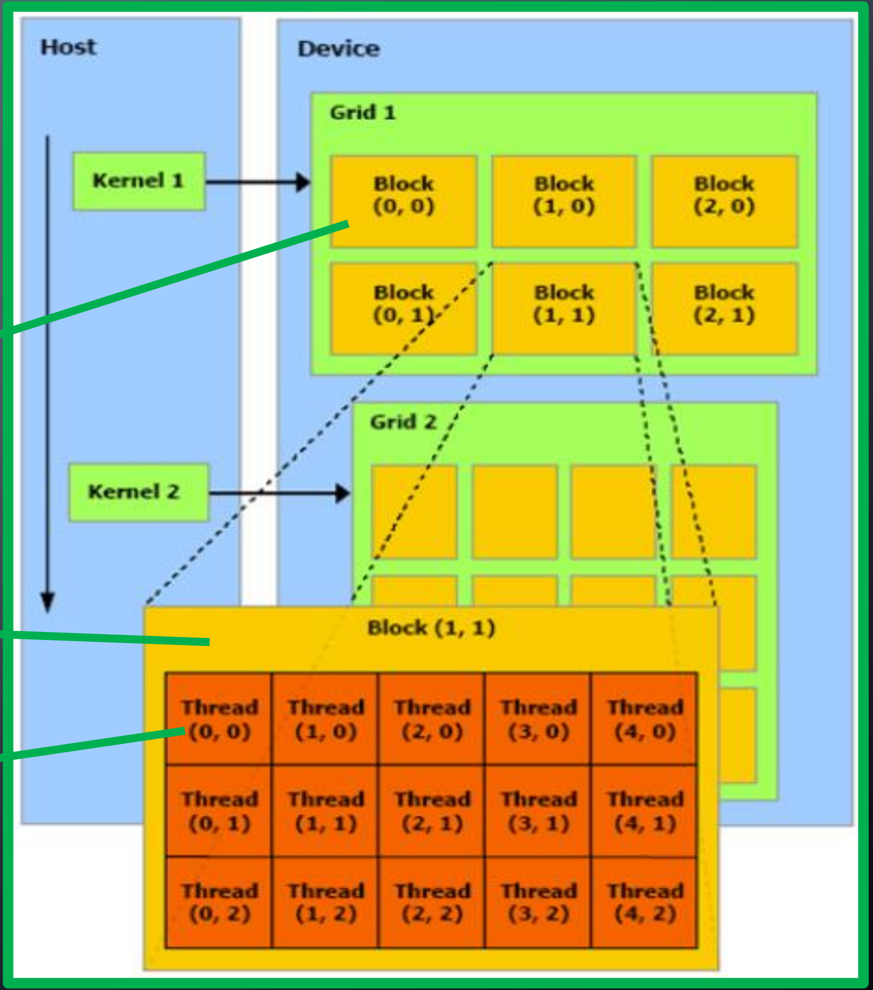
CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- 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



Picture from http://geco.mines.edu/tesla/cuda_tutorial_mio/

CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- A little more details

Quite fast, R/W, Only accessed by 1 thread – This is thread space

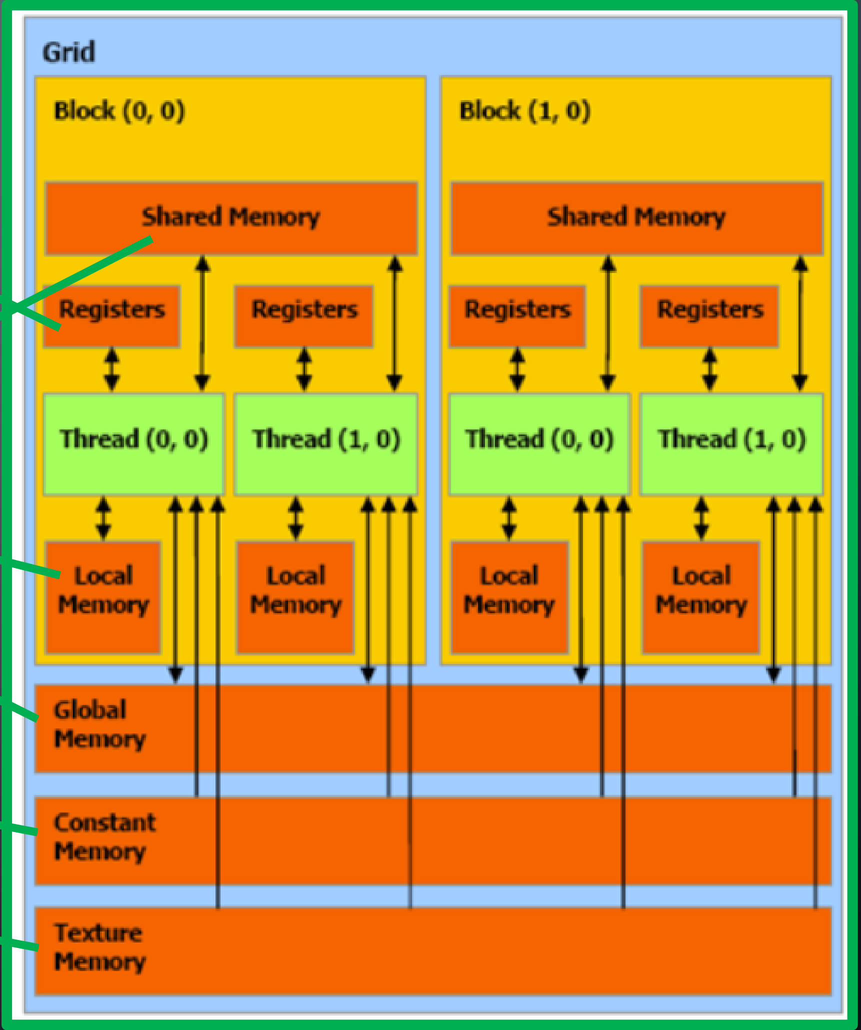
Fast, R/W, Accessed by all threads in a block (16 KB) – This is for thread collaboration

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

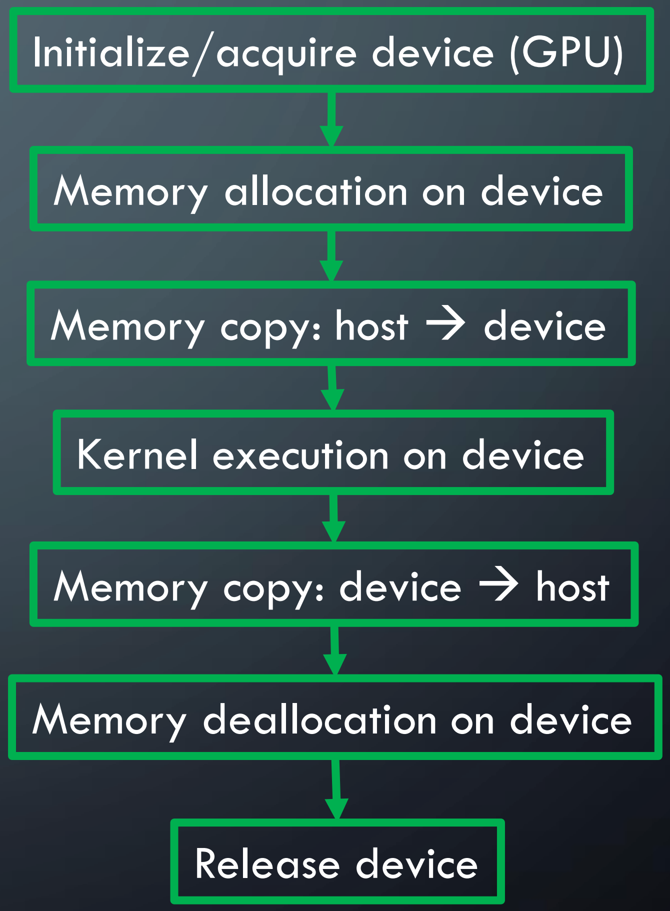
R, Accessed by all threads and CPU



Picture from http://geco.mines.edu/tesla/cuda_tutorial_mio/

CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

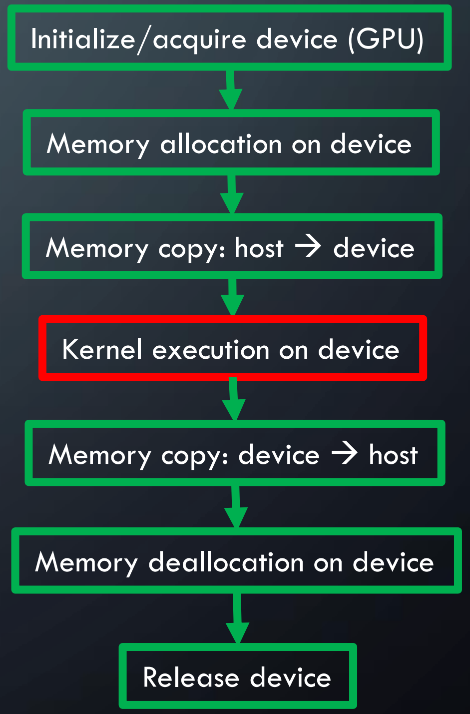
- Workflow [from http://geco.mines.edu/tesla/cuda_tutorial_mio/]



CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

```
// 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];
}
```



CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

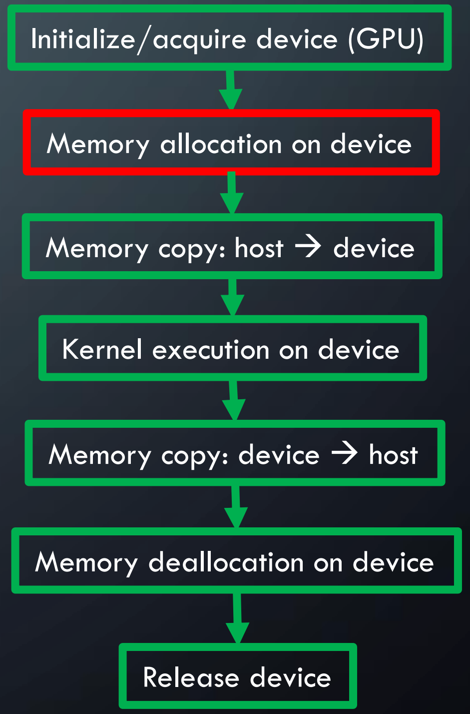
- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

```
// 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);
```

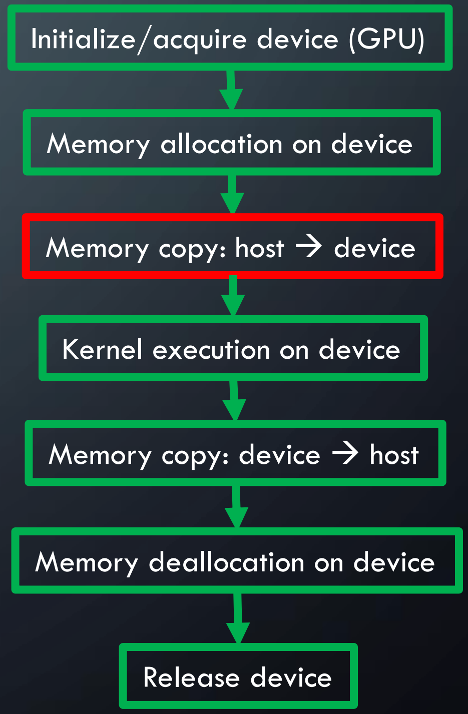


CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

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

// Invoke kernel
```

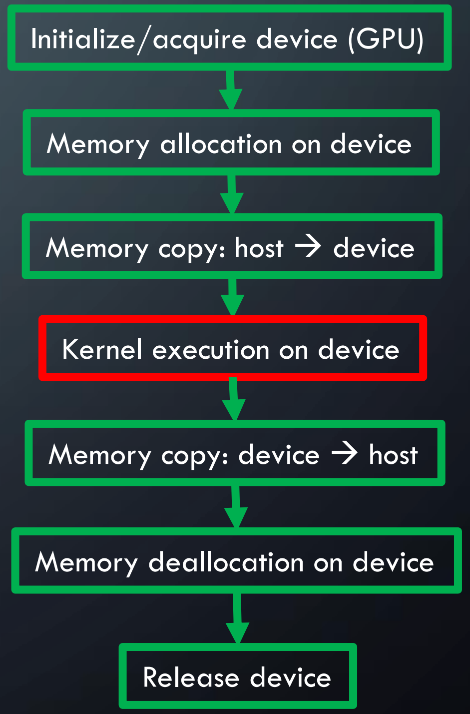


CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

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

```
// 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];
}
```



CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

```
// Copy result from device memory to host memory
// h_C contains the result in host memory
cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
```

Initialize/acquire device (GPU)

Memory allocation on device

Memory copy: host → device

Kernel execution on device

Memory copy: device → host

Memory deallocation on device

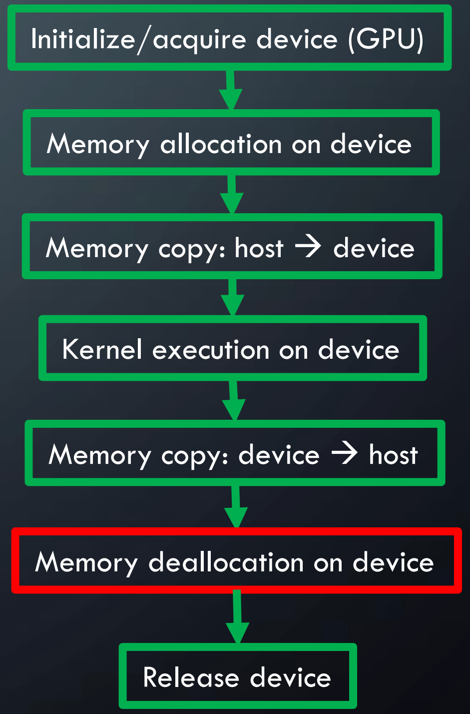
Release device

CPUs and GPUs: (A SHORT FLAVOR OF) CUDA

- Simple example: adding 2 arrays [from https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf]

```
    // 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

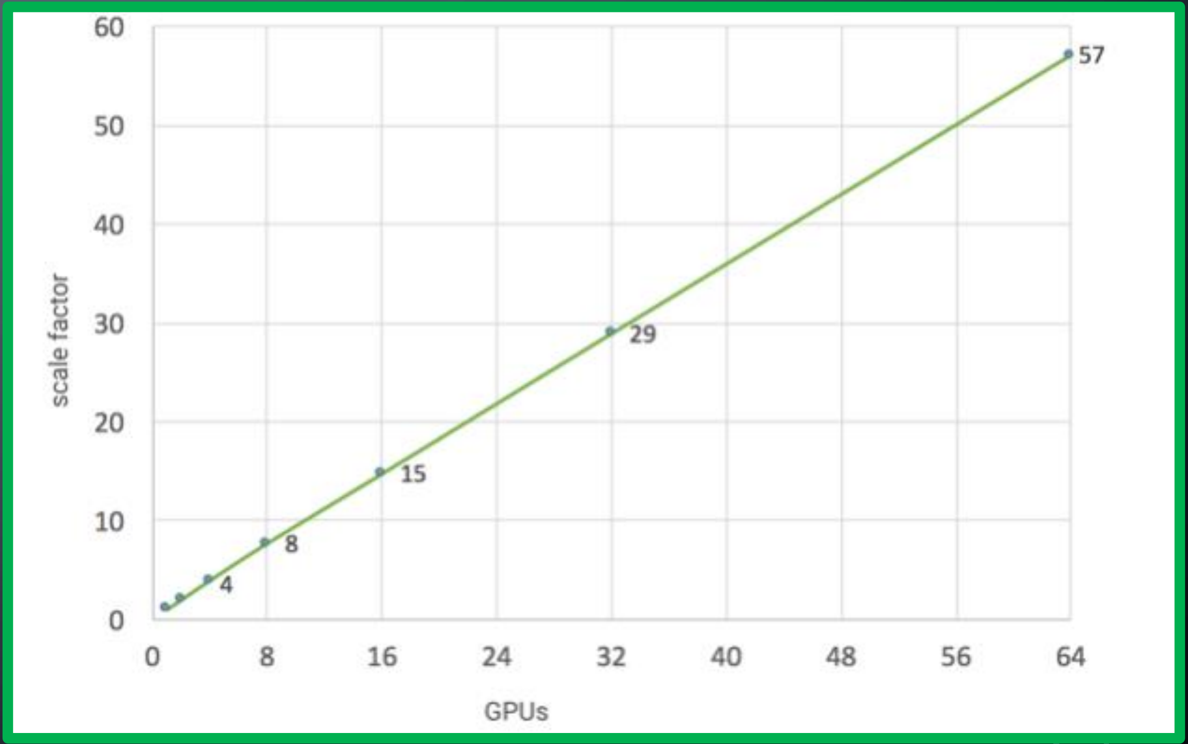
- Torch is Maintained by Facebook/Twitter/Google (DeepMind)
- Supported interfaces for Torch: C, C++, Lua
- PyTorch (open sourced in Jan. 2017 by Facebook) is not a Python binding into a C++ framework, it is built to be integrated in Python. Can be used naturally like numpy, scipy, …
- PyTorch Tensors 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

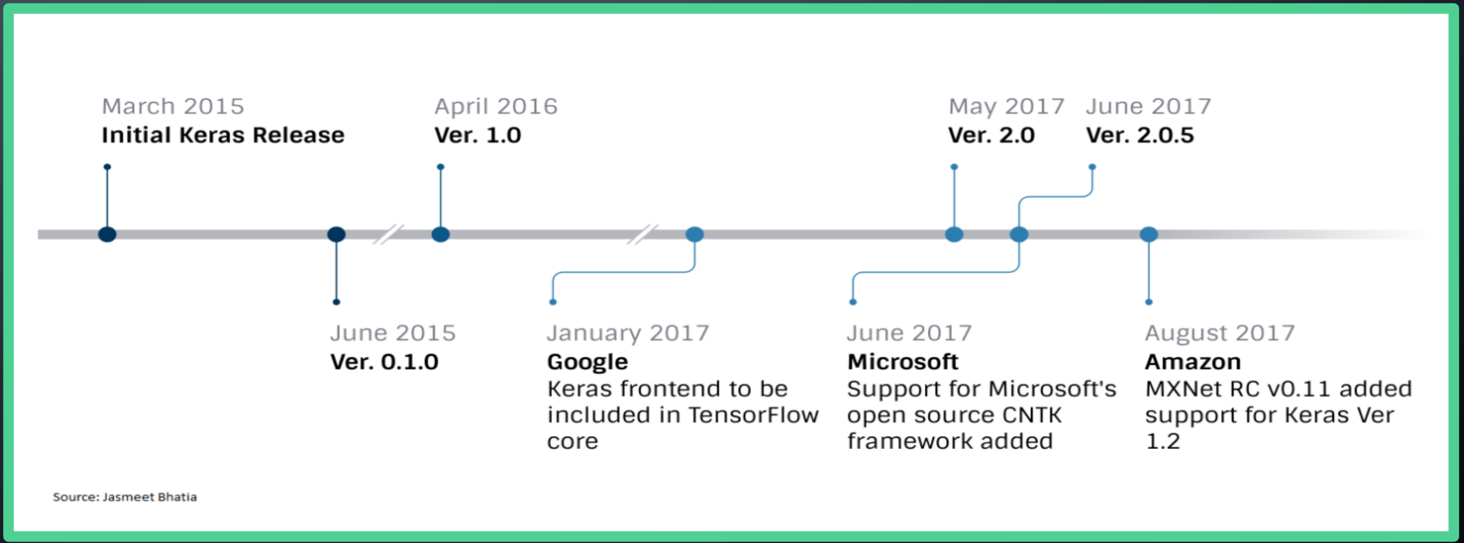
DL FRAMEWORKS: BIGDL

- 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

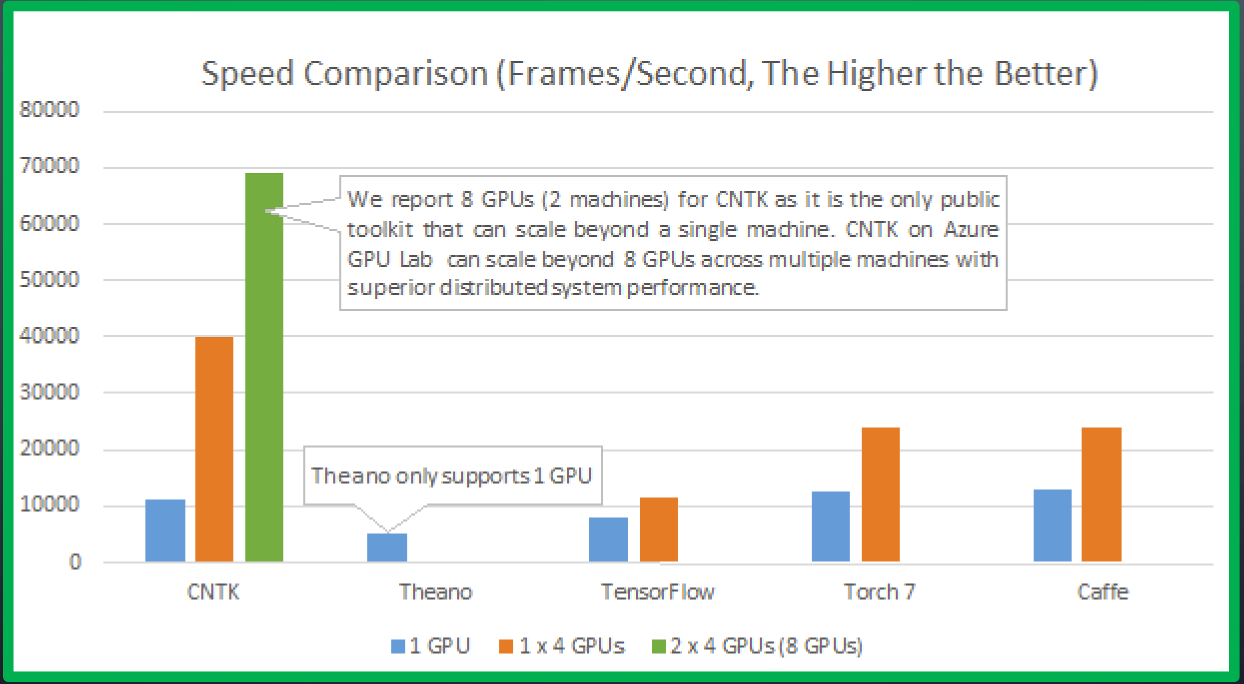
- An easier way to build your DL models:
  - Lasagne
    - Supported interface: Python
    - Can only use Theano (RIP) as backend
    - Not as good documentation as Keras
- Note: although these APIs make it easy to build DL models, they might not be as flexible as using the backend DL libraries directly.

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

- From http://dlbench.comp.hkbu.edu.hk/ and https://github.com/hclhkbu/dlbench

alexnet on K80

Tesla K80, CUDA: 8.0 CUDNN: v5.1 CUDA_DRIVER: 367.48 Network: alexnet

BatchSize	Caffe	CNTK	MXNET	TensorFlow	Torch
86	13.625ms(339.932s)	11.215ms(265.182s)	11.672ms(274.673s)	37.750ms(882.664s)	14.266ms(358.039s)
128	19.511ms(327.894s)	14.883ms(236.889s)	15.496ms(245.240s)	54.314ms(853.009s)	19.924ms(335.840s)
256	36.815ms(311.379s)	27.215ms(217.514s)	28.994ms(229.586s)	103.960ms(818.209s)	37.462ms(315.710s)
512	71.104ms(301.520s)	54.548ms(217.935s)	54.881ms(217.607s)	202.342ms(796.155s)	72.871ms(307.042s)
1024	140.007ms(297.183s)	103.069ms(206.114s)	105.975ms(210.728s)	398.114ms(783.322s)	143.613ms(302.626s)
2048	277.656ms(300.443s)	197.756ms(201.892s)	212.700ms(212.388s)	783.488ms(786.463s)	285.177ms(300.476s)

resnet on K80

Tesla K80, CUDA: 8.0 CUDNN: v5.1 CUDA_DRIVER: 367.48 Network: resnet

BatchSize	Caffe	CNTK	MXNET	TensorFlow	Torch
11	101.328ms(18984.640s)	77.144ms(14049.724s)	50.536ms(9191.549s)	108.827ms(19816.777s)	47.548ms(9141.084s)
16	109.304ms(14227.879s)	54.484ms(6831.130s)	59.298ms(7415.866s)	123.418ms(15453.923s)	58.778ms(7794.904s)
32	143.987ms(9561.758s)	81.470ms(5112.876s)	84.545ms(5287.561s)	181.404ms(11365.723s)	90.935ms(6021.554s)
64	225.325ms(7613.368s)	181.920ms(5710.661s)	147.274ms(4605.828s)	301.445ms(9453.074s)	164.761ms(5455.466s)
128	383.212ms(6553.839s)	288.178ms(2496.103s)	266.322ms(4164.944s)	563.190ms(8830.668s)	307.323ms(5079.835s)

DL FRAMEWORKS: BENCHMARKING EFFORTS

- From http://dlbench.comp.hkbu.edu.hk/ and https://github.com/hclhkbu/dlbench

Item in cell: batchTime(totalTime)

fcn5 on K80

Almost all of them are equally good for feedforward NNs

Tesla K80, CUDA: 8.0 CUDNN: v5.1 CUDA_DRIVER: 367.48 Network: fcn5

BatchSize	Caffe	CNTK	MXNET	TensorFlow	Torch
342	25.271ms(199.234s)	23.838ms(253.527s)	30.711ms(218.684s)	28.622ms(213.854s)	24.435ms(186.543s)
512	31.956ms(172.195s)	30.047ms(227.684s)	38.298ms(182.741s)	37.560ms(190.068s)	30.554ms(157.515s)
1024	55.329ms(151.994s)	51.038ms(205.680s)	60.448ms(144.837s)	62.044ms(159.047s)	52.154ms(135.299s)
2048	98.740ms(140.120s)	92.788ms(197.777s)	104.051ms(125.150s)	111.653ms(145.926s)	90.818ms(118.498s)
4096	184.107ms(132.149s)	175.332ms(190.883s)	182.601ms(110.287s)	211.366ms(138.959s)	168.122ms(110.418s)

lstm on K80

Tesla K80, CUDA: 8.0 CUDNN: v5.1 CUDA_DRIVER: 367.48 Network: lstm

BatchSize	CNTK	MXNET	TensorFlow	Torch
64	40.967ms(11922.492s)	87.171ms(1674.387s)	--	287.026ms(3412.095s)
128	42.736ms(6230.025s)	151.189ms(1451.132s)	--	565.879ms(3388.249s)
256	43.581ms(3187.478s)	288.142ms(1381.371s)	--	1130.606ms(3408.770s)
512	49.712ms(1826.834s)	560.380ms(1344.809s)	--	2312.802ms(3547.659s)
1024	63.695ms(1177.922s)	1101.696ms(1303.797s)	--	5073.561ms(4061.392s)

DL FRAMEWORKS: BENCHMARKING EFFORTS

- Shaohuai Shi et al, IEEE CCBD 2016 [https://arxiv.org/pdf/1608.07249v7.pdf]

Time per mini-batch. Table from Shaohuai et al, 2016

		Desktop CPU (Threads used)				Server CPU (Threads used)						Single GPU		
		1	2	4	8	1	2	4	8	16	32	G980	G1080	K80
FCN-S	Caffe	1.324	0.790	**0.578**	15.444	1.355	0.997	0.745	**0.573**	0.608	1.130	0.041	**0.030**	0.071
	CNTK	1.227	0.660	**0.435**	-	1.340	0.909	0.634	0.488	**0.441**	1.000	0.045	**0.033**	0.074
	TF	7.062	4.789	2.648	**1.938**	9.571	6.569	3.399	1.710	0.946	**0.630**	0.060	**0.048**	0.109
	MXNet	4.621	2.607	2.162	**1.831**	5.824	3.356	2.395	2.040	**1.945**	2.670	-	**0.106**	0.216
	Torch	1.329	0.710	**0.423**	-	1.279	1.131	0.595	0.433	**0.382**	1.034	0.040	**0.031**	0.070
AlexNet-S	Caffe	1.606	0.999	**0.719**	-	1.533	1.045	**0.797**	0.850	0.903	1.124	0.034	**0.021**	0.073
	CNTK	3.761	1.974	**1.276**	-	3.852	2.600	1.567	1.347	**1.168**	1.579	0.045	**0.032**	0.091
	TF	6.525	2.936	1.749	**1.535**	5.741	4.216	2.202	1.160	**0.701**	0.962	0.059	**0.042**	0.130
	MXNet	2.977	2.340	2.250	**2.163**	3.518	3.203	2.926	2.828	**2.827**	2.887	0.020	**0.014**	0.042
	Torch	4.645	2.429	**1.424**	-	4.336	2.468	1.543	1.248	**1.090**	1.214	0.033	**0.023**	0.070
RenNet-50	Caffe	11.554	7.671	**5.652**	-	10.643	8.600	6.723	**6.019**	6.654	8.220	-	**0.254**	0.766
	CNTK	-	-	-	-	-	-	-	-	-	-	0.240	**0.168**	0.638
	TF	23.905	16.435	10.206	**7.816**	29.960	21.846	11.512	6.294	**4.130**	4.351	0.327	**0.227**	0.702
	MXNet	48.000	46.154	44.444	**43.243**	57.831	57.143	54.545	54.545	**53.333**	55.172	0.207	**0.136**	0.449
	Torch	13.178	7.500	**4.736**	4.948	12.807	8.391	5.471	4.164	**3.683**	4.422	0.208	**0.144**	0.523
FCN-R	Caffe	2.476	1.499	**1.149**	-	2.282	1.748	1.403	1.211	1.127	**1.127**	0.025	**0.017**	0.055
	CNTK	1.845	0.970	0.661	**0.571**	1.592	0.857	0.501	0.323	**0.252**	0.280	0.025	**0.017**	0.053
	TF	2.647	1.913	1.157	**0.919**	3.410	2.541	1.297	0.661	0.361	**0.325**	0.033	**0.020**	0.063
	MXNet	1.914	1.072	0.719	**0.702**	1.609	1.065	0.731	0.534	0.451	**0.447**	0.029	**0.019**	0.060
	Torch	1.670	0.926	**0.565**	0.611	1.379	0.915	0.662	0.440	0.402	**0.366**	0.025	**0.016**	0.051
AlexNet-R	Caffe	3.558	2.587	**2.157**	2.963	4.270	3.514	3.381	**3.364**	4.139	4.930	0.041	**0.027**	0.137
	CNTK	9.956	7.263	**5.519**	6.015	9.381	6.078	4.984	**4.765**	6.256	6.199	0.045	**0.031**	0.108
	TF	4.535	3.225	1.911	**1.565**	6.124	4.229	2.200	1.396	1.036	**0.971**	**0.227**	0.317	0.385
	MXNet	13.401	12.305	12.278	**11.950**	17.994	17.128	16.764	**16.471**	17.471	17.770	0.060	**0.032**	0.122
	Torch	5.352	3.866	**3.162**	3.259	6.554	5.288	4.365	**3.940**	4.157	4.165	0.069	**0.043**	0.141
RenNet-56	Caffe	6.741	5.451	**4.989**	6.691	7.513	**6.119**	6.232	6.689	7.313	9.302	-	**0.116**	0.378
	CNTK	-	-	-	-	-	-	-	-	-	-	0.206	**0.138**	0.562
	TF	-	-	-	-	-	-	-	-	-	-	0.225	**0.152**	0.523
	MXNet	34.409	31.255	**30.069**	31.388	44.878	43.775	**42.299**	42.965	43.854	44.367	0.105	**0.074**	0.270
	Torch	5.758	3.222	**2.368**	2.475	8.691	4.965	3.040	**2.560**	2.575	2.811	0.150	**0.101**	0.301
LSTM	Caffe	-	-	-	-	-	-	-	-	-	-	-	-	-
	CNTK	0.186	0.120	**0.090**	0.118	0.211	0.139	0.117	0.114	**0.114**	0.198	0.018	**0.017**	0.043
	TF	4.662	3.385	1.935	**1.532**	6.449	4.351	2.238	1.183	0.702	**0.598**	0.133	**0.065**	0.140
	MXNet	-	-	-	-	-	-	-	-	-	-	0.089	**0.079**	0.149
	Torch	6.921	3.831	**2.682**	3.127	7.471	4.641	3.580	**3.260**	5.148	5.851	0.399	**0.324**	0.560

DL FRAMEWORKS: BENCHMARKING EFFORTS

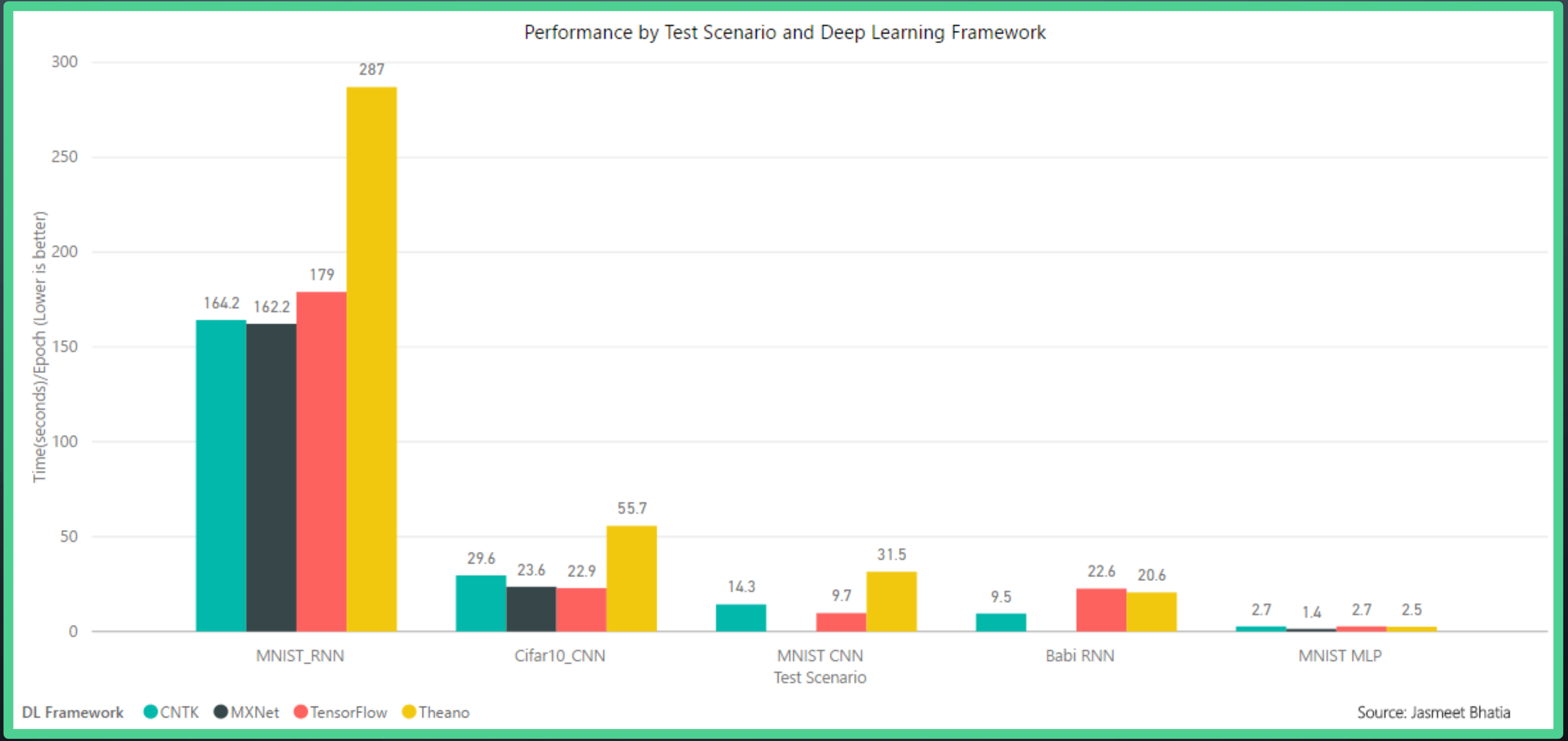
- TensorFlow vs MXNet on CIFAR-10, 8 GPUs [from https://medium.com/@julsimon/keras-shoot-out-tensorflow-vs-mxnet-51ae2b30a9c0]

```
Using MXNet backend.
X_train shape: (50000, 3, 32, 32)
50000 train samples
10000 test samples
[output removed]
Epoch 2/100
50000/50000 [==============================] - 25s - loss: 6.7589 - acc: 0.4978 - val_loss: 7.0586
Epoch 3/100
50000/50000 [==============================] - 25s - loss: 6.4458 - acc: 0.5799 - val_loss: 7.2990
Epoch 4/100
50000/50000 [==============================] - 25s - loss: 6.2015 - acc: 0.6478 - val_loss: 7.3075
Epoch 5/100
50000/50000 [==============================] - 25s - loss: 5.9922 - acc: 0.7095 - val_loss: 7.1960
Epoch 6/100
50000/50000 [==============================] - 25s - loss: 5.8184 - acc: 0.7620 - val_loss: 6.9669
Epoch 7/100
50000/50000 [==============================] - 25s - loss: 5.6454 - acc: 0.8120 - val_loss: 7.3894
Epoch 8/100
50000/50000 [==============================] - 25s - loss: 5.5012 - acc: 0.8554 - val_loss: 7.2800
Epoch 9/100
50000/50000 [==============================] - 25s - loss: 5.3884 - acc: 0.8847 - val_loss: 7.0814
Epoch 10/100
50000/50000 [==============================] - 25s - loss: 5.2744 - acc: 0.9147 - val_loss: 7.2281
```

```
Using TensorFlow backend.
X_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
[output removed]
2017-09-03 13:32:03.432572: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432584: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432600: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432605: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432611: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432616: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432620: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
2017-09-03 13:32:03.432629: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating Tenso
[output removed]
Epoch 2/100
50000/50000 [==============================] - 61s - loss: 6.8205 - acc: 0.4575 - val_loss: 6.8522
Epoch 3/100
50000/50000 [==============================] - 61s - loss: 6.6435 - acc: 0.5241 - val_loss: 6.7707
Epoch 4/100
50000/50000 [==============================] - 61s - loss: 6.5110 - acc: 0.5736 - val_loss: 6.7631
Epoch 5/100
50000/50000 [==============================] - 61s - loss: 6.3973 - acc: 0.6166 - val_loss: 6.6761
Epoch 6/100
50000/50000 [==============================] - 61s - loss: 6.2915 - acc: 0.6574 - val_loss: 6.6398
Epoch 7/100
50000/50000 [==============================] - 61s - loss: 6.1942 - acc: 0.6957 - val_loss: 6.6293
Epoch 8/100
50000/50000 [==============================] - 61s - loss: 6.1005 - acc: 0.7357 - val_loss: 6.6153
Epoch 9/100
50000/50000 [==============================] - 61s - loss: 6.0076 - acc: 0.7704 - val_loss: 6.6278
Epoch 10/100
50000/50000 [==============================] - 61s - loss: 5.9199 - acc: 0.8050 - val_loss: 6.7057
```

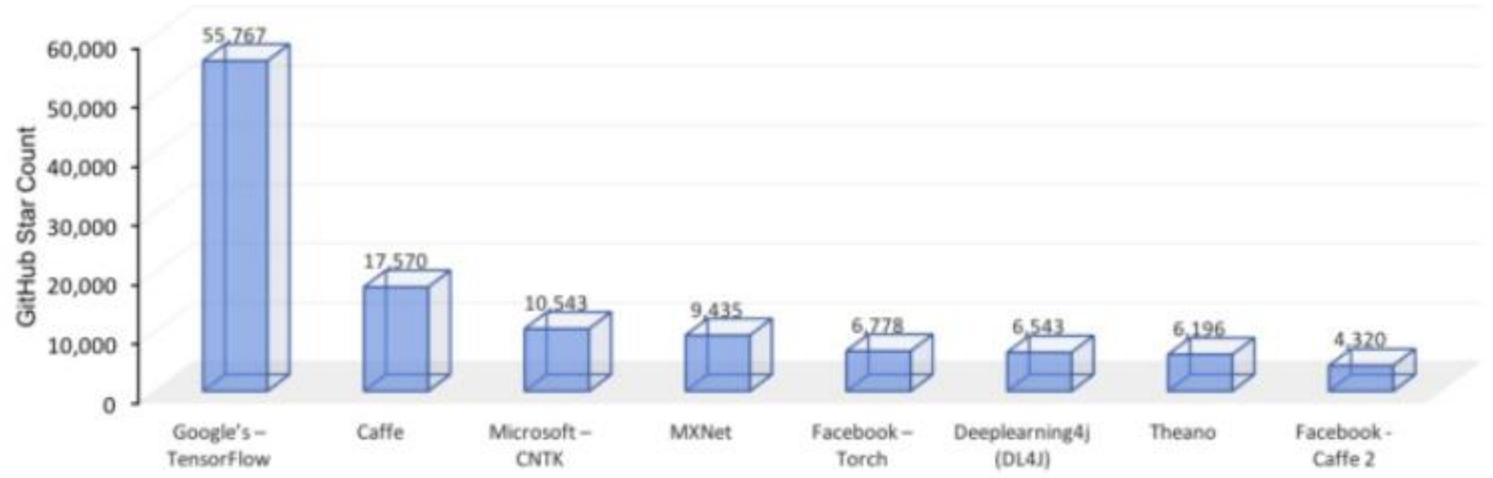
DL FRAMEWORKS: BENCHMARKING EFFORTS

- 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”

```
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 serialaization 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/), 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
  - Crayon: https://github.com/torrvision/crayon

DL FRAMEWORKS

- From https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Mac OS X, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	On roadmap[8]	Yes[9][10]	Computational Graph	Yes[11]	Yes	Yes	Yes	Yes[12]
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python, R	Only if using Theano or MXNet as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[13]	Yes	Yes	Yes	Yes[14]
MXNet	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, Mac OS X, Windows,[25][26] AWS, Android,[27] iOS, JavaScript[28]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[29]	Yes	Yes[30]	Yes[31]	Yes	Yes	Yes	Yes[32]
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	Yes	Yes	Yes	Yes	Yes
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, Mac OS X, Windows[33]	C++, Python	Python (Keras), C/C++, Java, Go, R[34]	No	On roadmap[35] but already with SYCL[36] support	Yes	Yes[37]	Yes[38]	Yes	Yes	Yes	Yes
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development[39]	Yes	Yes[40][41]	Through Lasagne's model zoo[42]	Yes	Yes	Yes	Yes[43]
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, Mac OS X, Windows,[44] Android,[45] iOS	C, Lua	Lua, LuaJIT,[46] C, utility library for C++/OpenCL[47]	Yes	Third party implementations[48][49]	Yes[50][51]	Through Twitter's Autograd[52]	Yes[53]	Yes	Yes	Yes	Yes[54]
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, Mac OS X, Linux, Cloud computing	C++	Wolfram Language	No	No	Yes	Yes	Yes[55]	Yes	Yes	Yes	Yes
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[16]	Yes	Windows, Linux[15] (OSX via Docker on roadmap)	C++	Python (Keras), C++, Command line,[17] BrainScript[18] (.NET on roadmap[19])	Yes[20]	No	Yes	Yes	Yes[21]	Yes[22]	Yes[22]	No[23]	Yes[24]
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, Mac OS X, Windows[2]	C++	Python, MATLAB	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?
Caffe2	Facebook	Apache 2.0	Yes	Linux, Mac OS X, Windows[5]	C++, Python	Python, MATLAB	Yes	Under development[6]	Yes	Yes	Yes[7]	Yes	Yes	No	Yes
MatConvNet	Andrea Vedaldi, Karel Lenc	BSD license	Yes	Windows, Linux[15] (OSX via Docker on roadmap)	C++	MATLAB, C++,	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Neural Designer	Artelnics	Proprietary	No	Linux, Mac OS X, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?
Gensim															
Paddle															
Pytorch															

DL FRAMEWORKS

- Microsoft Cognitive Toolkit (CNTK): https://www.microsoft.com/en-us/cognitive-toolkit/
- 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

- Microsoft Cognitive Toolkit (CNTK): https://www.microsoft.com/en-us/cognitive-toolkit/
- 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

Now What?!

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

Picture from http://www.mosharaf.com/wp-content/uploads/deepstack-hotos17.pdf

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

$\boxed{\text{PyTorch}}$

$\boxed{\text{MXNet}}$

$\boxed{\text{TensorFlow}}$

$\boxed{\text{Microsoft Cognitive Toolkit (CNTK)}}$

DL FRAMEWORKS: WHO ARE YOU?!

- Are you an organization or company that needs commercial support?

DL4J

DL FRAMEWORKS: WHO ARE YOU?!

- Are you doing computer vision?

$\boxed{\text{Caffe}}$

$\boxed{\text{Caffe2}}$

$\boxed{\text{MXNet}}$

$\boxed{\text{Torch}}$

$\boxed{\text{Microsoft Cognitive Toolkit (CNTK)}}$

DL FRAMEWORKS: WHO ARE YOU?!

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

$\boxed{\text{Microsoft Cognitive Toolkit (CNTK)}}$

$\boxed{\text{PyTorch}}$

THANK YOU!