The Anatomy of Deep Learning Frameworks*

***Everything you wanted to know about DL Frameworks but were afraid to ask**

$whoami

- Master's Student in CS @ ETH Zurich, bachelors from BITS Pilani
- Contributor to Theano
- Working on DL for Astronomy and GANs
- space.ml, Anatomy of Deep Learning Frameworks

What’s this talk about?

- Understanding the internals of DL frameworks
- Common components of all DLFs
- Old wine in a new bottle
- How you could DIY

Why do you need to know this?

- All DL work done in DLFs, no vanilla code
- TF, Keras, Theano backbone of DL research
- Any sufficiently advanced technology is indistinguishable from magic --Arthur C. Clarke
- Not voodoo
- Simple concepts, complex implementation details
- Once you know it, you get more control

Frameworks Galore

Many names, same concepts

- Why do we have so many frameworks?
  - Because, why not?
  - Theano – MILA, Torch – FB, CNTK – MS, TF – Google...
  - Does something well
- Are they all really that different?
  - NO
  - We'll cover this in the talk

Bare-bones DL Framework

- Components of any DL framework
  - Tensors
  - Operations
  - Computation Graph
  - Auto-differentiation
  - Fast and Efficient floating pt. Operations, GPU support
    - BLAS, cuBLAS, cuDNN

Example in TensorFlow

- Tensor: tf.Tensor
- Ops: tf.Operation
- Graph: tf.Graph
- Autodiff: tf.gradients et al.
- CuDNN, BLAS – See install notes

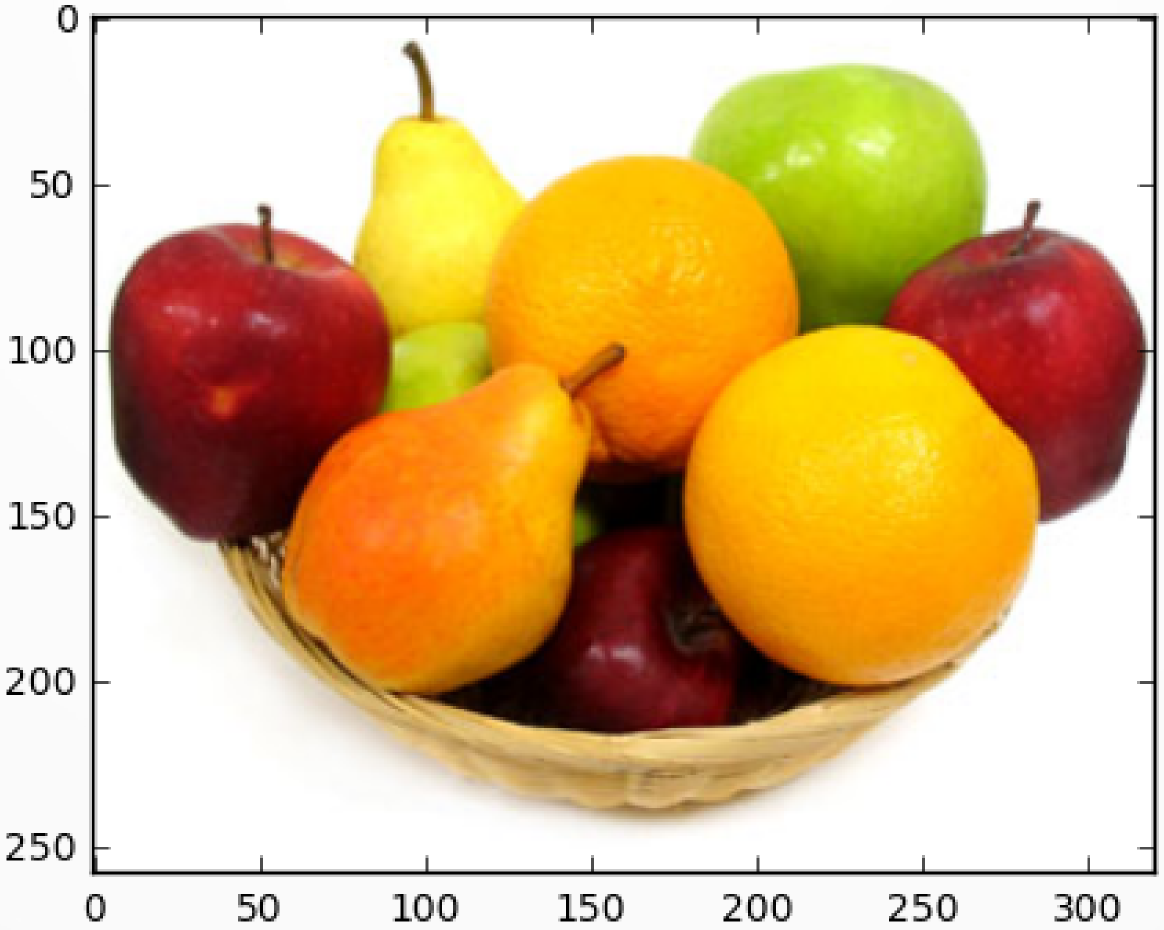
Example in Theano

- Tensor: theano.tensor
- Ops: theano.*
- Graph: theano.gof.graph
- Autodiff: theano.tensor.grad
- CuDNN / BLAS: GPU Backend

Tensors

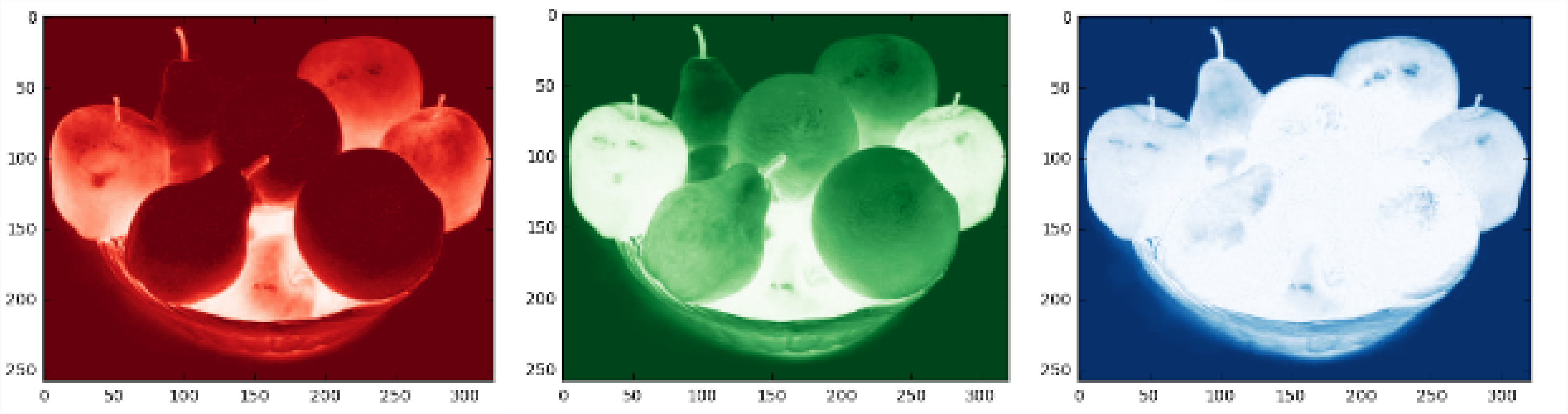
- Tensors – Mathematical objects
- Simply, N-Dimensional Arrays, like numpy.ndarray
- lingua franca in DL frameworks
- Data → Input Tensors → DNN → Output Tensors → Results
- Clean abstraction, allows use in different scenarios
- DNN sees only tensors, not images, audio, text...

Images to Tensors



A BMP Image

RGB Channels



3D Tensor

	0	1	2	3	4	5	6	7	8	9	...	310	311	312	313	314	315	316	317	318	319
0	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	...	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]
1	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	...	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]
2	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	...	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]
3	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	...	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]
4	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	...	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]	[1.0, 1.0, 1.0]

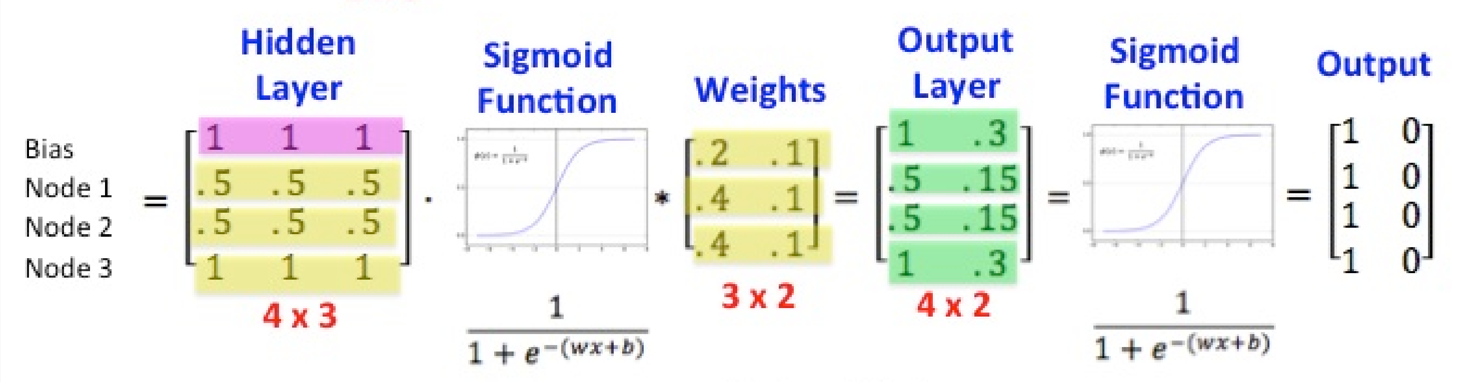
Other Examples

- Video – 4D Tensor (A video frame is an image)
- Words – Word2Vec
- Characters – 1-hot embeddings
- Audio – spectrograms .etc

Operations

- Operations on Tensors
- NNs are composition of Operations!
- Could let users implement
  - Suboptimal, prone to bugs, developer headaches
  - Can’t extend to new hardware and software versions
- Makes sense to support basic and widely used ops
  - Add, sub, mul, div, exp, log
  - Convolution, pooling, lstm units

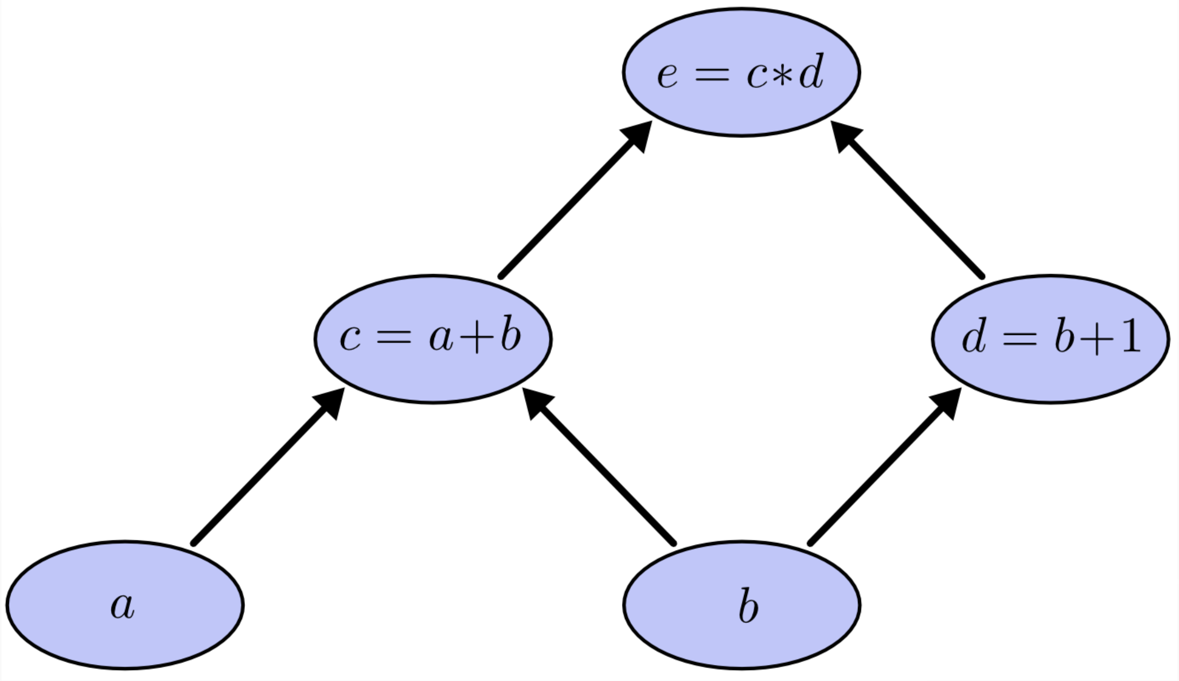
Example: Sigmoid layer



Sigmoid = 1 / (1 + np.exp(-1 * (np.dot(w.T,x)))

Computational Graph

- Combine multiple operations
- Graphical representation
- Similar to ASTs (Abstract Syntax Trees)



Need

- Helps to get a bigger picture of the network
- Allows us to run auto-diff on the network
- Helps in allocating resources to get best perf.
- Allows optimizations (two nodes with *2 → one with *4)
- Encapsulation, clean API
- Orchestration of operations

Make DNNs Learn Again!

- Beefed up backpropagation
- More general, easier to understand
- Calculus on computation graphs
- Chain Rule
- Symbolic differentiation and Autodifferentiation

Symbolic Differentiation

- Analytically find the gradients of each operation
- Chain Rule (and others)

$$z = f(y) \text{ and } y = g(x)$$

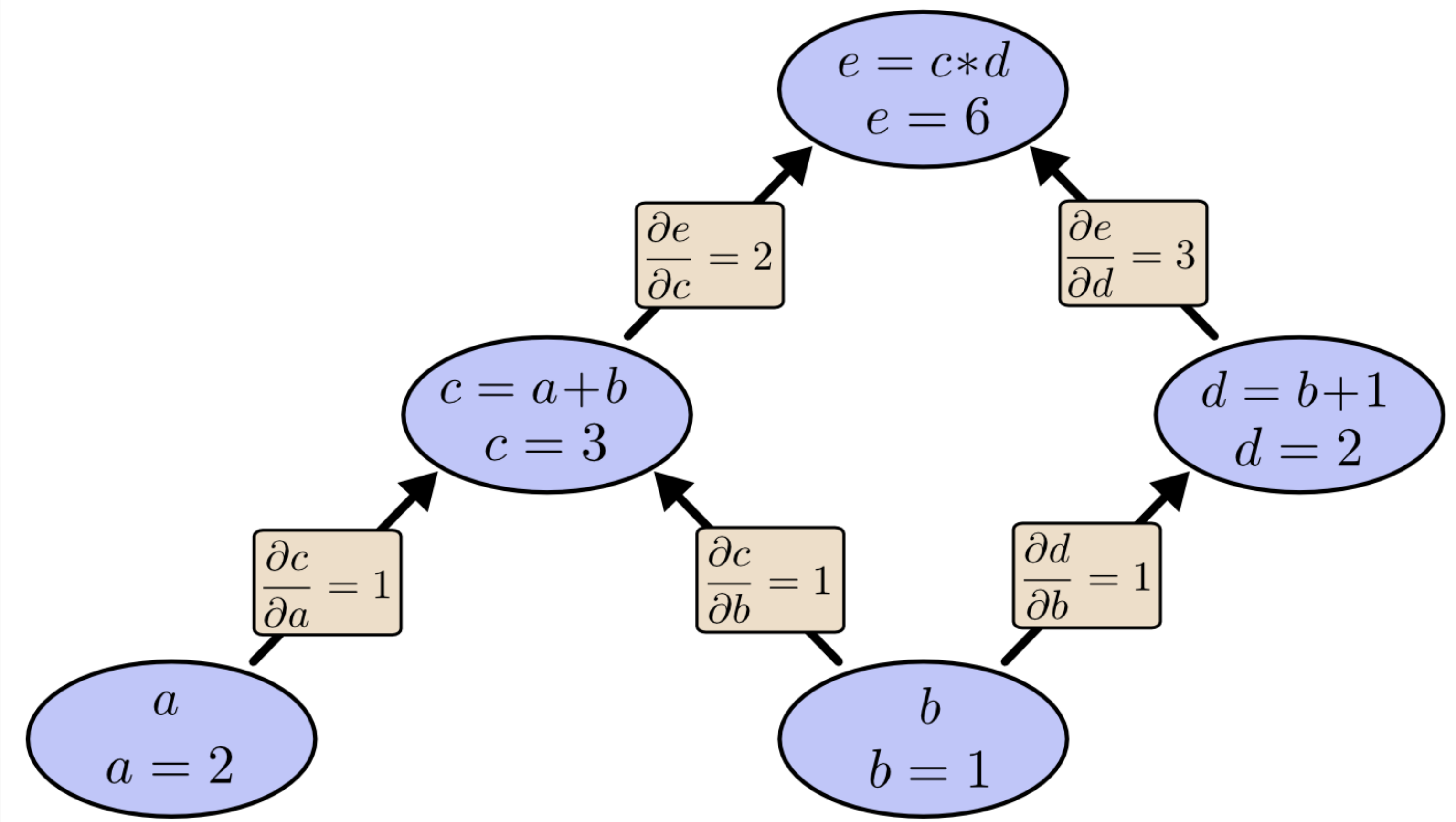
$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = f'(y)g'(x) = f'(g(x))g'(x)$$

- CAVEAT: Cannot calculate for all fn, too difficult, impoosible

Autodifferentiation

- Another approach to tackling the chain rule.
  - Compute the gradient for each Op (grad method)
  - Traverse the comp. graph,
  - collect gradients for each Op
  - Combine to get gradient
- Can be done in both forward and backward direction

Example

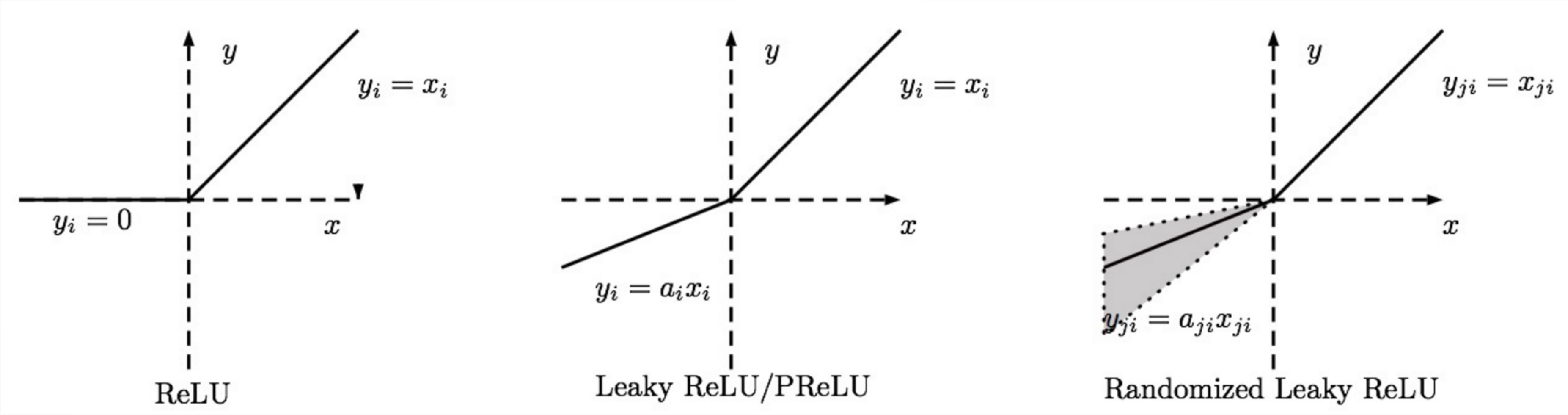


cf. http://colah.github.io/posts/2015-08-Backprop/

INTERMISSION

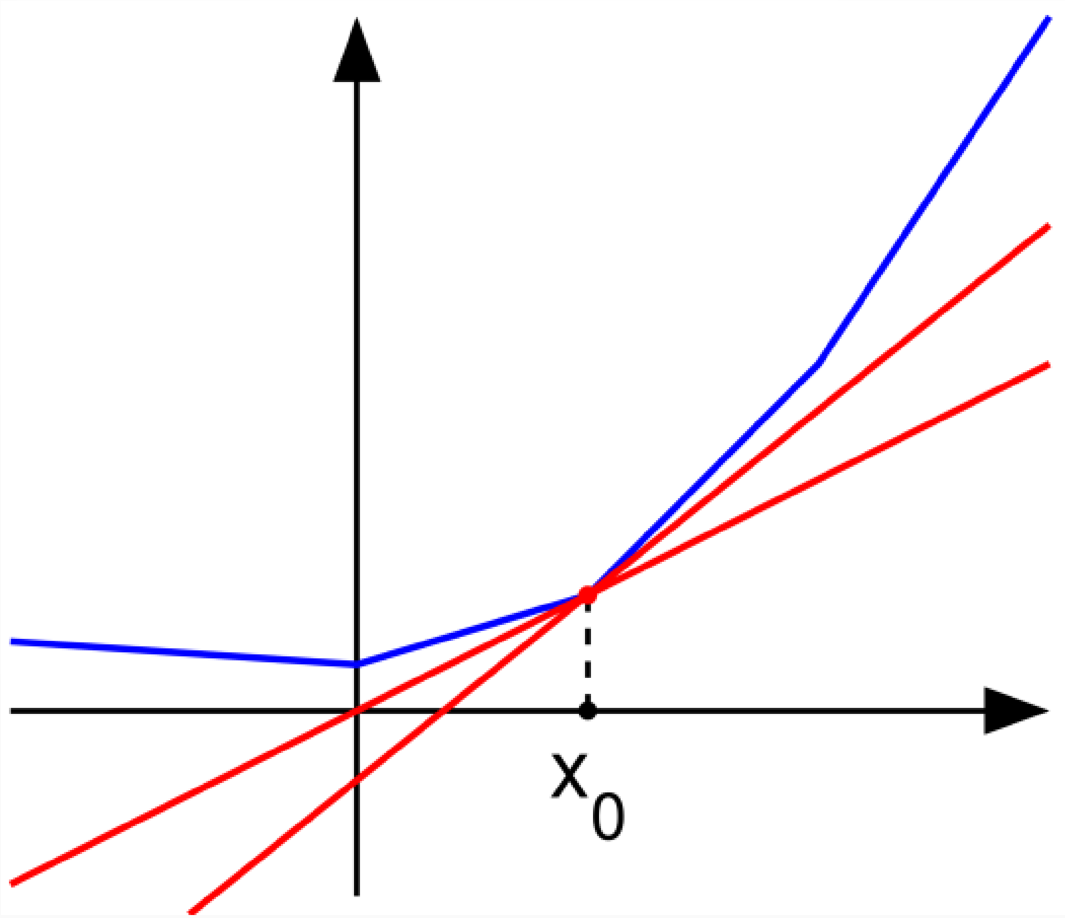
Aside: Subgradients

- ReLU, Leaky ReLU et al, not differentiable



- Can't differentiate at x = 0, approximate it!
- Subgrad: approximation to grads, greatest lower bound

Simple Example

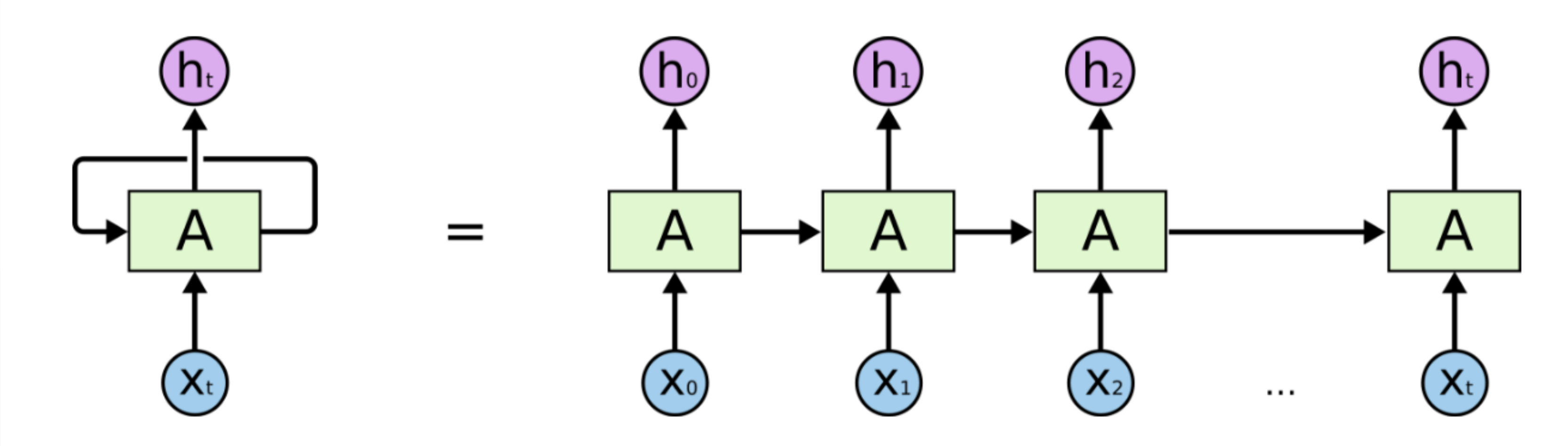


Aside II: RNNs

- RNNs have outputs of a layer as the input
  - h(t +1) = f(x, h(t))
- How do you run backprop?
  - Loop-the-loop
  - Multiply over and over again :(
- Results in Exploding / Vanishing Gradients

Backprop on RNNs

- Loop unrolling



- Are we done?
- NO, can still cause under and overflows
- Solution: Gradient Clipping (thresholds)

Time to get dirty*

* with the details ;)

Recap: DL Framework Components

- Components of any DL framework
  - Tensors
  - Operations
  - Computation Graph
  - Auto-differentiation
  - Fast and Efficient floating pt. Operations, GPU support
    - BLAS, cuBLAS, cuDNN

Functions or Classes?

- Should we define Ops as functions or classes?
- Functions → lesser memory footprint → logical mapping
- Classes → better encapsulation
  - Metadata like shape, size
  - Forward op and backward op have similar acces
- OOP → helps in scaling and extending
- But higher memory footprint
- Memory is cheap, dev time isn’t!
- Classes, FTW

Tensor Object

- Need to convert data to tensors and back
- Efficient storage of arrays
- Meta-data: shape, type, average, min, max
- Splicing and views
- Support for sparse matrices (ex. ReLU and variants)
- Integrity checks, GPU transfer, Compression

Op class

- Input sanity checks
- Optimized implementation
- Gradient Computation
- Shape of output tensor (sanity checks)
- Implementation in C++ / CUDA
- GPU / CPU?
- Parents and Children Ops – Useful for Computation graph

Graph Object

- Container class, refs to ops, and tensors
- Graph Traversal routines
- Device allocation and deallocation routines
- Methods to send inputs and get back results from devices
- Method to run autodiff

autodiff

- Don’t reinvent the wheel
- List: http://www.autodiff.org/?module=Tools
- Some examples in Python:
  - CGT: http://rll.berkeley.edu/cgt/
  - Autograd: https://github.com/HIPS/autograd
  - ad: http://pythonhosted.org/ad/
- Theano and TensorFlow use their own

Multicores? GPUs? Embedded?

- Laptop PC v. JBOGs v. Raspberry Pi
- Different hardware, different strengths
- Power-efficiency, Parallelism, Network Comm.
- Ex. Rpi
  - Low Power and Memory
  - Has GPU and supports HD video!
- Need to support multiple hardware opaquely!
- Use optimized numerical libraries

BLAS / LAPACK

- BLAS – **B**asic **L**inear **A**lgebra **S**ubprograms
- LAPACK – **L**inear **A**lgebra **PACK**age
- Written in Fortran or C, highly optimized!
  - Sometimes even assembly
- Can exploit multicore capabilities
- NumPy uses them
- Use routines for matrix ops instead of coding them

CUDA , cuBLAS

- CUDA – GPU Programming API
- Can be accessed in C, C++, Python (pycuda)
- Very low level
- Memory management, scheduling upto you
- Can lead to reduced perf.
- cuBLAS – BLAS in GPUs, very similar to BLAS API
- ALT: OpenCL

cuDNN

- Library with DL primitives
- eg. Convolution, LSTMs
- Built on top of CUDA
- Most DL frameworks use this in the background
- High-level, Highly Optimized
- Vanilla CUDA for initialization, cuDNN for compute

RECAP: It's all connected

- Tensors → For representing data
- Ops → To represent operations
  - Tensors → Ops → Tensors
- Computation Graph: Composition of Ops
  - Input → Op1 → Op2 → Op3 … → OpN → Output
- Autodiff: generalized backprop
- BLAS / CUDA / cuBLAS / cuDNN

What next?

- Know why your net is not training fast enough
- Go through the dev-docs, theano is pretty good
- DIY DLF, for fun and hopefully profit!
- Chutney: The IDLI DL Library ?

Questions?

Thanks!

- Malai
  - For proof-reading the article
  - Organizing and moderating this talk
- IDLI – Wonderful group, awesome discussions
- Fred Bastien: Theano Lead Dev
- Colah: InkScape guru _/_
- The DL community

Thanks!
Dankeschon!
Dhanyawaad!
Nandri!