Hardware Accelerators for Machine Learning

CS217

10:30am–11:50am

Packard 101

Staff

- Instructors
  - Kunle Olukotun
  - Ardavan Pedram
- TAs
  - Nandita Bhaskhar
  - Nathan Zhang
- Office hours, contact info, on course information sheet

CS217 is a New Course

- Goal
  - Design of hardware architectures for accelerating Machine Learning (ML)
- Approach
  - Understand key ML characteristics
  - Look at hardware to exploit
  - Guest lectures with industry and academic examples

CS217 Topics

- Machine Learning & Deep Learning from compute perspective
- Building blocks of traditional ML kernels
- Linear Algebra kernel characteristics
- Evaluating performance, parallelism, locality
- DNN Inference acceleration
- DNN Training acceleration and its challenges
- Scaling training
- ML Benchmarks

Administrivia

- Everything is on the class Web site

  piazza.com/stanford/fall2018/cs217/home

- Communication
  - Use Piazza, email, office hours
  - But definitely prefer Piazza!

Class Structure

- Class will be taught from notes
  - Research papers
- Do your own paper reviews
- Working in groups on programming assignments and project
  - Use research prototype language: *Spatial*
  - Optimize ML algorithms for parallelism and locality
  - Group size <= 4

Grading

- 30% Paper reviews
- 35% Programming assignments (3)
- 35% Final project

- See handout for late day policy

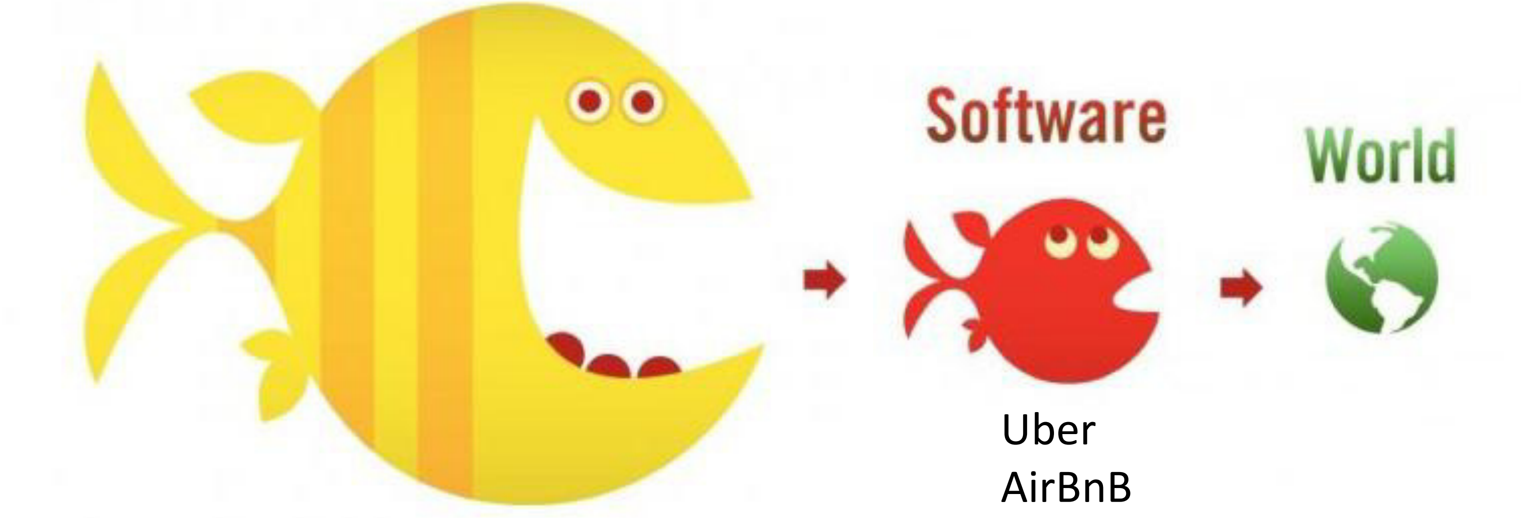
Prerequisites

- What the information sheet says
  - EE180 or CS149, CS229 is ideal but not required

- What we mean
  - Basic knowledge of architecture and performance programming and basics of machine learning
  - A certain programming maturity
    - Programs are modest
    - Comfortable picking up new languages/systems
    - Able to deal with "hardhat" debugging environments

… is Eating the World

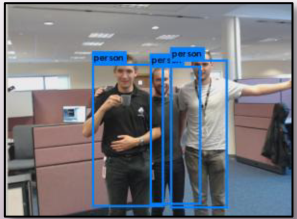
AI
Machine Learning



Uber
AirBnB
Tesla

Neural Networks: Many Applications

Image Classification



Object Detection

Semantic Segmentation

Computer Vision
CNNs

Speaker Diarization

Speech Recognition

Speech Recognition
RNNs, LSTMs

Translation

Sentiment Analysis

Natural Language Processing
Sequence to sequence

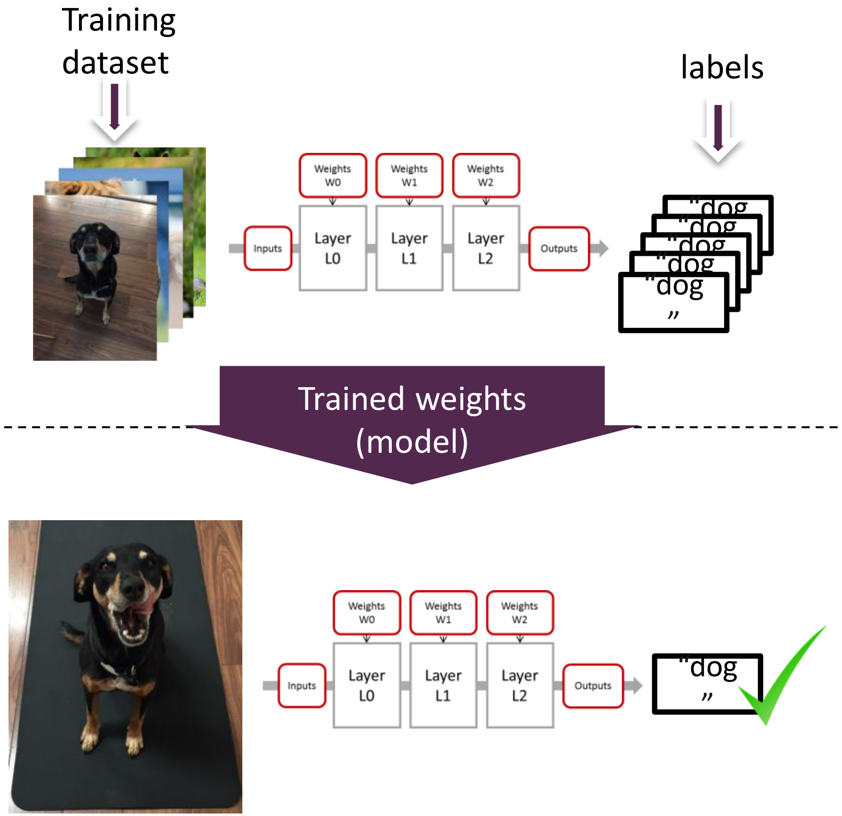
Recommender

GamePlay

Many more emerging…

Others

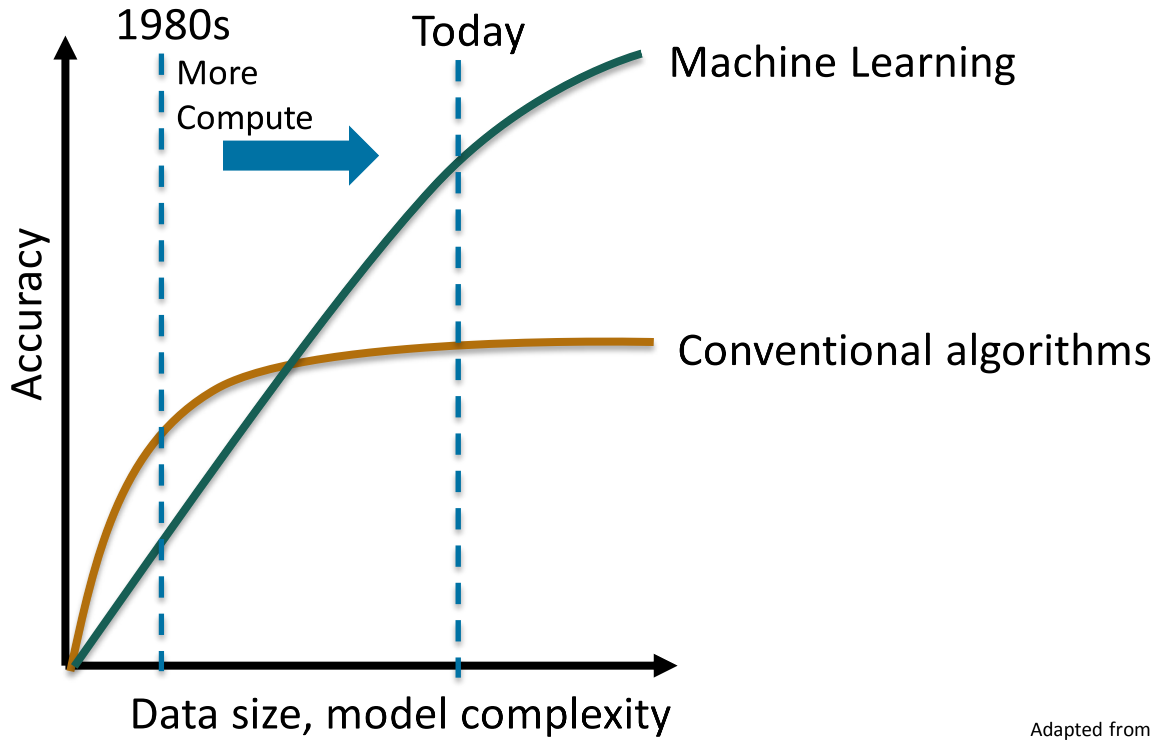
From Training to Inference



Training
Process for a machine to ***learn*** by optimizing models (weights) from labeled data

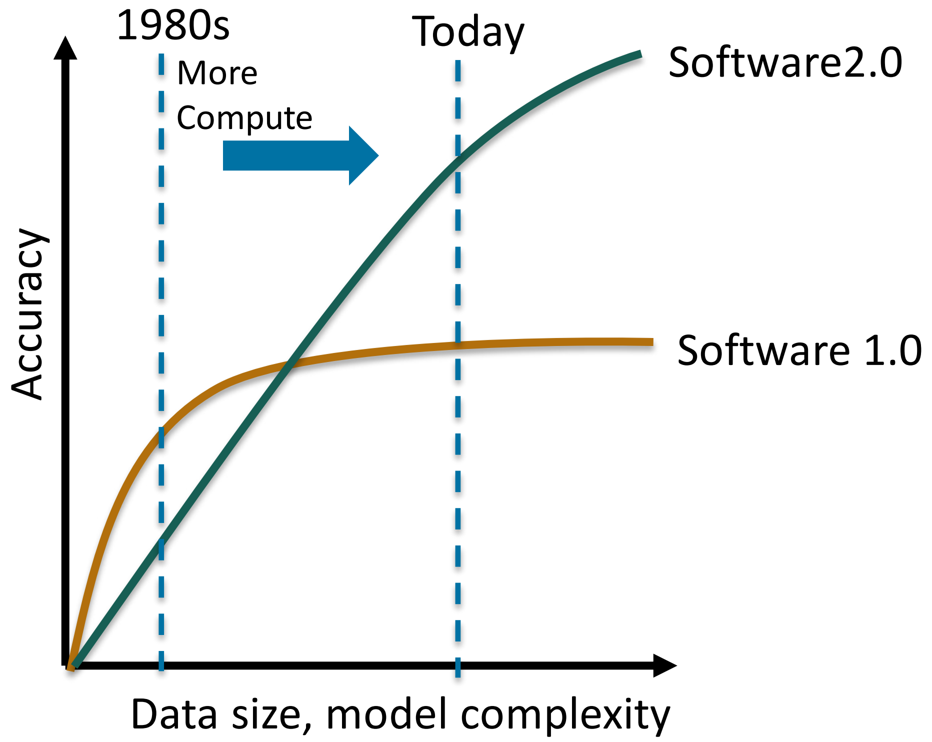
Inference
Using trained models to predict or estimate outcomes from new inputs

Machine Learning Today



Adapted from Jeff Dean
HotChips 2017

Machine Learning Today

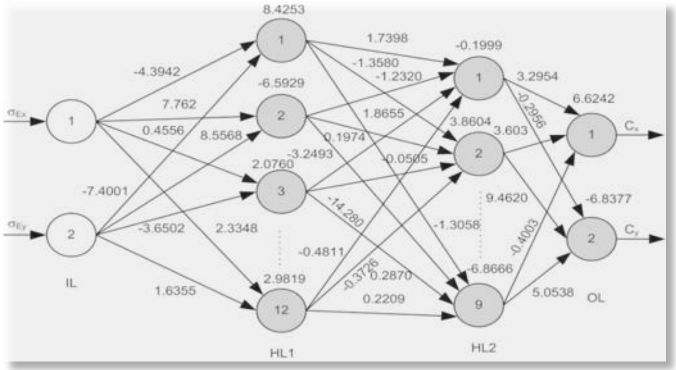


Adapted from Jeff Dean
HotChips 2017

Software 1.0 vs Software 2.0

```
template<typename InputIterator1, typename InputIterator2, typename NumericT>
NumericT  inner_product(InputIterator1 start1,
                        InputIterator1 end1,
                        InputIterator2 start2,
                        NumericT       startval) const {
  for (; start1 != end1; ++start1, ++start2)
    startval += ( (*start1) * (*start2) );
  return startval;
}
// ..
vector_type vec1, vec2;
vector_type::numeric_t val;
val = inner_product(vec1.begin(), vec1.end(), vec2.begin(),
                    vector_type::numeric_t(0));
```

- Written in code (C++, …)
- Requires domain expertise
    - Decompose the problem
    - Design algorithms
    - Compose into a system



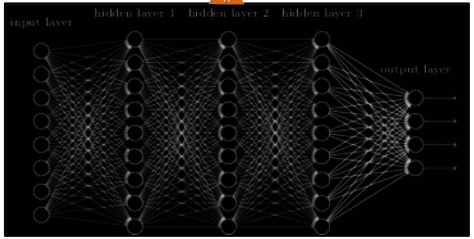
- Programmer input: training data
- Written in the weights of a neural network model by optimization

Andrej Karpathy. Scaled ML 2018 talk

Software 2.0 is Eating Software 1.0

Easier to Build and Deploy

- Build products faster
- Predictable runtimes and memory use: easier qualification



1000x Productivity: Google shrinks language translation code from 500k LoC to 500

Classical Problems

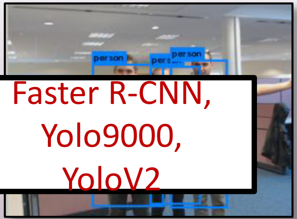
- Data cleaning (Holoclean.io)
- Self-driving DBMS (Peloton)
- Self-driving networks (Pensieve)

Popular Neural Networks

Computer Vision
CNNs


ResNet50, VGG, AlexNet, InceptionV3

Image Classification



Faster R-CNN, Yolo9000, YoloV2

Object Detection

Mask-R-CNN, SSD

Semantic Segmentation

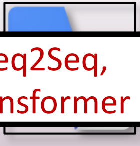
Speech Recognition
RNNs, LSTMs


Speaker Diarization


DeepSpeech2

Speech Recognition

Natural Language Processing
Sequence to sequence


Seq2Seq, Transformer

Translation


Seq-CNN

Sentiment Analysis

Others


NCF

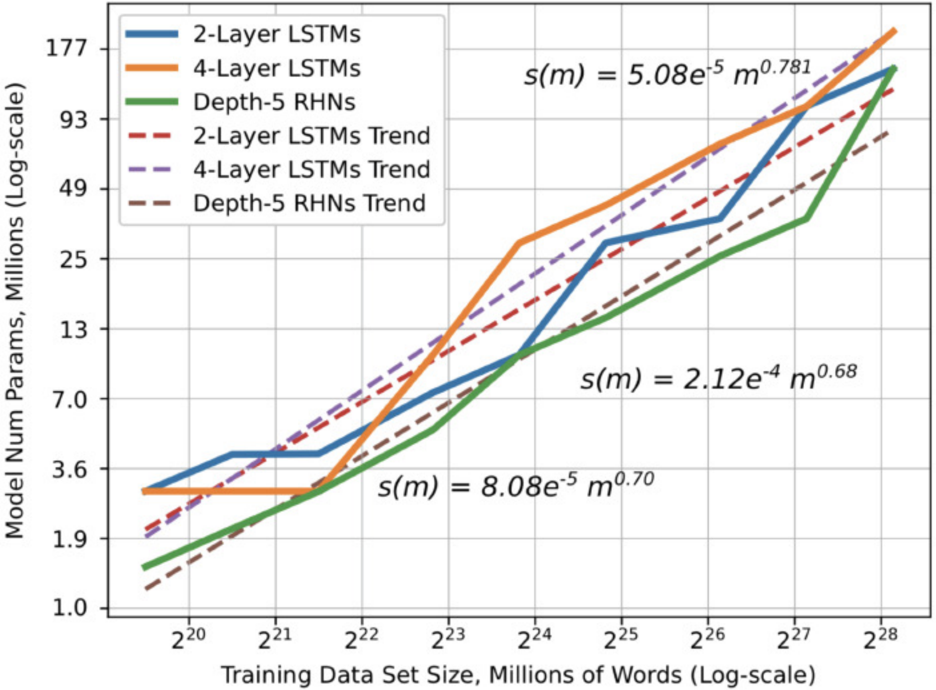
Recommender

MiniGo, DeepQ, A3C

GamePlay

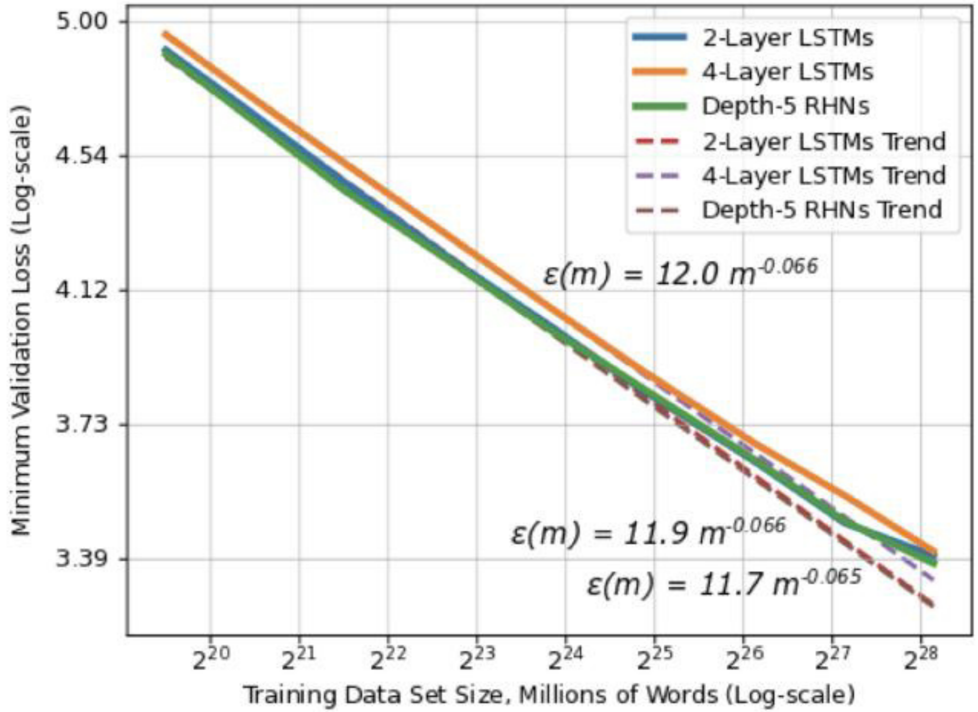
Adopted from MLPerf, Fathom, TDP

Deep Neural Networks: More Data ⇒ Larger Model



Hestness et al., "Deep Learning Scaling is Predictable, Empirically," arXiv:1712.00409

Deep Neural Networks: More Data ⇒ More Accuracy



Hestness et al., "Deep Learning Scaling is Predictable, Empirically," arXiv:1712.00409

Hardware Limits Development of ML

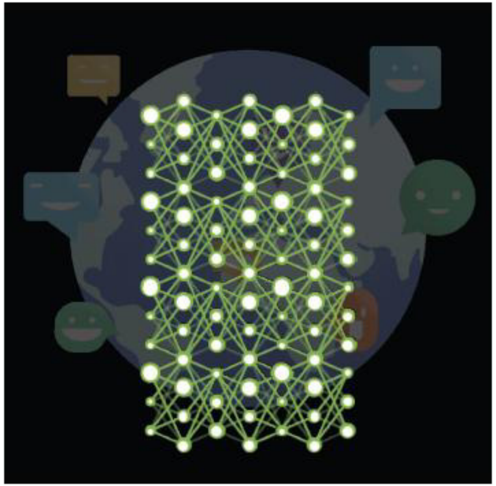
7 ExaFLOPS
60 Million Parameters

2015 - Microsoft ResNet
Superhuman Image Recognition

20 ExaFLOPS
300 Million Parameters

2016 - Baidu Deep Speech 2
Superhuman Voice Recognition

100 ExaFLOPS
8.7 Billion Parameters



2017 - Google Neural Machine Translation
Near Human Language Translation

ML Training is Limited by Computation

From EE Times – September 27, 2016

“Today the job of training machine learning models is limited by compute, if we had faster processors we’d run bigger models…in practice we train on a reasonable subset of data that can finish in a matter of months. We could use improvements of several orders of magnitude – 100x or greater.”

Greg Diamos, Senior Researcher, SVAIL, Baidu

Accelerators for ML



CPU	GPU	FPGA	TPU	Next
Threads SIMD	Massive Threads SIMD HBM	LUTs DSP BRAM	MM Unit BRAM	???

Power and Performance

Performance

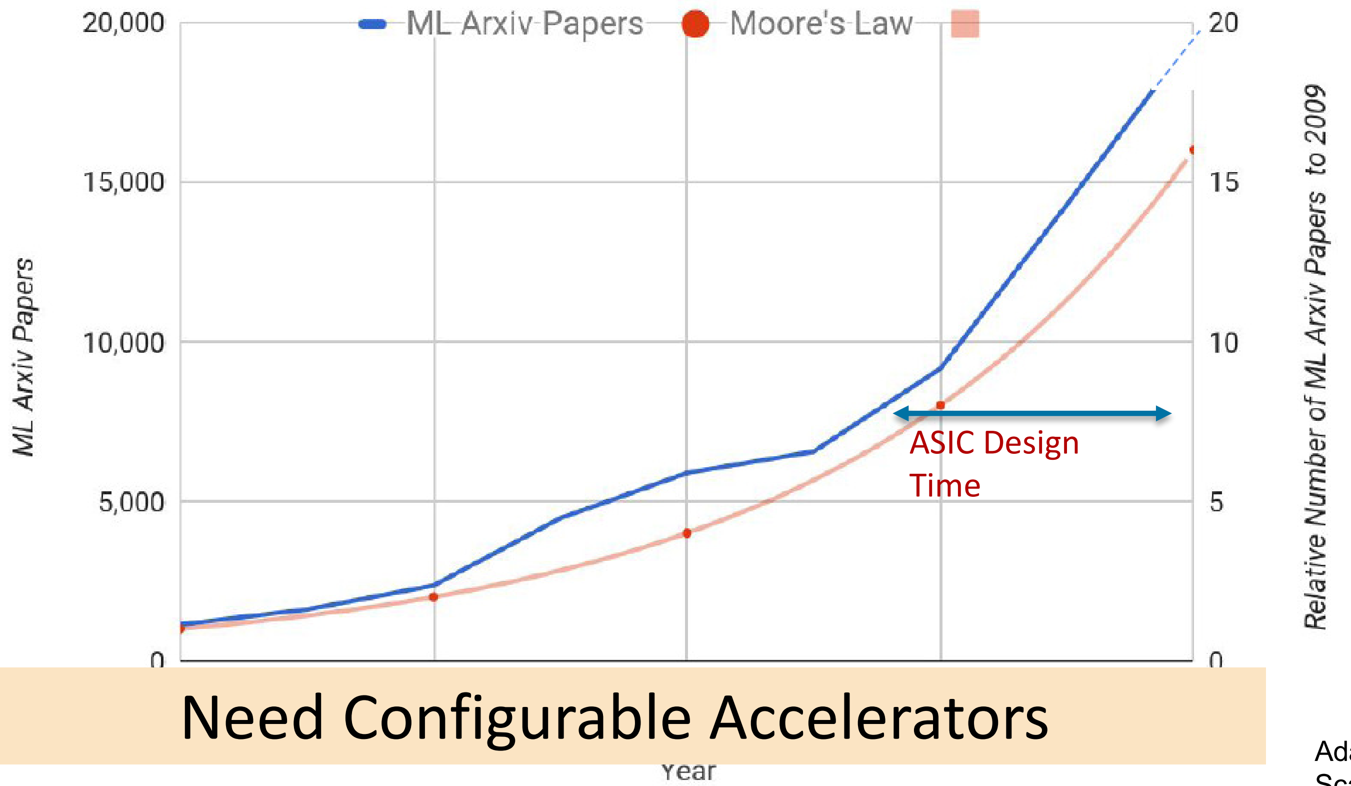
Energy efficiency

$$Power = \frac{Ops}{second} \times \frac{Joules}{Op}$$

FIXED

Specialization ⇒ better energy efficiency

What to Accelerate? ML Arxiv Papers Per Year



Need Configurable Accelerators

Adapted from Jeff Dean Scaled ML 2018

Key Questions

- How do we speed up machine learning by 100x?
  - Moore's law slow down and power wall
  - >100x improvement in performance/watt
  - Enable new ML applications and capabilities
  - Make ML easier to use (e.g. neural architecture search, Snorkel)

- How do we balance performance and programmability?
  - ASIC-like performance/Watt
  - Processor-like flexibility

- Need a "full-stack" codesign
  1. ML Algorithms
  2. Compilers
  3. Hardware

Computational Models

- Software 1.0 model
  - Deterministic computations with algorithms
  - Computation must be correct for debugging
- Software 2.0 model
  - Probabilistic machine-learned models trained from data
  - Computation only has to be statistically correct
- Creates many opportunities for improved performance

Relax, It's Only Machine Learning

- Lots of parallel computation!

- Relax precision: small integers are better
  - HALP [De Sa, Aberger, *et. al.*]
- Relax synchronization: data races are better
  - HogWild! [De Sa, Olukotun, Ré: *ICML 2016*, ICML Best Paper]
- Relax cache coherence: incoherence is better
  - [De Sa, Feldman, Ré, Olukotun: *ISCA 2017*]
- Relax communication: sparse communication is better
  - [Lin, Han et. al.: *ICLR 18*]