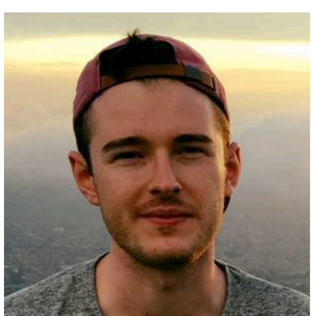
amazon web services™ Caltech UC IRVINE Carnegie Mellon University

SignSGD / Signum optimizers and deep learning with Gluon

Yu-Xiang Wang

Joint work with:

Jeremy Bernstein, Kamyar Azizzadenesheli, Anima Anandkumar

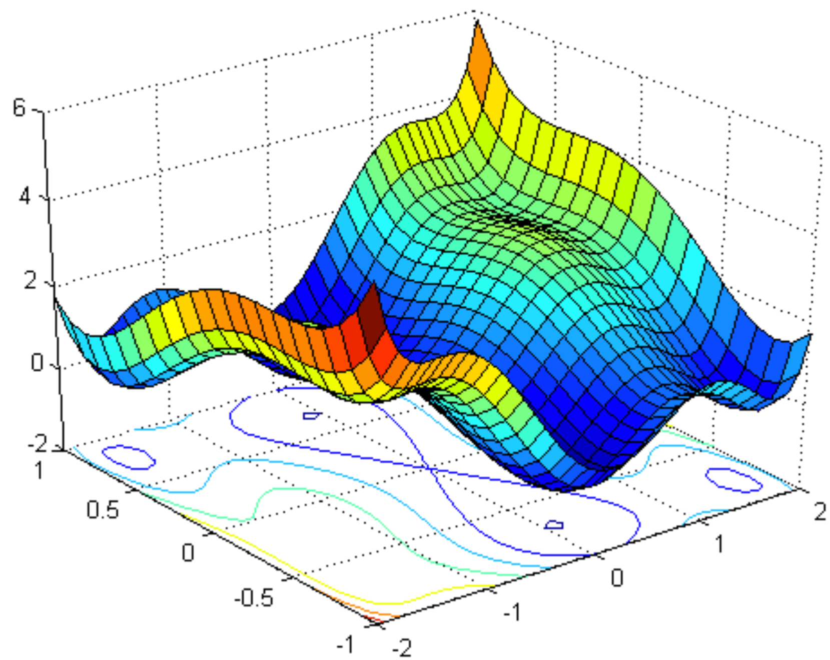


Outline

1. SignSGD / Signum optimizers
   - 1-bit updates for communication-efficiency
   - Nonconvex convergence rate
   - Escaping saddle point?

2. Hands-on Gluon tutorial: gluon.mxnet.io
   - Imperative vs. Symbolic
   - Train ConvNet with Gluon
   - Hybridization and Multi-GPU training

Part I: Nonconvex optimization with 1-bit updates: SignSGD and Signum



Zoos of algorithms: for nonconvex optimization theory

- Nonconvex SVRG [Reddi et. al., 2015, Allen-Zhu & Hazan, 2015]
- Noisy GD / SGD [Ge et. al.2015, Jin et. al. 2017]
- Trust-region method [Sun et. al., 2015]
- Natasha 1/2 [Allen-Zhu, 2017]

Detailed computational theory on convergence rate to **stationary points** and to **local minima**!

Zoos of algorithms: for deep learning practice

- SGD [Robbins and Monro, 1951]

- Momentum [Polyak, 1964; Nesterov, 1983]

- Adagrad [Duchi et. al., 2011] / Adam [Kingma & Ba, 2014]

- Rprop [Riedmiller&Braun, 1993] / RMSprop [Tieleman&Hinton, 2012]

Not well understood theoretically (perhaps except SGD).

Hammers and tricks

v.s.

- Variance reduction
- Active noise adding
- Hessian-vector product
- Cubic regularization

- Momentum
- Gradient clipping
- Batch normalization

Why don’t we listen to the practitioners for just once?
And try to understand how their approachs could work?

The Adam algorithm

Adam: A method for stochastic optimization

D **Kingma**, J Ba - arXiv preprint arXiv:1412.6980, 2014 - arxiv.org

Abstract: We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory

☆ 99 Cited by 5983 Related articles All 9 versions

[PDF] A stochastic approximation method

H **Robbins**, S **Monro** - The annals of mathematical statistics, 1951 - JSTOR

Let M (x) denote the expected value at level x of the response to a certain experiment. M (x) is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution x= θ of the equation M (x)= α, where α is a given constant. We

☆ 99 Cited by 5143 Related articles All 8 versions Web of Science: 1930 »

The Adam algorithm

- Adam update:

$$m_t = \beta m_{t-1} + (1-\beta) g_t$$

momentum

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t^2$$

variance

$$x_t = x_{t-1} + \eta \frac{m_t}{\sqrt{v_t}}$$

variance adjusted momentum update

Key idea: dividing the gradient (coordinate wise) by its magnitude!

From Adam to SignSGD to Signum

- SignSGD update: (Rprop [Riedmiller&Braun, 1993] in fact!)

$$x_t = x_{t-1} + \eta \frac{g_t}{|g_t|}$$

- Signum (SIGN momentUM):

$$m_t = \beta m_{t-1} + (1-\beta) g_t$$
$$x_t = x_{t-1} + \eta \frac{m_t}{|m_t|}$$

Under the standard assumptions

- Assumptions:
  - A1: Bounded from below $f(x) \geq f^*$
  - A2: Strong smoothness $\|g(y) - g(x)\|_2 \leq L\|y - x\|_\infty$
  - A3: Stochastic gradient access

$$\mathbb{E}\hat{g}(x) = g(x), \quad \mathrm{Var}(\hat{g}(x)[i]) \leq \sigma^2 \; \forall i = 1, ..., d.$$

Stationary point convergence of signSGD and Signum

Theorem: Take learning rate and minibatch size to be:

$$\delta_k = \frac{\delta}{\sqrt{k+1}} \qquad\qquad n_k = k+1$$

Then we have:

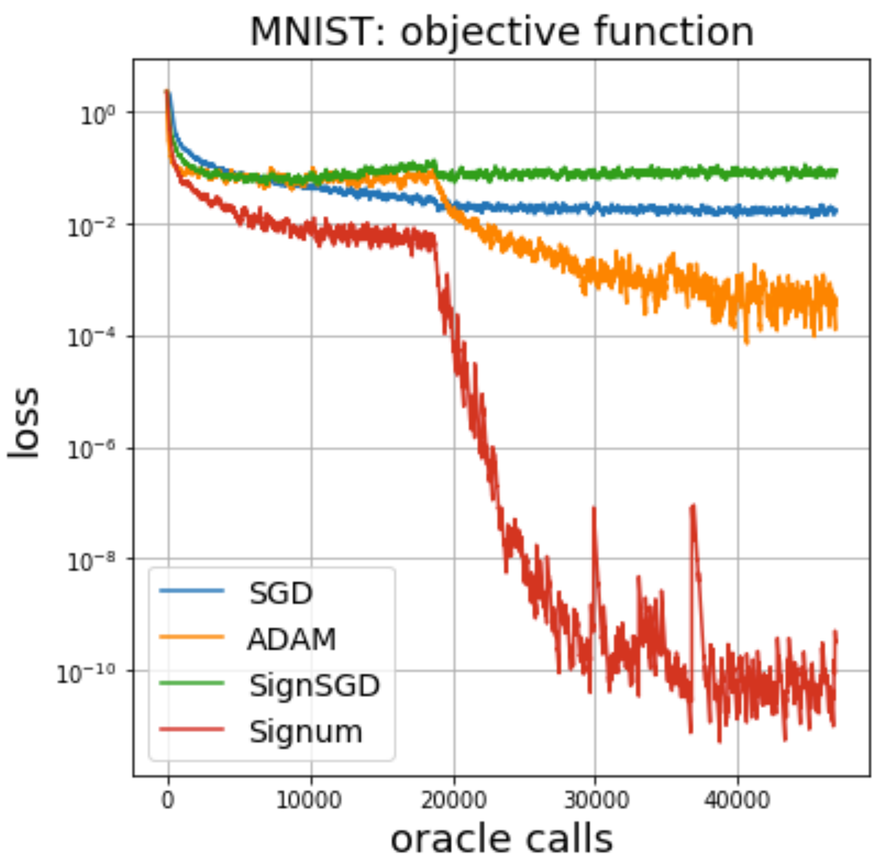
$$\min_{C \leq k \leq K-1} \mathbb{E}[\|g_k\|_1]^2 = \; O(1/\sqrt{N})$$

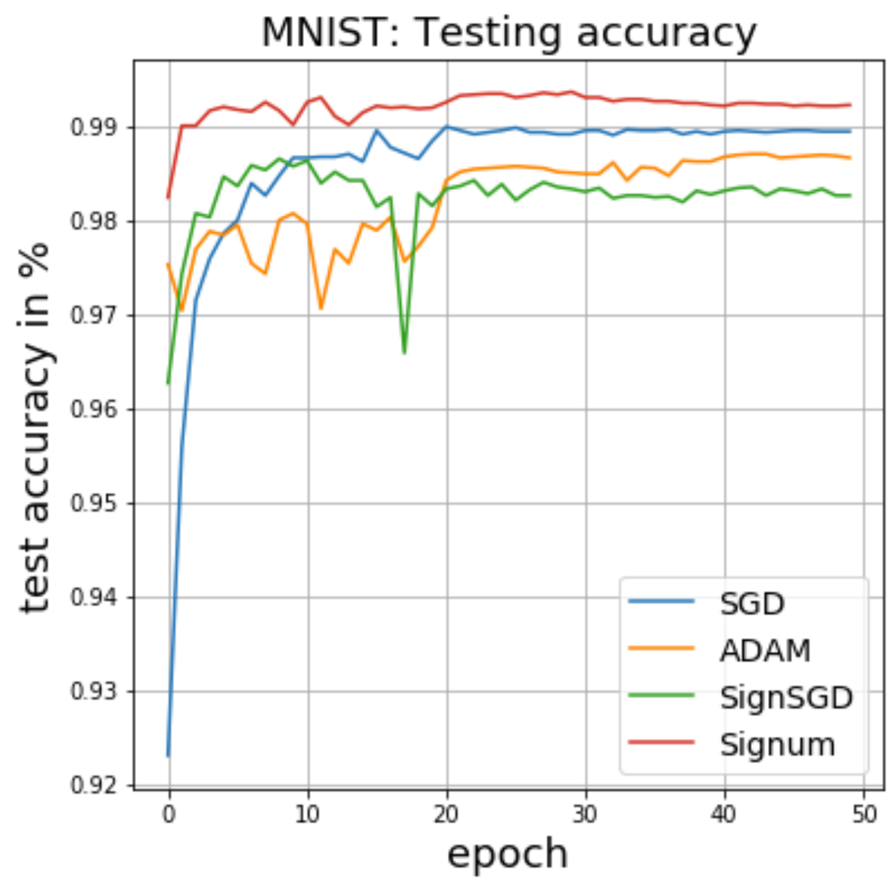
- Burn-in period: C=1 for SignSGD and O(1) for momentum.
- Const. hides dimension / smoothness constant and beta.

Novelties in the proof

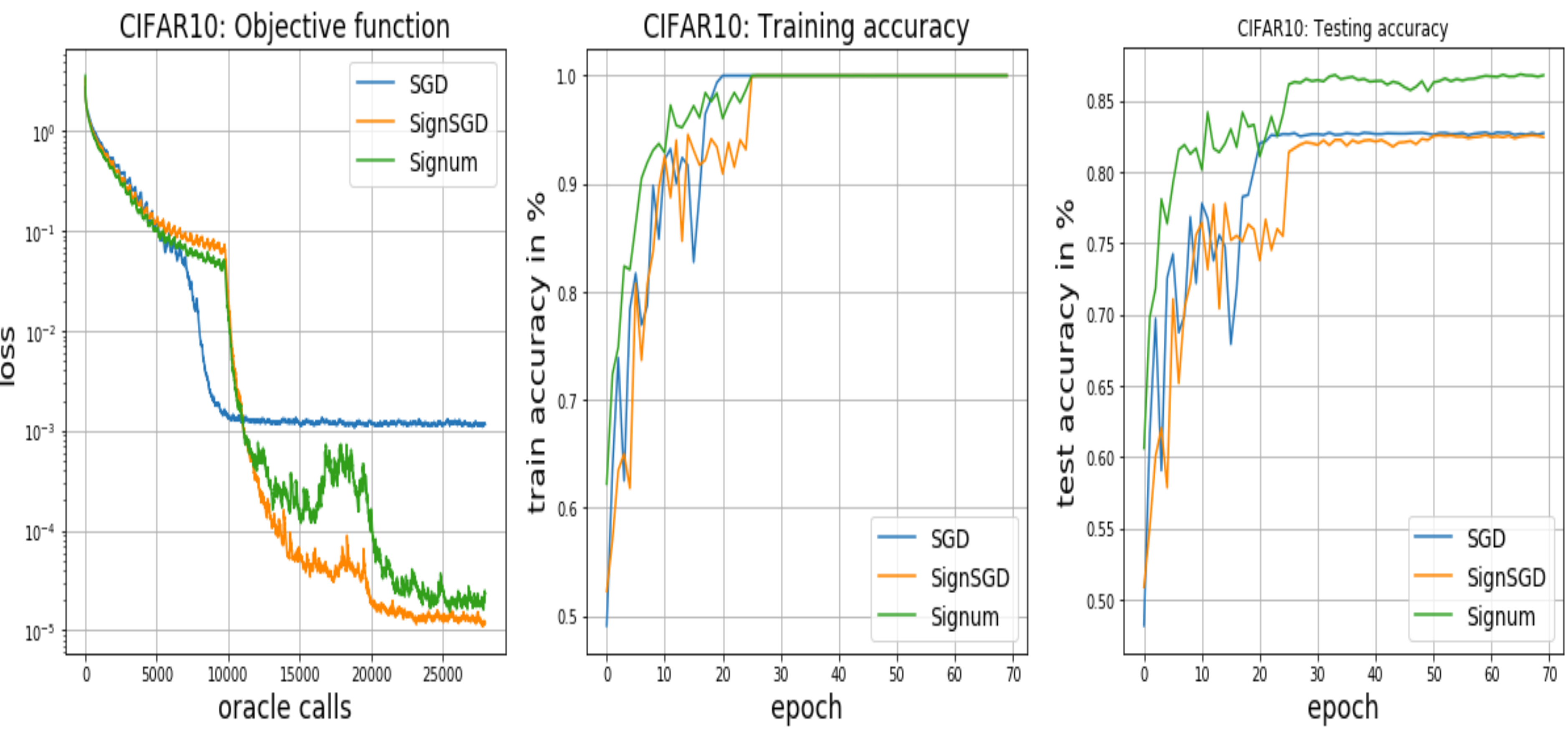
- Use L1 norm for the nonlinear mapping of sign
- Biased gradient
- Handling momentum
- A general recipe of approximate signed updates.
    - Easy to handle delayed gradients
    - Variance reduction techniques

Real data experiments: MNIST



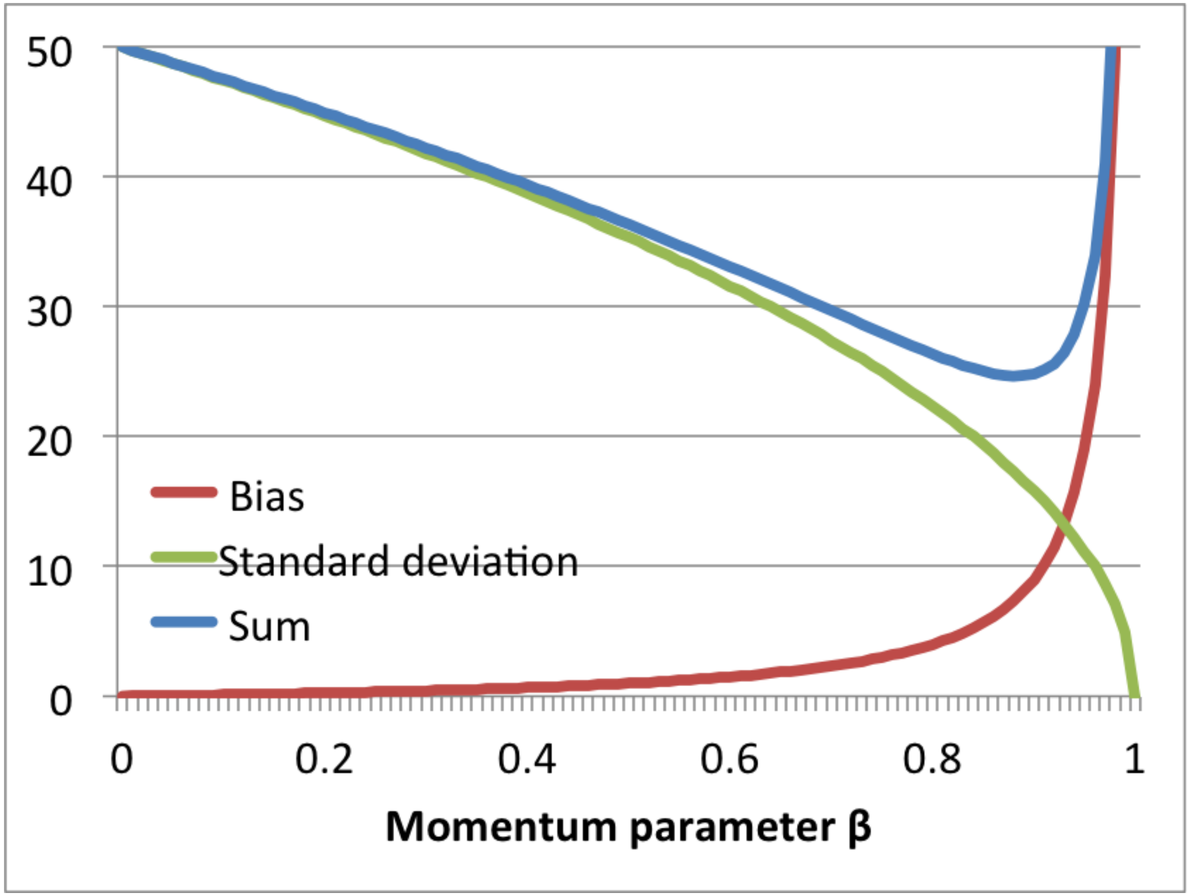


Real data experiments: CIFAR-10

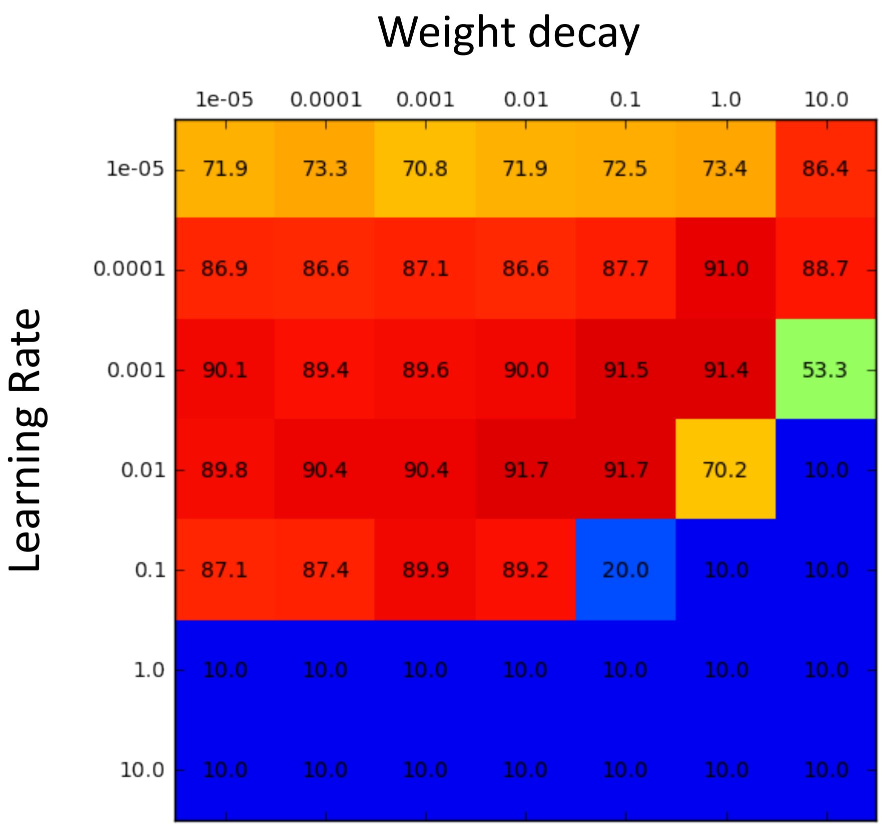


Why does Signum work better?

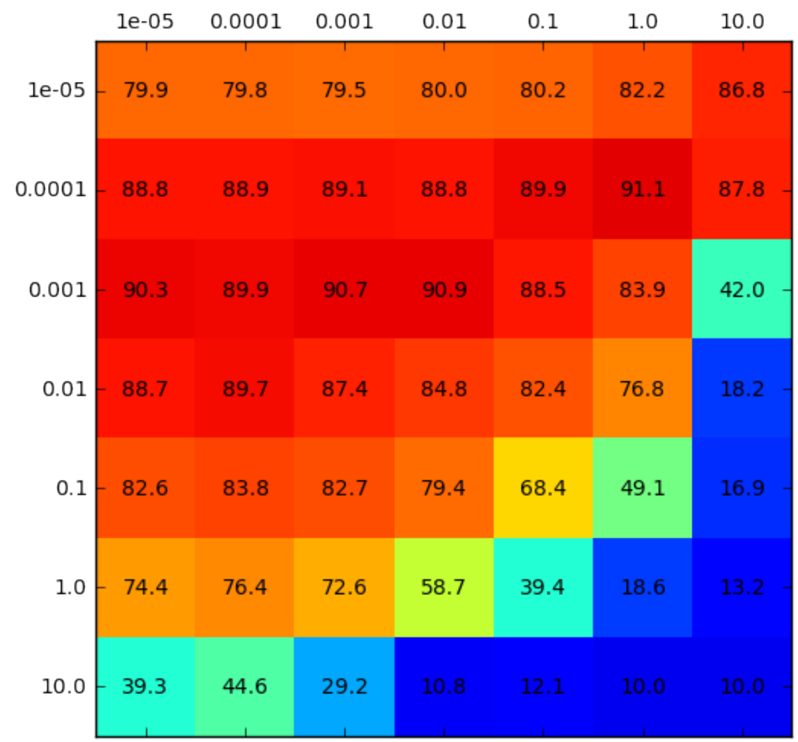
- One plausible explanation: bias-variance trade-off



Stability to hyper-parameter choices

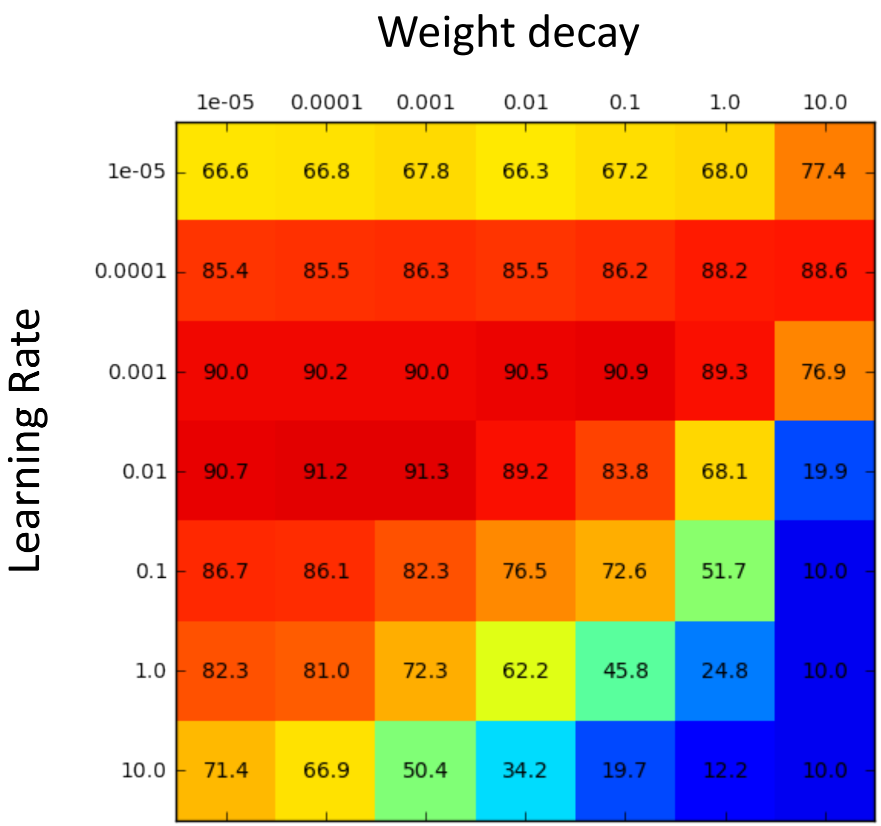


SGD /w momentum

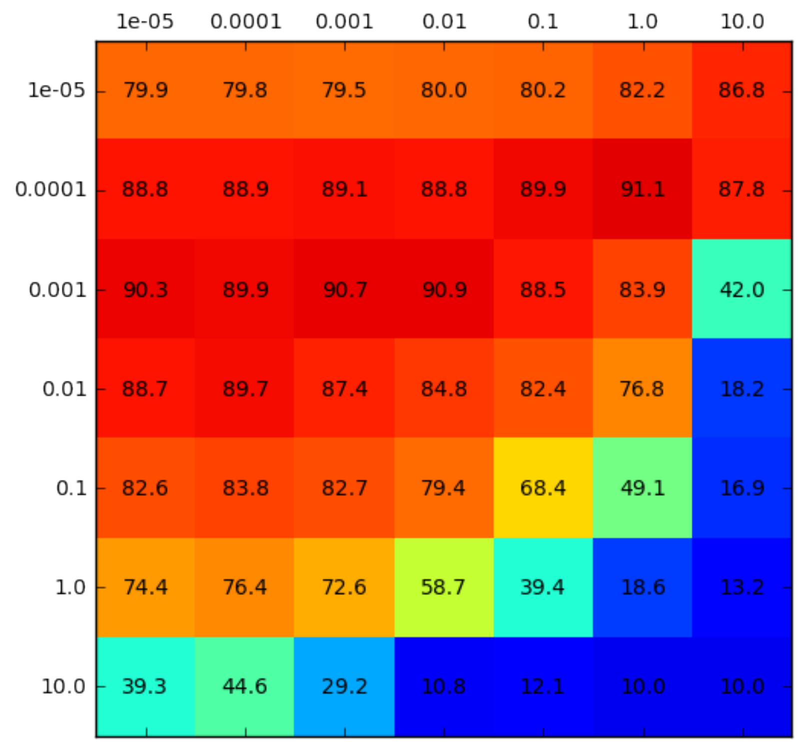


Signum

Stability to hyper-parameter choices



Adam



Signum

Other interesting properties

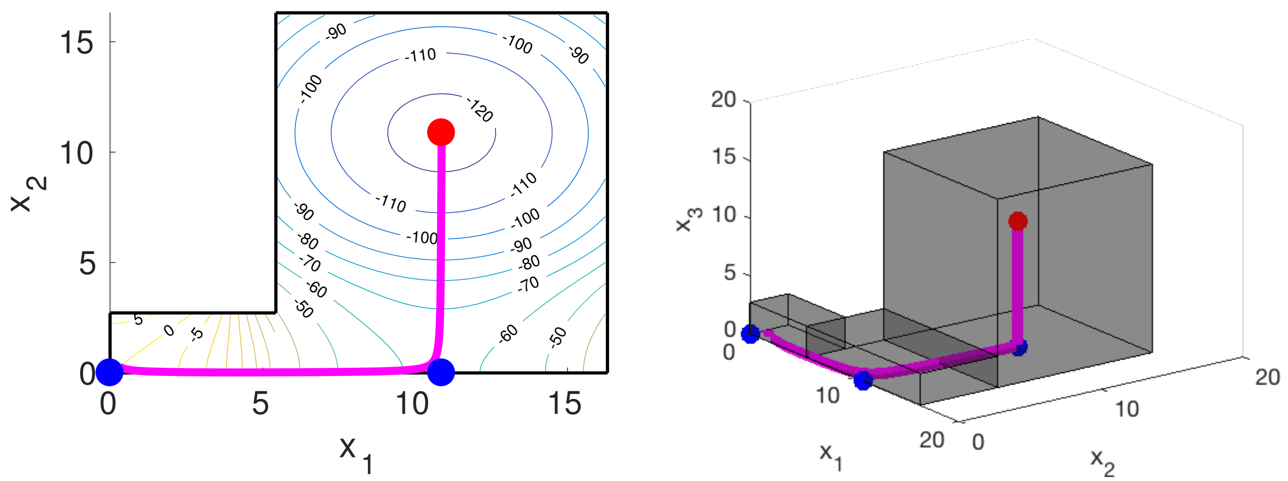
- 1-bit update
    - Gradient compression? [Seide et. al., 2014]
    - Communication savings in multi-machine training.

- Move by a fixed amount each step
    - Solve the “Vanishing and exploding gradient” problem?
    - Think “plataeu and cliff”

- Never collapse to a point
    - Push you away from saddle point!

The tube example

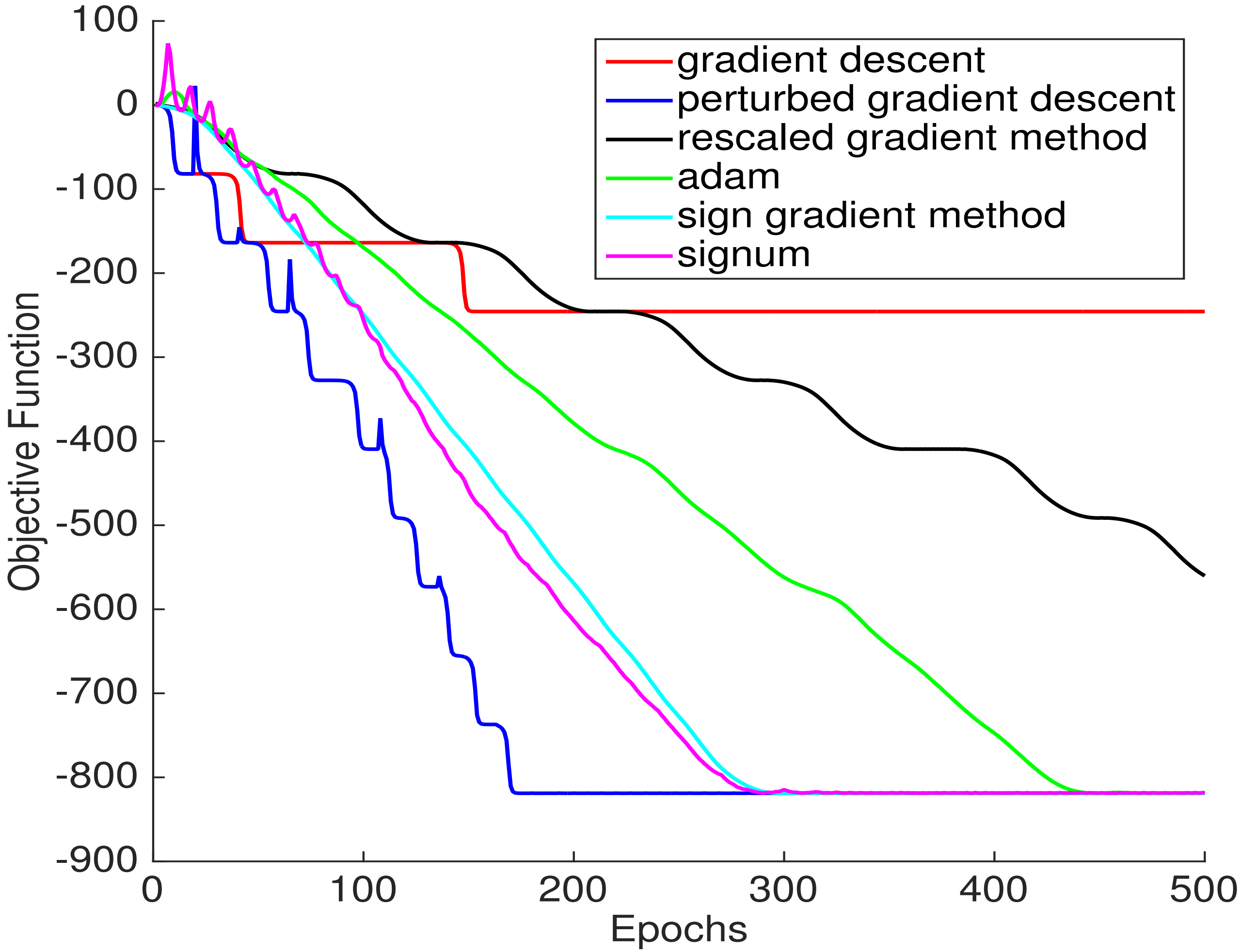
- Extracted from [Du et. al. 2017]: Gradient descent "take forever" to escape saddle point!



(a) Contour plot of the objective function and tube defined in 2D.

(b) Trajectory of gradient descent in the tube for $d = 3$.

Is SignSGD/Signum escaping saddle points?



Conclusion

- SignSGD and Signum optimizers seem to be an interesting class of algorithm
- Very practical, insensitive to tuning parameters
- Towards understanding the success of Adam
- Link to the paper: https://jeremybernste.in/projects/amazon/signum.pdf

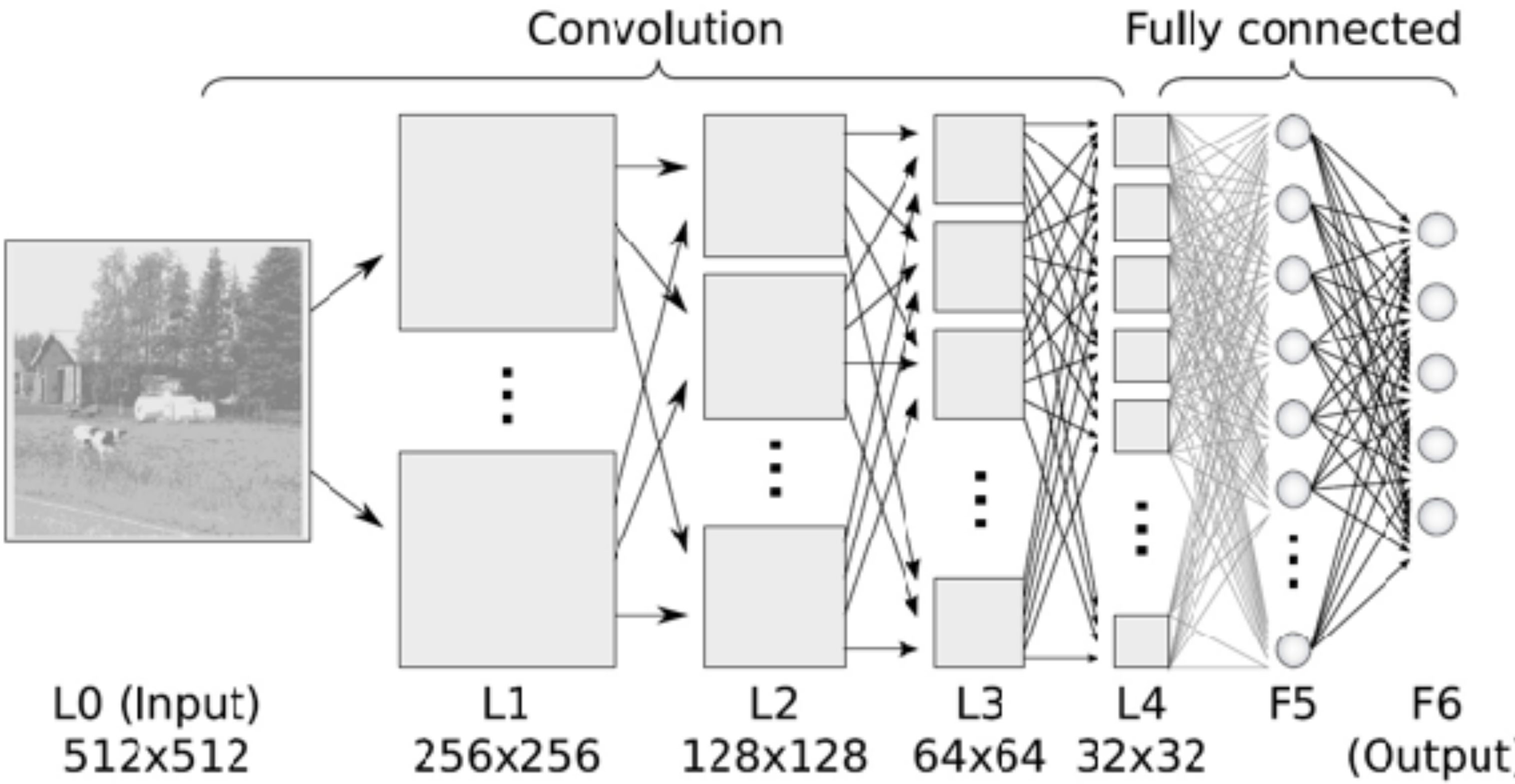
Part II:
Effortless deep learning with

GLUON

gluon.mxnet.io

Deep Learning: Applications

A typical ConvNet model



Motivation

- Deep learning seems to be quite useful and popular.
- Need to deal with GPU, multithreading, network bandwidth; consider data iterator, and so on.
- Can be very intimidating for folks with little CS-system experience.

As researchers, we like writing

- Matlab code
- R code
- Python code (numpy)

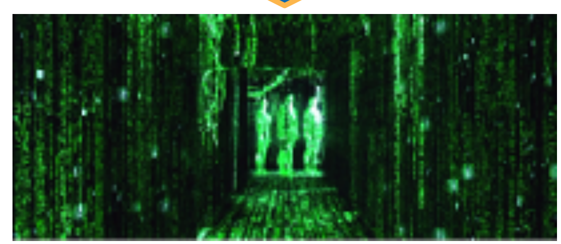
The languages we like are slow…

Call down to fast libraries

mxnet TensorFlow

PYTORCH theano

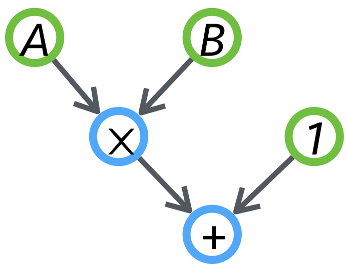
`model.forward(data)`



[crazy GPU stuff]

amazon web services

Declarative (Symbolic) Programs




```
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*

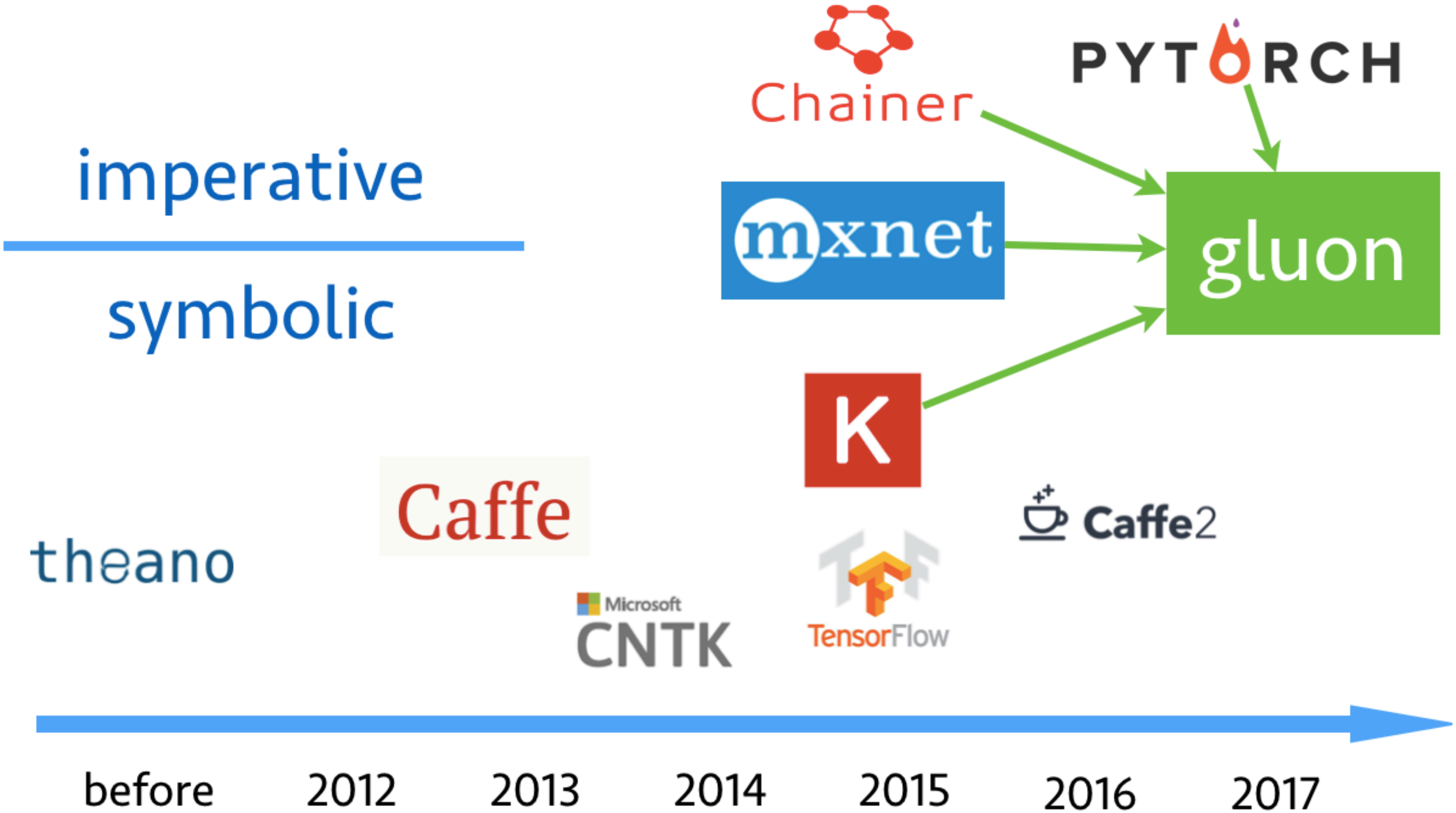
- **Advantages**:
  - Opportunities for optimization
  - Easy to serialize models
  - More portable across languages
- **Disadvantages**:
  - Hard to debug
  - Unsuitable for dynamic graphs
  - Can't use native code

Imperative Programs

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

Easy to debug, easy to code

- **Advantages**
  - Straightforward and flexible
  - Use language natively (loops, control flow, debugging)
- **Disadvantages**
  - Hard to optimize
    - **Fixed with JIT compiler! (covered in this tutorial)**


imperative
symbolic
Chainer
PYTORCH
mxnet
gluon
K
Caffe
theano
Caffe2
Microsoft
CNTK
TensorFlow
before
2012
2013
2014
2015
2016
2017

What am I gonna do?

- Installation

- Let's look at **actual code** in Jupyter notebooks

- If you have a laptop with you:
  - Go to http://gluon.mxnet.io/
  - And follow along!

Installation

```
pip install --upgrade pip
pip install --upgrade setuptools
pip install mxnet -pre --user

# if  the machine has GPU running CUDA

pip install mxnet-cu80
```

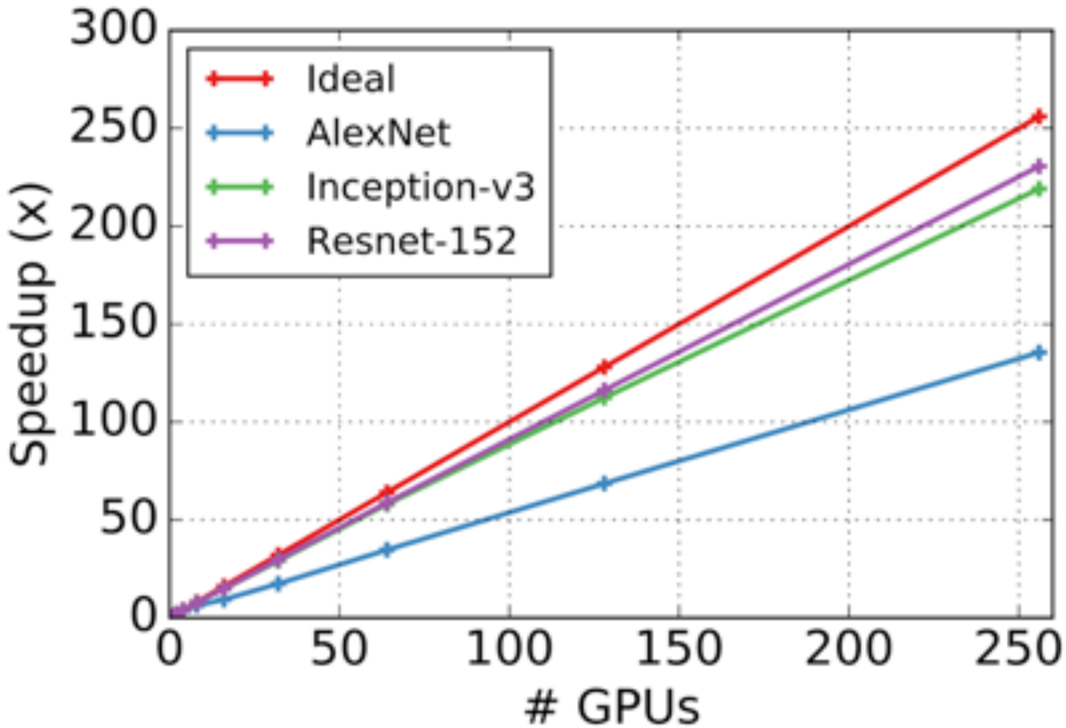
Download the notebooks

git clone https://github.com/zackchase/mxnet-the-straight-dope.git

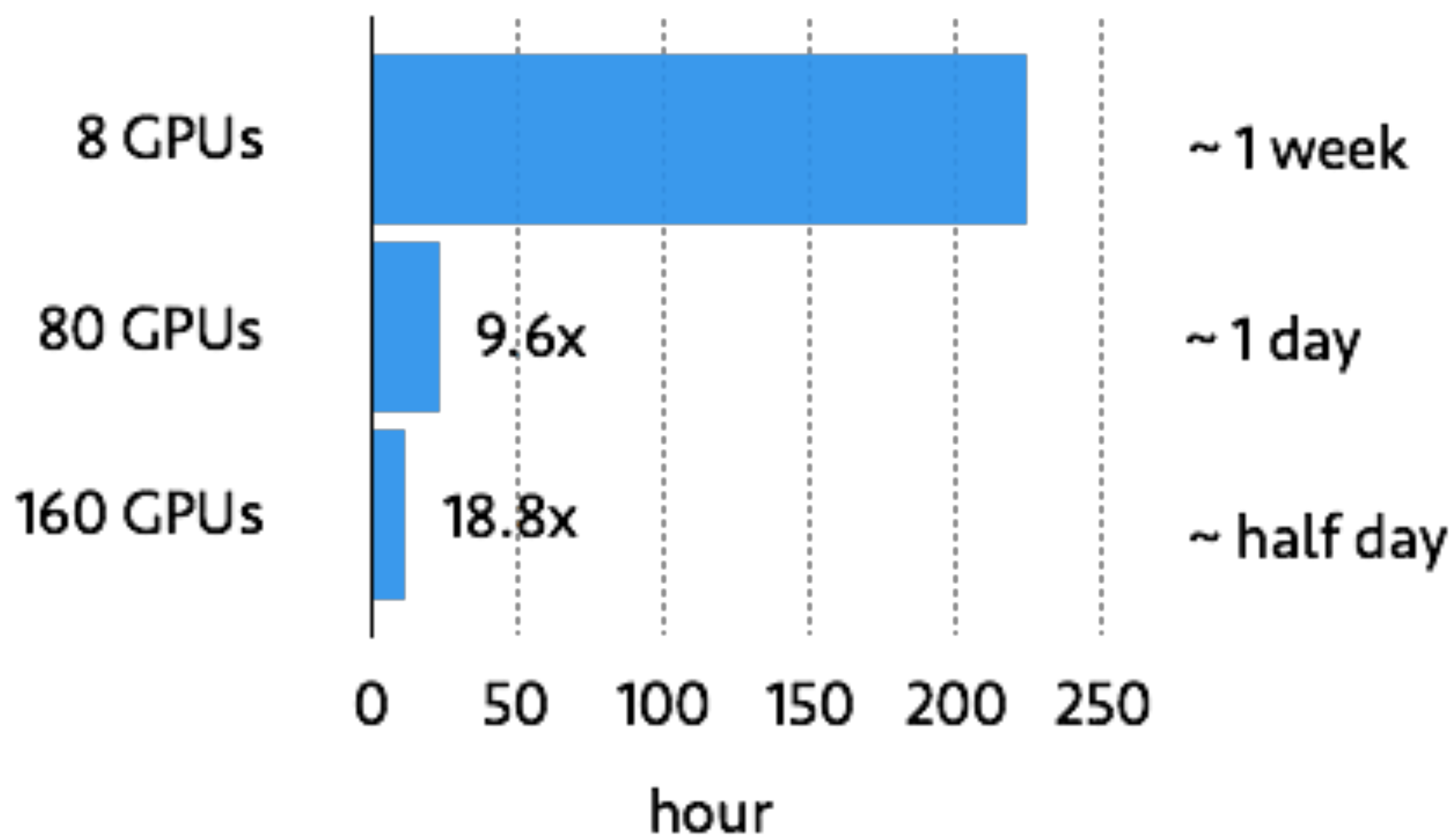
On our list today:

- ConvNet with Gluon
- Hybridization and MultiGPU training.

Multi-machine training: Linear scaling with the number of GPUs



Time to achieve the State-of-The-Art Performance on ImageNet



Conclusion

- About one week of getting used to…
- Flexible and versatile for research purposes
  - Can use things that are build-in
  - Also implement things from scratch
- “hybridization” and “multi-gpu” support makes things faster

Acknowledgement

- Gluon API is primarily developed by Eric Junyuan Xie, Amazon Scientist
- Open Book Project: “Deep learning: The Straight Dope” is led by: Zack Chase Lipton (joining CMU Tepper as an Asst. Prof)
- The Apache Mxnet project is led by Mu Li (PhD CMU, now Amazon Principal Scientist).
- Plus a lot of community effort.

Resources

- Gluon book: https://gluon.mxnet.io
  - Github: https://github.com/zackchase/mxnet-the-straight-dope
- Apache Mxnet:
  - https://github.com/apache/incubator-mxnet/
- English-version discussion forum: https://discuss.mxnet.io/
- Chinese-version discussion forum: https://discuss.gluon.ai/

Sketch of proof (Part I)

- Taylor expansion

$$f_{k+1} - f_k \le g_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|_\infty^2$$
$$= -\delta_k g_k^T \mathrm{sign}(v_k) + \delta_k^2 \frac{L}{2}$$

- Error decomposition

$$-\delta_k \|g_k\|_1 + 2\delta_k \sum_{i=1}^{d} |g_k[i]| \, \mathbb{I}[\mathrm{sign}(v_k[i]) \ne \mathrm{sign}(g_k[i])]$$

Sketch of proof (Part II)

- Take expectation and control the error

- Telescoping sum