AI-Systems
Hardware For Machine Learning
(294-162)

Amir Gholami & Joseph E. Gonzalez

Acknowledgments

Many slides from Prof. Kurt Keutzer, Suresh Krishna, Prof. Patterson, Michael Pellauer (Nvidia), Prof. Sophia Shao, Naveen Kumar (Google), Pallas Group

Agenda for Today

- 1:10-2:00: Preliminary Lecture on Hardware for ML
- 2:00-2:45: PC Meeting Discussions
- 2:45-3:00: Break
- 3:00-4:00: Guest Lecture by Dr. Reynold Xin (Databricks)

Objectives For Today

- Key Trends in Hardware and motivation for DSAs
- Reduced Precision Training
- Different DataFlow Patterns in Hardware
- How to evaluate a HW with Roofline Model
- Important context for papers and what to expect

Key Drivers for Neural Network Success

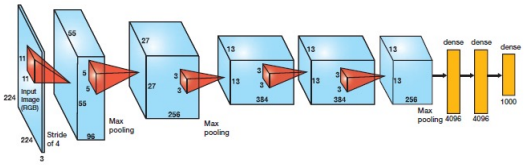
DARPA Neural Network Study Final Report (606 pages):

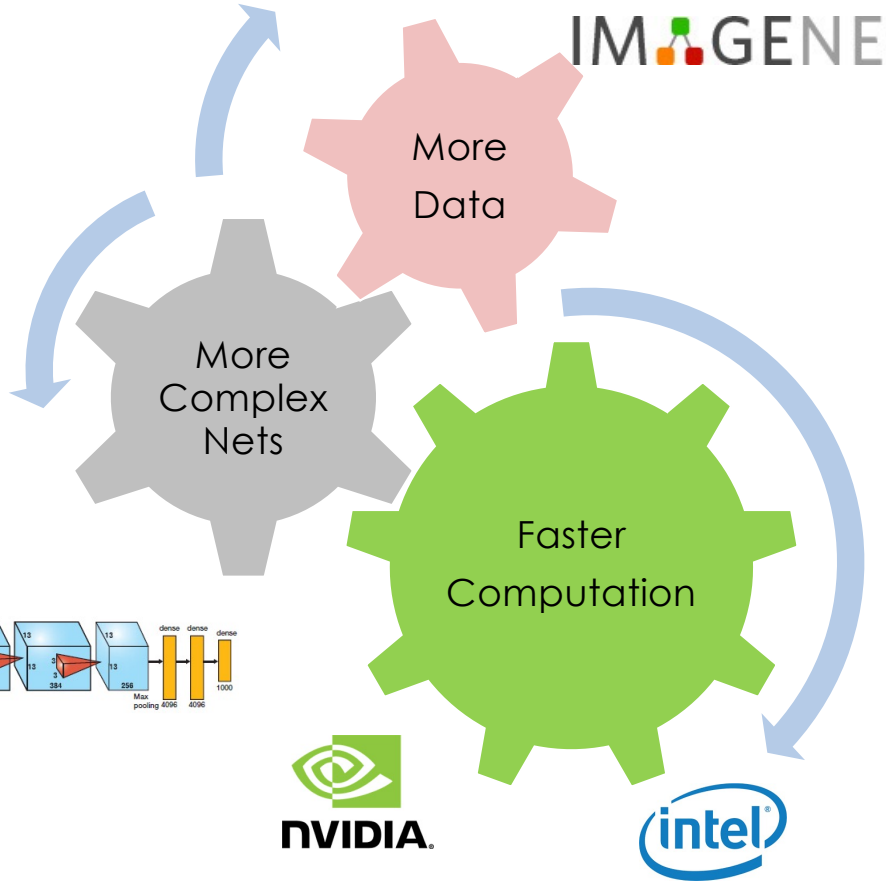
"After participating in this Study, my personal view is that **neural networks will provide the next major advance in computing technology.**"

Dr. Jasper Lupo

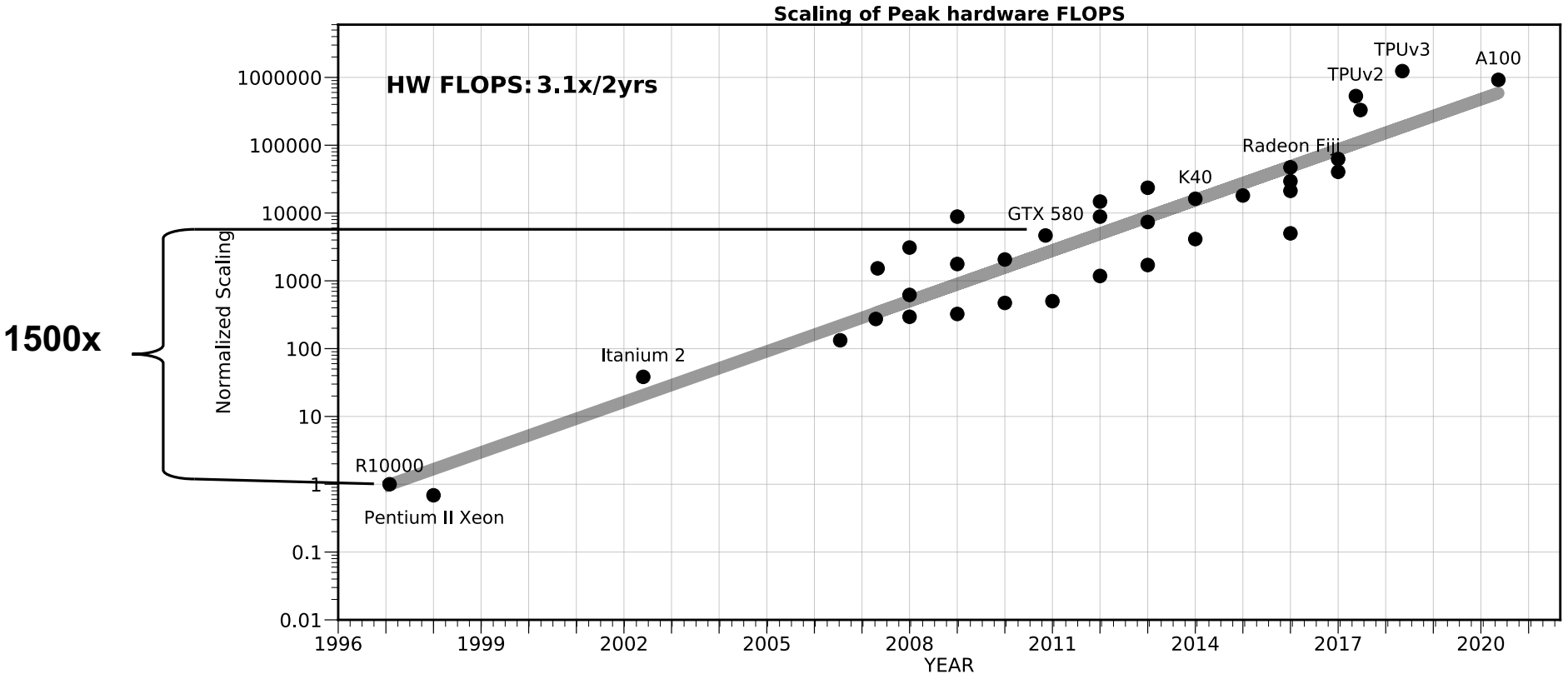
DARPA, Washington, DC

June, **1988**



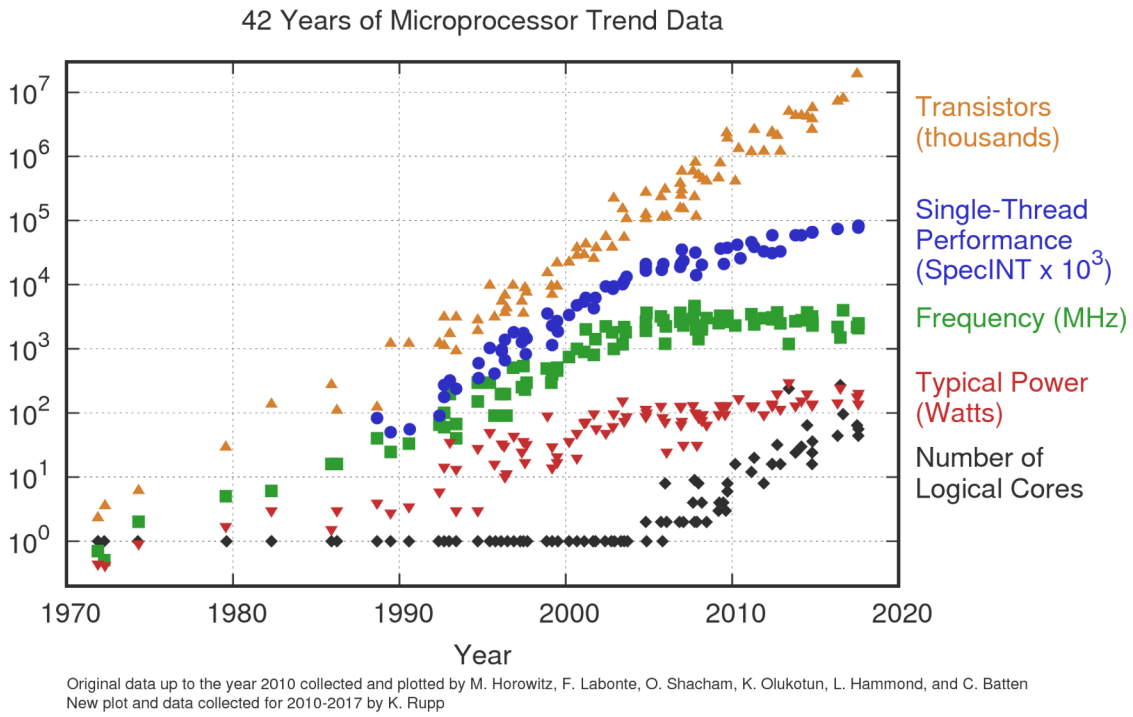


AlexNet vs Lenet5: 1000x More Compute



Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

General Purpose Hardware Trend

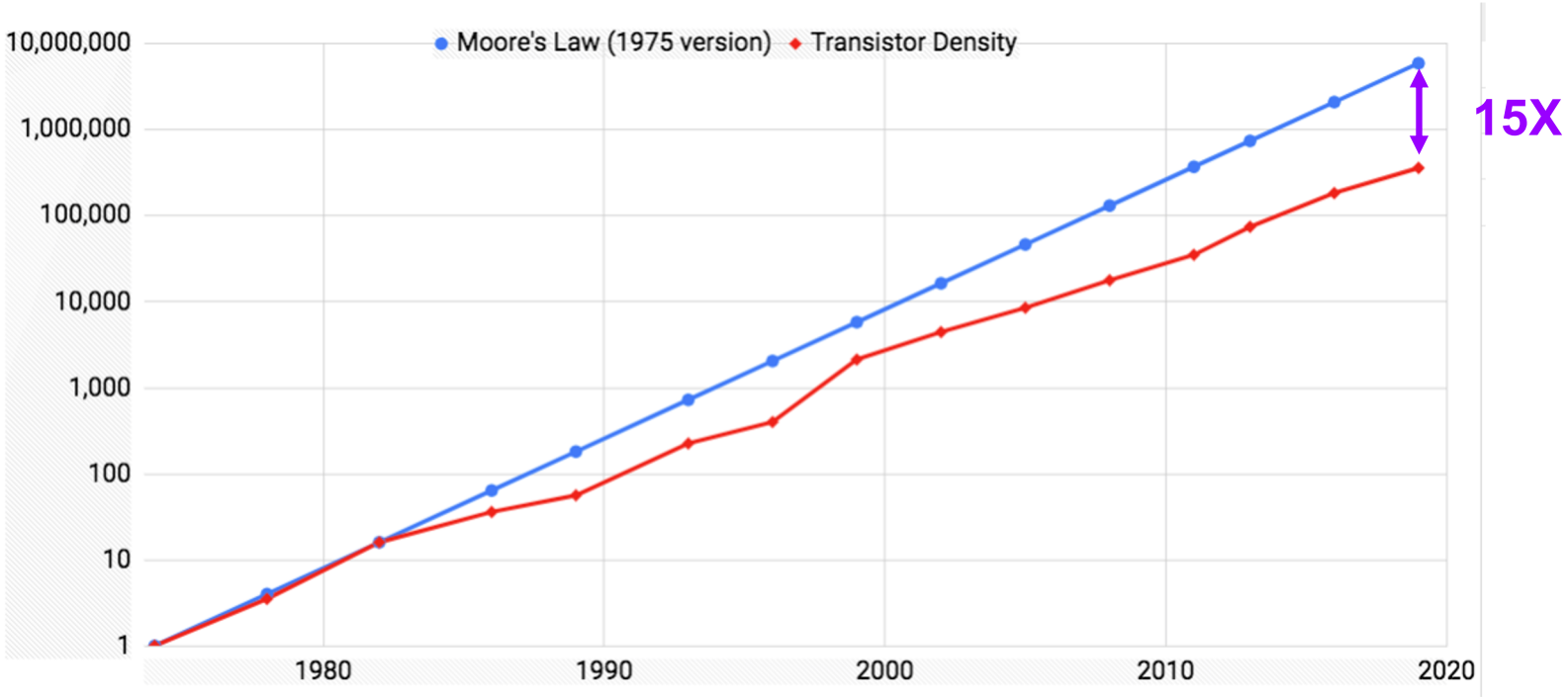


[42 Years of Microprocessor Trend Data](), Karl Rupp

Key Observations

- # Transistors still increasing
- Single Core Performance Plateauing
- End of Dennard Scaling
- Distributed Computing

Common Fallacy: Moore's Law is not Dead



Moore, Gordon E. "No exponential is forever: but 'Forever' can be delayed!" *Solid-State Circuits Conference, 2003.*

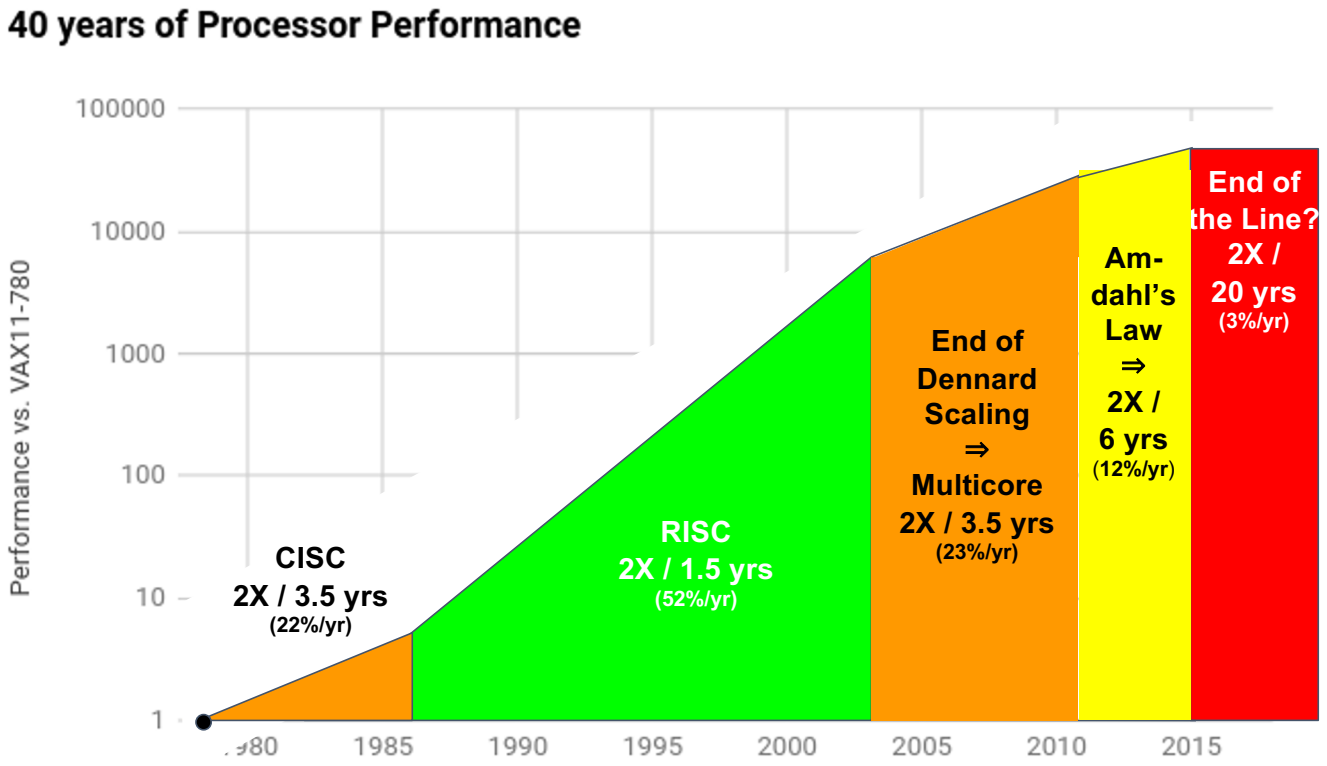
Slide from Prof. Patterson

It is becoming increasingly difficult to push the boundary

Building a 3nm fab costs around $20B. This is still economical given the $600B ARR for the semi-conductor industry, but it is questionable how much farther we can push the limit.

Source: high end performance packaging 3d/2.5d integration report, Yole, Development, 2020.

But It has Slowed Down



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

Domain Specific Accelerators

- John Hennessy and David Patterson, "A New Golden Age for Computer Architecture," *Communications of the ACM*, February 2019

Domain Specific Accelerators

SnapDragon 835 (→ 845 → 855)

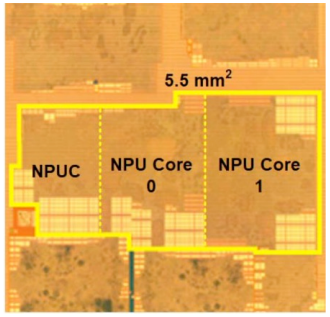
~3.23 – 4 MOPS/mW (835)

11– 16.6 GFLOPS SGEMM (835)

2.5K – 30K X increase in MOPs/mW

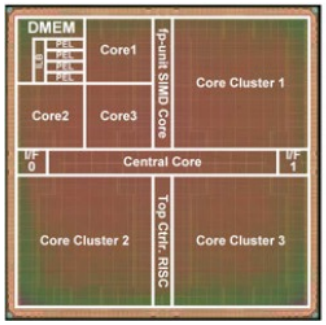
11,500 MOPS/mW

7.1 An 11.5TOPS/W 1024-MAC Butterfly Structure Dual-Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC



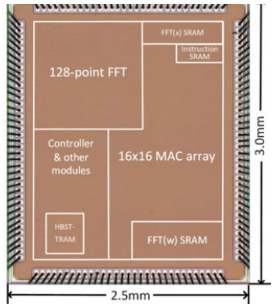
25,300 MOPS/mW

7.7 LNPU: A 25.3TFLOPS/W Sparse Deep-Neural-Network Learning Processor with Fine-Grained Mixed Precision of FP8-FP16



140,300 MOPS/mW

7.5 A 65nm 0.39-to-140.3TOPS/W 1-to-12b Unified Neural-Network Processor Using Block-Circulant-Enabled Transpose-Domain Acceleration with 8.1× Higher TOPS/mm² and 6T HBST-TRAM-Based 2D Data-Reuse Architecture



AI Chip Landscape

basicmi.github.io/AI-chip

Tech Giants/Systems

Google
Microsoft
aws
IBM
facebook
Apple
Tesla
HUAWEI
Baidu 百度
Alibaba Group 阿里巴巴集团
FUJITSU
NOKIA
TOSHIBA
Hewlett Packard Enterprise
DELL

IC Vender/Fabless

intel
SAMSUNG
NVIDIA
QUALCOMM
AMD
NXP
ST
XILINX
MEDIATEK
BROADCOM
MARVELL
Rockchip 瑞芯微电子

IP/Design Service

arm
SYNOPSYS
Imagination
cadence
CEVA
VeriSilicon
alchip
GUC
FARADAY
eSilicon

Startup in China

Cambricon 寒武纪科技
地平线 Horizon Robotics
BITMAIN
intellifusion 云天励飞
ChipIntelli
Think Force
Canaan
云知声 Unisound
AISPEECH 思必驰 专注人性化的智能语音
Rokid
肇观电子 NextVPU
Enflame

Startup Worldwide

cerebras
Wave Computing
Graphcore
habana
SambaNova SYSTEMS
thinci
LIGHTELLIGENCE
HAILO Empowering Intelligence
KALRAY
Tenstorrent
MYTHIC
Preferred Networks
brainchip
PEZY Computing
GREENWAVES TECHNOLOGIES
AIMOTIVE
KONIKU
Tachyum
flexlogix
SYNTIANT
gyrfalcon technology
NOVUMIND

more on https://basicmi.github.io/AI-Chip/

Compiler

TensorFlow
PyTorch
NVIDIA TensorRT
tvm
nGraph Compiler stack (Beta)
plaidML

Benchmarks

MLPerf
AI - Benchmark
AI Matrix.

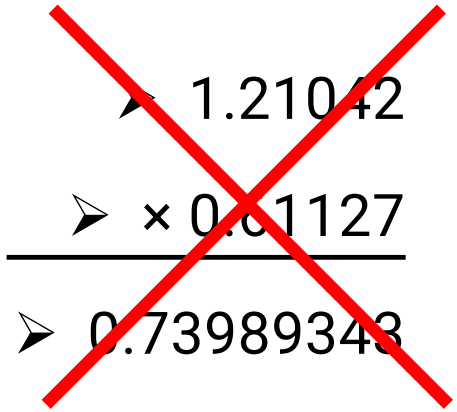
Two Important Innovations in AI Hardware That We Will Cover Today

- Low Precision Arithmetic Units
- Custom DataFlow Design

Low Precision Training

Low Precision Training

- For a lot of the operations in ML we use randomized algorithms. So why not use low precision instead of FP32 or FP64?



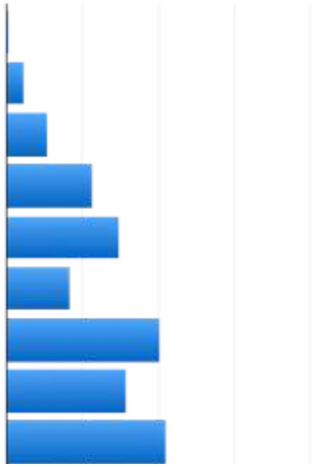
=>

- about 1.2
- × about 0.6
- about 0.7

- First, let's see how much performance gain we can get
- Second, there are subtleties associated with limited precision that need to be resolved

How much is on the table with low precision?

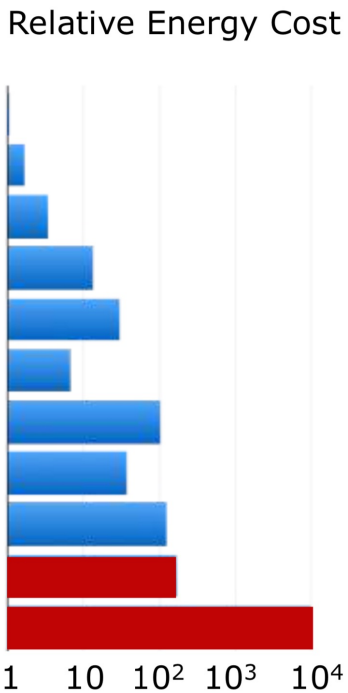
Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7



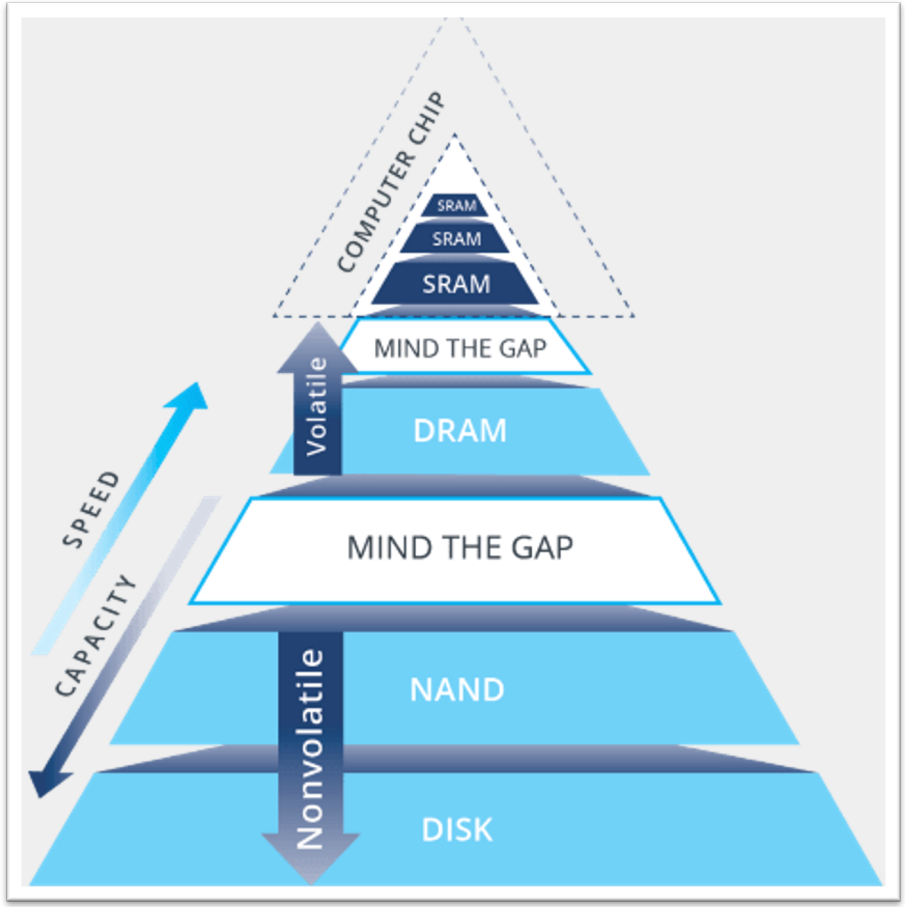
- At least 2x reduction in energy with FP16 compute

How much is on the table with low precision?

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



[**Horowitz**, *ISSCC* 2014]

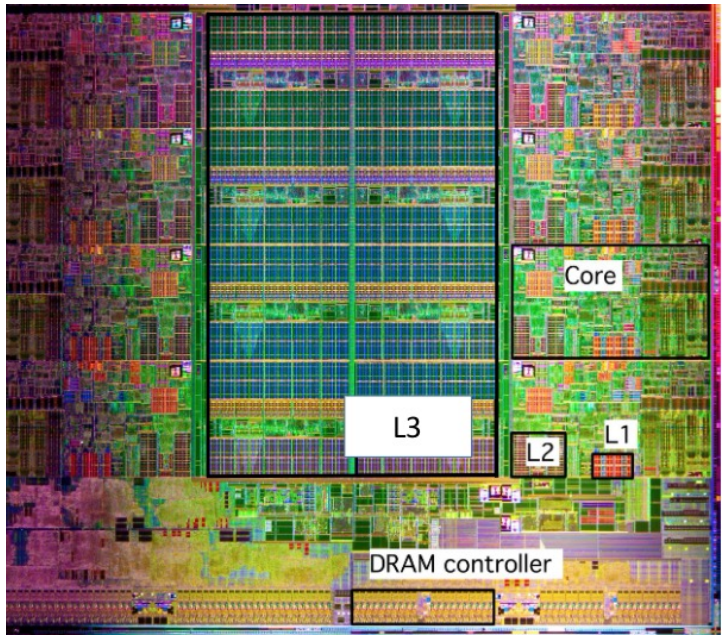


Memory hierarchy from Andrew Walker, EE Times.

Why can't we just increase SRAM?

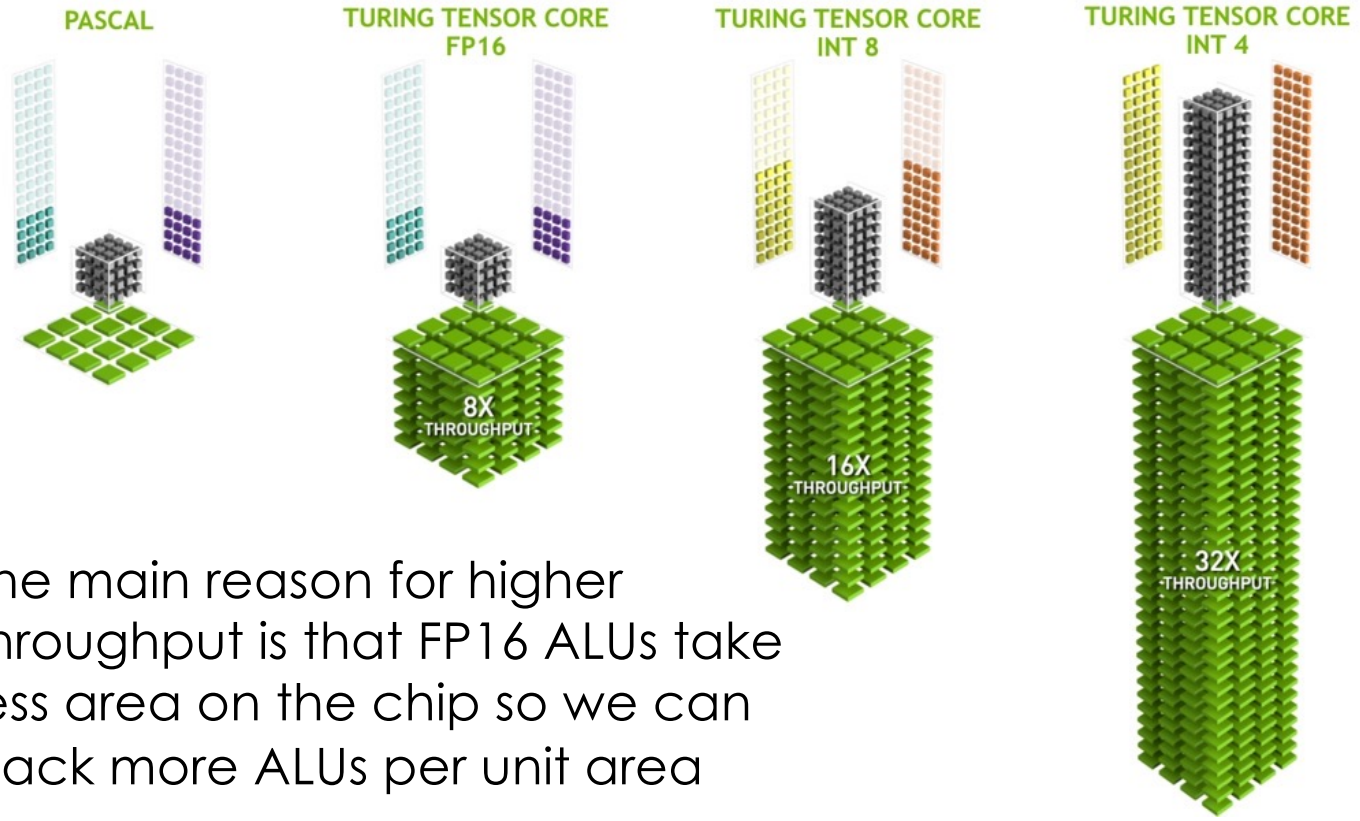
- SRAM is 100x more expensive than DRAM and importantly occupies more area of the chip!
- Using FP16 instead of FP32/64 reduces access volume without having to incur this cost

	Transistors per bit	Access Time	Persistent?	Sensitive?	Price
SRAM	6	1X	Yes	No	100X
DRAM	1	100X	No	Yes	1X



Intel Sandy Bridge-E Processor Die (Credit Rui Pan)

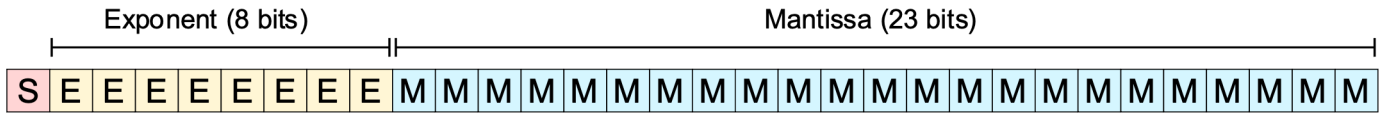
Faster Compute with FP16



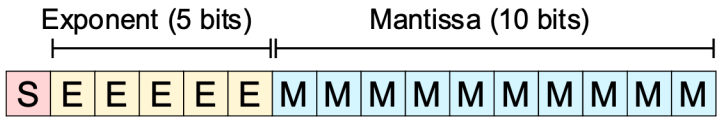
- The main reason for higher throughput is that FP16 ALUs take less area on the chip so we can pack more ALUs per unit area

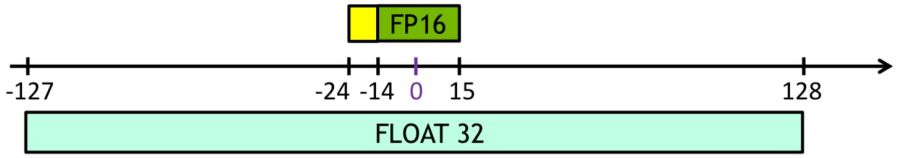
16 Bit Representation

Float32: (range ~$1e^{-38}$ to ~$3e^{38}$)

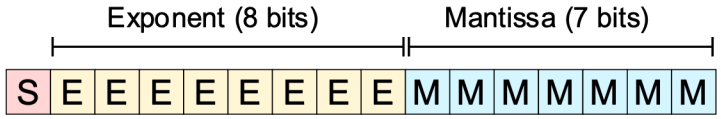


Float16: (range ~$5.96e^{-8}$ to 65,504)



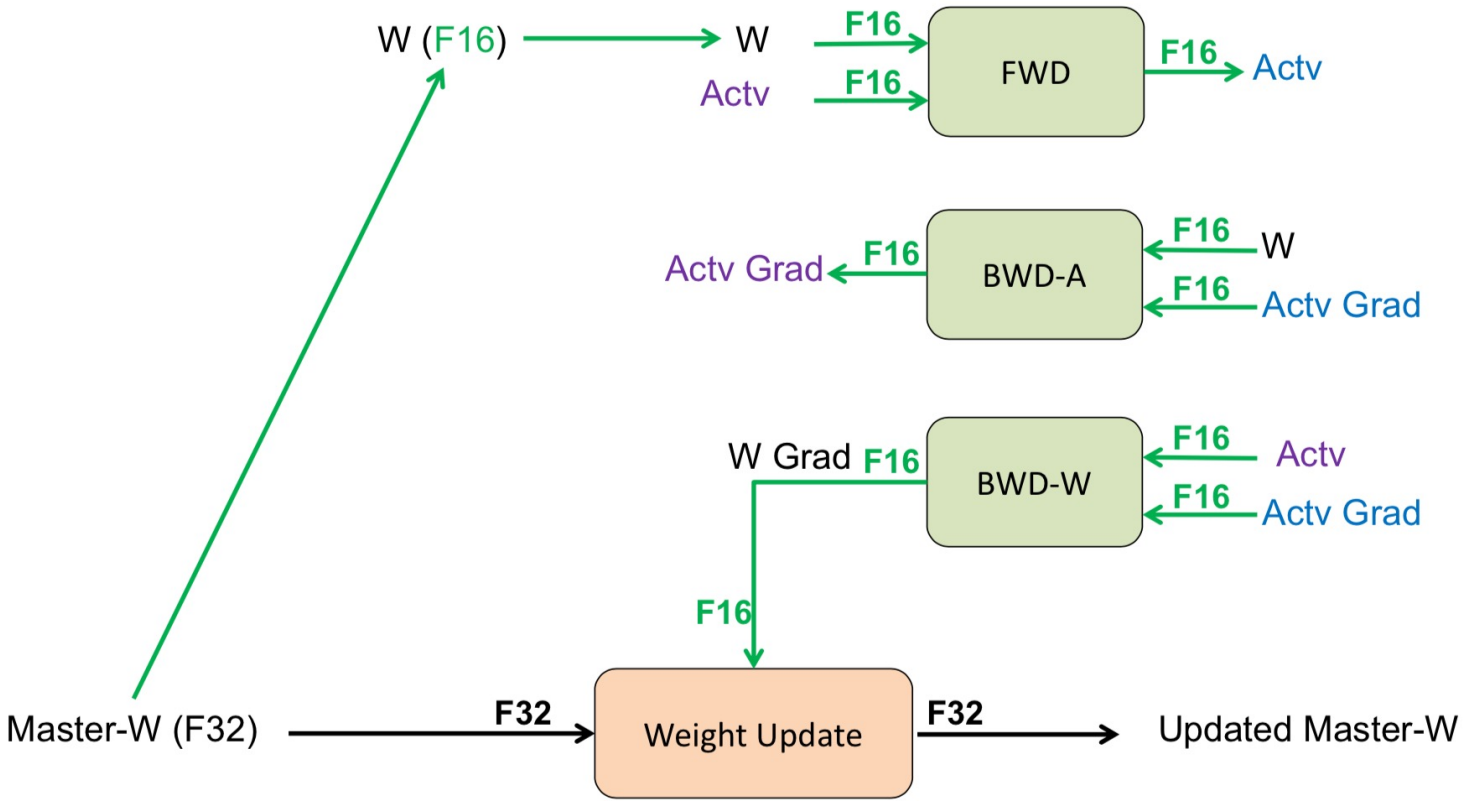


BFloat16: (range ~$1e^{-38}$ to ~$3e^{38}$)



*** Google largely avoided low precision training problems with BFLOAT16**

FP16/32 Training Workflow



ML Challenges with FP16 Training

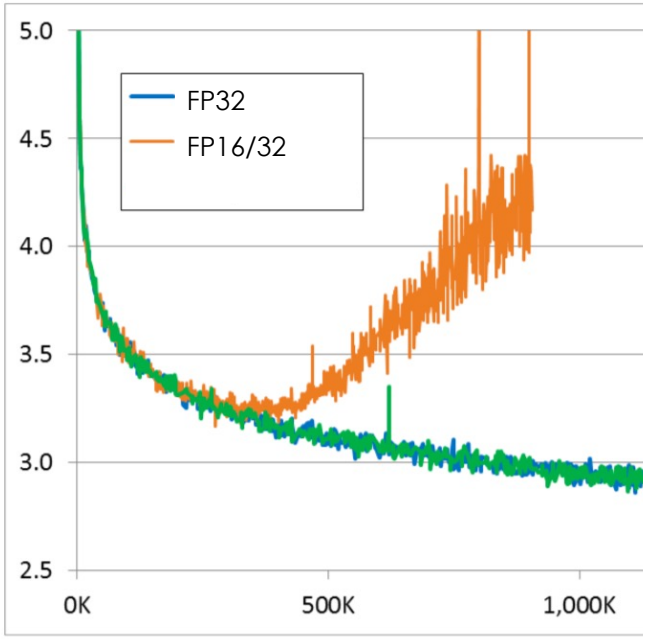
Naïve FP16 training leads to divergence/poor accuracy

FP16 has very limited dynamic range:

- Max =65504 , Min Normal Number = 2^{-14}

This creates several challenges in training NN:

- Loss of precision during weight update
- Exacerbates ‘‘vanishing and exploding gradients’’
- Overflow/underflow for layer activations during forward/backward pass

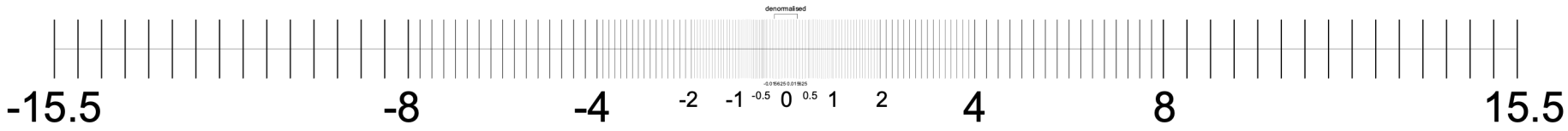


Loss curve for training AlexNet on ImageNet

P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, H. Wu. Mixed precision training. ICLR’18.
B. Ginsburg, S. Nikolaev, A. Kiswani, H. Wu, A. Gholami, S. Kierat, M. Houston, A. Fit-Flores. Tensor processing using low precision format. US Patent 15/624577.

Limitations of Floating Points

- There are only a **finite number** of different floats
- Floats are discrete but **not equidistant**



A hypothetical 8-bit floating point representation with 1 bit sign, 3 bit exponent, 4 bit mantisa
Notice that the representable numbers are not equidistant

Teaser question: What is the best way to sum an array with a lot of small values and some very large values to minimize numerical error?

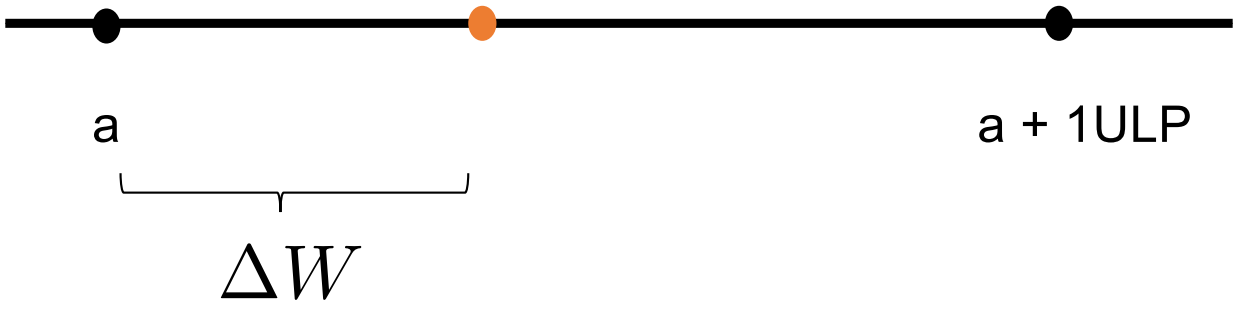
In IEEE representation $(a + b) + c \neq a + (b + c)$ no associative property

Figure Reference: volkerschatz.com

Unit in the Last Place (ULP)

- Floating points can only represent specific real valued numbers

$$W = W + \Delta W$$

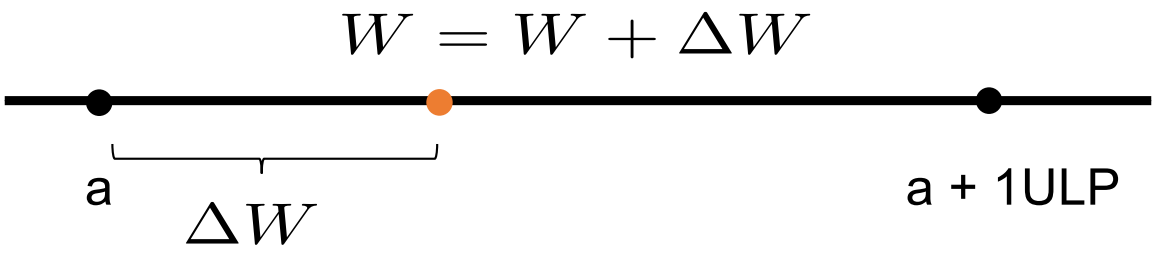


- Can lead to stalling at one point if the gradients are small

Stochastic Rounding

- One popular solution proposed to avoid this stalling is stochastic rounding

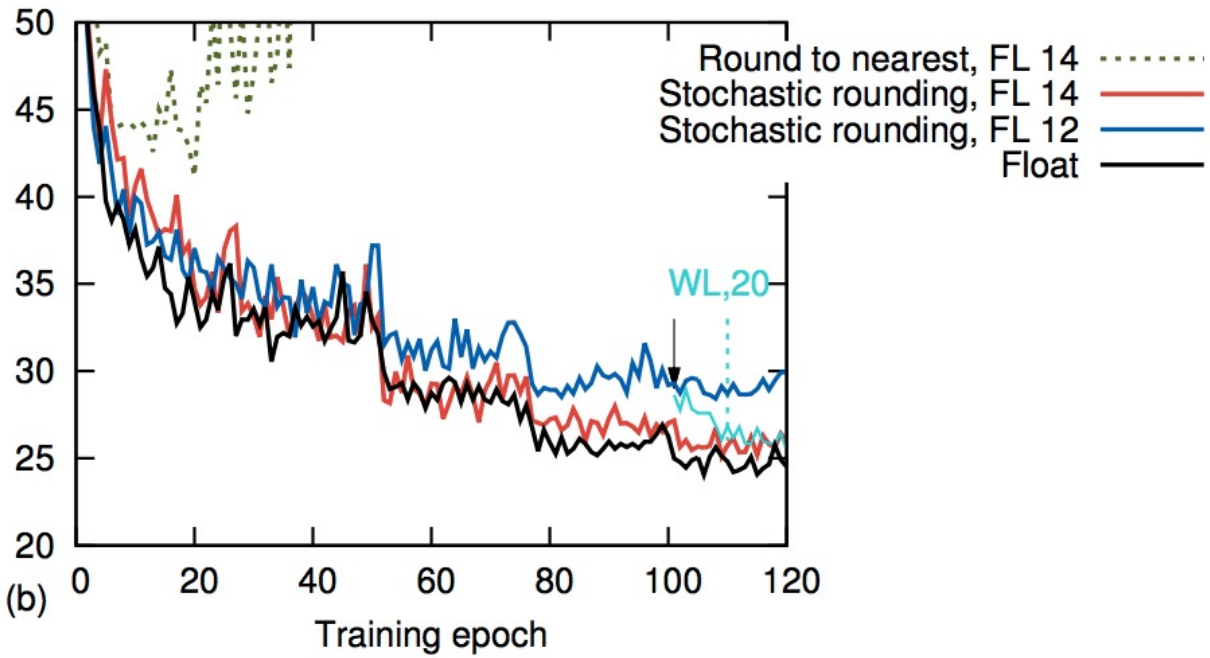
$$SR(x) = \begin{cases} \lfloor x \rfloor, & \text{w.p. } 1 - \frac{x - \lfloor x \rfloor}{\epsilon} \\ \lfloor x \rfloor + \epsilon, & \text{w.p. } \frac{x - \lfloor x \rfloor}{\epsilon} \end{cases}$$



Gupta S, Agrawal A, Gopalakrishnan K, Narayanan P. Deep learning with limited numerical precision. InInternational Conference on Machine Learning 2015 Jun 1 (pp. 1737-1746).
Ho¨hfeld, Markus and Fahlman, Scott E. Probabilistic rounding in neural network learning with limited precision. Neurocomputing, 4(6):291–299, 1992.

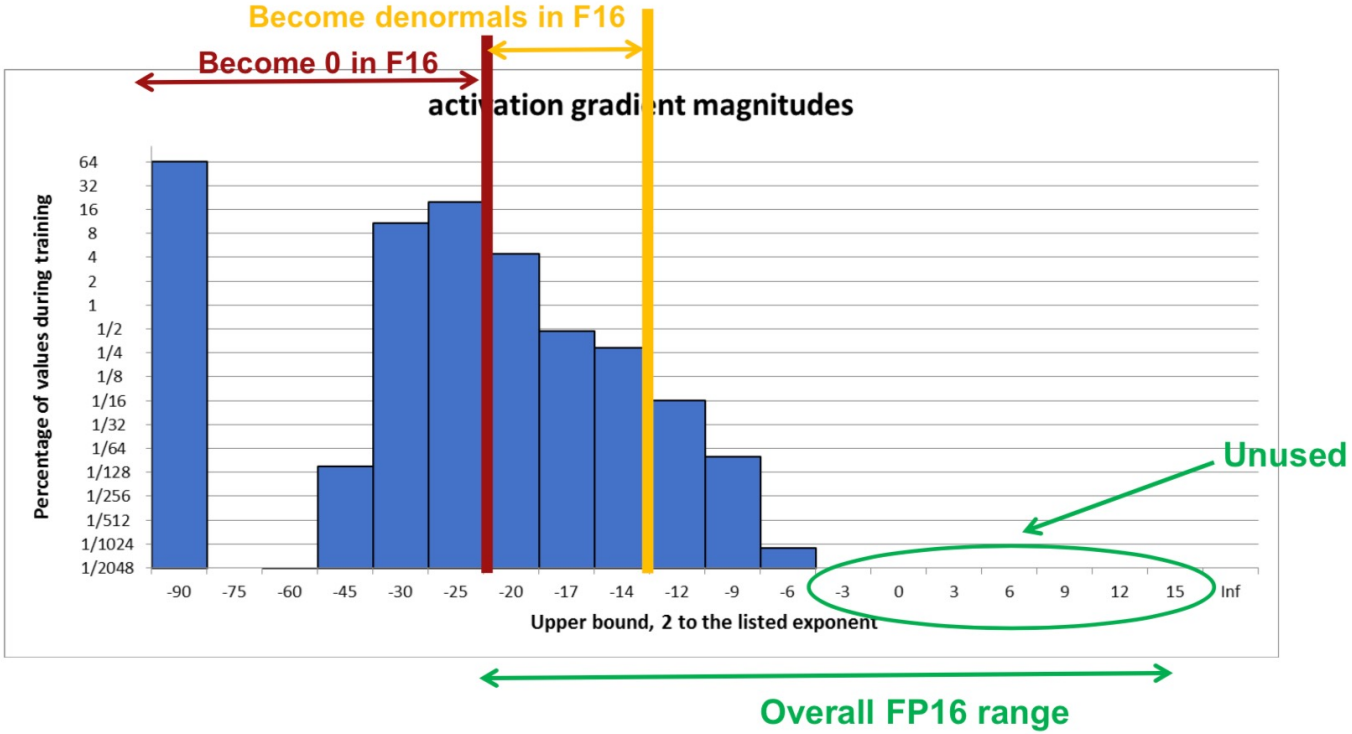
Stochastic Rounding

- About 5 percent accuracy loss for a 3 layer CNN on Cifar-10



Gupta S, Agrawal A, Gopalakrishnan K, Narayanan P. Deep learning with limited numerical precision. In International Conference on Machine Learning 2015 Jun 1 (pp. 1737-1746).
Ho¨hfeld, Markus and Fahlman, Scott E. Probabilistic rounding in neural network learning with limited precision. Neurocomputing, 4(6):291–299, 1992.

Vanishing Gradient Problem



- Main problem is in representing gradients
- Solution to be discussed during next week's PC meeting for [Micikevicius, 2018] paper!

Further Reading (INT8 Training?)

Several recent papers have proposed INT8 training. The ideas are interesting but:

- Many layers need to be kept at higher precision (FP16 or FP32)
- Master copy of weights needed at higher precision
- Lots of Hyper-parameter tuning is needed

Interesting papers to read:

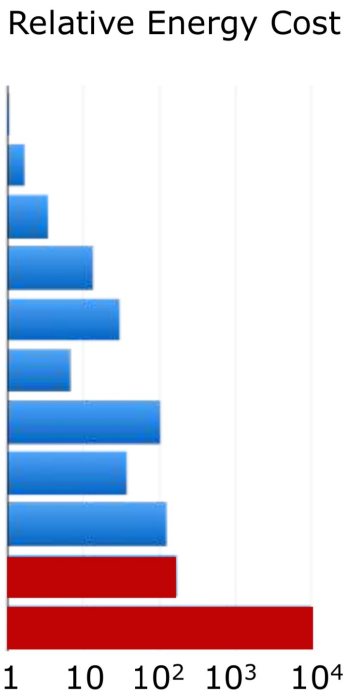
- Training DNNs with 8-bit FPs [NeurIPS'18]
- HybridFP8 Training [NeurIPS19]
- Shifted and Squeezed 8-bit Floating Point [ICLR'20]

Custom DataFlow

Motivation: Use A Custom DataFlow to minimize DRAM Accesses

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

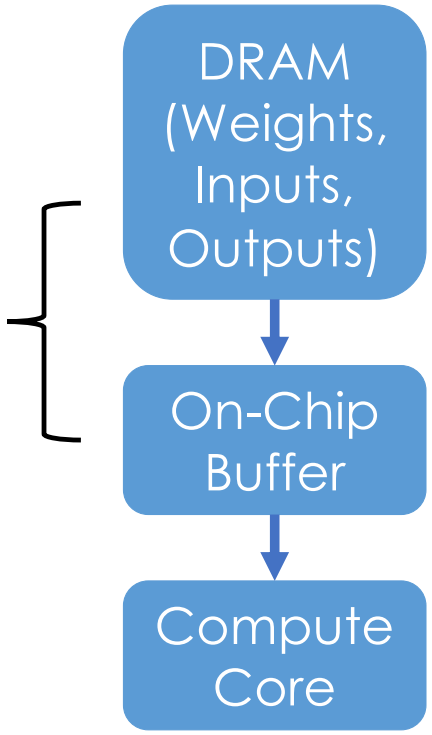
[**Horowitz**, *ISSCC* 2014]

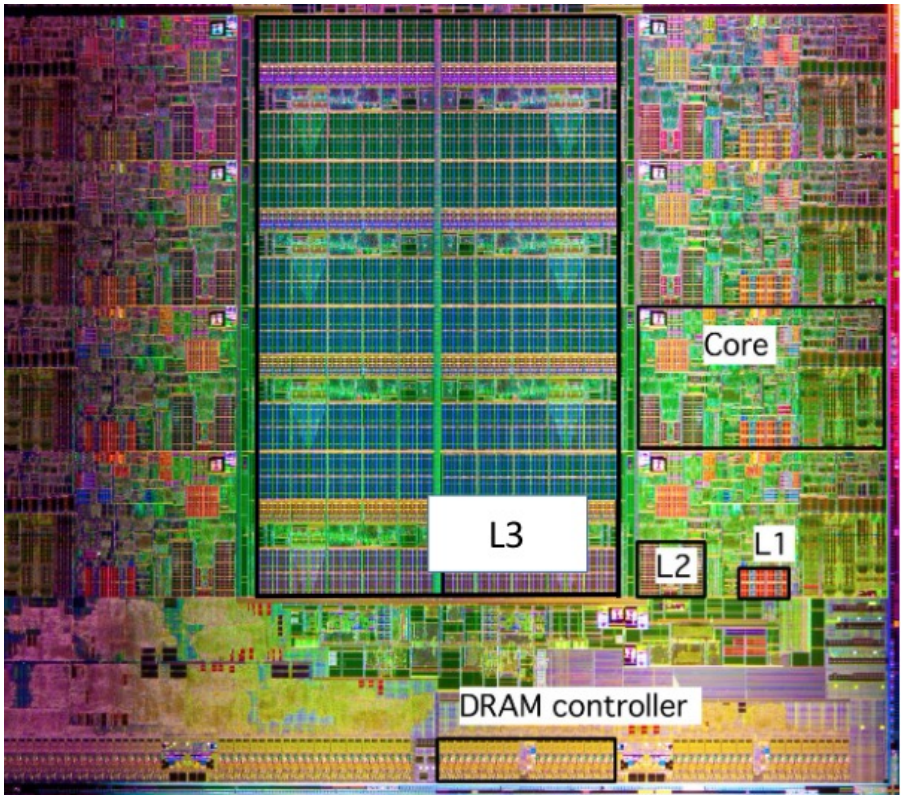


- **DRAM access are 100x costlier than SRAM.**
- **The number of times that we access DRAM depends on the order of the computations in our code!**
- **The goal is to find an order/dataflow to minimize this cost**

Custom DataFlow Motivation

Minimize data fetching from DRAM

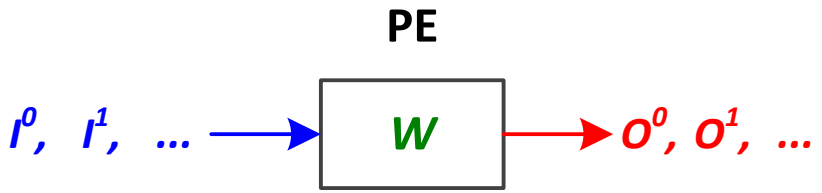




Intel Sandy Bridge-E Processor Die (Credit Rui Pan)

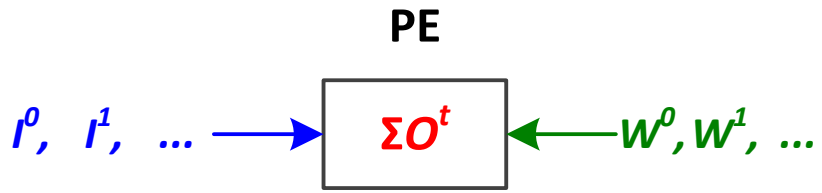
Taxonomy of Dataflow in NN Accelerators

We can classify NN accelerators based on spatial/systolic architecture according to the type of data each PE locally reuses.



Weight Stationary (WS)

- A PE reuses a kernel weight across different input activations.
- Examples: TPU[2], CNP[3]

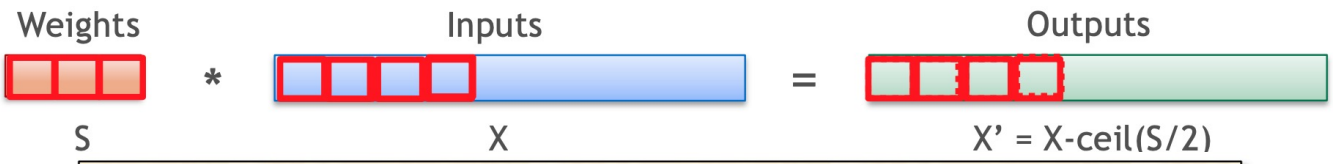


Output Stationary (OS)

- A PE accumulates partial products.
- Examples: ShiDianNao (SOC-MOP)[4], Envision (MOC-MOP)[5], DNA[6]

[1] Y.-H. Chen, et. al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," 2016.
[2] N. Jouppi, et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit," 2017.
[3] C. Farabet, et al., "CNP: An FPGA-based processor for Convolutional Networks," 2009.
[4] Z. Du, et al., "ShiDianNao: Shifting vision processing closer to the sensor," 2015.
[5] B. Moons, et al., "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," 2017.
[6] F. Tu, et al., "Deep Convolutional Neural Network Architecture With Reconfigurable Computation Patterns," 2017.C

Weight Stationary DataFlow (TPU)




```
int i[X];      # Input activations
int w[S];      # Filter weights
int o[X'];     # Output activations

for (s = 0; s < S; s++) {
    for (x = 0; x < X'; x++) {
        o[x] += i[x+s]*w[s];
}
```

How often are outputs/inputs accessed?

Every Cycle

How about the weights?

Every X' Cycles

What do we mean by Stationary?

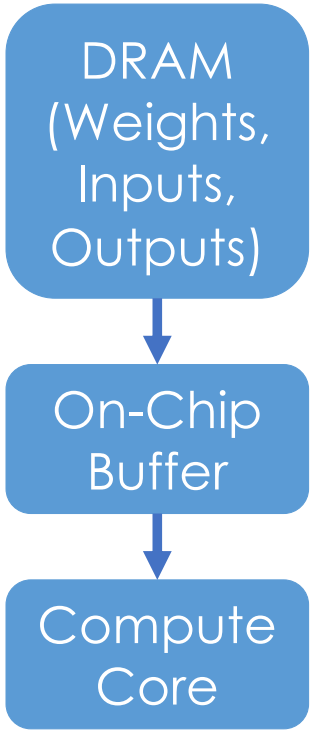
The datatype (and dimension) that changes most slowly

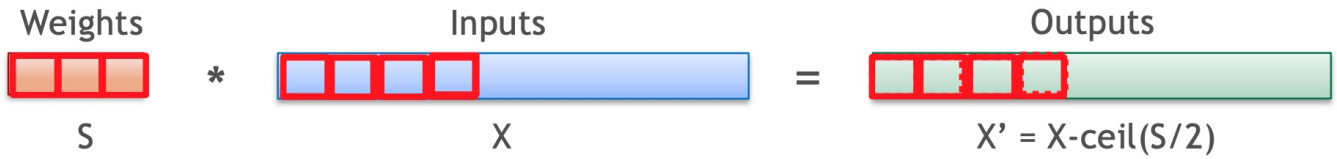
Imprecise analogy: think of data transfers as a wave with "amplitude" and "period"

- The stationary datatype has the **longest** period (locally held tile changes most slowly)
- Later we will see how intermediate staging buffers reduce both bandwidth and energy

Often corresponds to datatype that is "done with" earliest without further reloads

Impact of Local Buffer






```
for (s = 0; s < S; s++) {
        for (x = 0; x < X'; x++) {
            o[x] += i[x+s]*w[s];
}
```

- **Buffer size be to minimize refetching data from DRAM?**

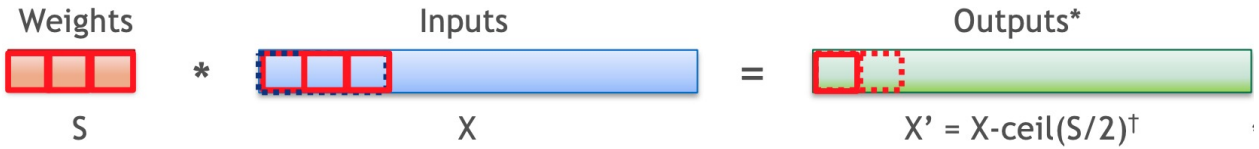
Dataflow	Outputs	Inputs	Weights
Weight Stationary (WS)	X'	X	1

- **What is the energy cost of accessing the buffers?**

$$SX'[f(X') + f(X') + f(X) + f(1)]$$

Where f is the energy of accessing a buffer of size x. See PPT notes for derivation.

Output Stationary DataFlow




```
int i[X];      # Input activations
int w[S];      # Filter weights
int o[X'];     # Output activations

for (x = 0; x < X'; x++) {
        for (s = 0; s < S; s++) {
            o[x] += i[x+s]*w[s];
}
```

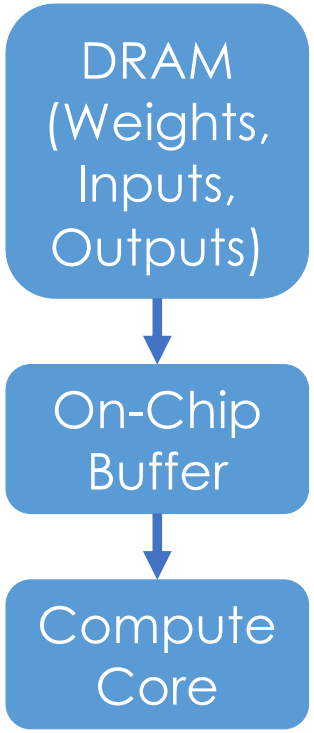
How often are outputs accessed?

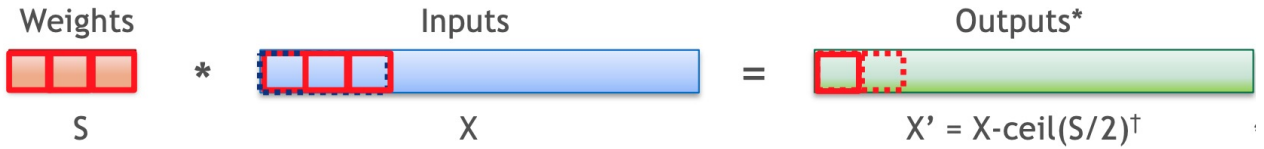
Every S Cycles

How about weights/inputs?

Every Cycle

Impact of Local Buffer: OS






```
for (x = 0; x < X'; x++) {
        for (s = 0; s < S; s++) {
            o[x] += i[x+s]*w[s];
}
```

- **Buffer size be to minimize refetching data from DRAM?**

Dataflow	Outputs	Inputs	Weights
Output Stationary (OS)	1	S	S

- **What is the energy cost of accessing the buffers?**

$$SX'[f(1) + f(1) + f(S) + f(S)]$$

Where f is the energy of accessing a buffer of size x. See PPT notes for derivation.

Weight Stationary vs Output Stationary

Buffer size be to minimize refetching data from DRAM

Dataflow	Outputs	Inputs	Weights
Weight Stationary (WS)	X'	X	1
Output Stationary (OS)	1	S	S

Very different energy cost of accessing the local buffers by just a change in the loop order

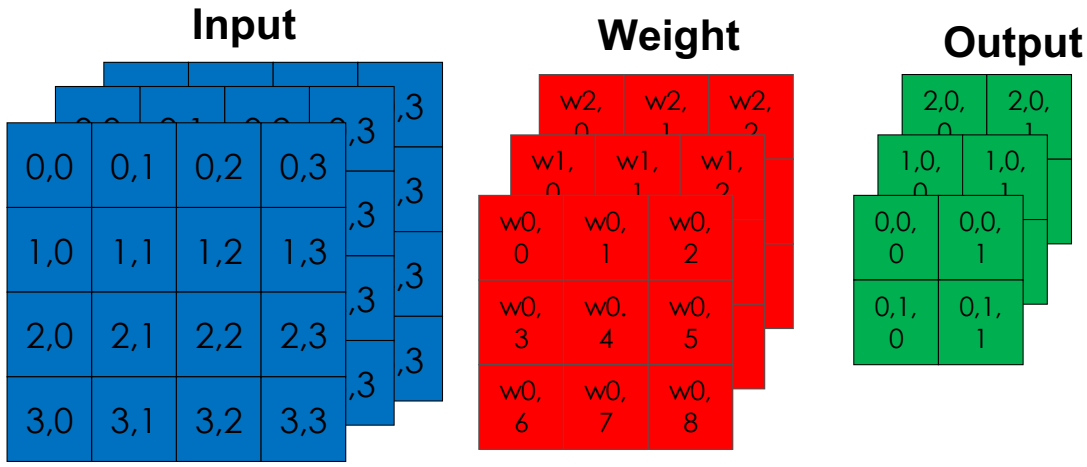
WS: $$SX'[f(X') + f(X') + f(X) + f(1)]$$

OS: $$SX'[f(1) + f(1) + f(S) + f(S)]$$

Convolution: Restructured as Matrix Multiplication

In practice we deal with more complex loop structures.

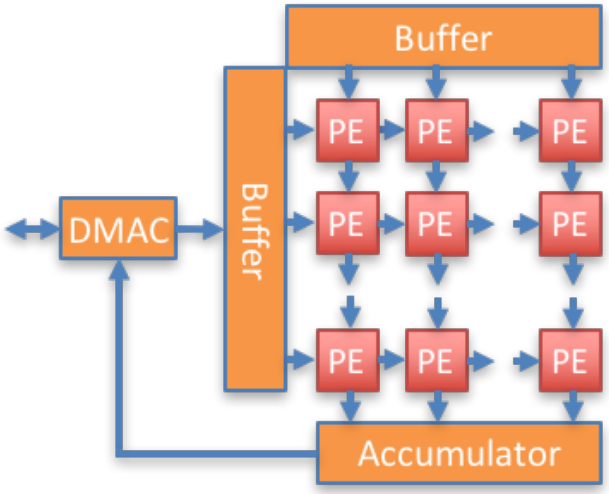
Stay tuned for the compilers lecture on how ML could be used to optimize the dataflow pattern for these cases.

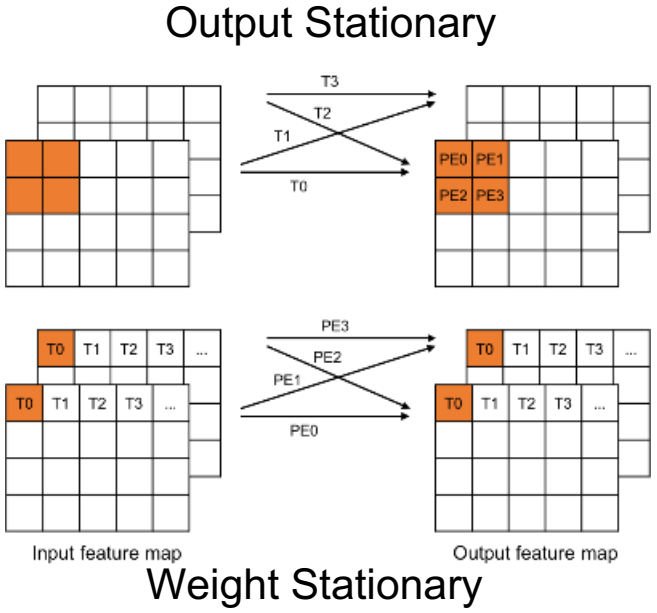



```
for k ← 0 to C_o do ;                    // Output Channels
 for y ← 0 to H do ;                      // H: Height
  for x ← 0 to W do ;                     // W: Width
   O[k][y][x] = 0;
   for c ← 0 to C_i do ;                  // Input Channels
    for j ← 0 to K_h do ;                 // Filter Size
     for i ← 0 to K_w do
      O[k][y][x] += I[c][y+j][x+i] * W[k][c][j][i]
```

Squeezelerator: Hybrid WS/OS (Berkeley, Samsung)

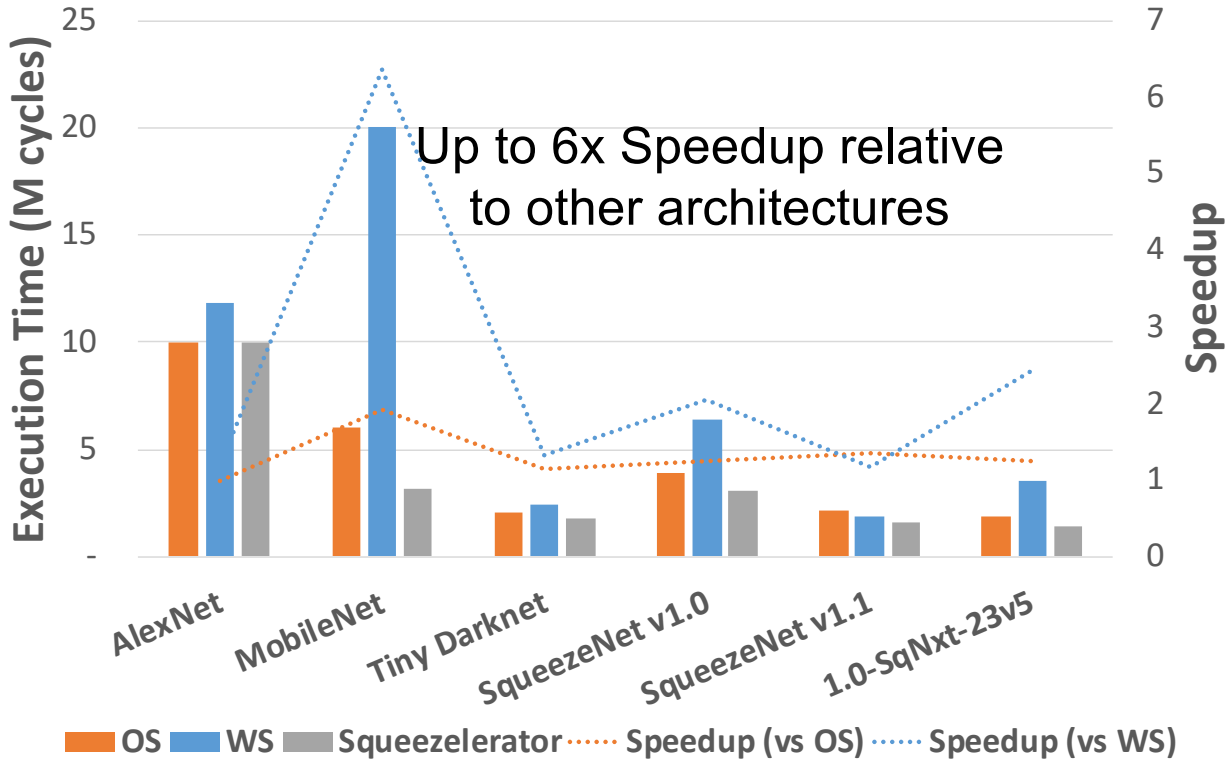
- Squeezelerator: Supports hybrid WS and OS data flow: up to 6x reduction in energy over OS, WS
  - Determines execution flow statically based on trained weights (one time setup cost per network)





Kwon, Kiseok, Alon Amid, Amir Gholami, Bichen Wu, Krste Asanovic, and Kurt Keutzer. "Co-design of deep neural nets and neural net accelerators for embedded vision applications." In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1-6. IEEE, 2018.

Squeezelerator: Hybrid OS/WS



Kwon, Kiseok, Alon Amid, Amir Gholami, Bichen Wu, Krste Asanovic, and Kurt Keutzer. "Co-design of deep neural nets and neural net accelerators for embedded vision applications." In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1-6. IEEE, 2018.

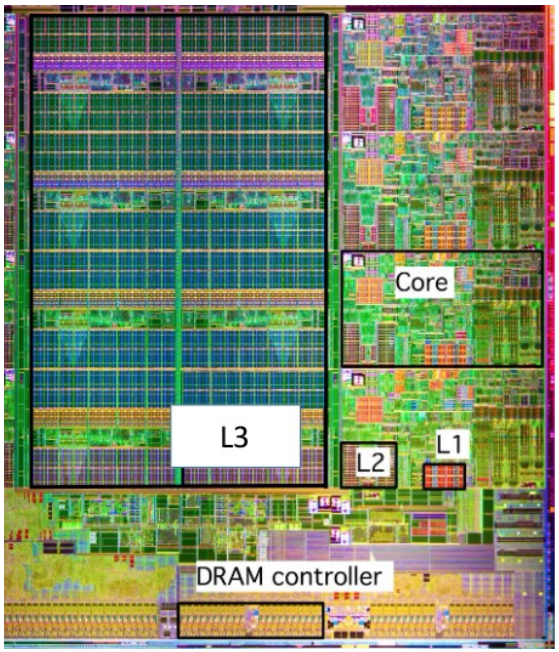
Gholami A, Kwon K, Wu B, Tai Z, Yue X, Jin P, Zhao S, Keutzer K. Squeezenext: Hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 2018 (pp. 1638-1647).

How to Evaluate HW Performance

FLOPs is not the same as wall clock time

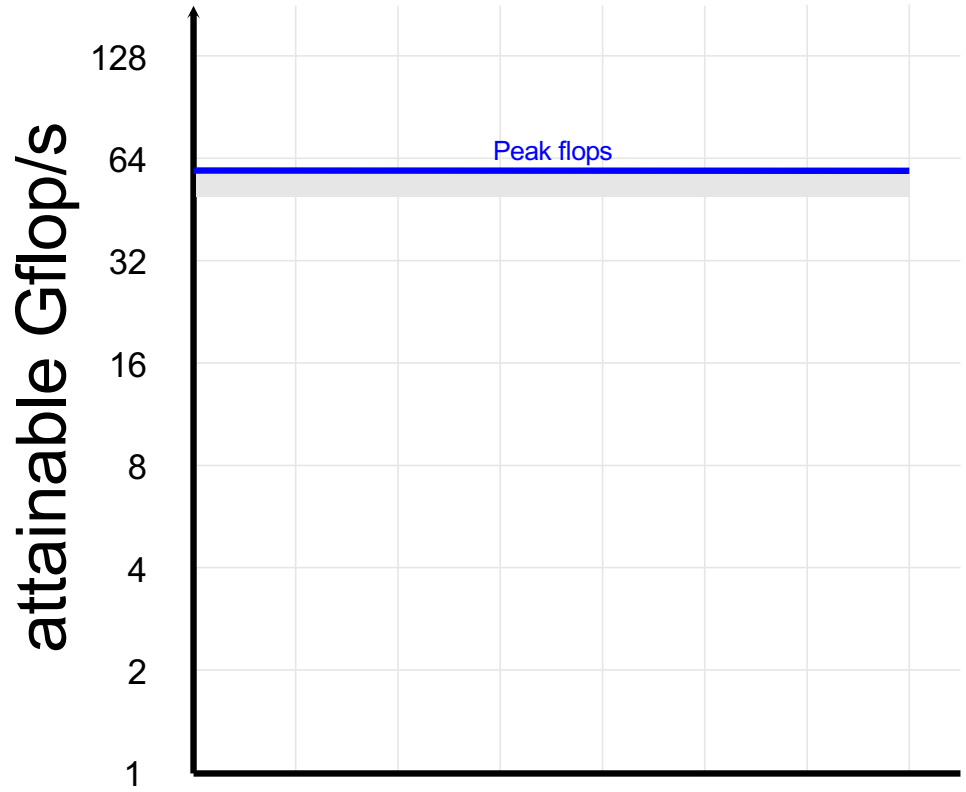
Williams Roofline Model

- For decades programmers and architects did back-of-the-envelope calculations on compute vs communication at various levels of the memory hierarchy
  - Processor to register file
  - On-chip L1, L2 (L3?) caches
  - Off-chip DRAM
  - Interprocessor communication
- UC Berkeley grad student Sam Williams gave a simple model, known as the Roofline Model, for reasoning about these issues



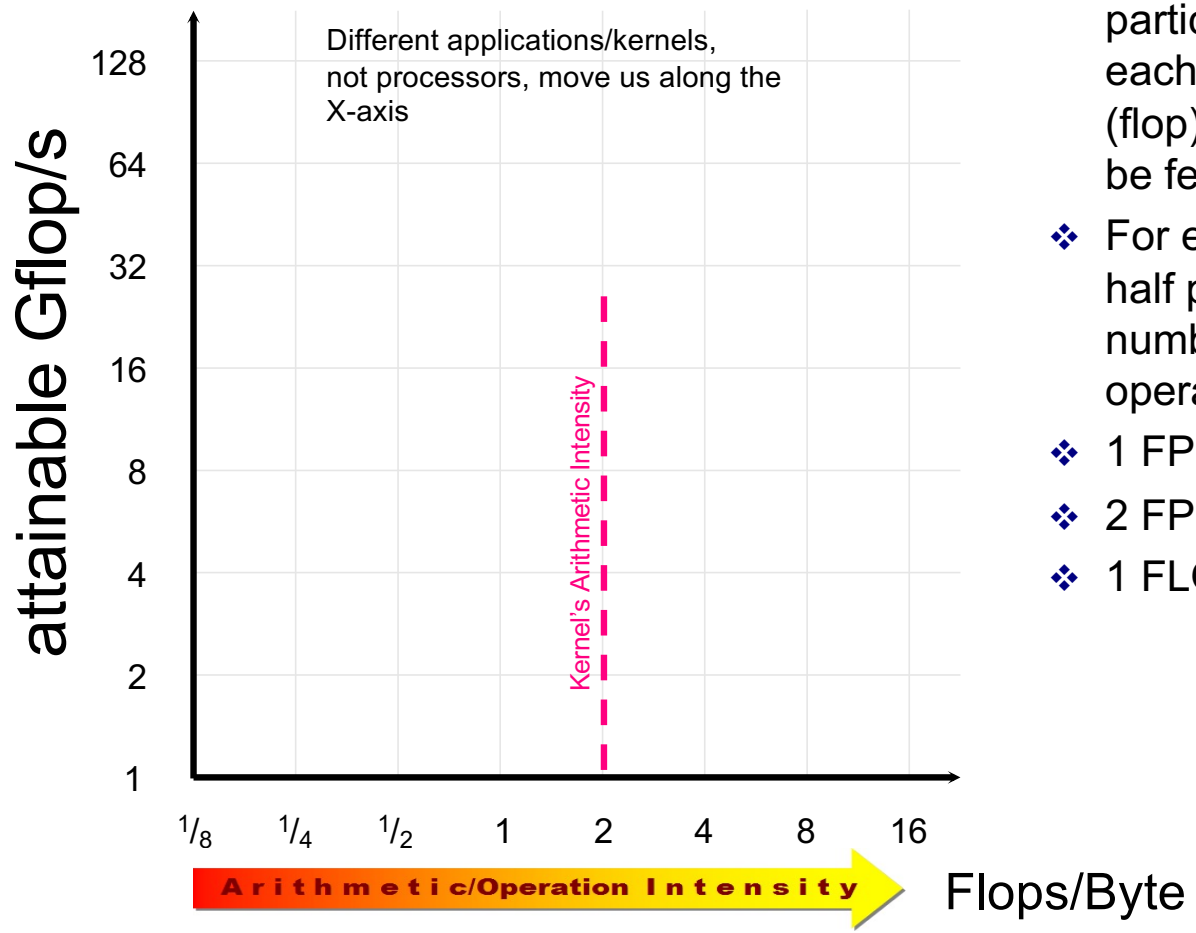
Williams, Samuel, Andrew Waterman, and David Patterson. *Roofline: An insightful visual performance model for floating-point programs and multicore architectures*. No. LBNL-2141E. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009.

Roofline Model: y-axis



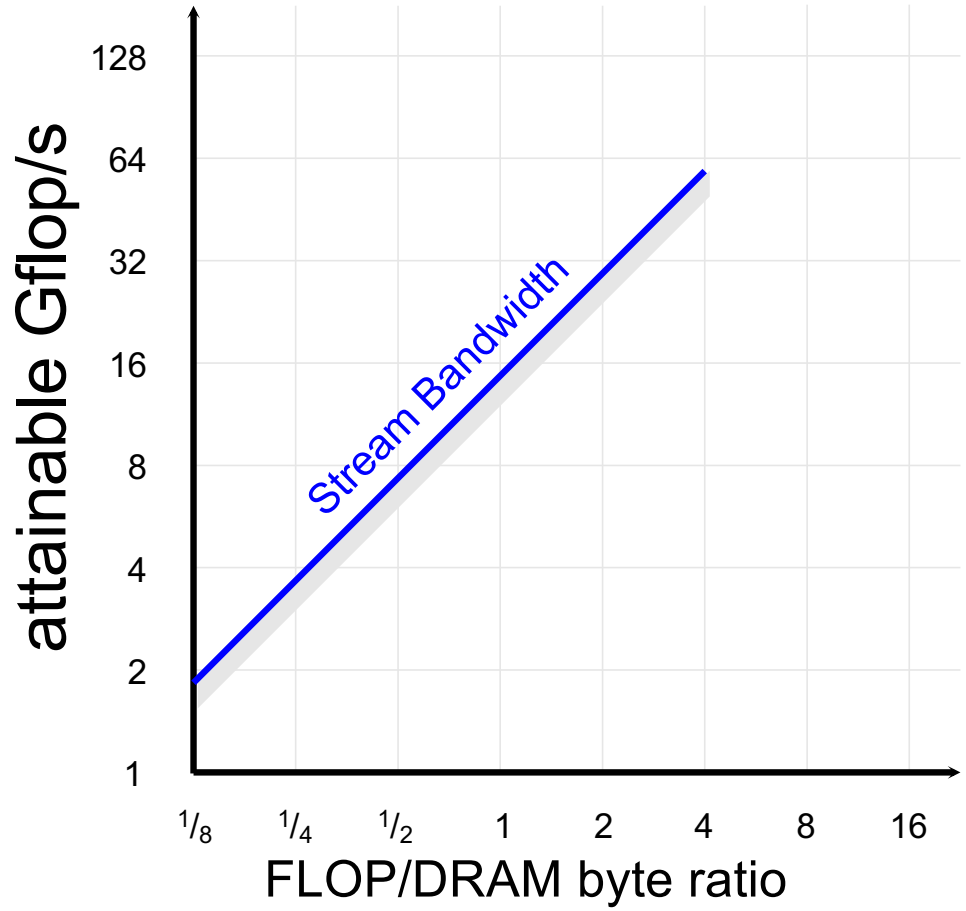
- The y-axis describes the attained performance
- It's easy to add the "peak performance" as an upper bound

Roofline Model: x-axis



- The x-axis tells indicates for this particular application/kernel, for each floating-point operation (flop), how many bytes (B) must be fetched
- For example if we have to fetch half precision floating-point numbers for each floating point operation, then:
- 1 FP16: 2 Bytes (16 bits)
- 2 FP16: 4 Bytes (32 bits)
- 1 FLOP requires 4 DRAM Bytes

Bandwidth as Slope



Bandwidth is represented as a slope of

Peak Flops/ AI

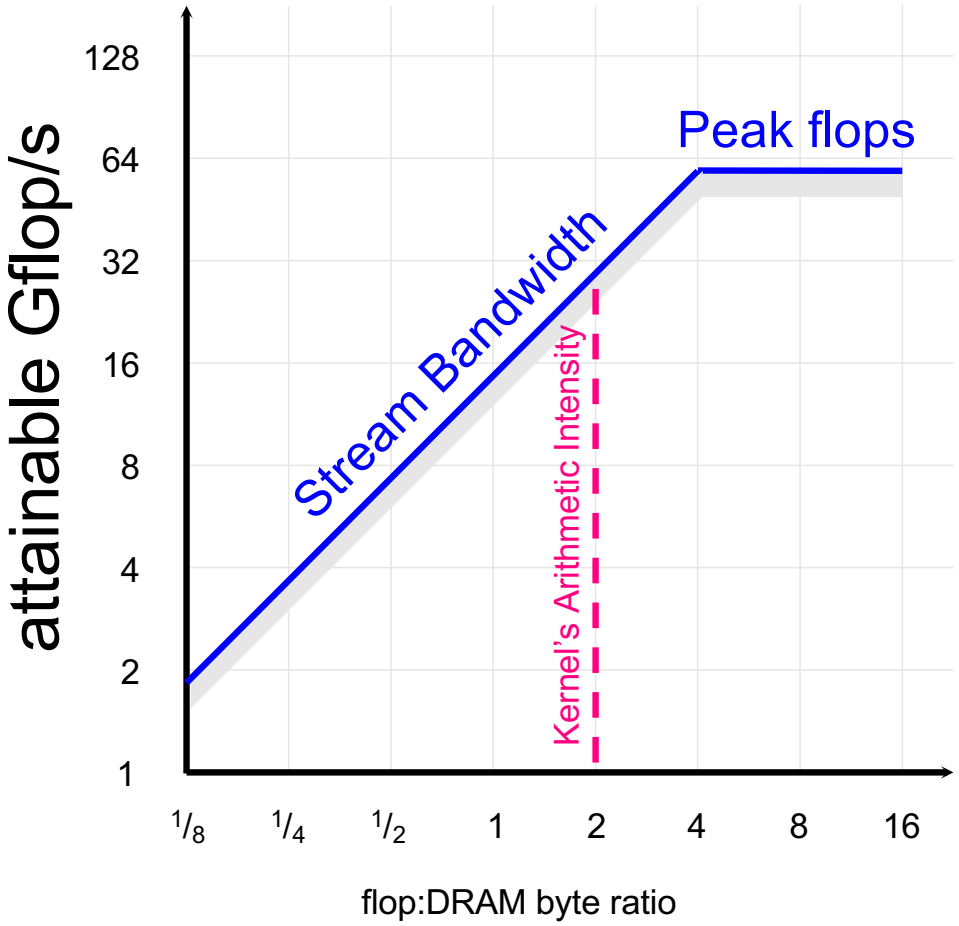
It is a given by the system configuration/architecture

- Remember $m = \frac{y}{x}$

Example

- m = y/x = 16
- y= $\frac{2G\ Flop}{second}$
- x = $\frac{1\ Flop}{4\ bytes}$

Bandwidth Meets Arithmetic Intensity



$$FLOPS \neq Time$$

Designing an accelerator

1) Accelerators are Only the First 80% of the Problem

The other 80%: Full system design

The remaining 200%: SW development

2) HW design shouldn't be about what can be built, rather what can be programmed

Stay tuned for the Lecture on AI Frameworks

3) Deploy at scale? Stay tuned for the next two lectures on scaling training

Research Challenges (if you are interested definition checkout EE 290 course by Prof. Shao)

- Abstraction Levels, DSLs
- Coarse Grained vs Fine Grain Acceleration
- Programming Languages, Programming Models
- Optimization Techniques, Runtimes
- HW Definition Languages
- Co-Design Methodology
- Datacenter / Design, Deployment
- Datacenter / Heterogeneity

Naveen Kumar (Google)

What Does the Future Look Like?

The right dataflow, precision, and many other parameters heavily depend on the workload.

If you were to design a HW today, it would at least take **~2years** for its tape out, with a cost of **~$500M** (estimate for 3nm)

It must remain relevant through ~5 years to justify the huge upfront investment

What do you think is the right workload for the future to bet on?

Next week's readings

Reading for Next Week

- Mixed precision training. ICLR'18
  - Discusses reduced precision training with IEEE FP16.
- Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks [SIGRAPH'16]
  - Discusses impact of dataflow on AI Hardware performance, introduces a new dataflow called row stationary
- Interstellar: Using Halide's Scheduling Language to Analyze DNN Accelerators (formerly: DNN Dataflow Choice Is Overrated) [ASPLOS'20]
  - Studies the impact of different dataflow patterns
- Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration [DAC'21 [Best Paper Award]]
  - Introduces an AI accelerator designed by UC Berkeley team with superior performance even compared to SOTA HW used in industry

Extra Suggested Reading

- Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures [CACM'08]
  - Introduces a very simple method to evaluate/understand HW performance
- A new golden age for computer architecture [CACM'19]
  - Prof. Patterson and Prof. Hennessy's great article on the motivations and opportunities for Domain Specific Accelerators
- In-Datacenter Performance Analysis of a Tensor Processing Unit [ISCA'17]
  - Design choices and performance analysis of TPU