MXNet: Lightweight, Flexible, and Efficient Deep Learning Library

Naiyan Wang

TuSimple

MXNet is developed by over 100 collaborators

Special thanks to

Tianqi Chen
UW

Mu Li
CMU/Amazon

Bing Xu
Turi

Chiyuan Zhang
MIT

Junyuan Xie
UW

Yizhi Liu
MediaV

Tianjun Xiao
Microsoft

Yutian Li
Stanford

Yuan Tang
Uptake

Qian Kou
Indiana University

Hu Shiwen
Shanghai

Chuntao Hong
Microsoft

Min Lin
Qihoo 360

Naiyan Wang
TuSimple

Tong He
Simon Fraser University

Minjie Wang
NYU

Valentin Churavy
OIST

Ali Farhadi
UW/AI2

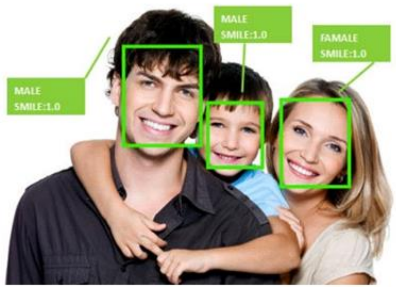
Carlos Guestrin
UW/Turi

Alexander Smola
CMU/Amazon

Zheng Zhang
NYU Shanghai

Deep Learning

Image Understanding



Speech Recognition

Natural Language Processing

“Deep Learning” trend in the past 10 years

Packages

TensorFlow

torch

Chainer

dmlc mxnet

theano

Caffe

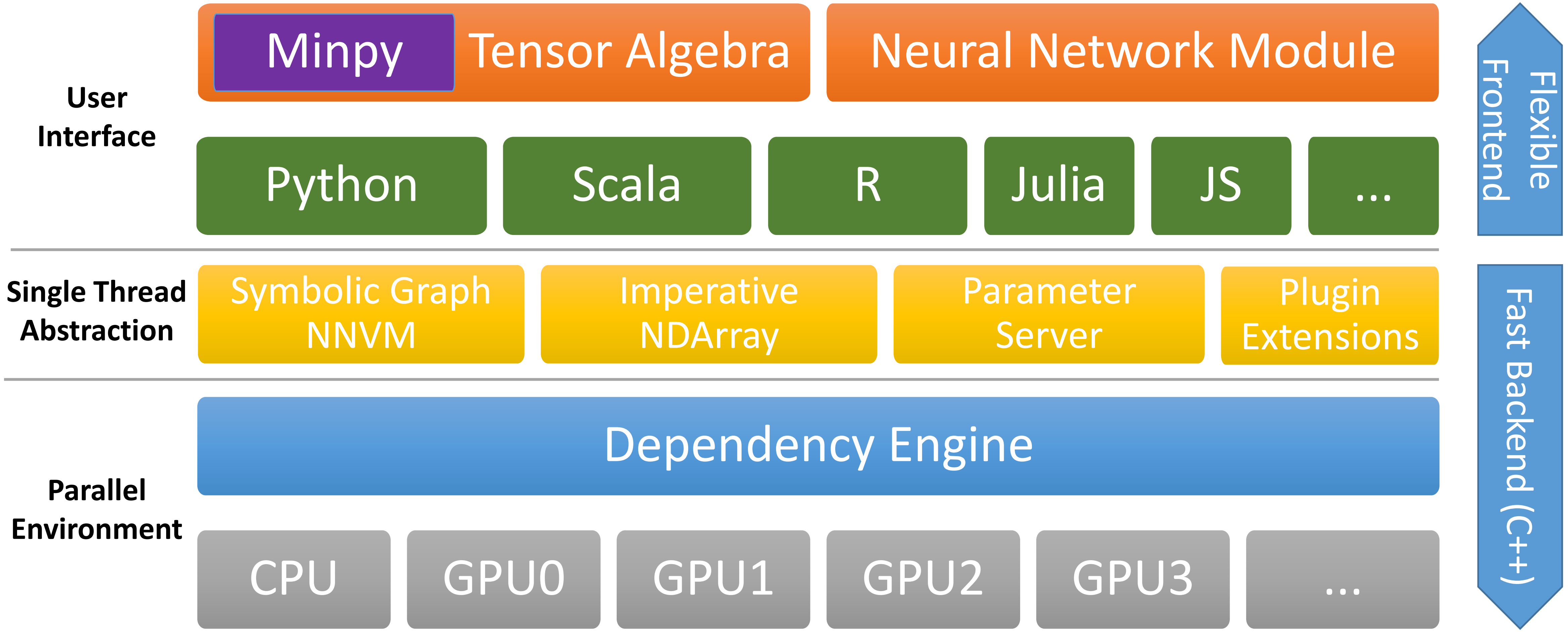
K

Microsoft CNTK

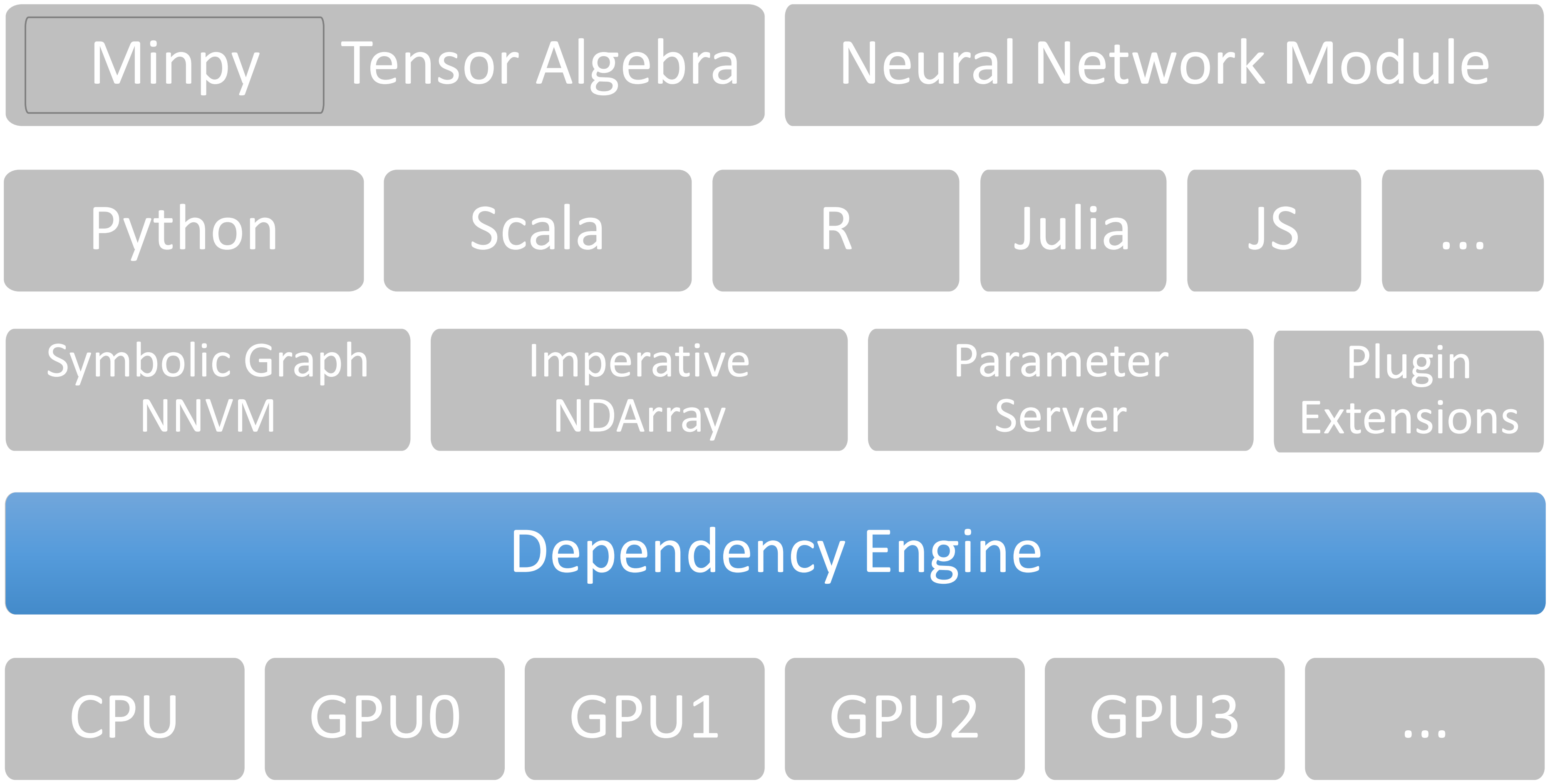
MXNet's Approach

- Lightweight
- User defined flexibility-efficiency trade-off
- Transparent scaling
- Deploy everywhere
- Modular and extendable
- Mixed declarative and imperative programming
- Community driven open source

The MXNet Stack

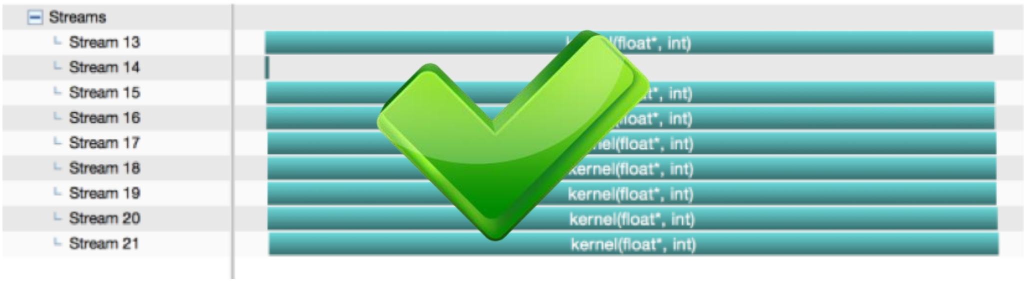


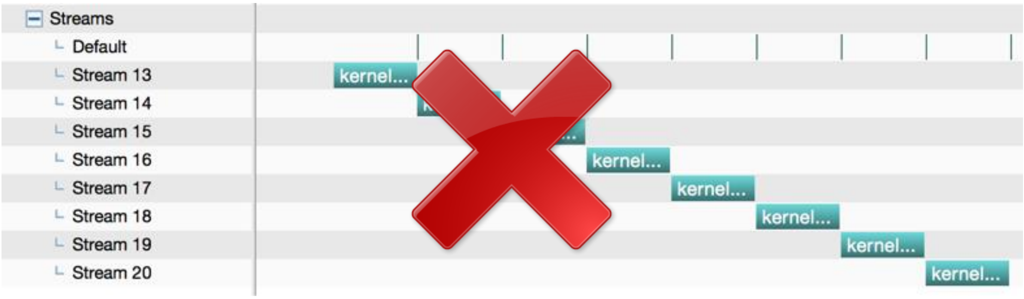
Dependency Scheduling Engine



Need for Parallelism

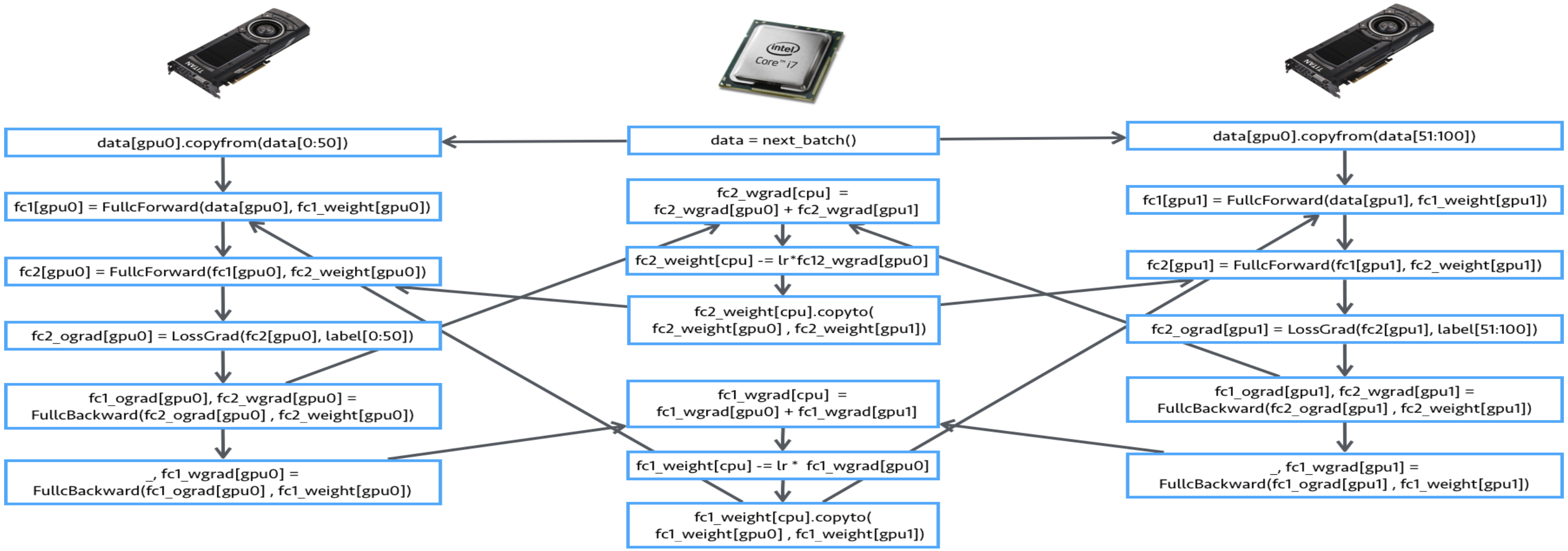
- Speed is critical to deep learning
- Parallelism leads to higher performance
  - Parallelization across multiple GPUs
  - Parallel execution of small kernels
  - Overlapping memory transfer and computation
  - …





Parallel Programs are Painful to Write…

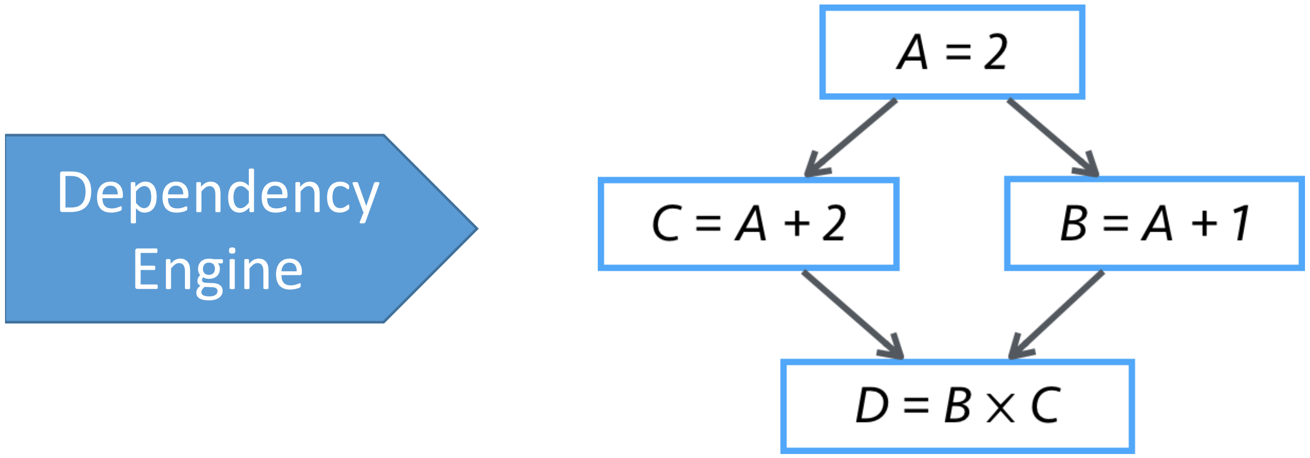
- … because of dependencies



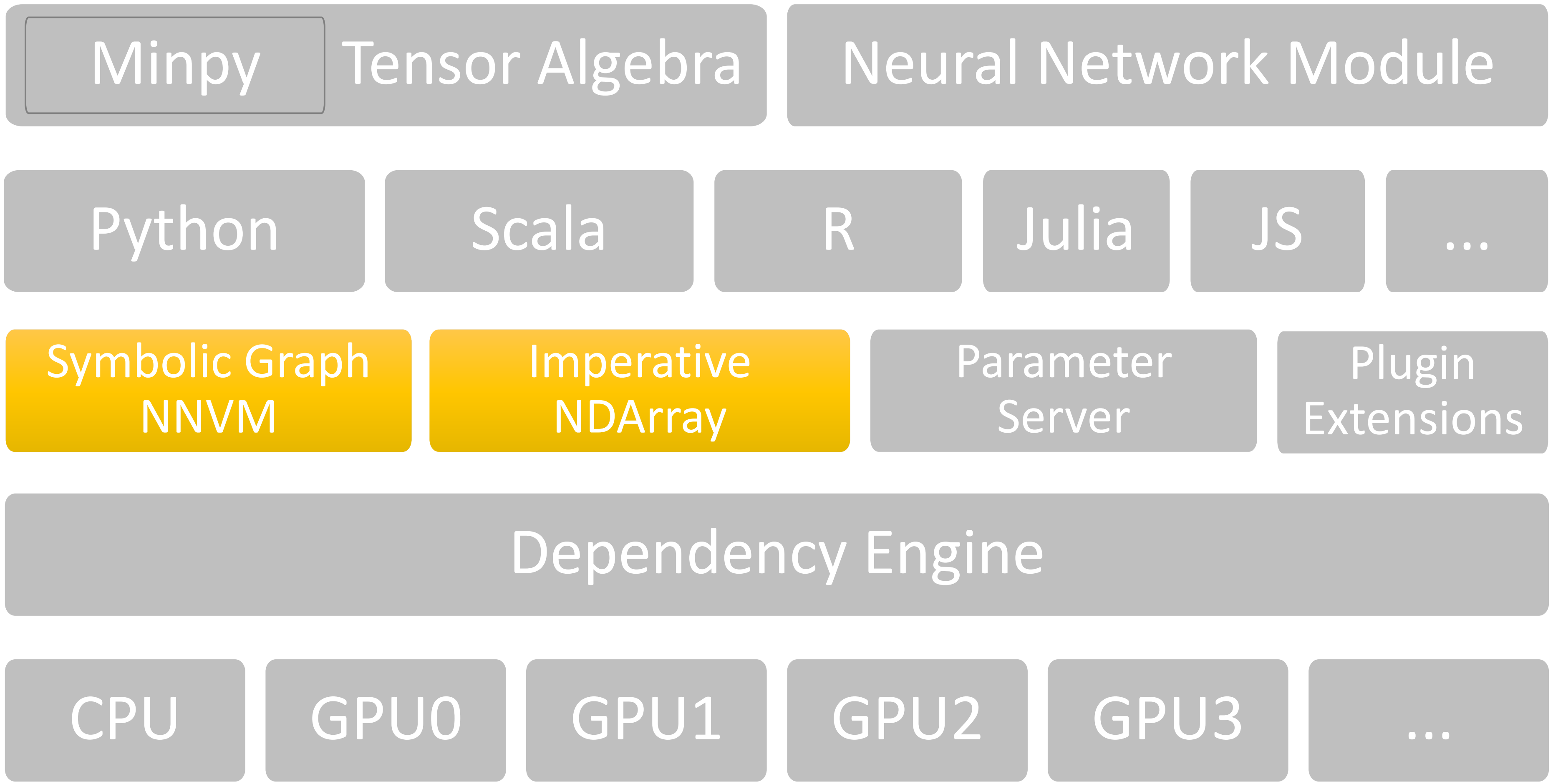
Solution: Auto Parallelization with Dependency Engine

- Single thread abstraction of parallel environment

```python
import mxnet as mx
A = mx.nd.ones((2,2)) *2
C = A + 2
B = A + 1
D = B * C
```

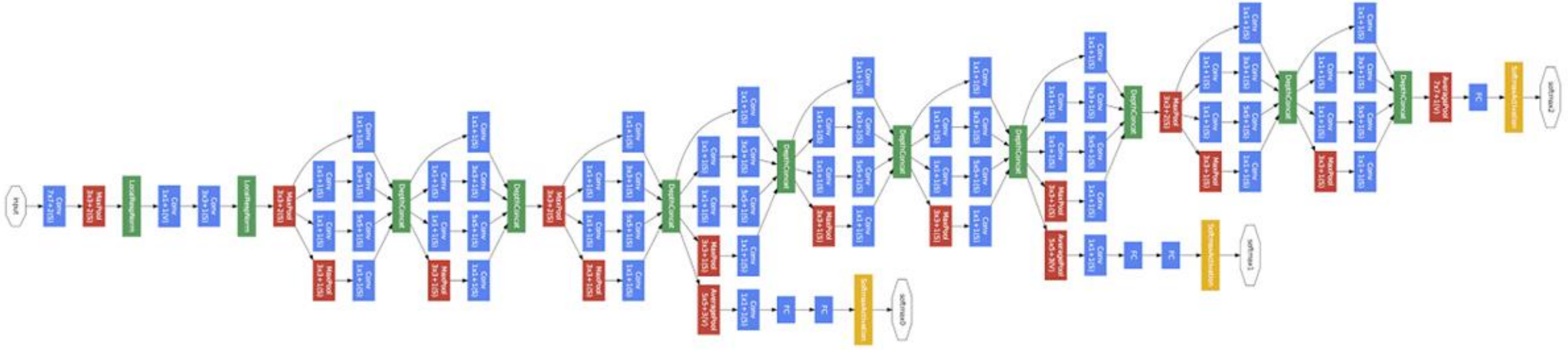


Symbolic vs. Imperative Deep Learning



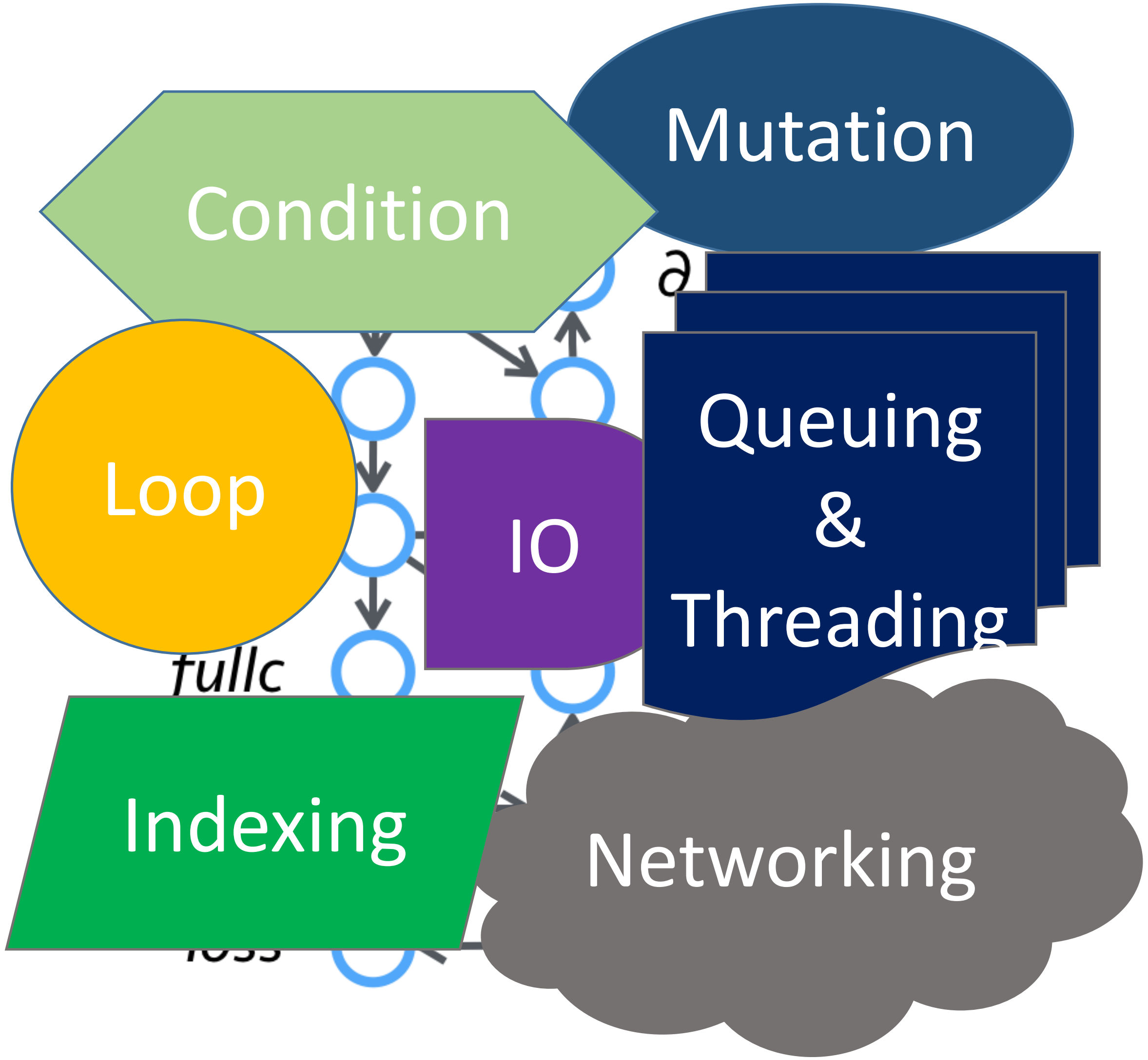
Neural Networks as Symbolic Graph

- Most packages represent networks as graphs
  - MXNet, Caffe, Theano, Tensorflow, CNTK, …
- Easy to store, port, and optimize
  - Parallelization, buffer sharing, operator fusion, serialize and deploy, …



Deep Learning is More Than DAG

- All is good …
  - … until you start adding too many things to it …



You just reinvented programming language

...

and lost most advantages of using graph

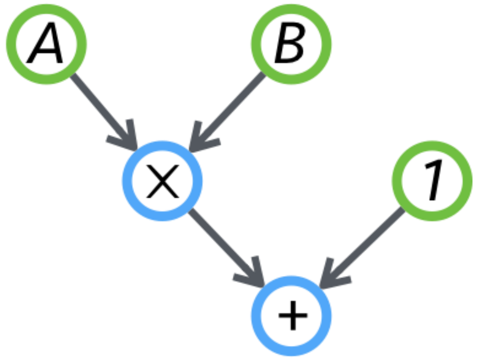
Neural Networks as Imperative Program

- DL, or ML in general, is largely tensor algebra.
  - Torch, Chainer, Matlab, R, Numpy, …
- Imperative programs are flexible but hard to optimize:

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
print(c)
d = c + 1
```

Easy to tweak with python codes

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10), B=np.ones(10)*2)
```



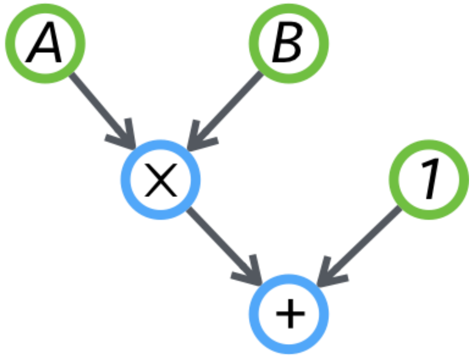
Neural Networks as Imperative Program

- DL, or ML in general, is largely tensor algebra.
  - Torch, Chainer, Matlab, R, Numpy, …
- Imperative programs are flexible but hard to optimize:

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

c cannot share memory with *d*, because it could be used in future

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10), B=np.ones(10)*2)
```



C can share memory with *D*, because C cannot be seen by user

MXNet’s Approach: Mixed Programming

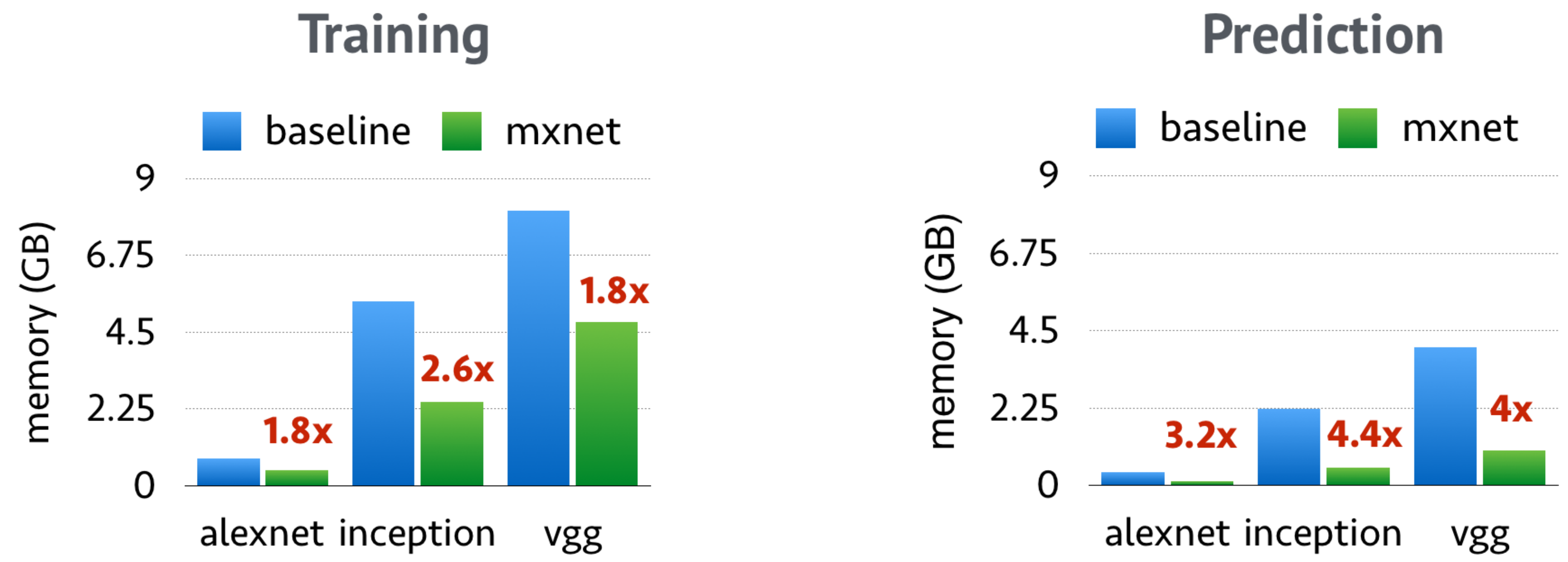
- Mix symbolic and imperative operations to get benefit of both.
- Symbolic graph: heavy, standard operations.
  - >90% of runtime, <10% of coding time.
- Imperative program: light, project specific operations.
  - <10% of runtime, >90% of coding time.
- Dependency Engine allows seamless combination of two parts.
- Only optimize the bottleneck!

MXNet’s Approach: Static Graph

- In MXNet, graphs are simple and static:
  - Simple DAG without fancy logic
  - Fixed topology and tensor shapes
- Static graphs are:
  - Faster to build: multiple graphs instead of loop and condition.
  - Easier to optimize: Static memory planning, asynchronous execution, …
- Flexibility compensated by imperative operations.

Smallest GPU Memory Footprint

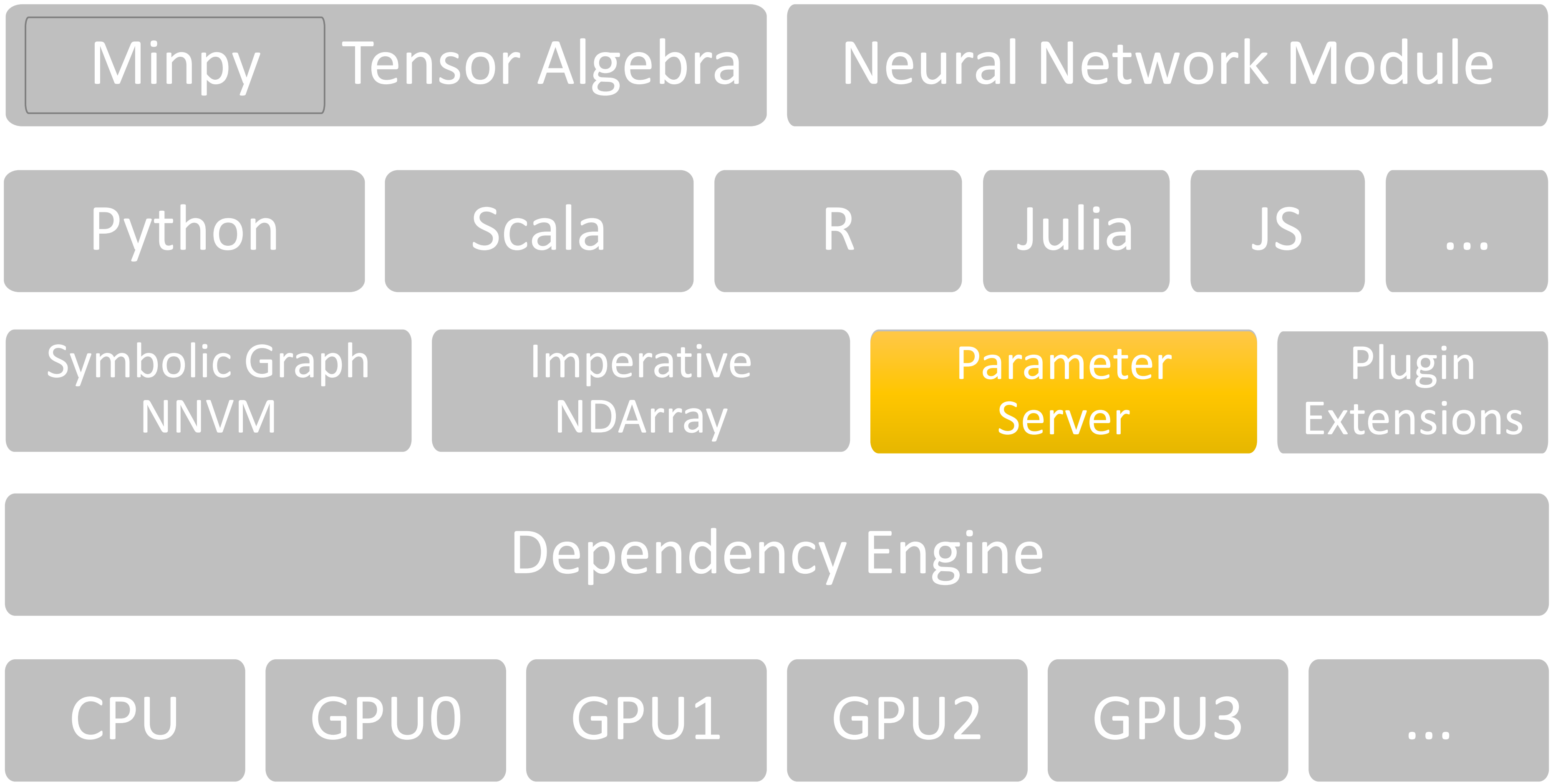
- Static graph enables aggressive memory sharing.



Trade Speed for Memory

- MXNet Mirror:
  - Discard result of small Ops on forward pass (ReLU, BN, etc).
  - Re-compute on backward pass.
  - 30%-50% memory saving at 90% speed.
- MXNet Memmonger:
  - Only keep result of $\sqrt{N}$ (anchor layers) out of N layers on forward pass.
  - Re-compute $\sqrt{N}$ layers between two anchor layers on backward pass.
  - Train $\mathrm{O}(\sqrt{N})$ times larger model at 75% speed.

Parallel & Distributed Training

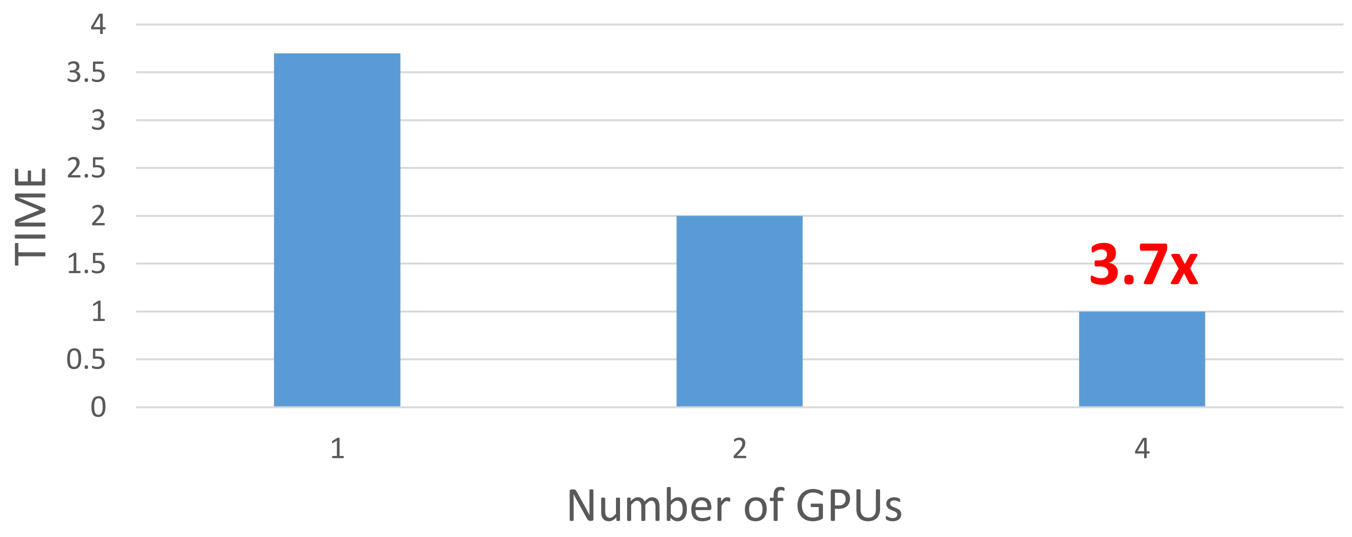


Drop-in Parallel Training

- Scaling to Multi-GPU machine as easy as one line change:

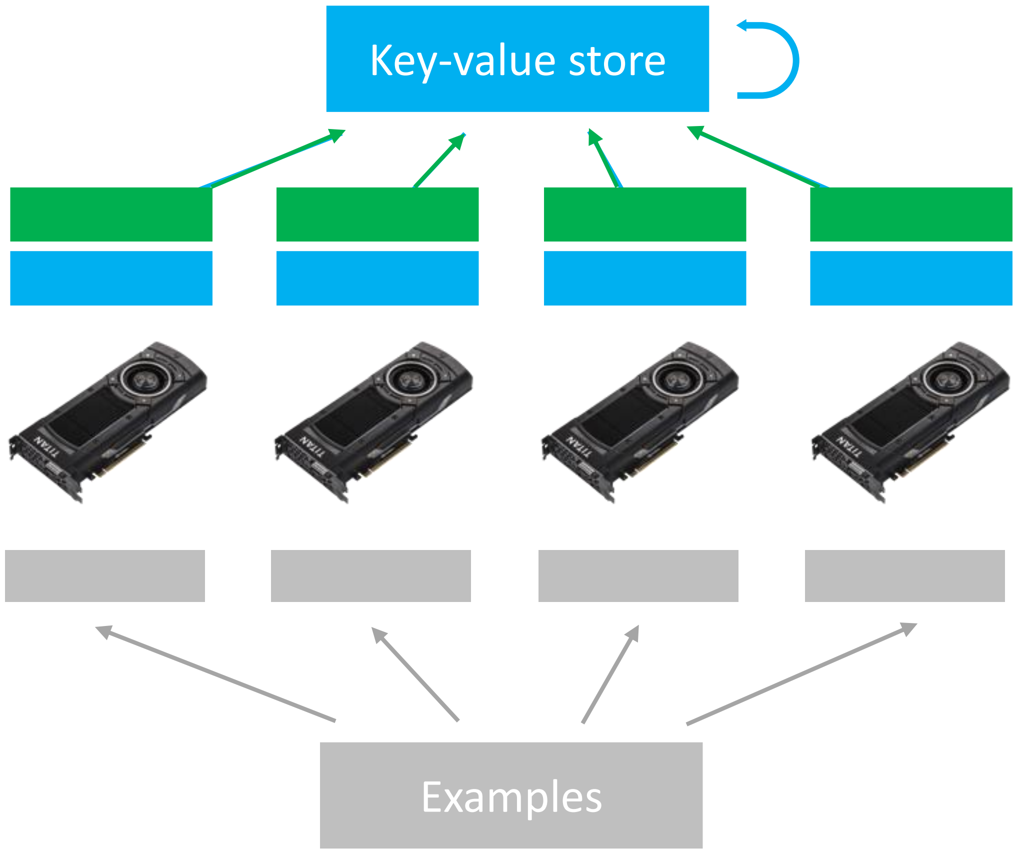
```
model = mx.mod.Module(net, ctx=mx.gpu(0))
-> model = mx.mod.Module(net, ctx=[mx.gpu(0), mx.gpu(1)])
```

- Near linear speedup on a single machine:



Parallel Training: Under the Hood

- Read a data partition
- Pull the weight
- Compute the gradient
- Push the gradient
- Update the weight



Distributed Training

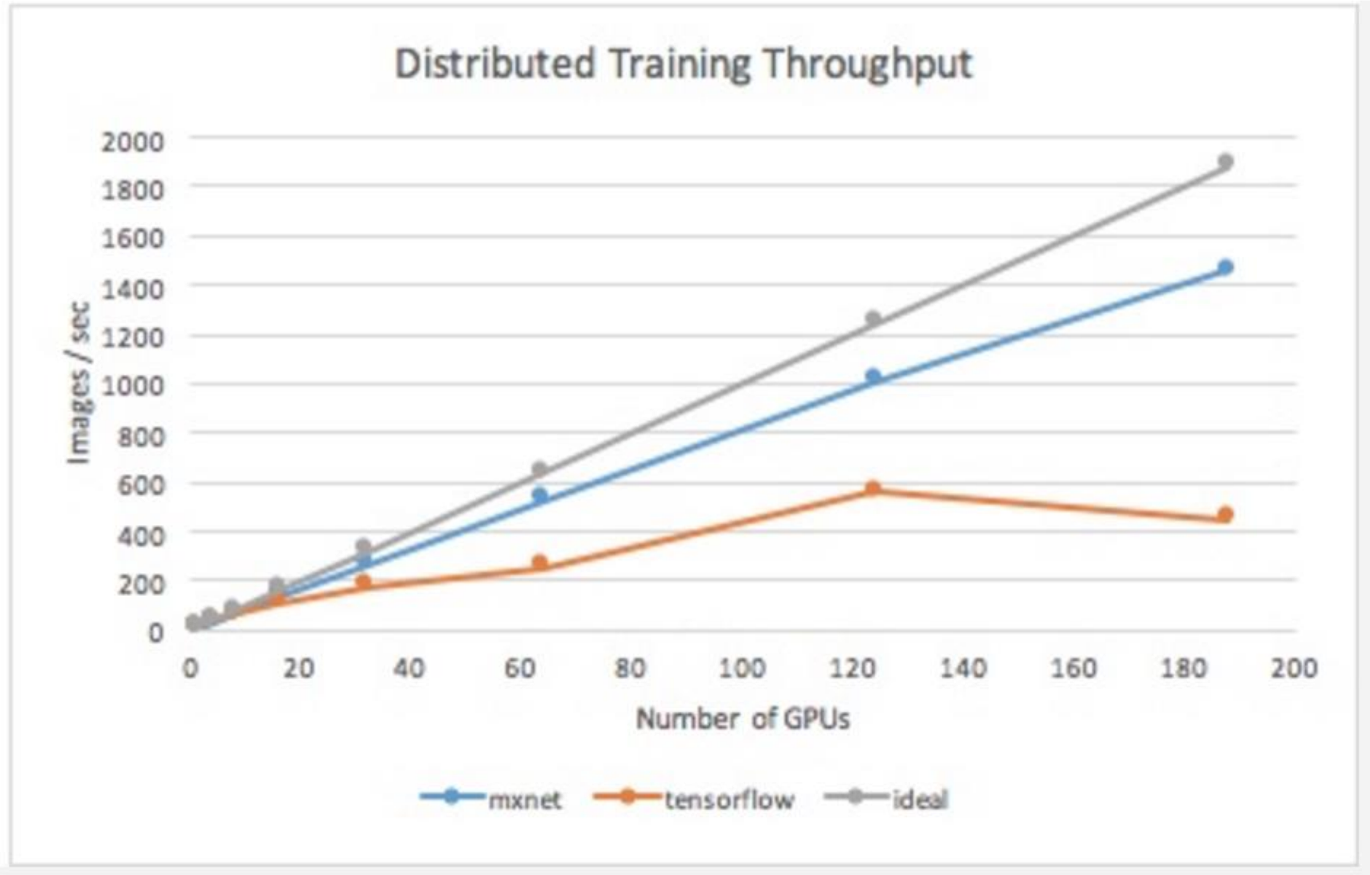
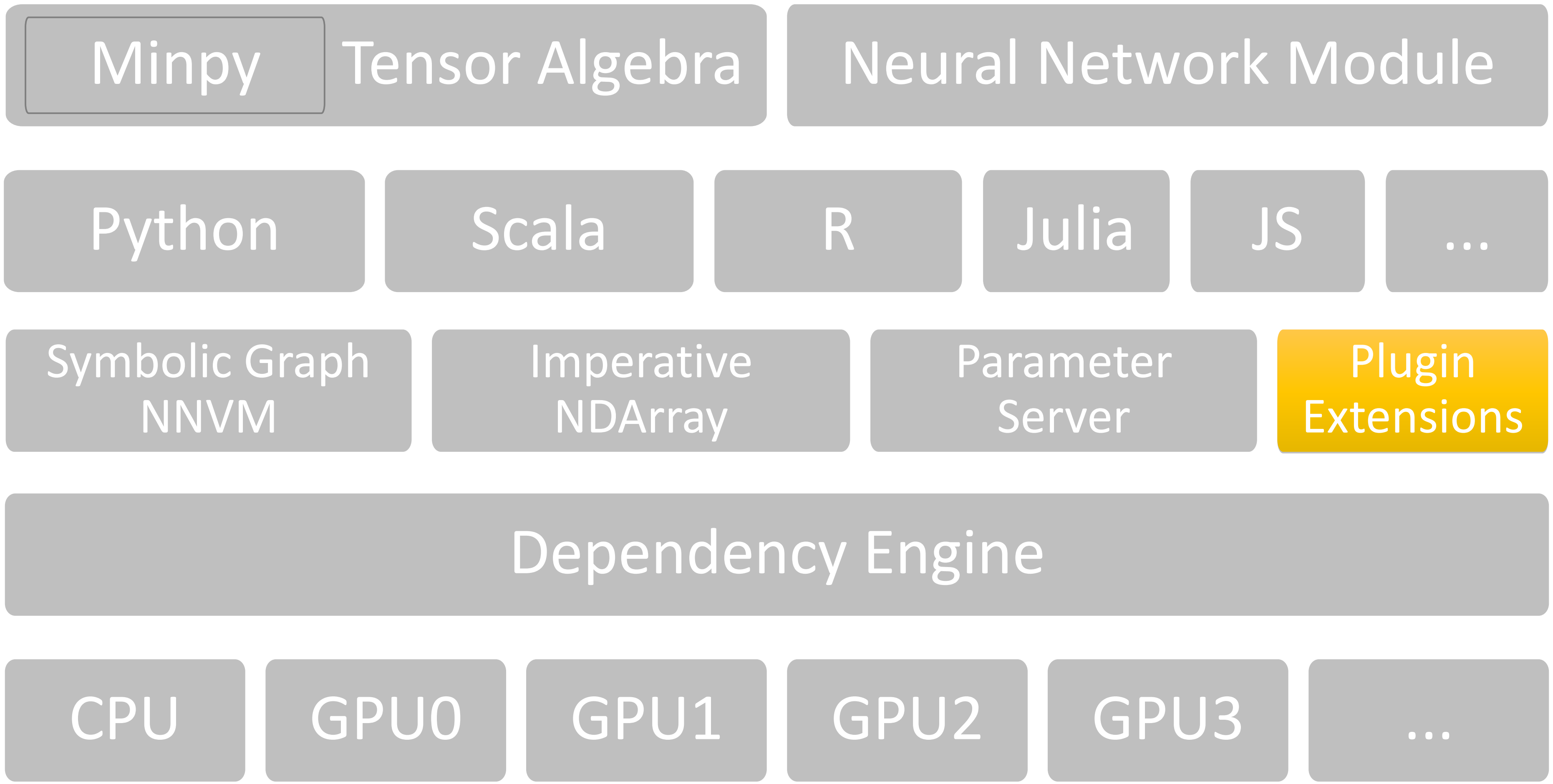


Figure by Carlos Guestrin @Turi

- Scale to multi-machine with the same key-value store interface.
- 2x faster than Tensorflow on >10 4GPU machines
- Latest Update: 74x acceleration on 10 machines with 8GPUs in each machine.

Parallel & Distributed Training



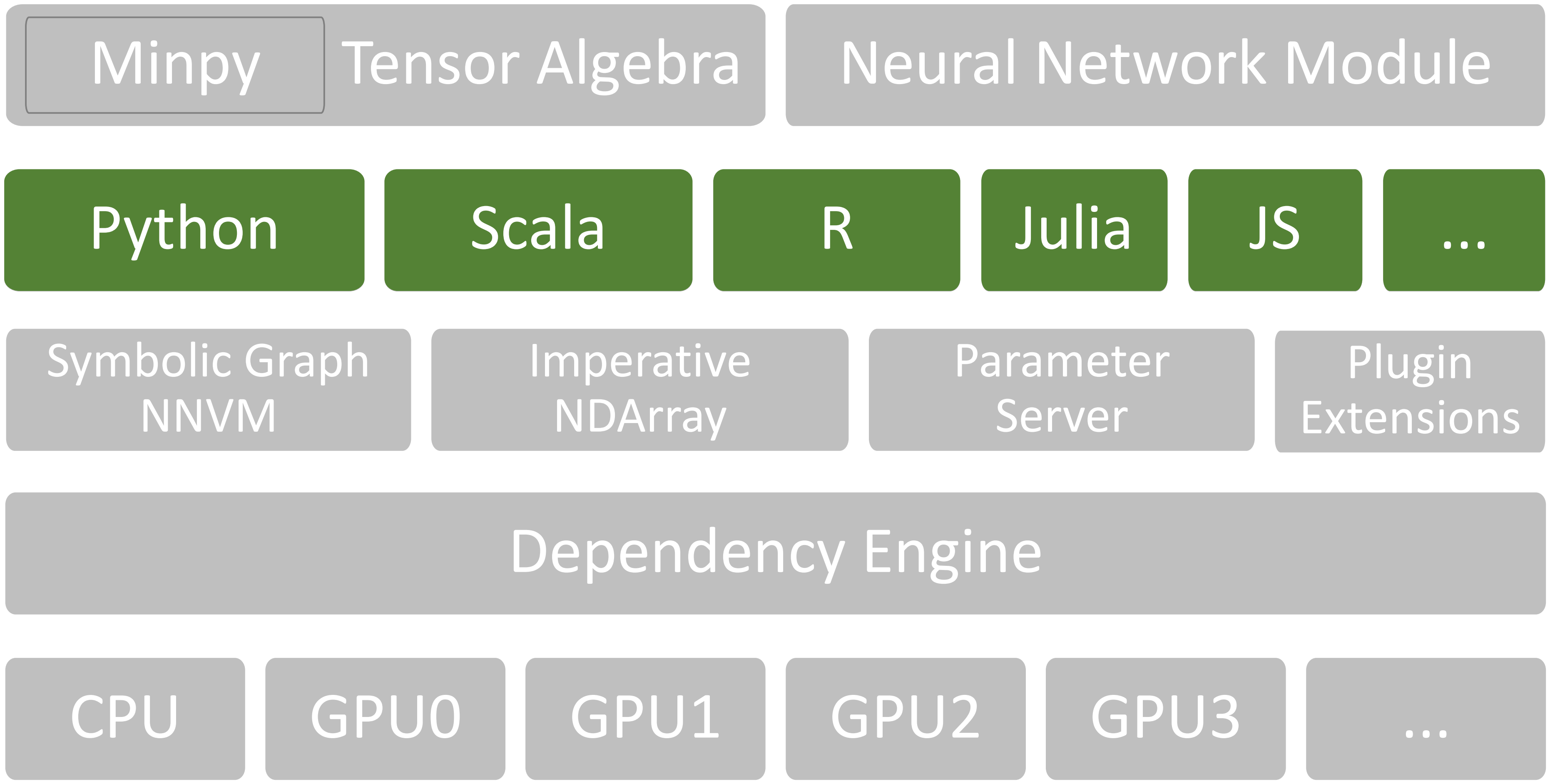
Plugin Extensions

- TorchModule:
  - Use Torch NN layers and tensor functions in MXNet graph.
    fc1 = mx.sym.TorchModule(lua_string='nn.Linear(784, 128)', …)
- CaffeOp:
  - Use Caffe layers in MXNet graph.
    fc1 = mx.symbol.CaffeOp(prototxt="layer{type:\"InnerProduct\”
    inner_product_param{num_output: 128} }”, …)
- WrapCTC:
  - Use Baidu’s CTC module for sequence learning in MXNet.
- OpenCV:
  - Multi-threaded OpenCV interface that by pass GIL for fast image IO.

Mainstream Applications in Vision/NLP/Speech

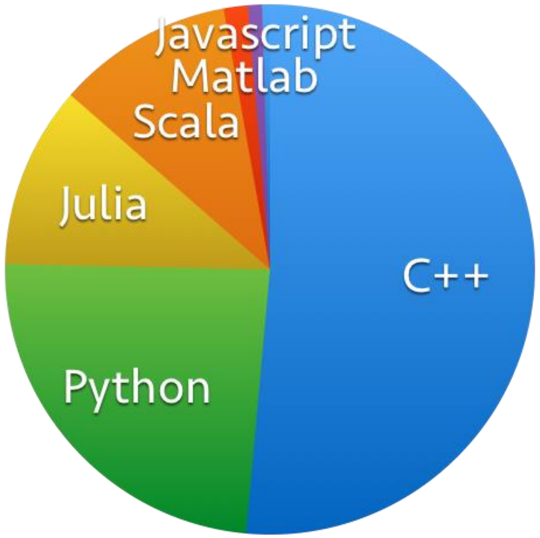
- Image Classification
  - Inception/ResNet
- Object Detection
  - Faster RCNN
- Image Segmentation
  - FCN/Deeplab
- OCR
  - Warp-CTC
- Char LSTM/Char CNN/Speech Acoustic Modeling/Neural Art…

Runs Everywhere



Code with Any Language

Lines of code



Train in the Cloud

Load data from distributed filesystems

HDFS

S3

Blob

⋮

multithreaded read/write
to hide network latency

Launch distributed jobs

SSH

MPI

qsub

Yarn

⋮

easily extend to other cluster
resource management software

Deploy Everywhere

Beyond

Amalgamation

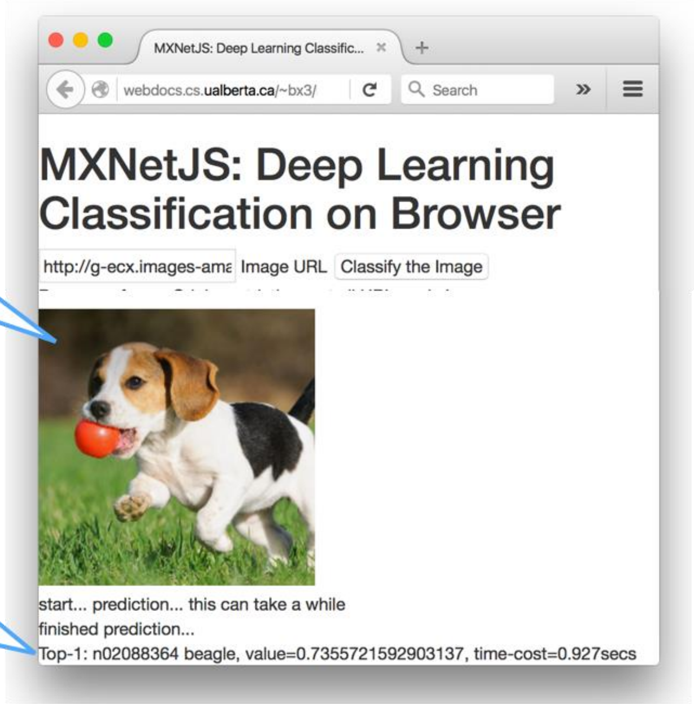
- Fit the core library with all dependencies into a single C++ source file
- Easy to compile on …

BlindTool by Joseph Paul Cohen, demo on Nexus 4

Runs in browser with Javascript

The first image for search “dog” at images.google.com

Outputs “beagle” with prob = 73% within 1 sec



Thanks