AI at Facebook Datacenter Scale

Misha Smelyanskiy
Facebook AI System SW/HW Co-design Team
Nov-29 2018

Hardware Accelerators for Machine Learning (CS 217)
Stanford University, Fall 2018

AI System Co-design (ASyC) Team

Our mission: high performance numerical and architectural SW optimizations, HW performance modeling and recommendations through Machine Learning-driven Co-design

What we do

- Server-side algorithmic, numerical and performance optimizations
- Drive AI hardware roadmap and co-design new AI hardware
- Performance modeling and simulation tools

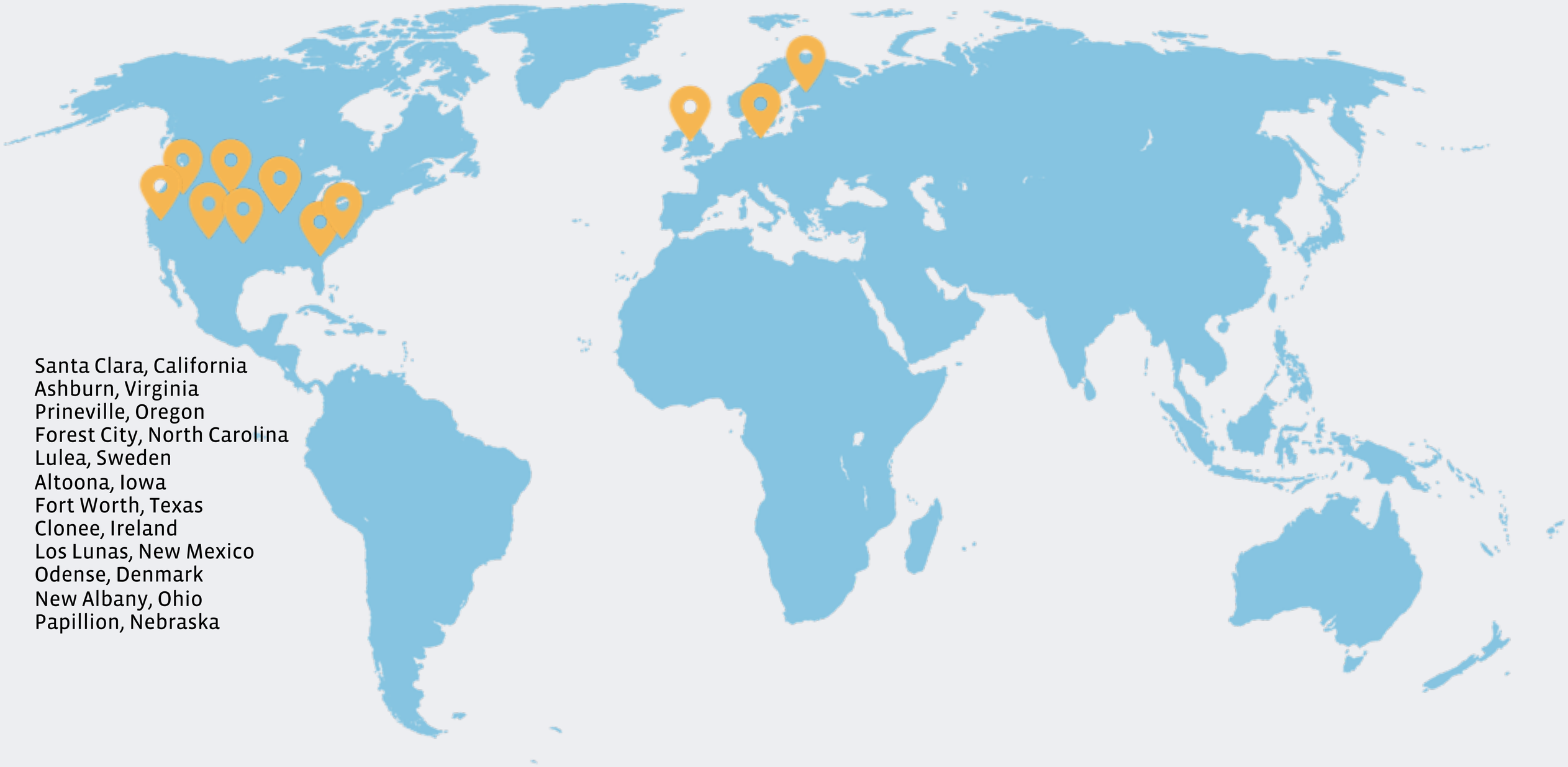
Introduction


How do you scale to over
2 Billion People
?

Scaling Challenges / Opportunities

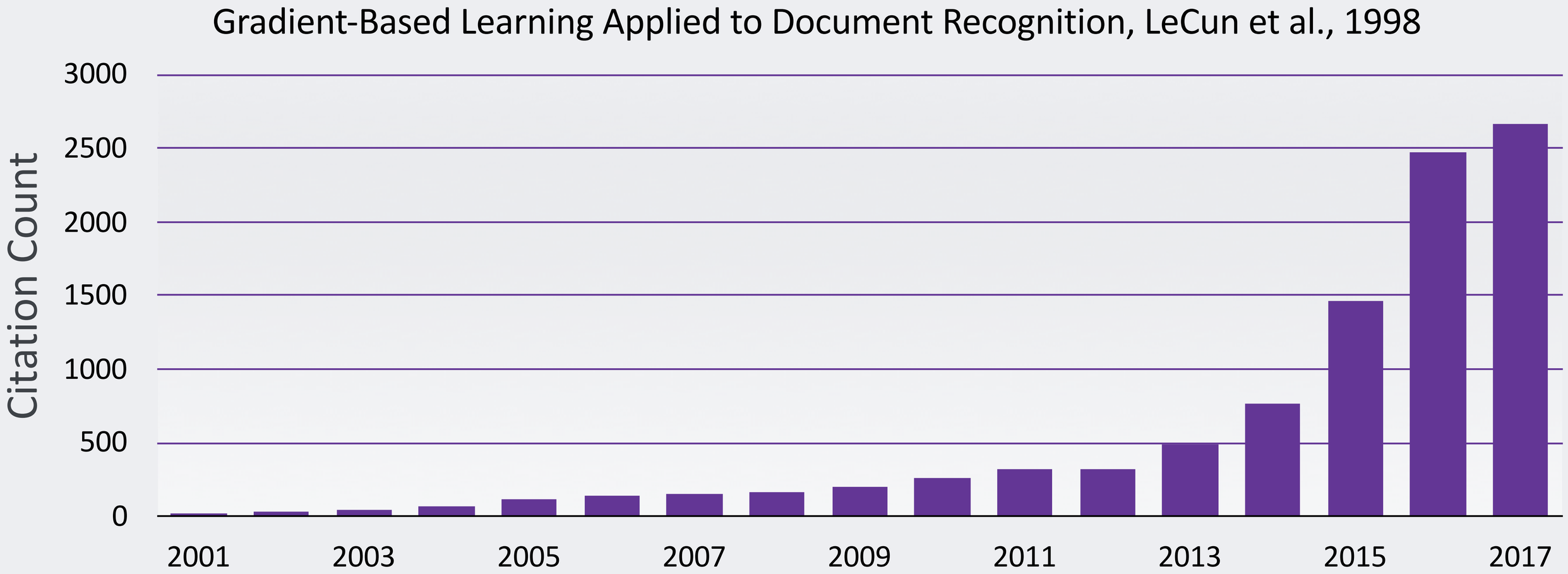
Lots of Data

Lots of Compute

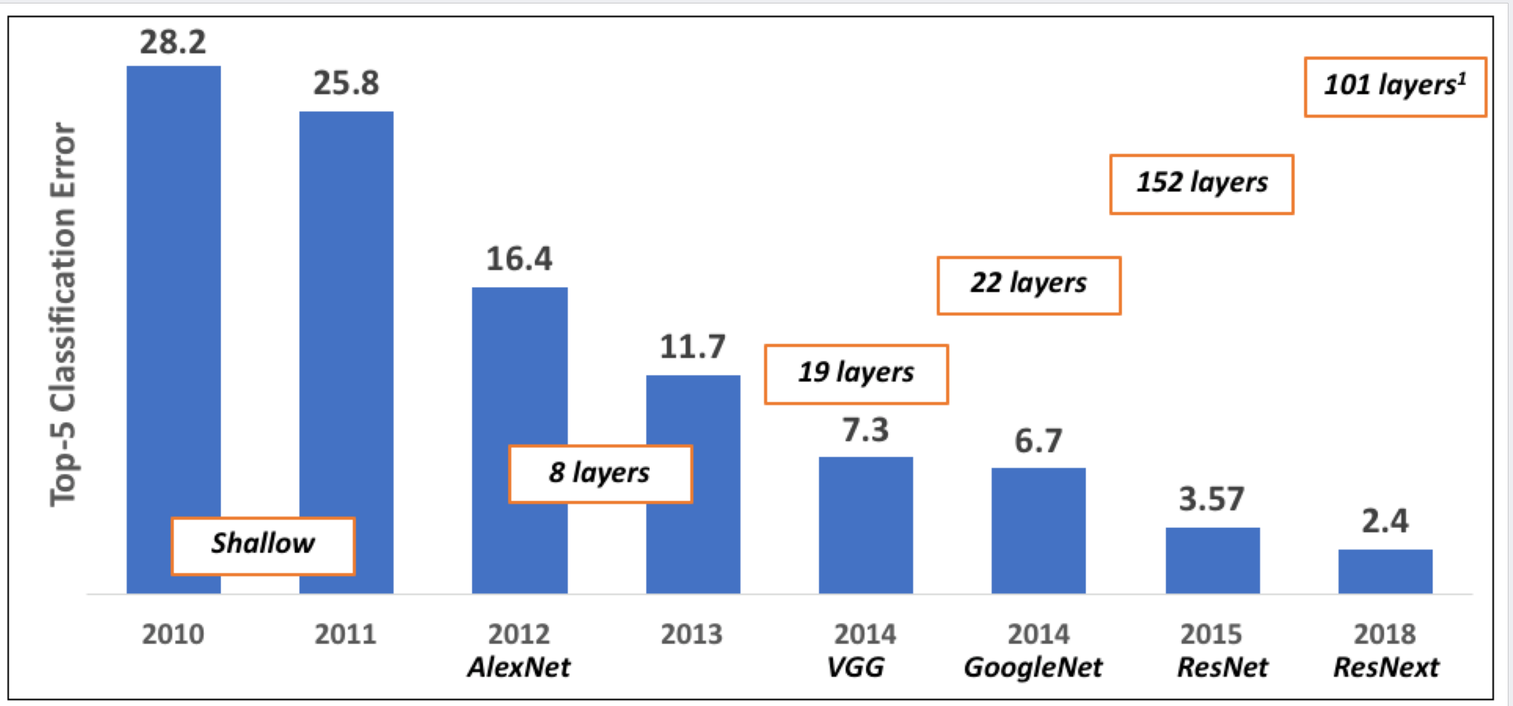


Santa Clara, California
Ashburn, Virginia
Prineville, Oregon
Forest City, North Carolina
Lulea, Sweden
Altoona, Iowa
Fort Worth, Texas
Clonee, Ireland
Los Lunas, New Mexico
Odense, Denmark
New Albany, Ohio
Papillion, Nebraska

Exponential Growth of AI



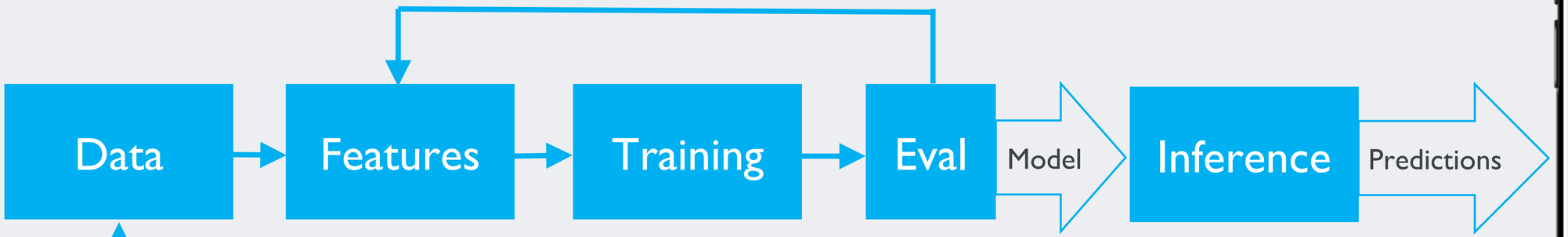
AI Growth and Its Drivers

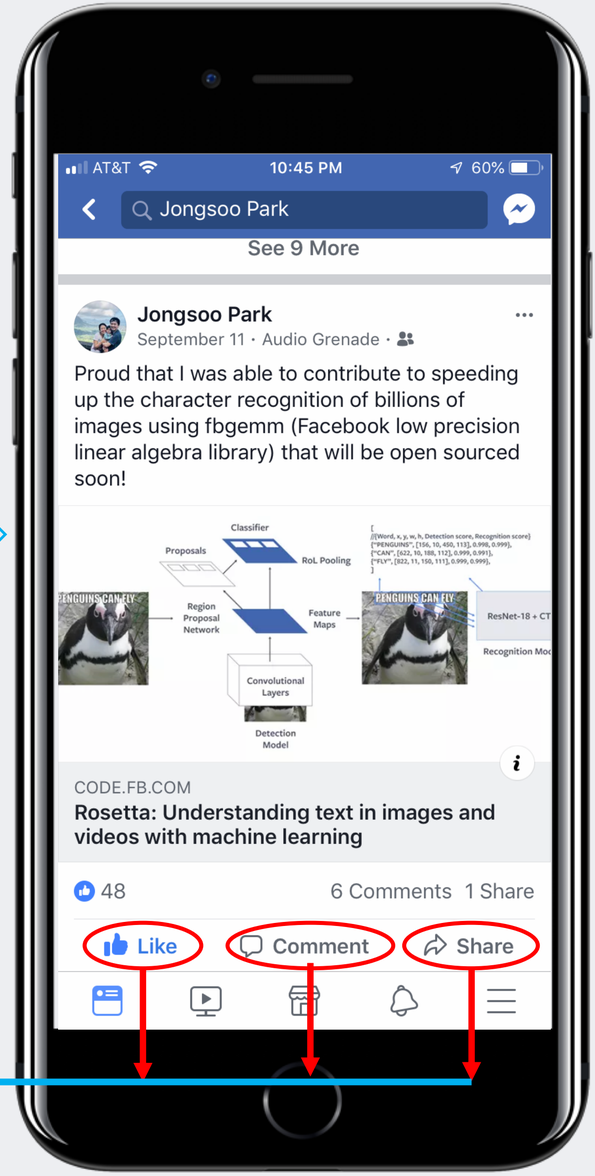


- Big and better data
- Better algorithms
- More compute

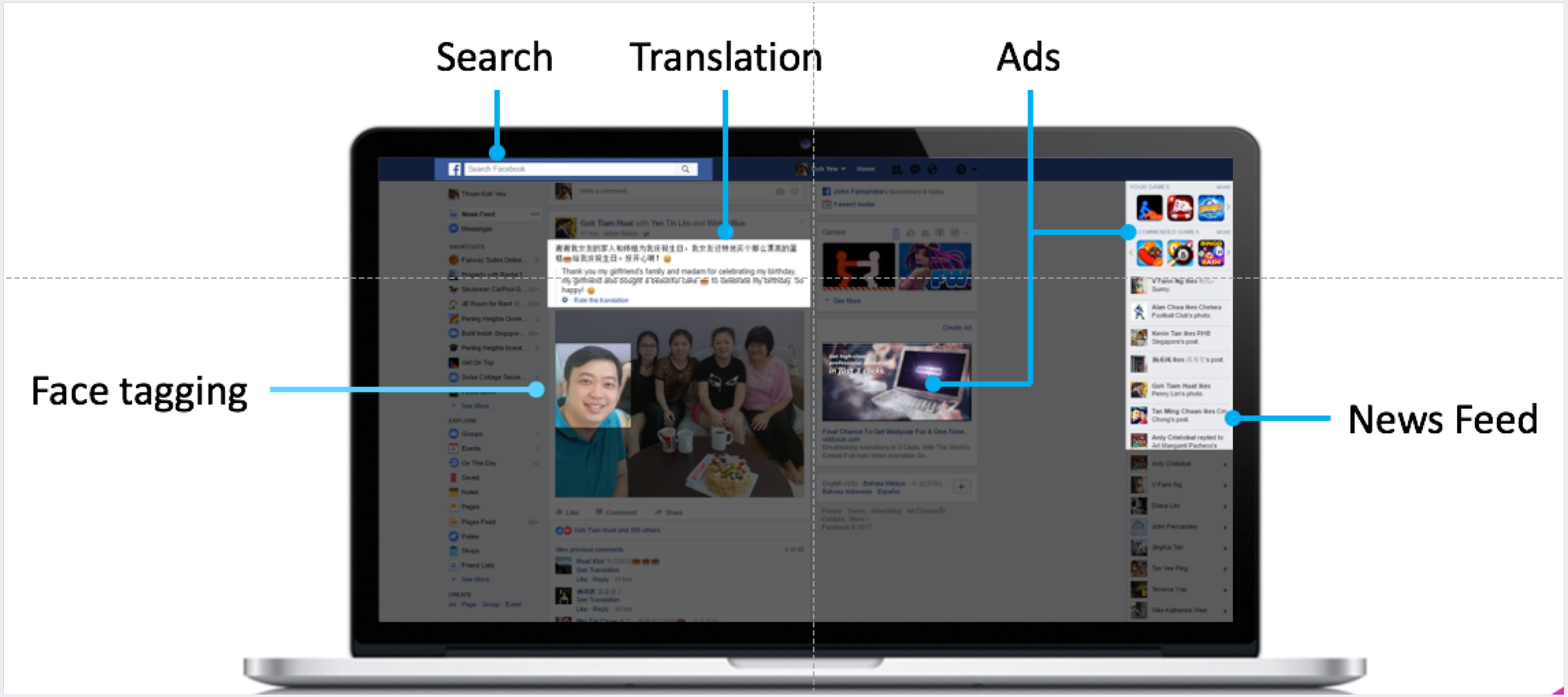
[1] Trained on 1B Instagram images

Machine Learning Execution Flow





How Does FB Use Machine Learning?



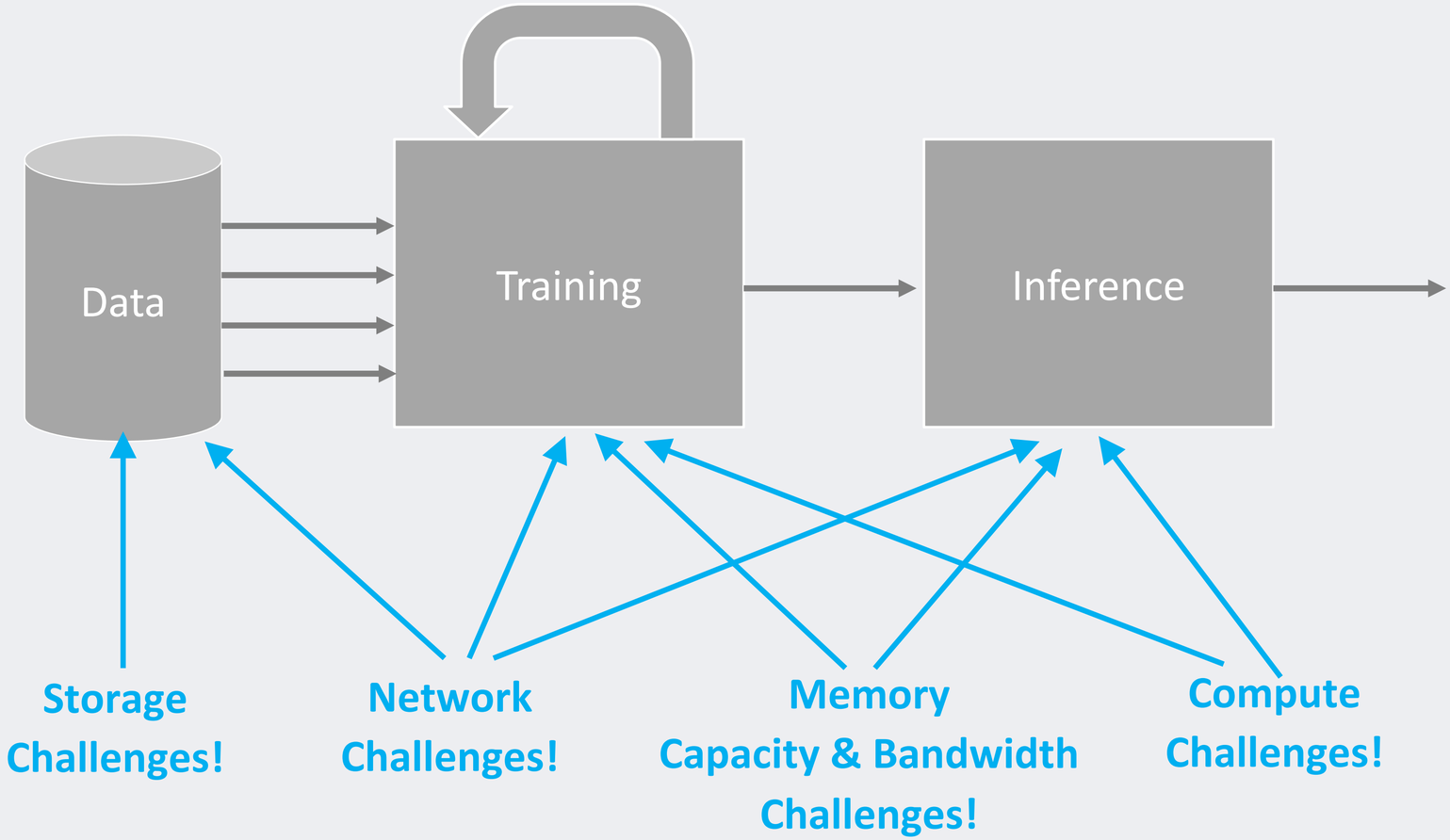
Major Services and Use Cases

Support Vector Machines	Gradient-Boosted Decision Trees	Multi-Layer Perceptron	Convolutional Neural Nets	Recurrent Neural Nets
SVM	GBDT	MLP	CNN	RNN
Facer	Sigma	News Feed	Facer	Language Translation
		Ads	Lumos	Speech Rec
		Search		Content Understanding
		Sigma		

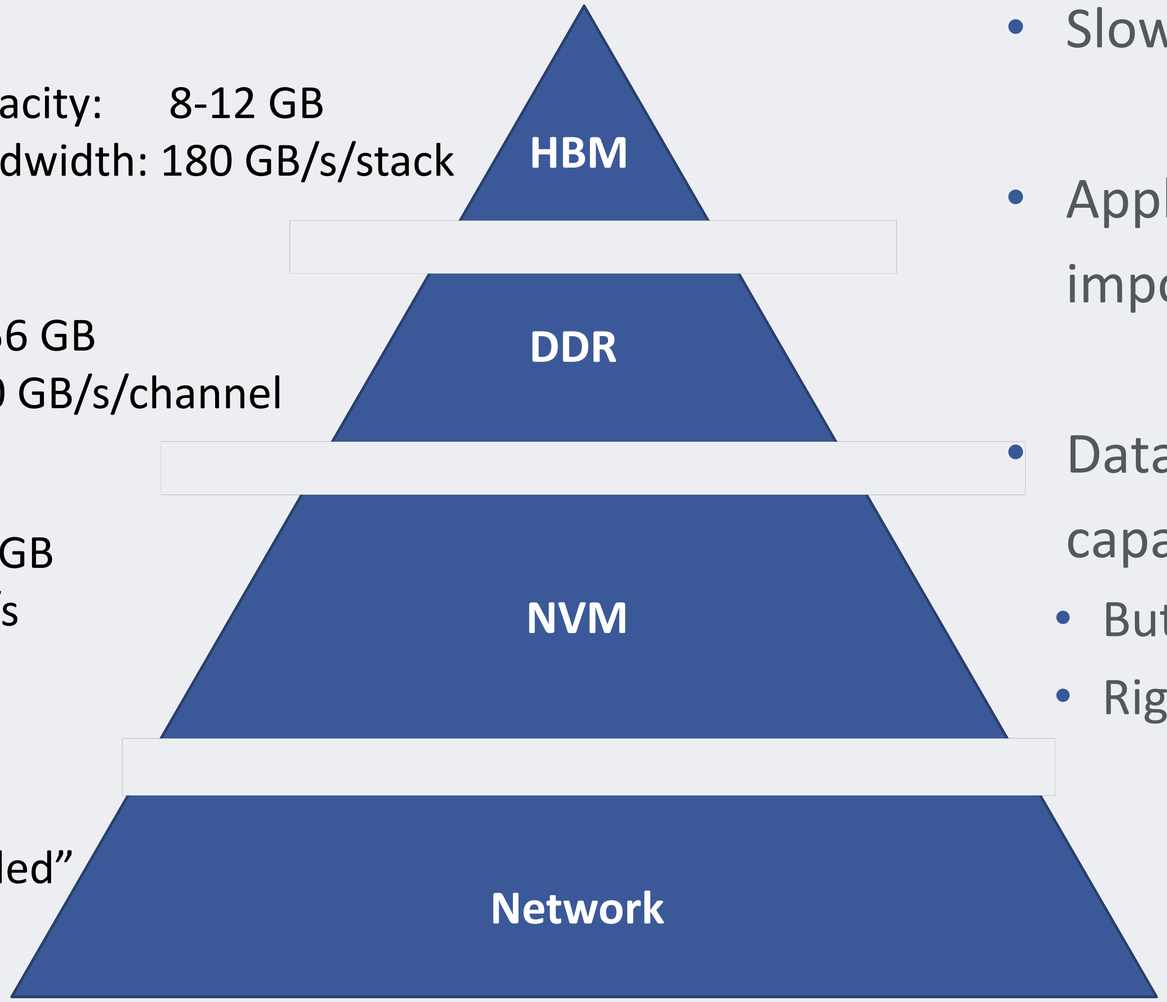
Facebook ML Models

	Training		Inference	
Services	**Resources**	**Duration**	**Resources**	**Relative Capacity**
News Feed	Dual-Socket CPUs	many hours	CPUs	100X
Lumos	GPUs	many hours		10X
Facer	GPUs + Single-Socket CPUs	sub-second		10X
Language Translation	GPUs	days		1X

Infrastructure Challenge



DC “Aggregate” Capacity & Bandwidth



Capacity: 8-12 GB
Bandwidth: 180 GB/s/stack

Capacity: 256 GB
Bandwidth: 20 GB/s/channel

Capacity: 2000 GB
Bandwidth: 3 GB/s

Capacity: ”unbounded”
Bandwidth: ≤ 3 GB/s

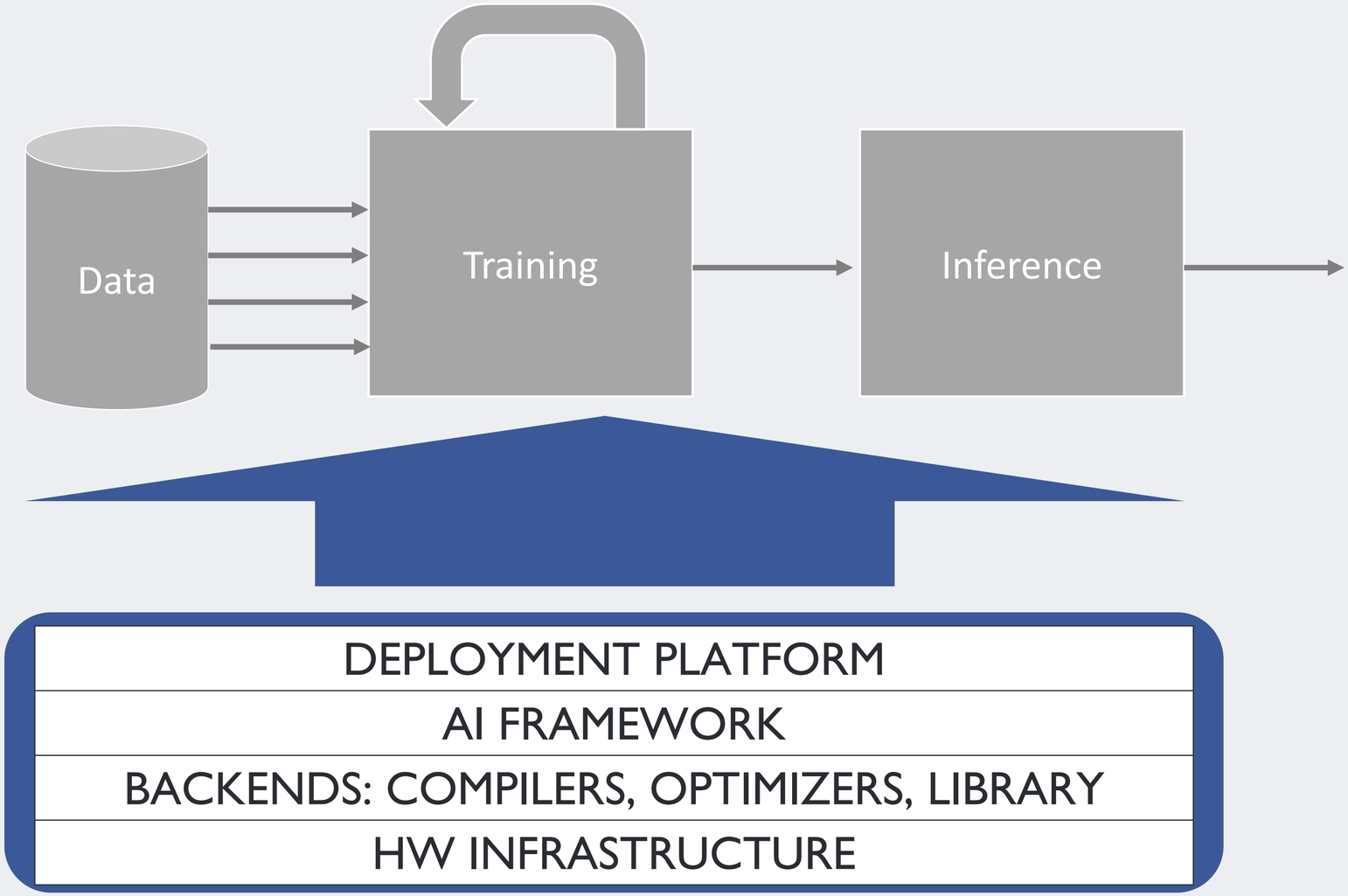
- Faster memory has lower capacity
- Slower memory has less bandwidth

- Application-aware data placement is important

- Datacenter has high distributed aggregate capacity and bandwidth
  - But have to pay net communication cost
  - Right balance is application dependent

FB AI Development Ecosystem

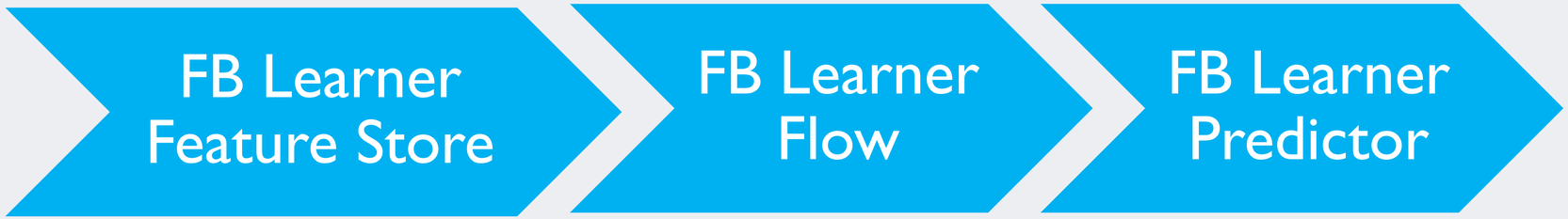
What Lies Beneath?



DEPLOYMENT PLATFORM
AI FRAMEWORK
BACKENDS: COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

FB Learner Platform

- AI workflow for automated model management and deployment
- Programmable, reusable, distributed training pipeline
- Library of reusable ML algorithms
- Provide sharable and reusable history of past experiments



FB AI PyTorch Framework

DEPLOYMENT PLATFORM
AI FRAMEWORK
BACKENDS: COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

Prototype

PyTorch

Developer Efficiency for Research

- Flexible
- Fast Iteration
- Highly Debuggable
- Less Robust

Transfer

Deploy

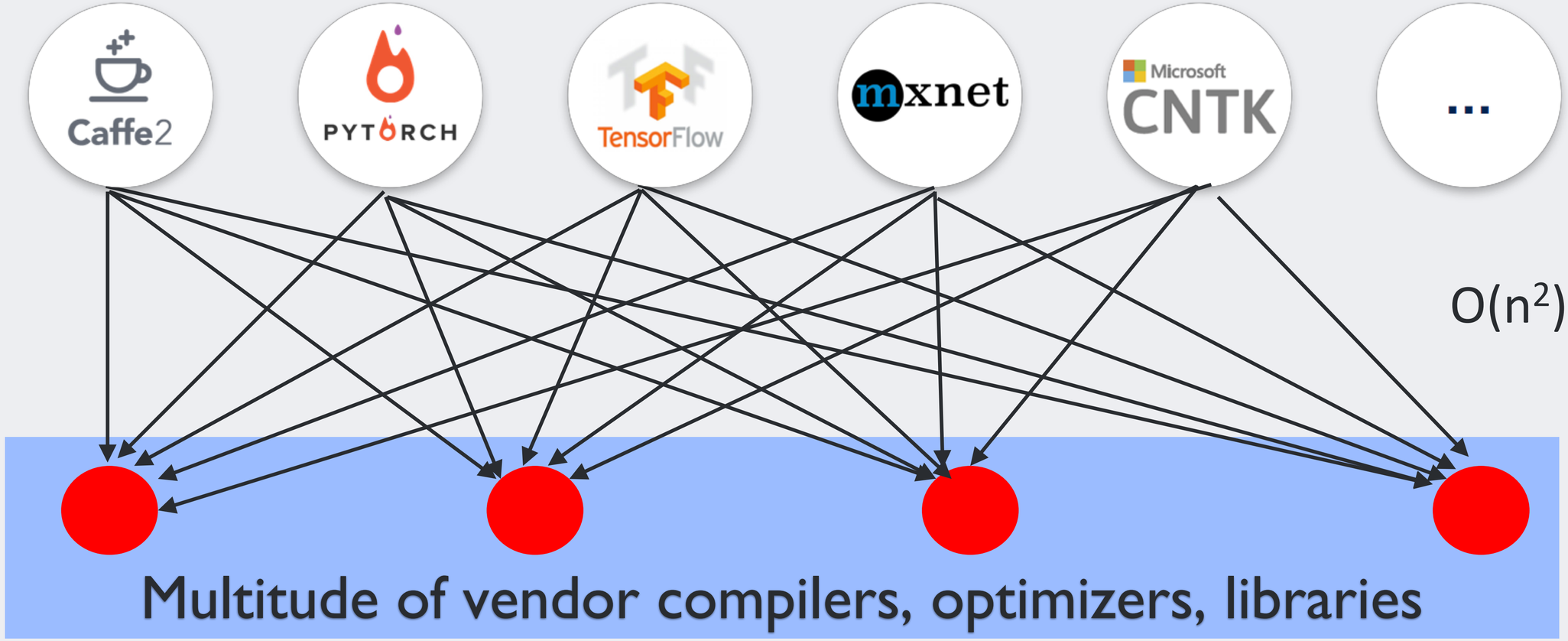
Caffe2

Infra Efficiency for Production

- Stability
- Scale & Speed
- Data Integration
- Relatively Fixed

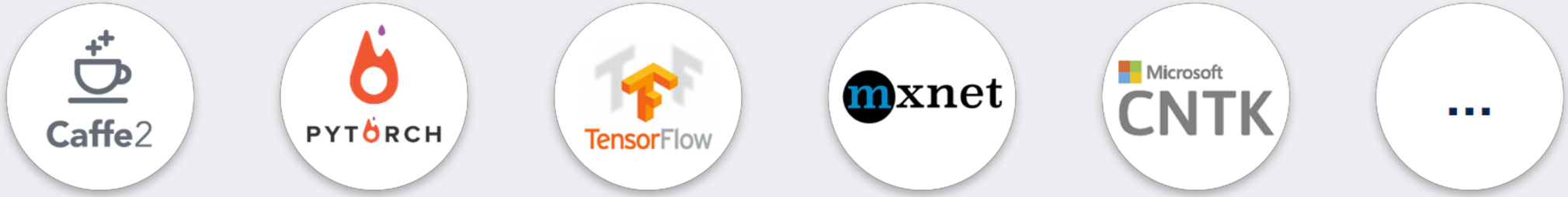
DEPLOYMENT PLATFORM
AI FRAMEWORK
BACKENDS: COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

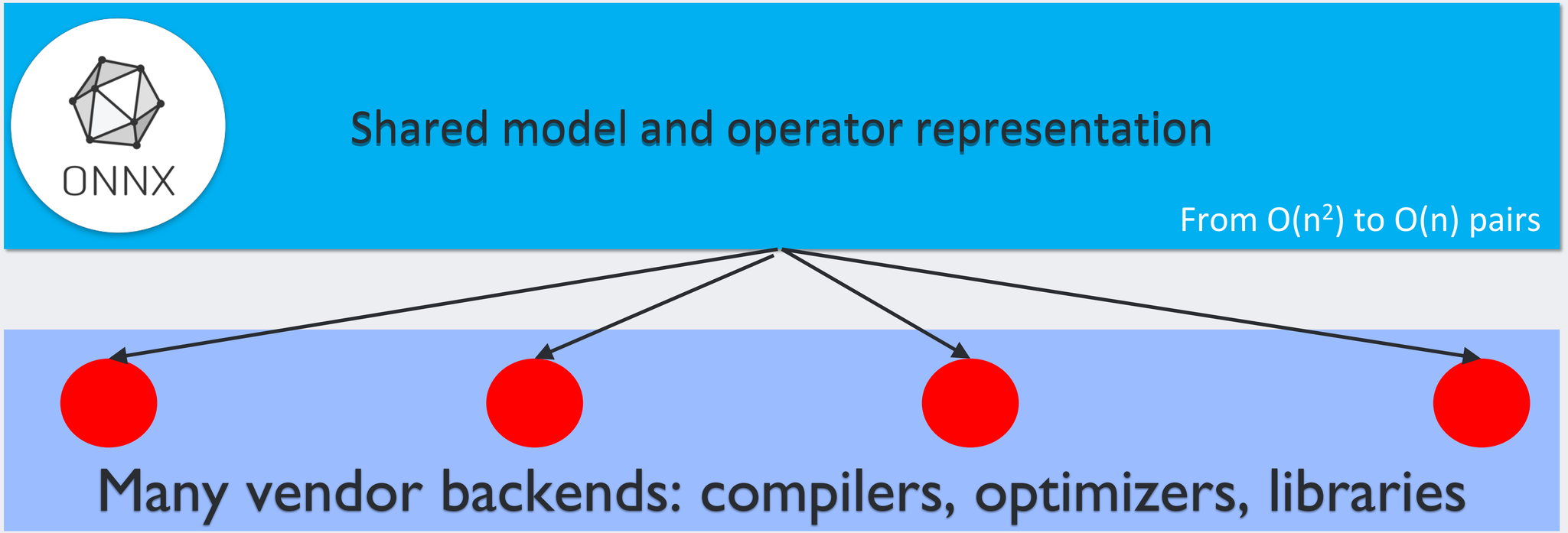
$O(n^2)$ Problem



DEPLOYMENT PLATFORM
AI FRAMEWORK
BACKENDS: COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

Open NN Exchange (**ONNX**)

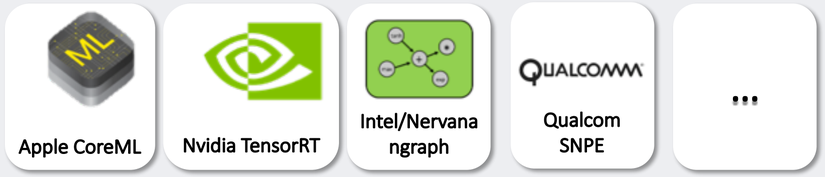




DEPLOYMENT PLATFORM
AI FRAMEWORK
BACKENDS: COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

Backends

- Glow ML Compiler

- Vendor optimizers
  - 

- ML Libraries
  - QQNPACK for mobile CPU
  - FBGEMM, Intel MKL for server CPU
  - CUDNN for GPU

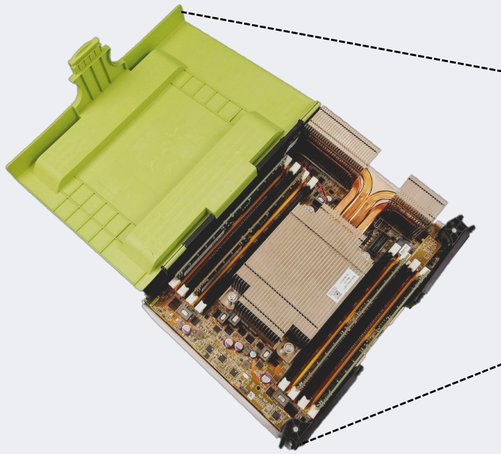
DEPLOYMENT PLATFORM
AI FRAMEWORK
COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

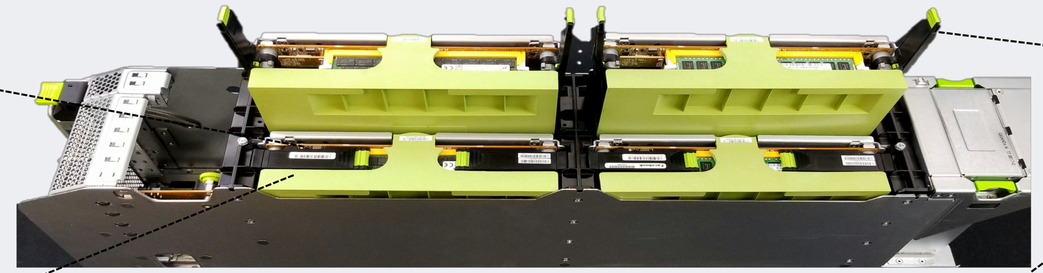
Facebook Hardware

- Facebook designs its own hardware since 2010
- All designs released through open compute!
- Facebook Server Design Philosophy
  - Identify a **small number** of **major services** with unique resource requirements
  - Design servers for those major services

DEPLOYMENT PLATFORM
AI FRAMEWORK
COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

Yosemite V2: Modular Platform





Twin Lakes

single socket, 18c Intel Skylake,
4-channel DDR4 memory,
Boot drive

Yosemite V2 Chassis

with 4 Twin Lakes, or
6x M.2 Glacier Point,
Shared 50 Gbps NIC

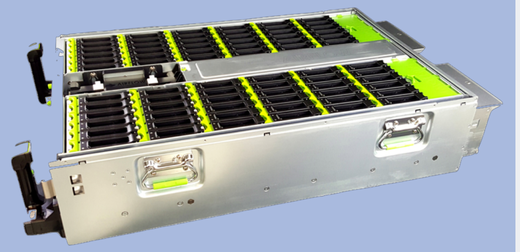
Rack

Chassis combined into rack
Used for Web Tier and other
“stateless” services

DEPLOYMENT PLATFORM
AI FRAMEWORK
COMPILERS, OPTIMIZERS, LIBRARY
HW INFRASTRUCTURE

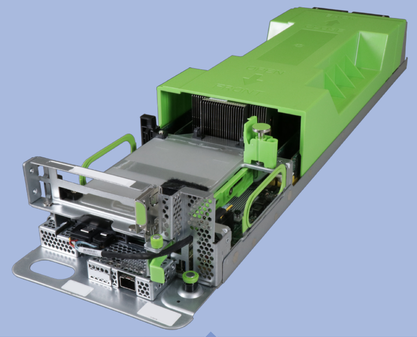
ML Hardware

Bryce Canyon
- High density storage
- Storage-heavy tasks

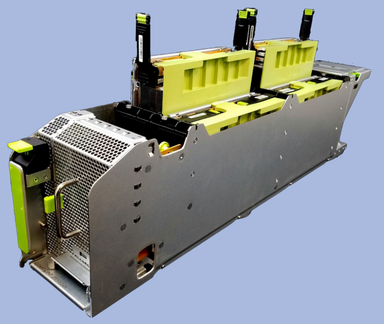


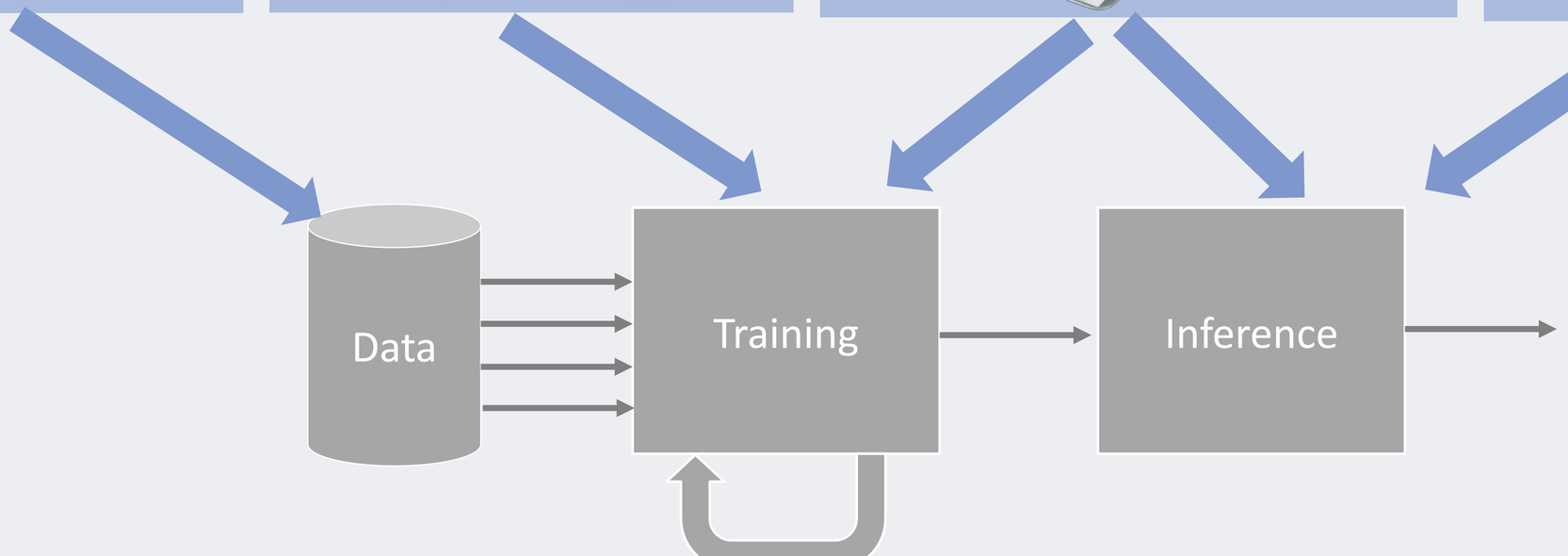
Big Basin
- 8 NVIDIA P100/V100's
- Very compute-heavy tasks

Tioga Pass
- Dual socket, many cores Skylake CPU
- Compute and mem-heavy tasks



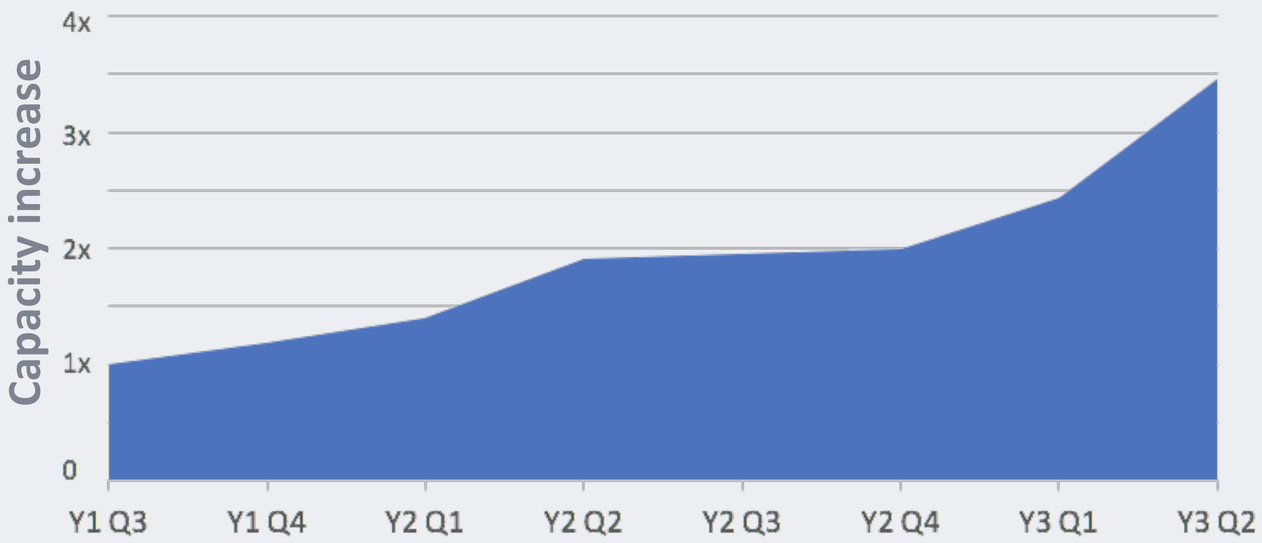
Twin Lakes
- Single socket, fewer cores Skylake CPU
- Compute and low latency tasks





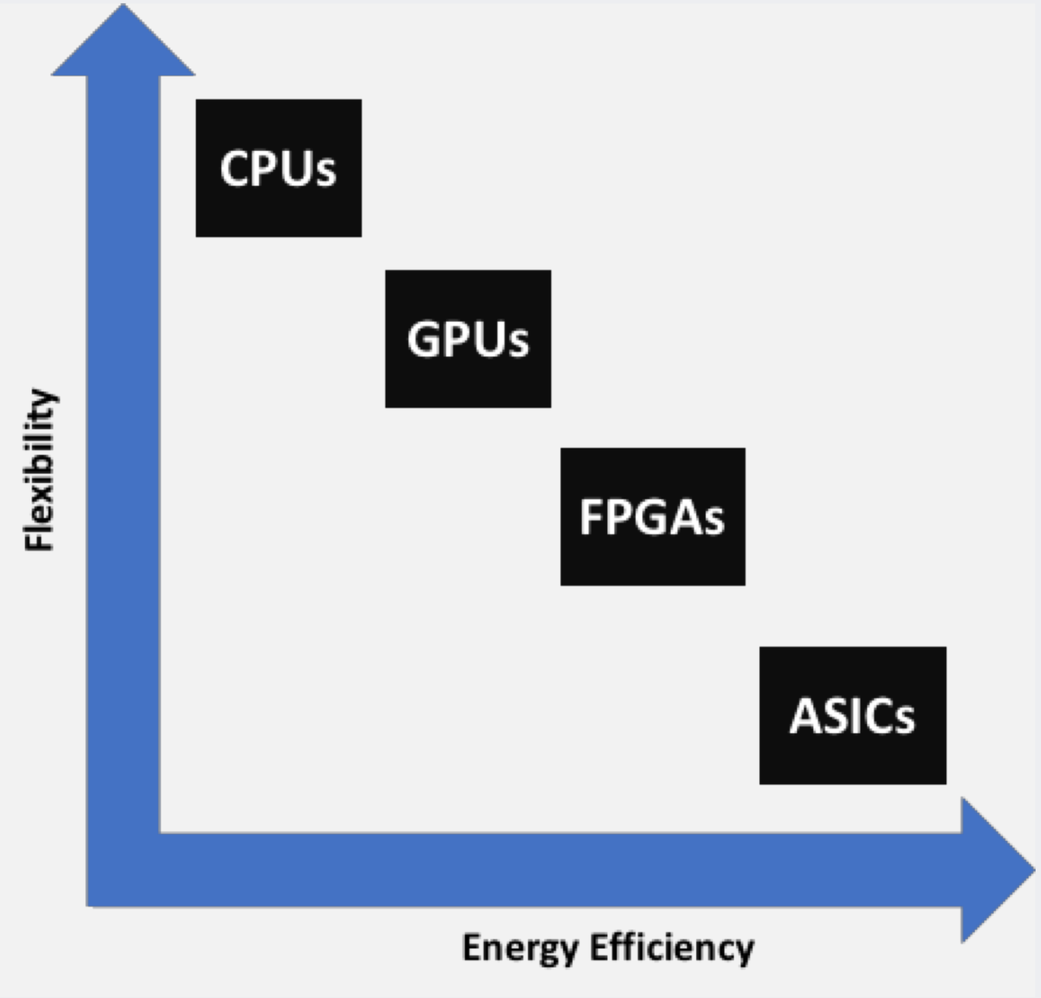
Inference ASIC Hardware

- AI demands have been growing



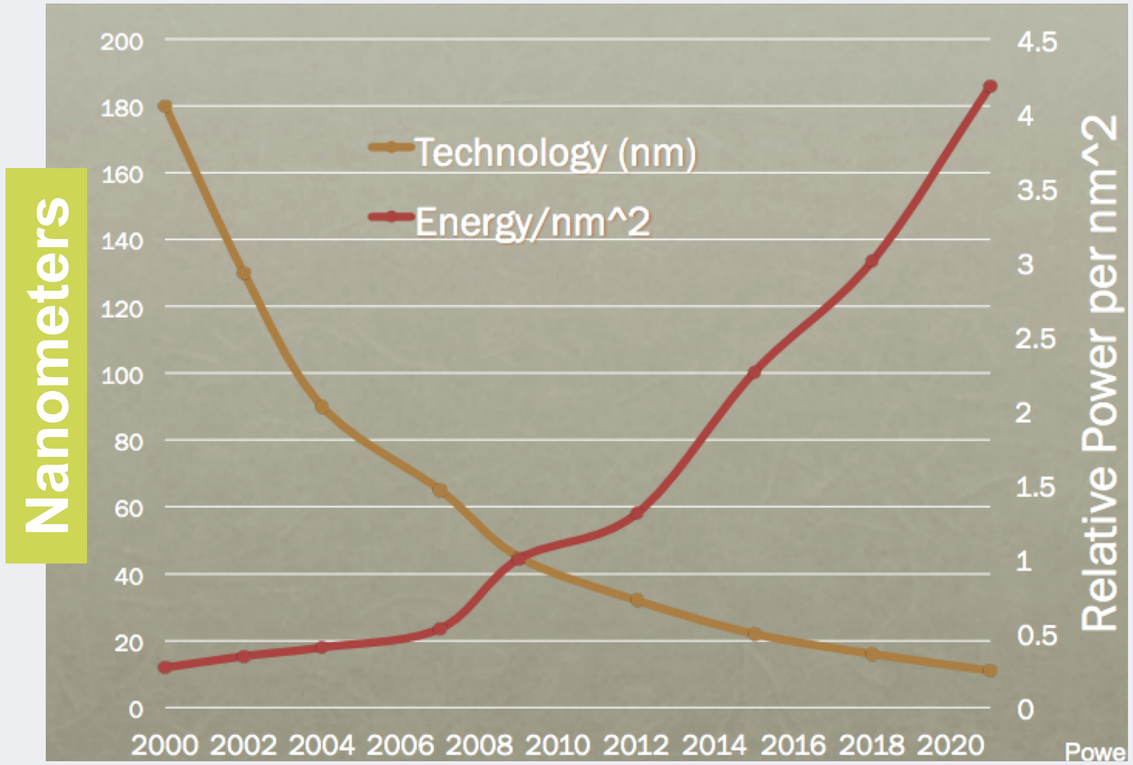
Increase of server capacity for DL inference, Xiaodong Wang

- How can we meet demand?



General-Purpose Compute

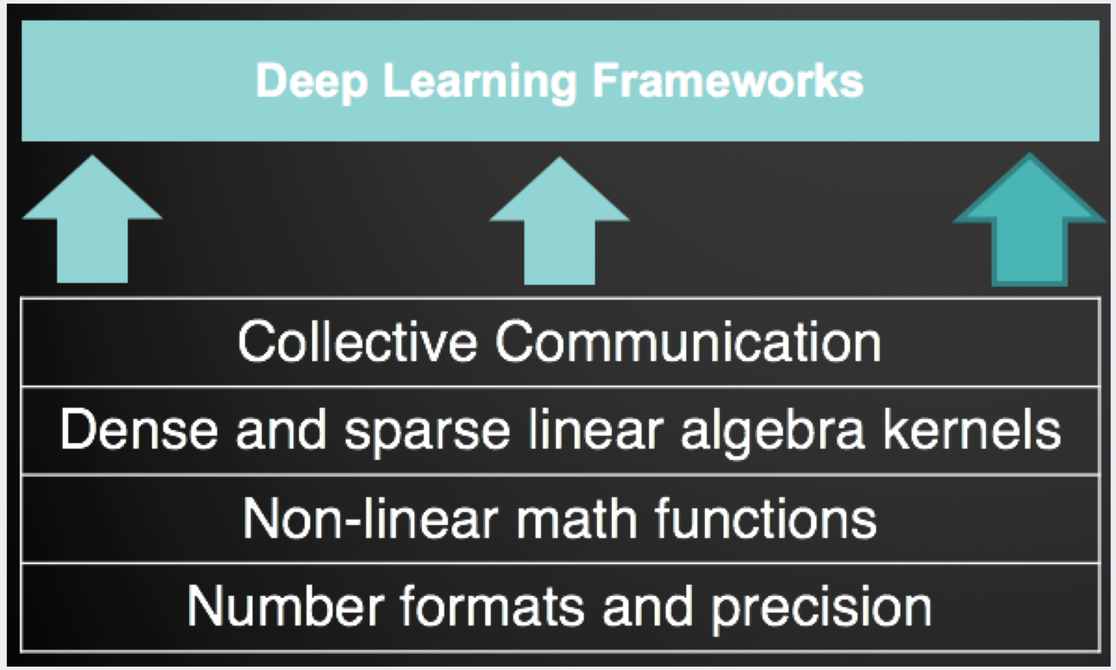
Technology, Energy, and Dennard Scaling



Future of Computing, John Hennessey

- Dennard scaling is long dead
- Moore's Law is slowing down
- Thus increase in AI demand → commensurate fleet size increase

Deep Learning – Fixed Function Workload



- DL is fixed fixed function workload
- Spends most time in matrix multiply
- Maps well to specialized ASIC
- Fast, energy-efficient (high reuse), co-designed

Inference ASIC at Facebook

- Many companies are building an AI chip
- Start-ups, public cloud providers, etc.
- Facebook is partnering with HW vendors to build inference ASIC
- Done via co-design with FB workloads in mind

Deep-dive: Training

Asynchronous and Synchronous SGD

Asynchronous SGD

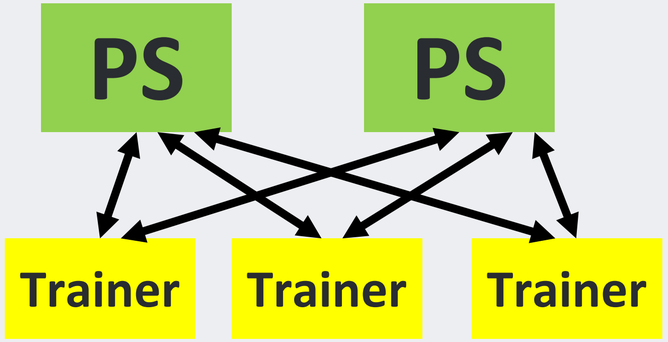
- Parameters are updated by parallel workers asynchronously
- Updates may be delayed or “lost’
- Number of parameter servers manage parameters

Pros

- Used for very large models that do not fit in one machine
- Can scale to very large clusters with slow network

Cons

- Staleness of updates sensitive to network characteristics



Synchronous SGD

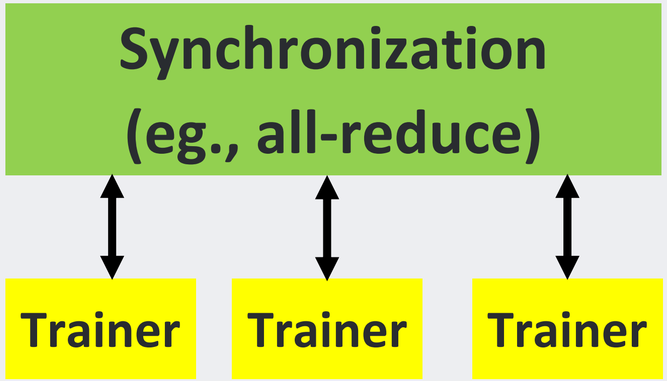
- Workers sync parameter gradients after each iteration
- Updates are always in sync
- Computation is function of the total batch size only

Pros

- Reproducible results up to machine precision
- Scales well if large batch size is used

Cons

- Does not scale well to large clusters



GLOO Communication Library for ML

June, 2017

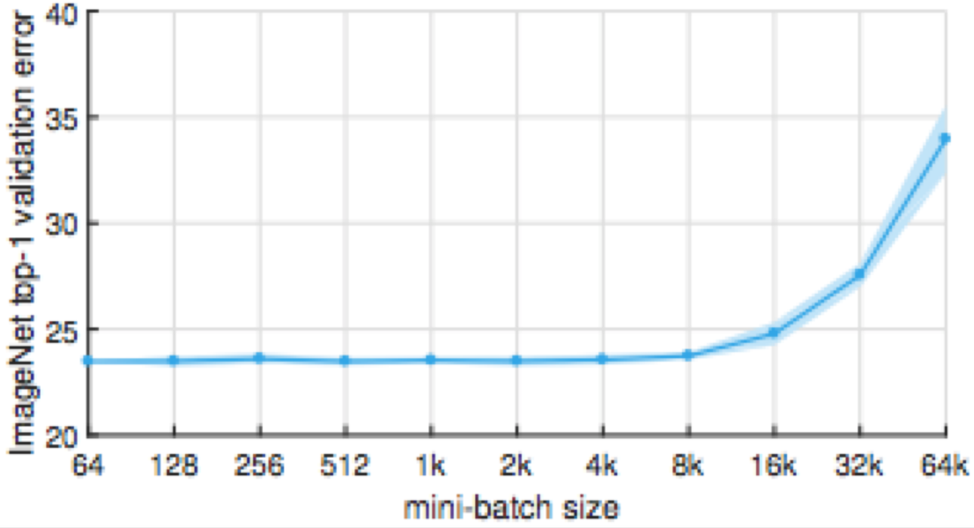
Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

Abstract

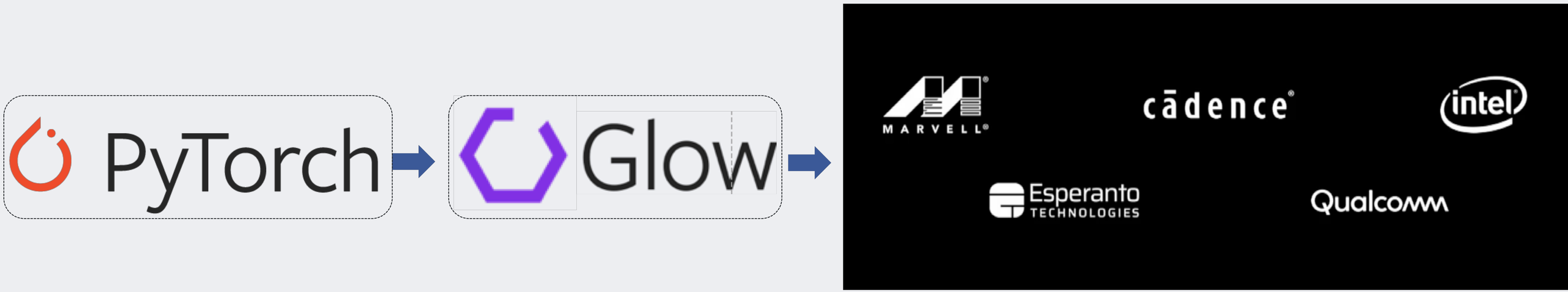
Deep learning thrives with large neural networks and large datasets. However, larger networks and larger datasets result in longer training times that impede research and development progress. Distributed synchronous SGD offers a potential solution to this problem by dividing SGD minibatches over a pool of parallel workers. Yet to make this scheme efficient, the per-worker workload must be large, which implies nontrivial growth in the SGD mini-



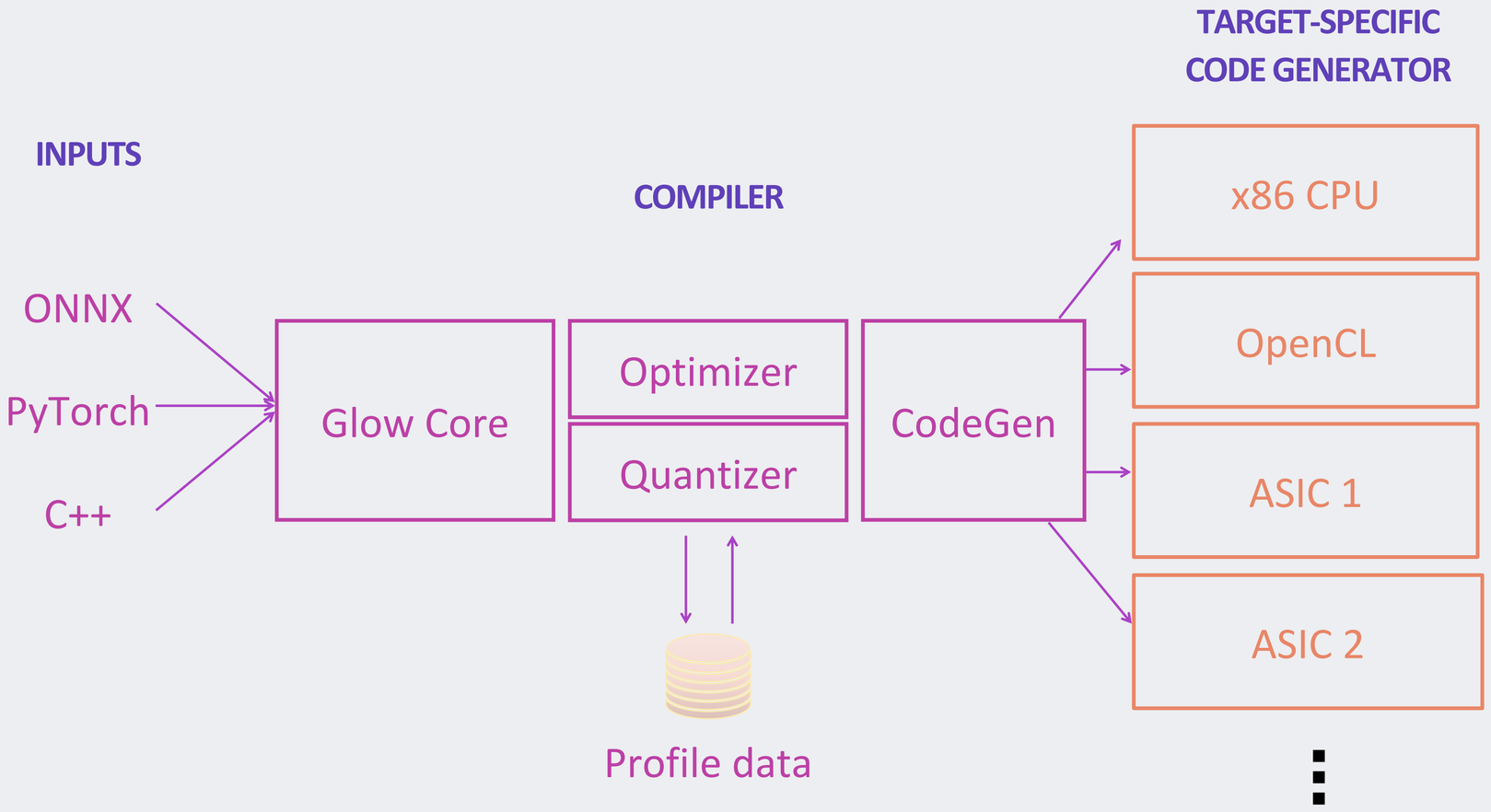
Deep-dive: Glow ML Compiler

ML HW Acceleration Ecosystem

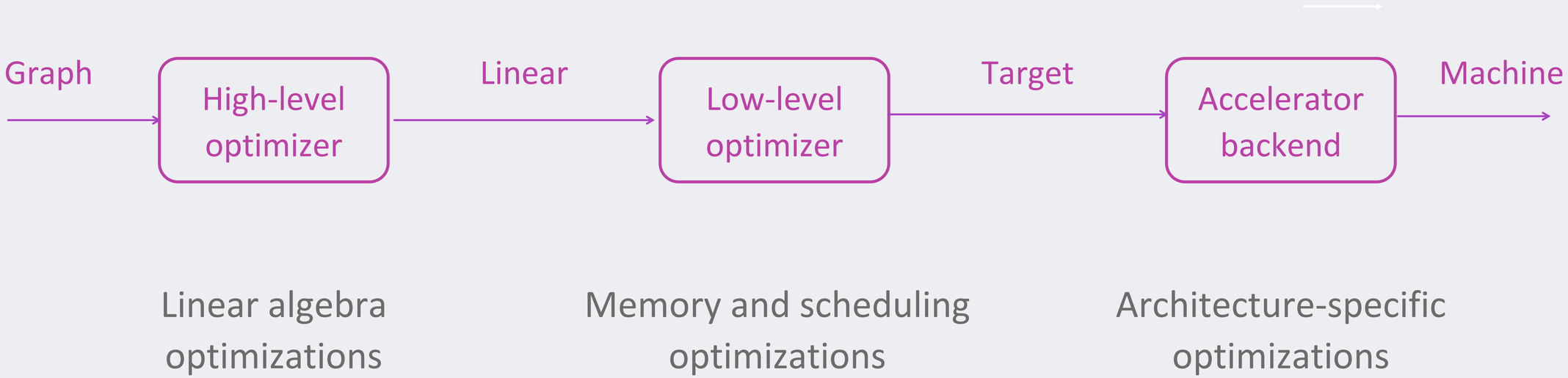
- Facebook is building ML HW acceleration using Glow compiler



Glow Compiler Design



Compilation Pipeline



- Optimize NN using layers of intermediate representations (IR)

High-Level Graph

Pure data-flow graph

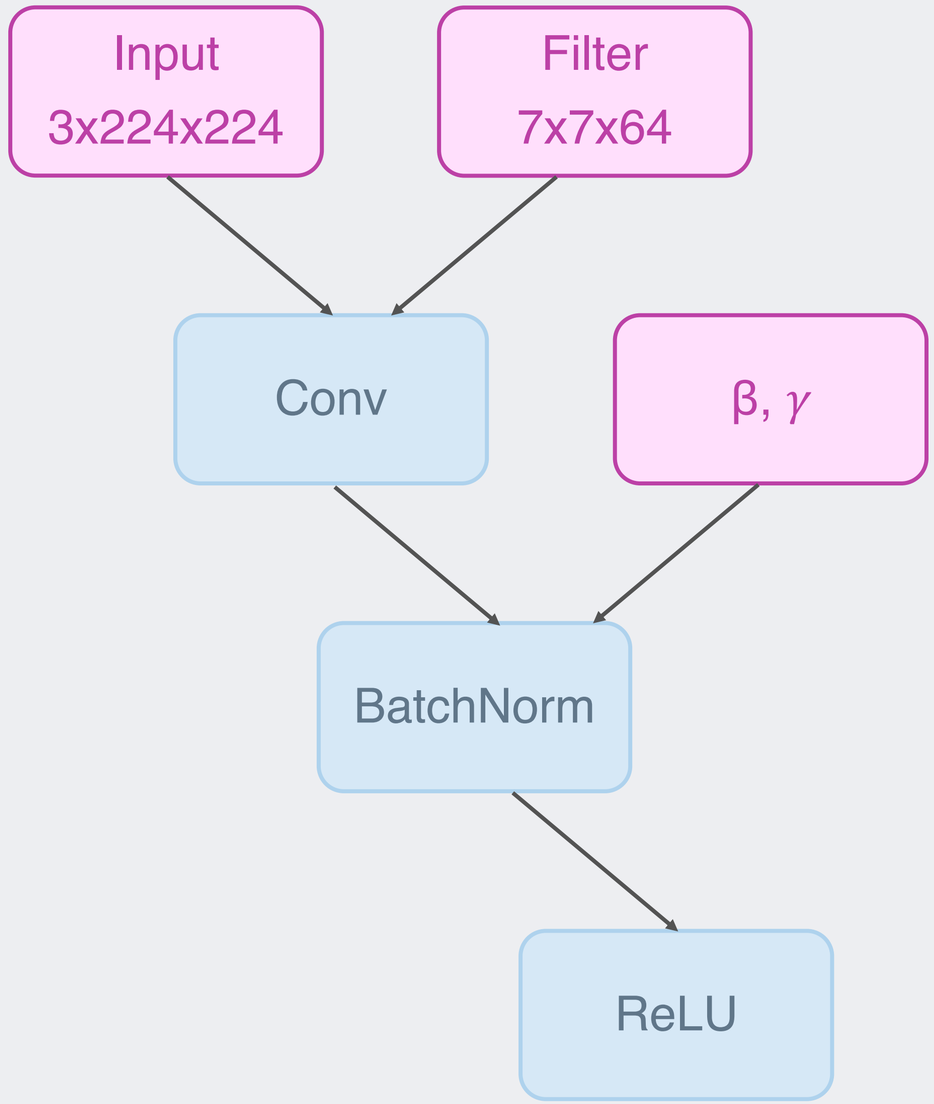
Static types

Domain specific optimization

- Fuse batchnorm+convolution
- Eliminate transpose

Classical compiler optimization

- Dead code elimination
- Constant propagation



Low Level Optimizer: Linear IR

Sequence of macro instructions

- Linearized schedule
- Not SSA (static single assignment)

Typed memory regions

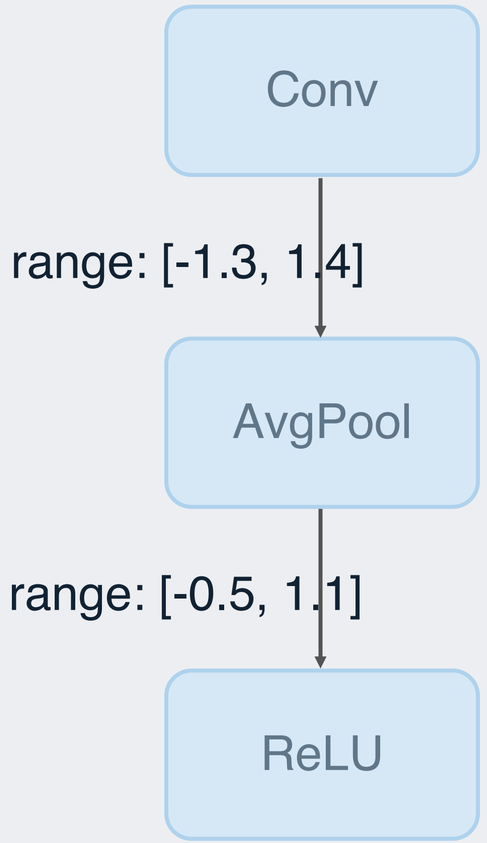
Optimize memory

- Re-use regions
- Concat in place
- Minimize memory footprint

```
declare {
  %input = WeightVar float<10 x 5000> mutable
  %state = WeightVar float<10 x 20> mutable
  %rnn_Whh = WeightVar float<20 x 20> mutable
  %result = WeightVar float<100 x 500> mutable
}
program {
  %X0 = allocactivation
        extracttensor @out %X0, @in %input
  %X2 = allocactivation
        splat @out %X2
        inserttensor @inout %X2, @in %X_0
  %X5 = allocactivation
        matmul @out %X5, @in %X2, @in %Wxh
        deallocactivation @out %X2
        batchedadd @out %X5, @in %X5, @in %Bxh
  %X9 = allocactivation
        matmul @out %X9, @in initial, @in %Whh
        batchedadd @out %X9, @in X9, @in %rnn_Bhh
}
```

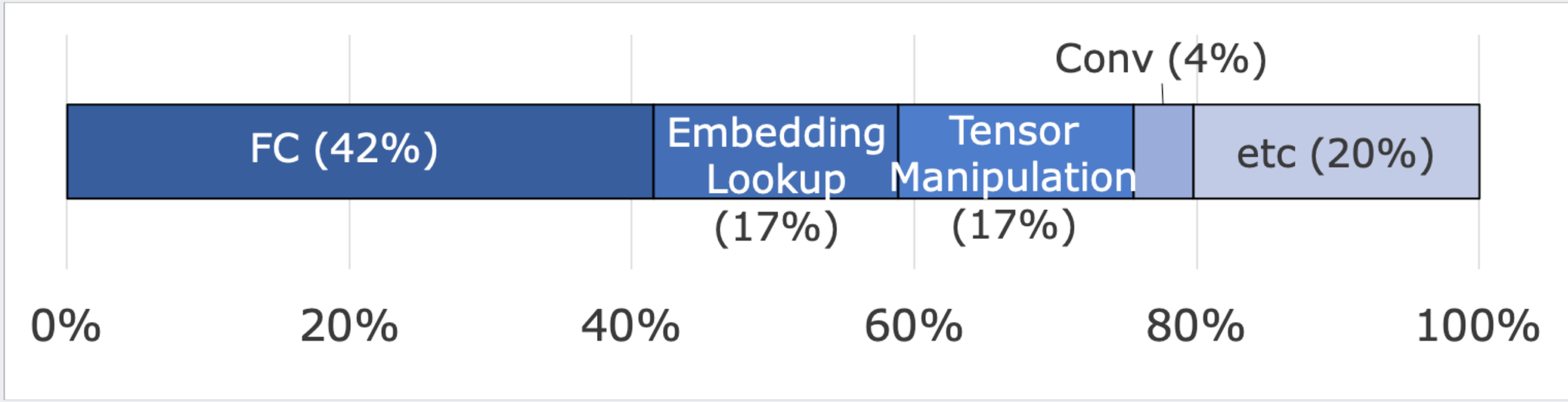
Quantization

- Converts fp32 or fp16 trained network to int
- Glow uses profiling to estimate ranges
  - Instrument network
  - Run with data
  - Record histogram at each edge
- Recompile
  - Quantize types
  - Optimize scaling



Deep-dive: FBGEMM

Fleet-wide DL Execution Time Breakdown

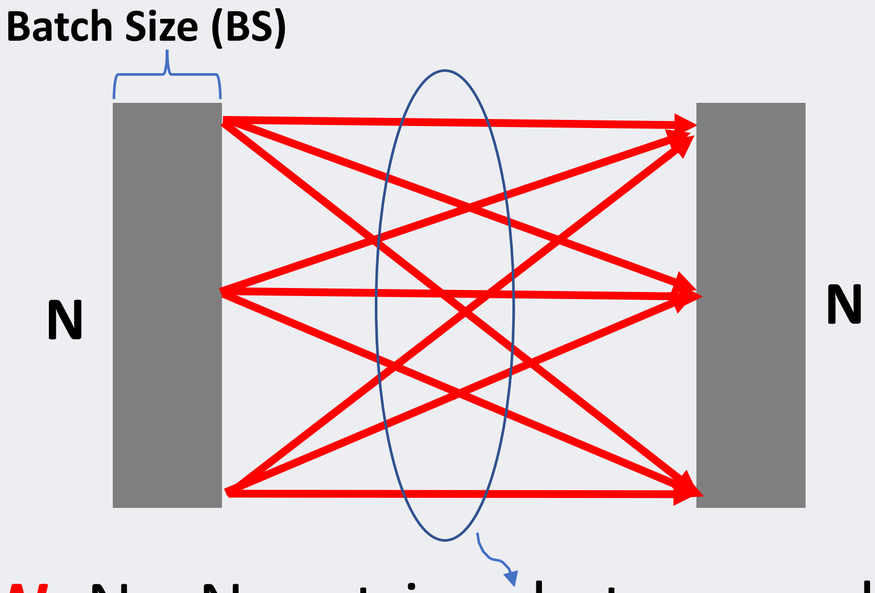


- FC is the most time consuming followed by embedding
- Conv is only 4%
- Tensor manipulation (concat, split, transpose…): good graph-level optimization targets

FBGEMM

- GEMM is very important
- Last few year invested into quantization for CPU
  - Provides high-performance
  - Getting us prepared for accelerators
- Developed high-performance reduced-precision FBGEMM library for servers

Why Quantizing GEMM for CPU?



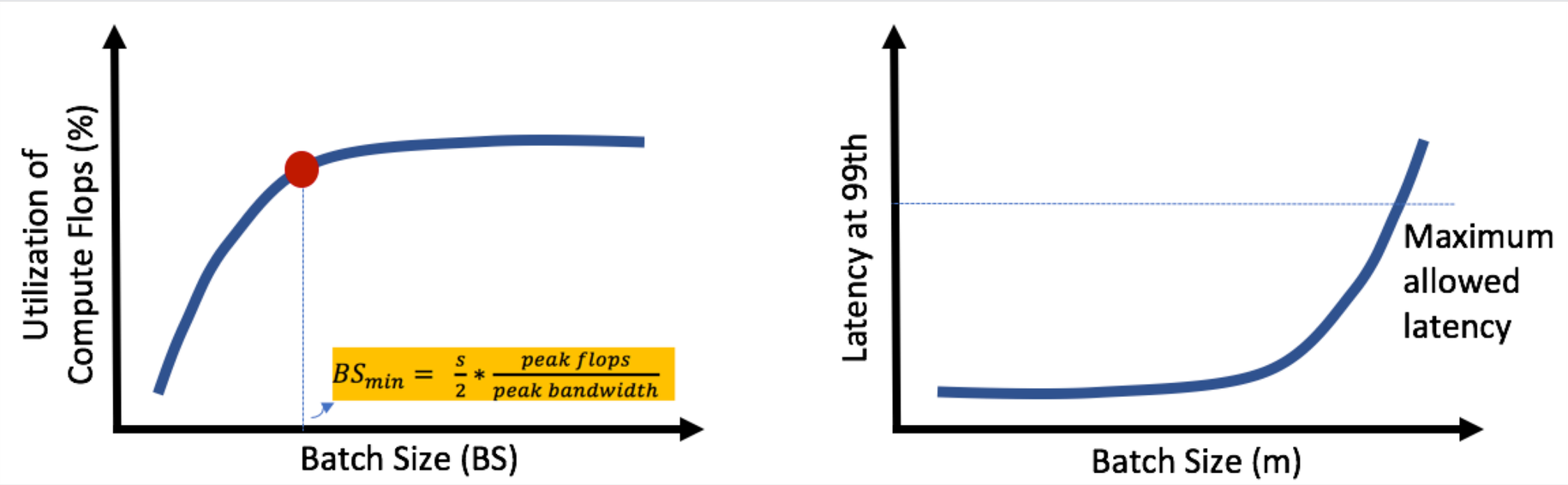
W: N x N matrix, s bytes per element

Time to move data	$\frac{s * N * N}{peak\ \boldsymbol{bandwidth}}$
Time to compute	$\frac{2 * BS * N * N}{peak\ \boldsymbol{flops}}$
flops:byte Ratio (aka arithmetic intensity)	$\frac{2 * BS}{s}$

To be compute bound, we need

$$BS_{min} = \frac{s}{2} * \frac{peak\ flops}{peak\ bandwidth}$$

Batching



- Increase bandwidth (more DRAM channels, fit problem into SRAM, etc.)
- Reduce “s” (**reduced precision**, sparsity, code book, etc…)

FBGEMM

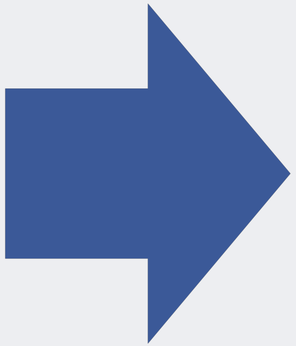
https://code.fb.com/ml-applications/fbgemm/

- Especially optimized for low-precision
- Fast general matrix-matrix multiplication (GEMM)
  - Optimized for low precision
  - Optimized for small batch sizes
  - Built-in support for accuracy-loss-minimizing techniques such as row-wise quantization and outlier-aware quantization.
- Exploits fusion of bandwidth-bound pre- and post-GEMM operations

The Sacred Art of Writing GEMMs

Simple

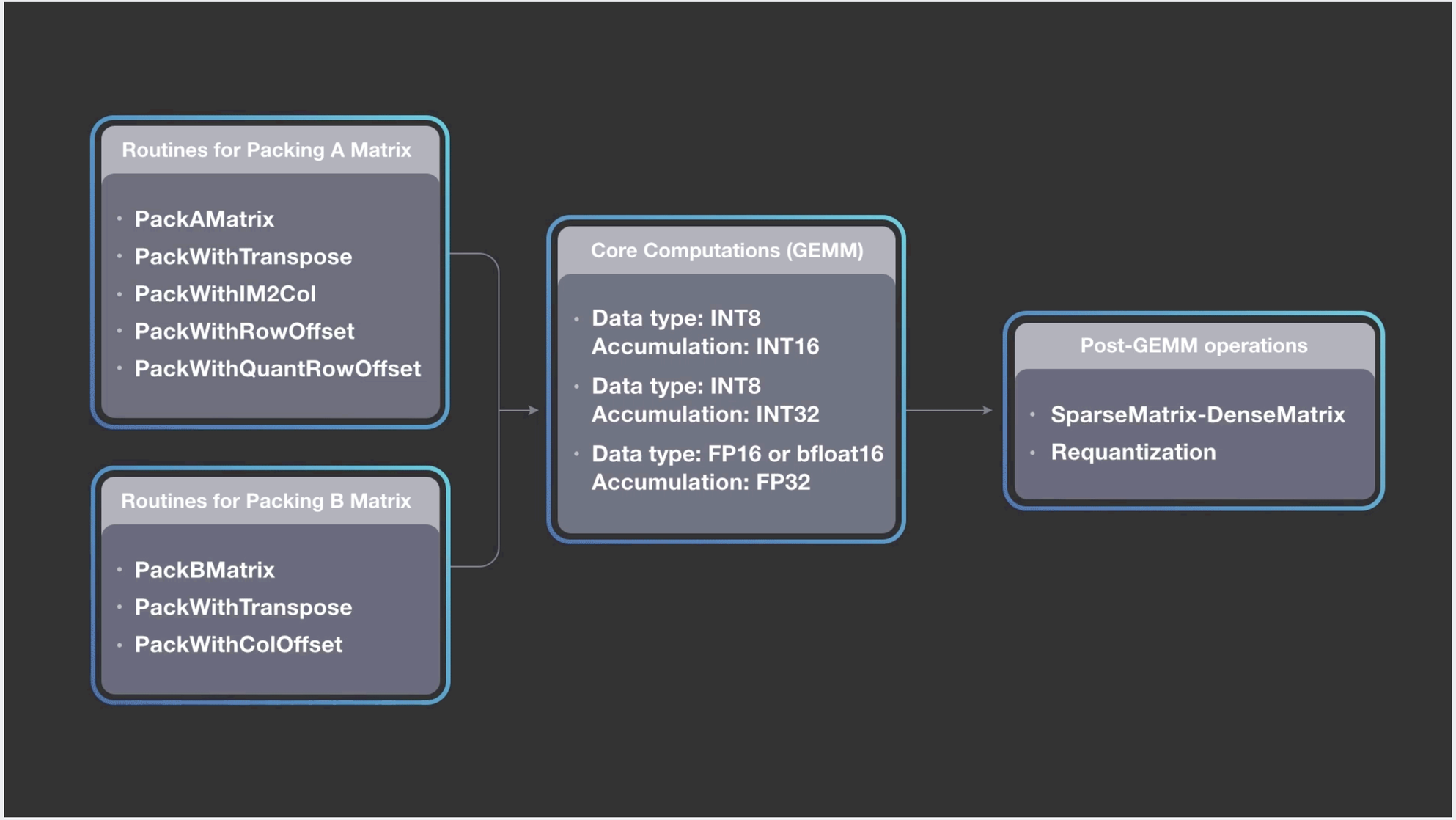
```
for (i=0; i<m; i++) {
  for (j=0; j<n; j++) {
    a[i][j]=0;
    for (k=0; k<p; k++) {
      a[i][j] += b[i][k]*c[k][j] ;
    }
  }
}
```



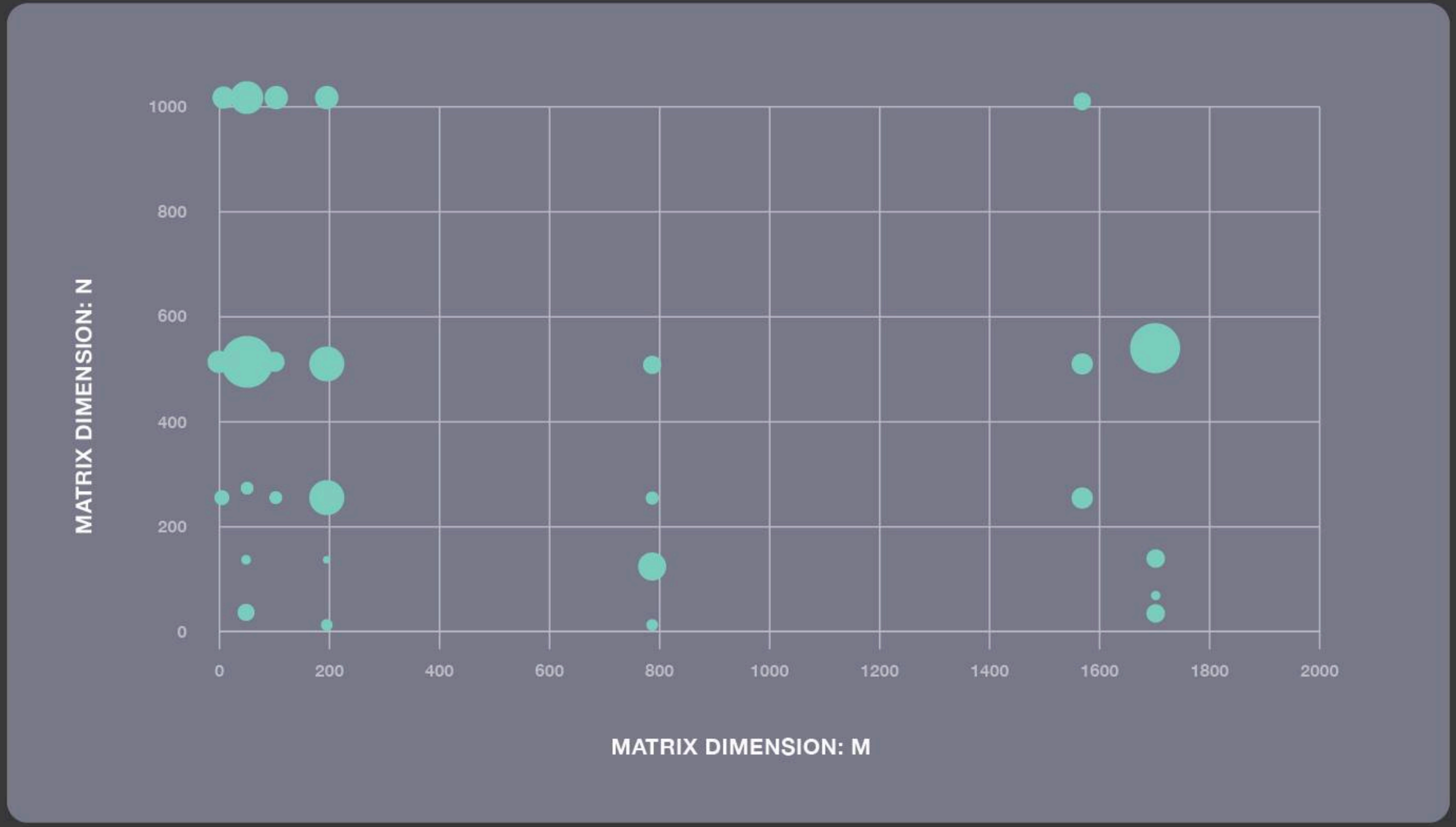
Optimized for CPU

```
Loop1 for ic = 0 to M-1 in steps of MCB
Loop2   for kc = 0 to K-1 in steps of KCB
          //Pack MCBxKCB block of A
Loop3     for jc = 0 to N-1 in steps of NCB
            //Pack KCBxNCB block of B
//--------------------Macro Kernel-----------
Loop4       for ir = 0 to MCB-1 in steps of MR
Loop5         for jr = 0 to NCB-1 in steps of NR
//--------------------Micro Kernel-----------
Loop6           for k = 0 to KCB-1 in steps of 1
                  //update MRxNR block of C matrix
```

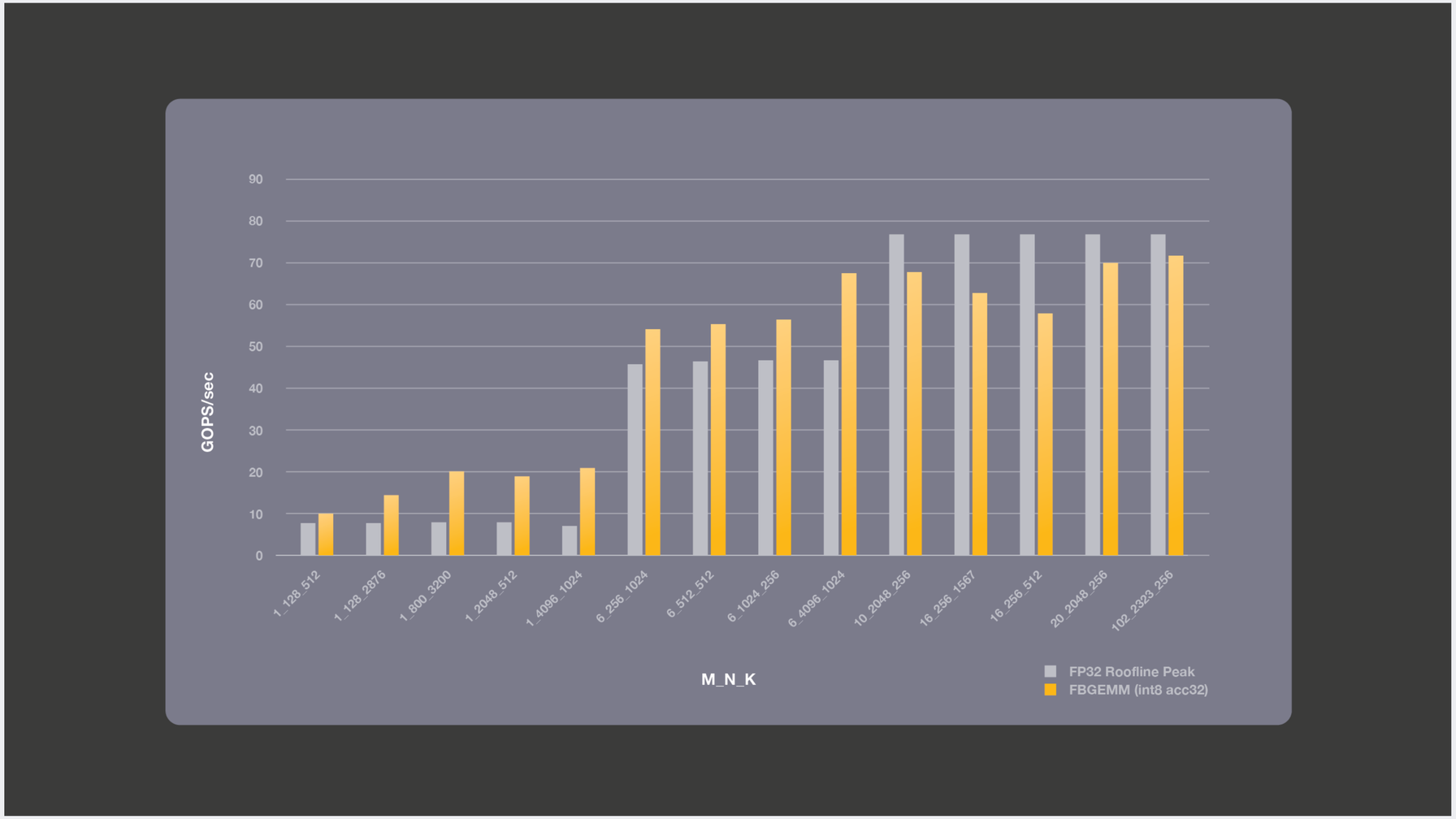
FBGEMM Library Choices



Matrices Come in All Shapes and Sizes



Performance for Small Batch



Deep-dive: Deep Learning Inference

DL Inference in Facebook Data Centers

- Used for core services: personalization and integrity/security
- Diverse data types: images, videos, multi-lingual contents
- Scale to billions of users

- Application domains
  1. Ranking and recommendation: ads, feed, and search
  2. Computer vision: image classification, object detection, video understanding
  3. Language: translation, content understanding
  4. Interactions among these

Domain 1: Ranking and Recommendation

- Embedding tables demand
  - High memory capacity (>10s of GBs)
  - High memory bandwidth (low arithmetic intensity)
- HBMs are too small
- NVMs are too slow

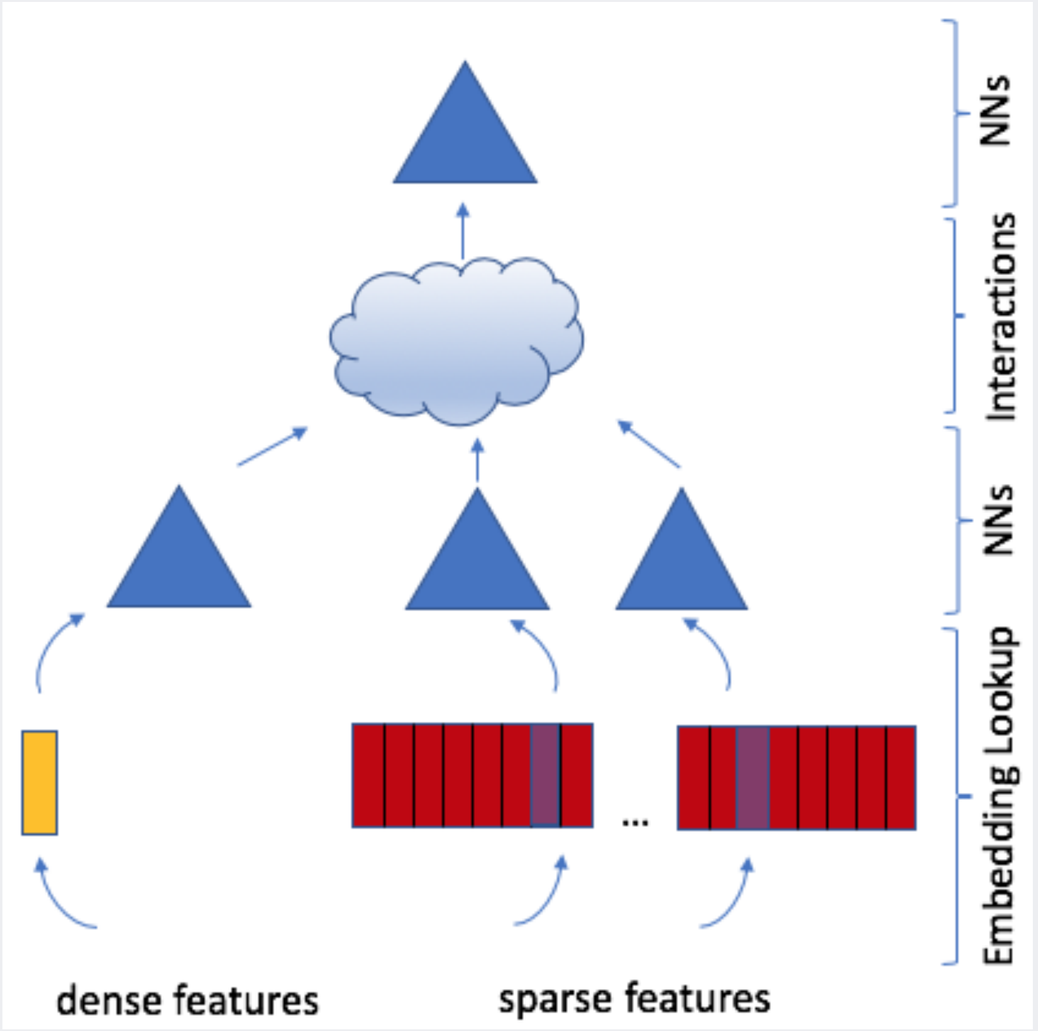
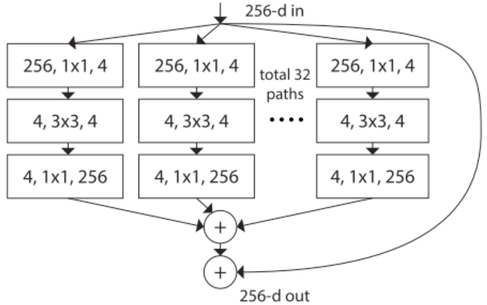
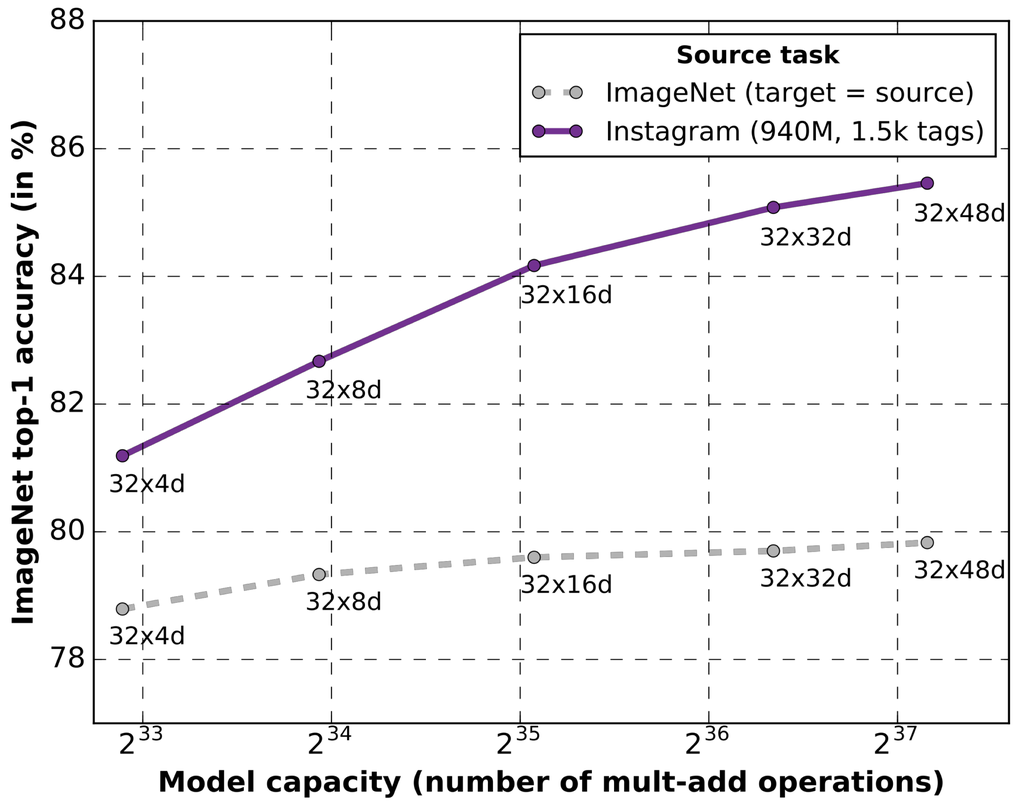


Figure credit: Maxim Naumov

Domain 2: Computer Vision

- Classification
  - Bigger model + bigger data → higher accuracy





Domain 2: Computer Vision

- Classification
  - Bigger model + bigger data → higher accuracy

- Object detection and video understanding
  - Bigger inputs than classification
  - FLOP-efficient models like ShuffleNet with depth-wise convolutions [2]

[1] Exploring the limits of weakly supervised pretraining. Mahajan et al.

[2] Rosetta: understanding text in images and videos with machine learning. Sivakumar et al.

Domain 3: Language Models

- Small batch size for latency constraints
- Attention only models
- Multilingual models

Resource Requirements

Category	Model Types	Model Size (# params)	Maximum Live Activations	Op. Intensity (w.r.t. weights)
Recommendation	FCs	1-10M	> 10K	20-200
	Embeddings	>10 Billion	> 10K	1-2
Computer Vision	ResNeXt101-32x4-48	43-829M	2-29M	avg. 380 Min. 100
	Faster-RCNN (with ShuffleNet)	6M	13M	Avg. 3.5K Min. 2.5K
	ResNeXt3D-101	21M	58M	Avg. 22K Min. 2K
Language	seq2seq	100M-1B	>100K	2-20

Observation 1: big embedding with low op. intensity

Category	Model Types	Model Size (# params)	Maximum Live Activations	Op. Intensity (w.r.t. weights)
Recommendation	FCs	1-10M	> 10K	20-200
	Embeddings	**>10 Billion**	> 10K	**1-2**
Computer Vision	ResNeXt101-32x4-48	43-829M	2-29M	avg. 380 Min. 100
	Faster-RCNN (with ShuffleNet)	6M	13M	Avg. 3.5K Min. 2.5K
	ResNeXt3D-101	21M	58M	Avg. 22K Min. 2K
Language	seq2seq	100M-1B	>100K	2-20

- Interesting challenge for future memory system designs

Observation 2: bigger models and activations

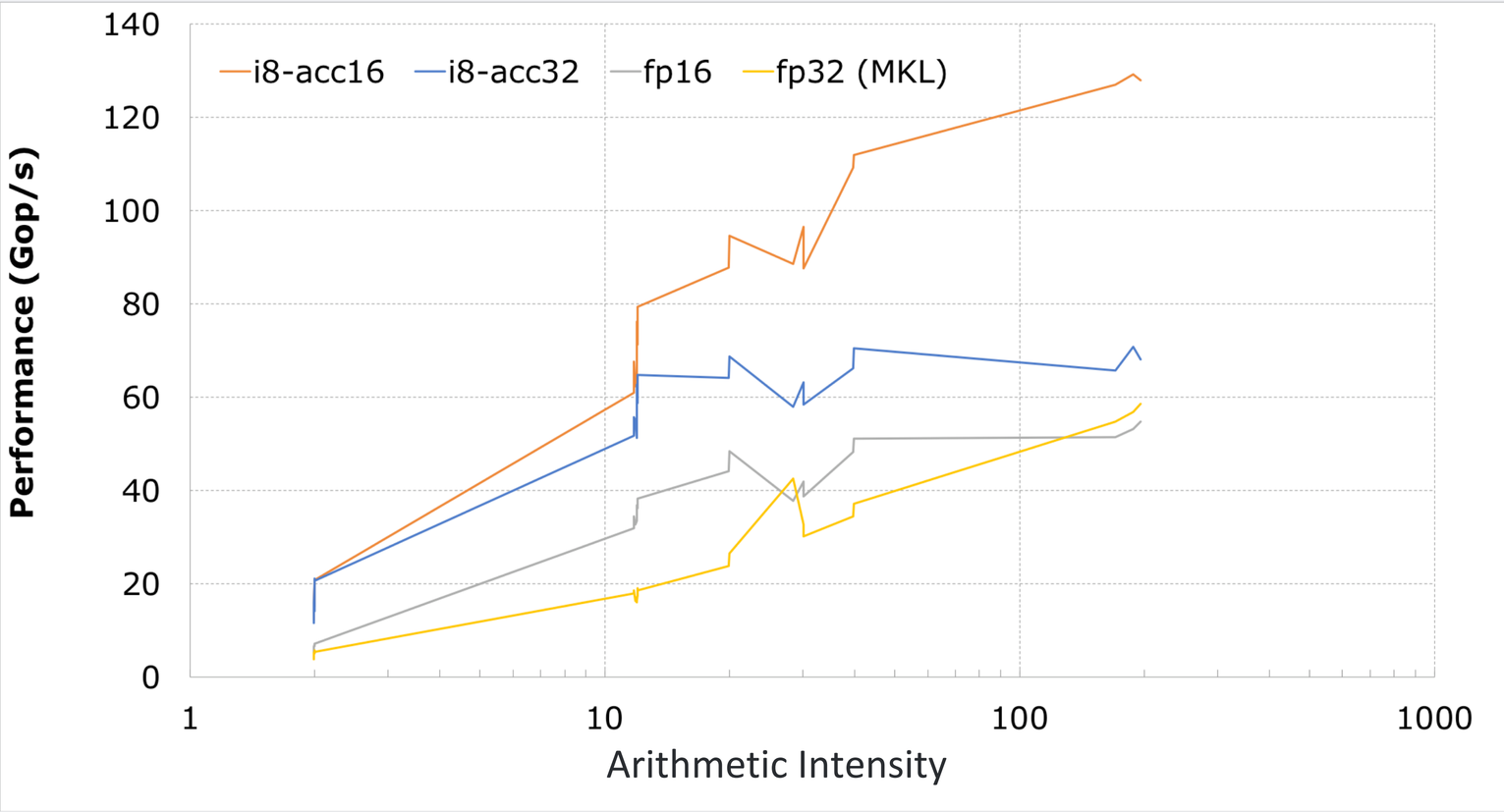
Category	Model Types	Model Size (# params)	Maximum Live Activations	Op. Intensity (w.r.t. weights)
Recommendation	FCs	1-10M	> 10K	20-200
	Embeddings	>10 Billion	> 10K	1-2
Computer Vision	**ResNeXt101-32x4-48**	**43-829M**	**2-29M**	avg. 380 Min. 100
	Faster-RCNN (with ShuffleNet)	6M	**13M**	Avg. 3.5K Min. 2.5K
	ResNeXt3D-101	21M	**58M**	Avg. 22K Min. 2K
Language	**seq2seq**	**100M-1B**	>100K	2-20

- Need large on-chip memory. Otherwise off-chip memory BW bound for small batch.

Reduced-precision Inference

- Performance challenges in current Intel CPUs
  - 8-bit multiplication with 32-bit accumulation instruction throughput not much higher than fp32 (until VNNI is available)
- Accuracy challenges
  - Strict accuracy requirements in data center DL inference

16-bit accumulation for high op. intensity cases



- Measured with 1 core of Intel Xeon E5-2680 v4 with turbo mode off
- i8-acc32 for low op. intensity case and i8-acc16 for high op. intensity case
- 1.7x in resnet50 and 2.4x in Rosetta (Faster-RCNN-ShuffleNet) over fp32

Accuracy improving techniques

- **Outlier-aware quantization**
- L2 error minimizing quantization: find a scale and zero_point that minimizes L2 error (similar to Nvidia TensorRT's KL divergence minimization)
- Fine-grain quantization: per output feature quantization (FC), per output channel quantization (Conv), per-entry quantization (Embedding)
- Quantization-aware training: fake quantization (similar to TF)
- Selective quantization: skip layers with high quantization errors (e.g., first Conv layer)
- Net-aware quantization: propagation range constraints (e.g., operators followed by ReLU or sigmoid)

Outlier-aware Quantization

$$Y = X * W^T = X * (W\_1 + W\_2)^T$$

$$W\_1(i, j) = W(i, j) \text{ if } |W(i, j)| < \text{outlier\_threshold, else } 0$$

$$W\_2(i, j) = W(i, j) \text{ if } |W(i, j)| >= \text{outlier\_threshold, else } 0$$

- W_1 : dense matrix with small values. Can compute with 16-bit accumulation
- W_2 : sparse matrix with big values. Compute with 32-bit accumulation

Inference in DC vs. Inference at Edge Devices

	Data Center	Edge Devices
Reduced Precision	Wants to maintain accuracy. Fp16 fallback can be useful	Trade-off accuracy for energy-efficiency and latency constraints
Model Pruning	Should focus on speeding up inference (exception: embeddings)	Should focus on model size

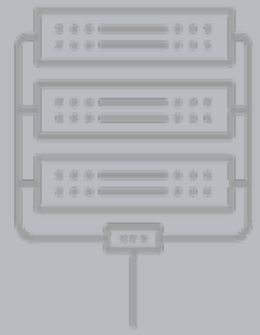
Key Takeaways

Facebook AI

Lots of Data

Wide variety of models

Full stack challenges



Global scale

Acknowledgments

- Thanks to many awesome colleagues for letting me reuse/adapt their slides
  - **Kim Hazelwood**, AI Infra
  - **Yangqing Jia**, AI Infra
  - **Pieter Noordhuis**, AI Infra
  - **Joe Spisak**, Applied ML
  - **Jongsoo Park**, AI System Co-Design
  - **Nadav Rotem**, Glow Runtime
  - **Bert Maher**, Glow Runtime
  - **Whitney Zhao**, Hardware Engineering

Q&A