In-Data Center Performance Analysis of a Tensor Processing Unit^{TM}

Norman P. Jouppi, **Cliff Young**, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg,Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon

cliffy@google.com

Hot Chips, August 22, 2017

A Golden Age in Microprocessor Design

- Stunning progress in microprocessor design 40 years ≈ 10^6x faster!
- Three architectural innovations (~1000x)
  - Width: 8->16->32 ->64 bit (~8x)
  - Instruction level parallelism:
    - 4-10 *clock cycles per instruction* to 4+ *instructions per clock cycle* (~10-20x)
  - Multicore: 1 processor to 16 cores (~16x)
- Clock rate: 3 to 4000 MHz (~1000x thru technology & architecture)
- Made possible by IC technology:
  - **Moore's Law**: growth in transistor count (2X every 1.5 years)
  - **Dennard Scaling**: power/transistor shrinks at same rate as transistors are added (constant per mm^2 of silicon)

Source: John Hennessy, "The Future of Microprocessors," Stanford University, March 16, 2017

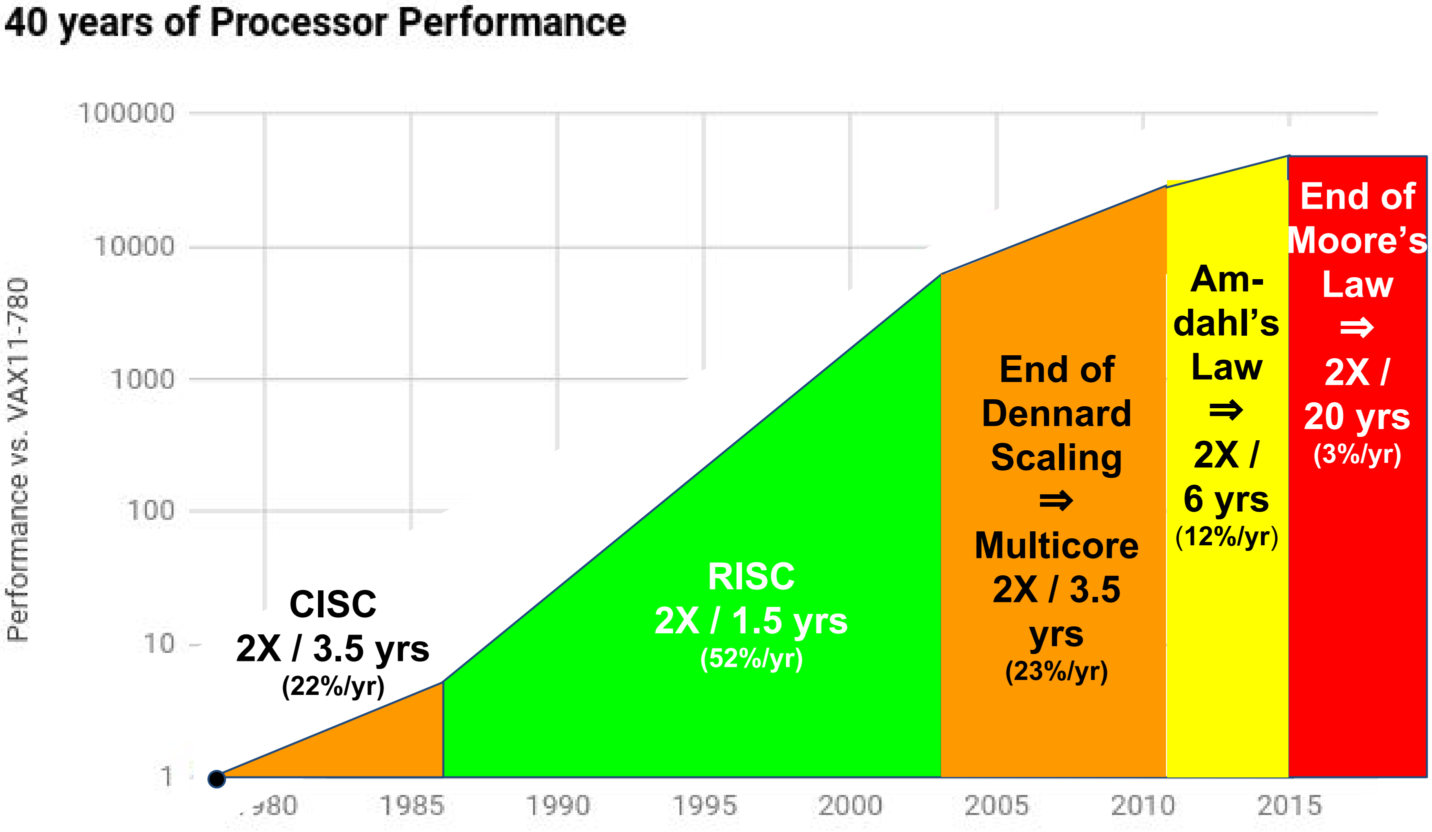
Changes Converge

- Technology
  - End of Dennard scaling: power becomes the key constraint
  - Slowdown (retirement) of Moore’s Law: transistors cost

- Architectural
  - Limitation and inefficiencies in exploiting instruction level parallelism end the uniprocessor era in 2004
  - Amdahl’s Law and its implications end “easy” multicore era
- Products
  - PC/Server ⇒ Client/Cloud

Source: John Hennessy, “The Future of Microprocessors,” Stanford University, March 16, 2017

End of Growth of Performance?



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

What’s Left?

Since

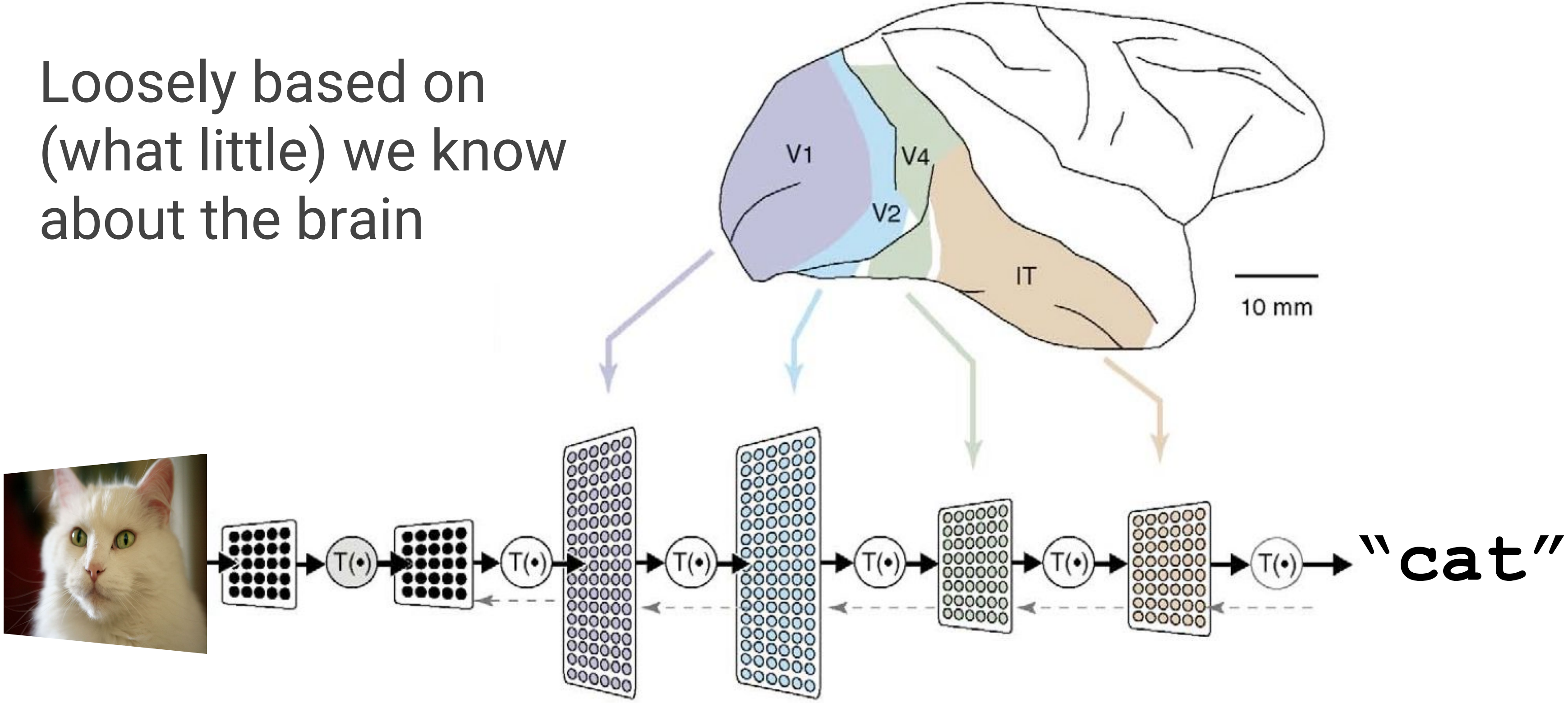
- Transistors not getting much better
- Power budget not getting much higher
- Already switched from 1 inefficient processor/chip to N efficient processors/chip

Only path left is *Domain Specific Architectures*

- Just do a few tasks, but extremely well

What is Deep Learning?

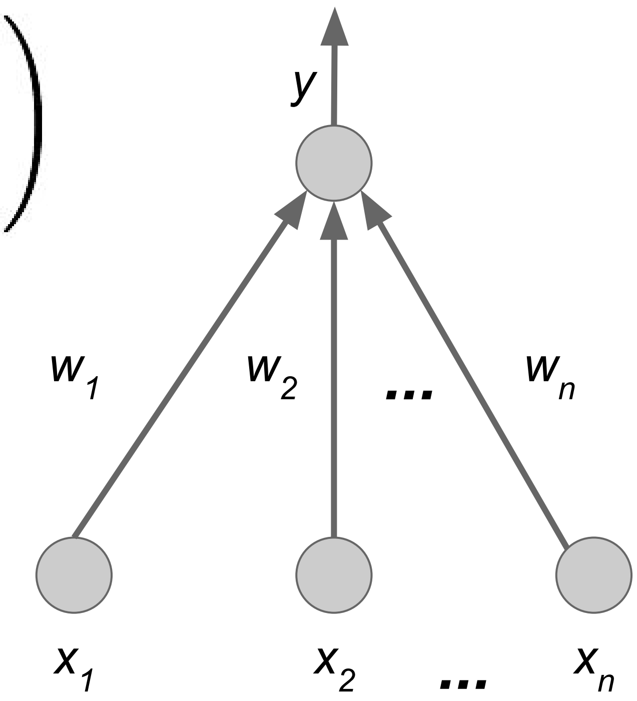
- Loosely based on (what little) we know about the brain

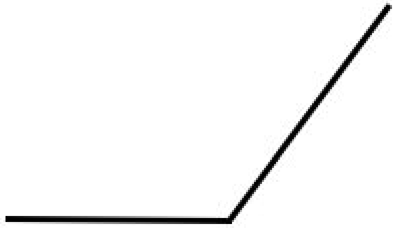


Slide from "Large-Scale Deep Learning with TensorFlow for Building Intelligent Systems," by Jeff Dean, ACM Webinar, 7/7/16

The Artificial Neuron

$$y = F\left(\sum_i w_i x_i\right)$$



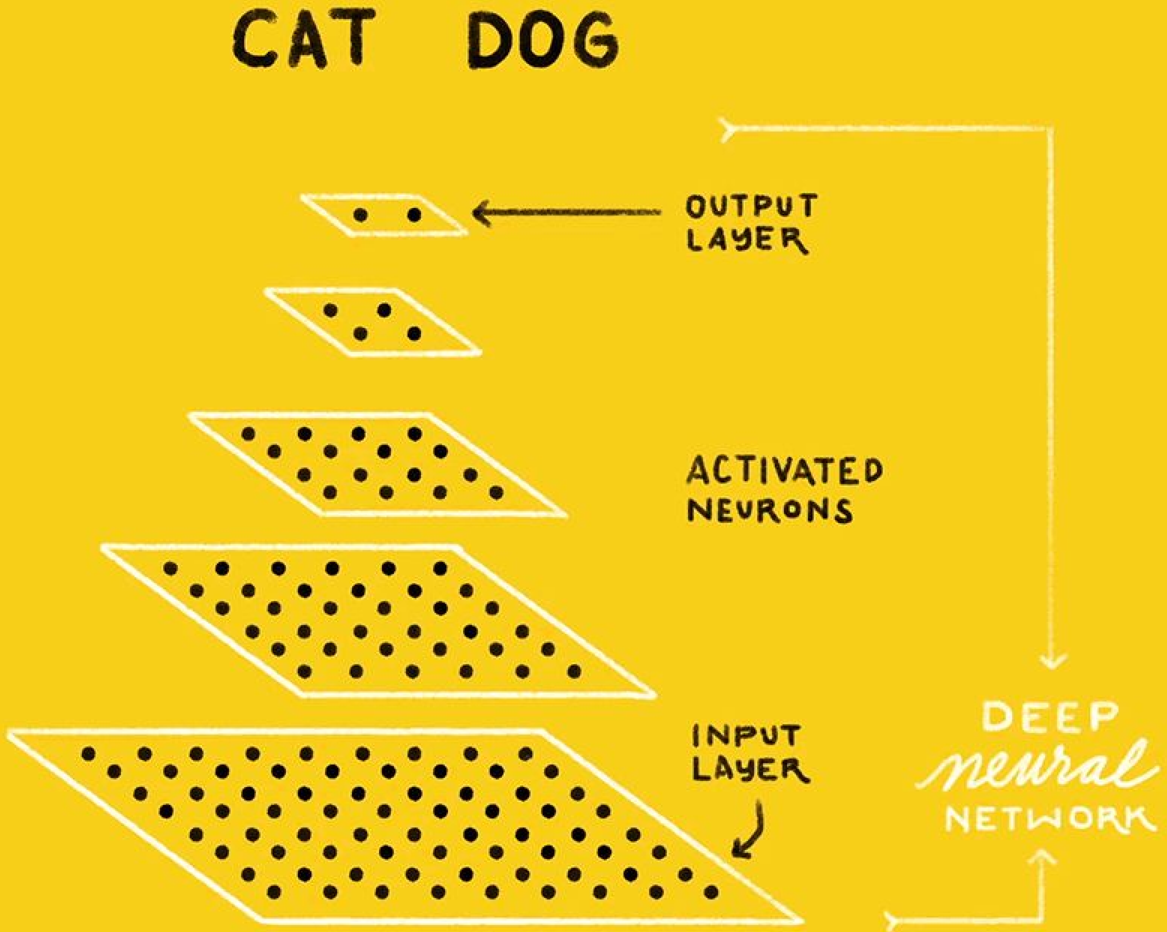


$F(x) = \max(0, x)$

F: a nonlinear differentiable function

Slide from "Large-Scale Deep Learning with TensorFlow for Building Intelligent Systems," by Jeff Dean, ACM Webinar, 7/7/16

IS THIS A CAT or DOG?



Slide from "Large-Scale Deep Learning with TensorFlow for Building Intelligent Systems," by Jeff Dean, ACM Webinar, 7/7/16

Key NN Concepts for Architects

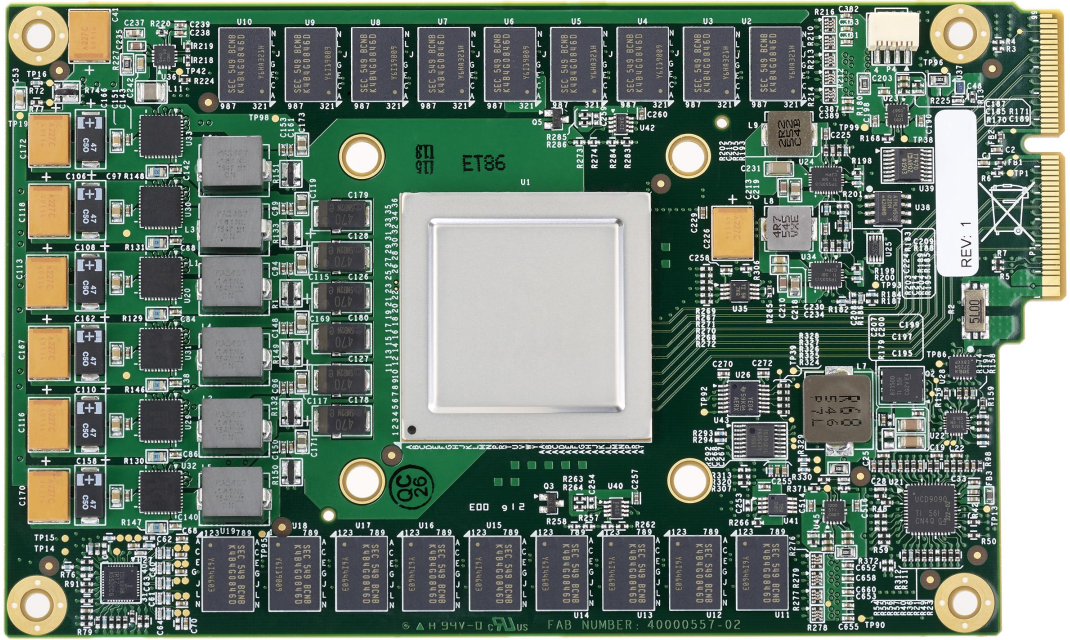
- *Training* or learning (development)
  vs. *Inference* or prediction (production)
- *Batch size*
  - Problem: DNNs have millions of weights that take a long time to load from memory (DRAM)
  - Solution: Large batch ⇒ Amortize weight-fetch time by inferring (or training) many input examples at a time
- Floating-Point vs. Integer (“*Quantization*”)
  - Training in Floating Point on GPUs popularized DNNs
  - Inferring in Integers faster, lower energy, smaller

TPU Origin Story

- 2013: Prepare for success-disaster of new DNN apps
  - Scenario with users speaking to phones 3 minutes per day:
    If only CPUs, need 2X-3X times whole fleet
  - Unlike some hardware targets, DNNs applicable to a wide range of problems, so can reuse for solutions in speech, vision, language, translation, search ranking, ...
- Custom hardware to reduce the TCO of DNN inference phase by 10X vs. GPUs
  - Must run existing apps developed for CPUs and GPUs
- A very short development cycle
  - Started project 2014, running in datacenter 15 months later:
    Architecture invention, compiler invention, hardware design, build, test, deploy
- Google CEO Sundar Pichai reveals Tensor Processing Unit at Google I/O on May 18, 2016 as “10X performance/Watt”
  cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html

TPU Card & Package

- TPU Card to replace a disk
- Up to 4 cards / server



3 Types of NNs

1. Multilayer Perceptrons
   - Each new layer applies nonlinear function F to weighted sum of all outputs from prior layer (“fully connected”) $x_n = F(Wx_{n-1})$
2. Convolutional Neural Network
   - Like MLPs, but same weights used on nearby subsets of outputs from prior layer
3. Recurrent NN/“Long Short-Term Memory”
   - Each new layer a NL function of weighted sums of past *state* and prior outputs; same weights used across time steps

Inference Datacenter Workload (95%)

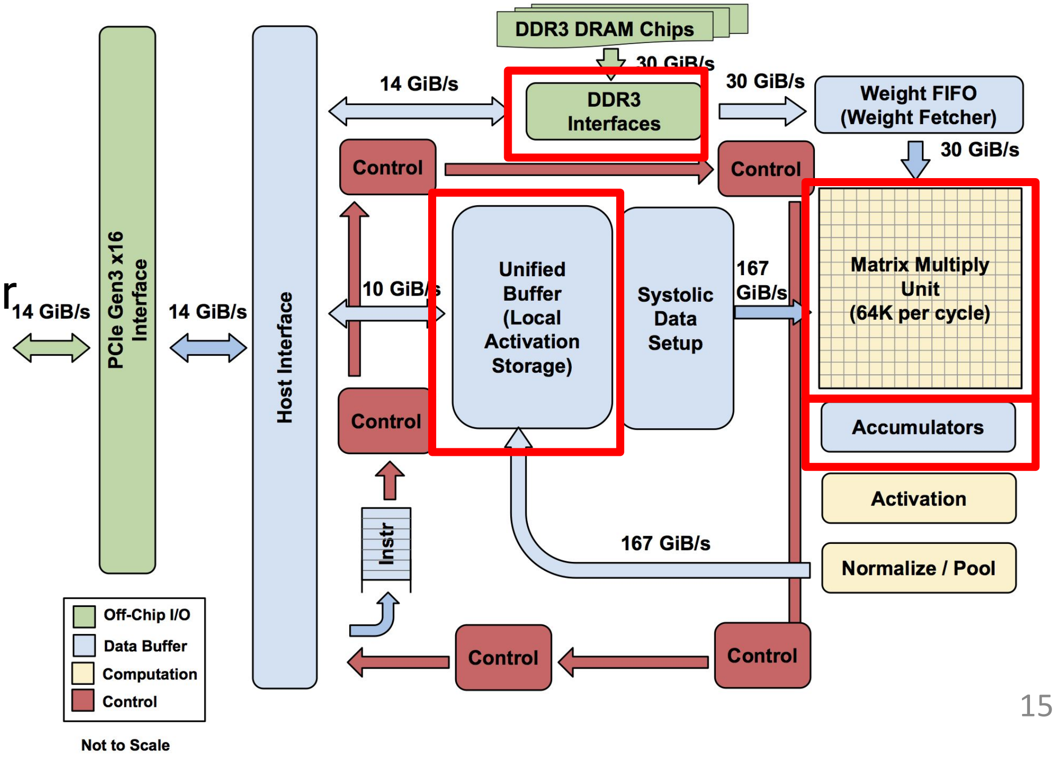
Name	*LOC*	*Layers*					*Nonlinear function*	*Weights*	*TPU Ops / Weight Byte*	*TPU Batch Size*	*% Deployed*
		FC	*Conv*	*Vector*	*Pool*	*Total*					
MLP0	0.1k	5				5	ReLU	20M	200	200	61%
MLP1	1k	4				4	ReLU	5M	168	168	
LSTM0	1k	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1.5k	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1k		16			16	ReLU	8M	2888	8	5%
CNN1	1k	4	72		13	89	ReLU	100M	1750	32	

TPU Architecture and Implementation

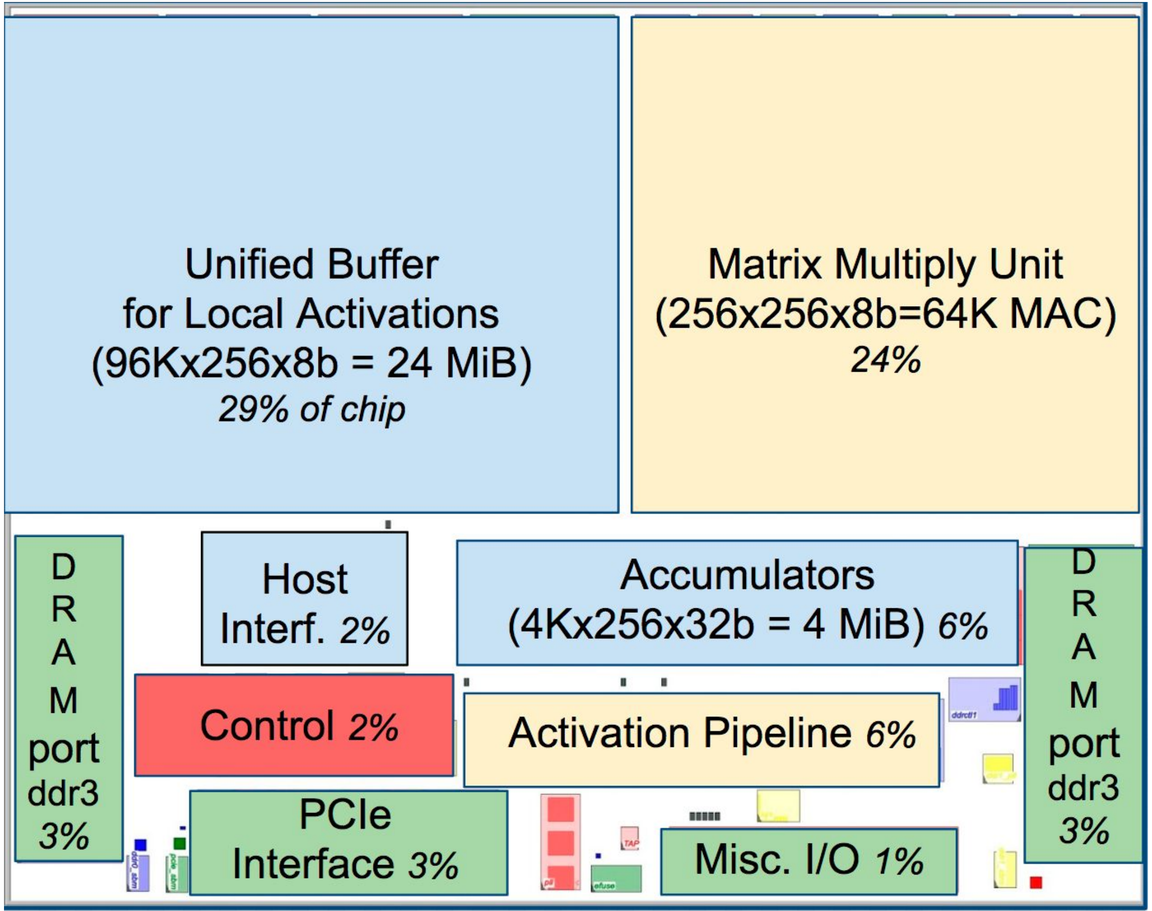
- Add as accelerators to existing servers
  - So connect over I/O bus (“PCIe”)
  - TPU ≈ matrix accelerator on I/O bus
- Host server sends it instructions like a Floating Point Unit
  - Unlike GPU that fetches and executes own instructions

TPU: High-level Chip Architecture

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory



TPU: a Neural Network Accelerator Chip



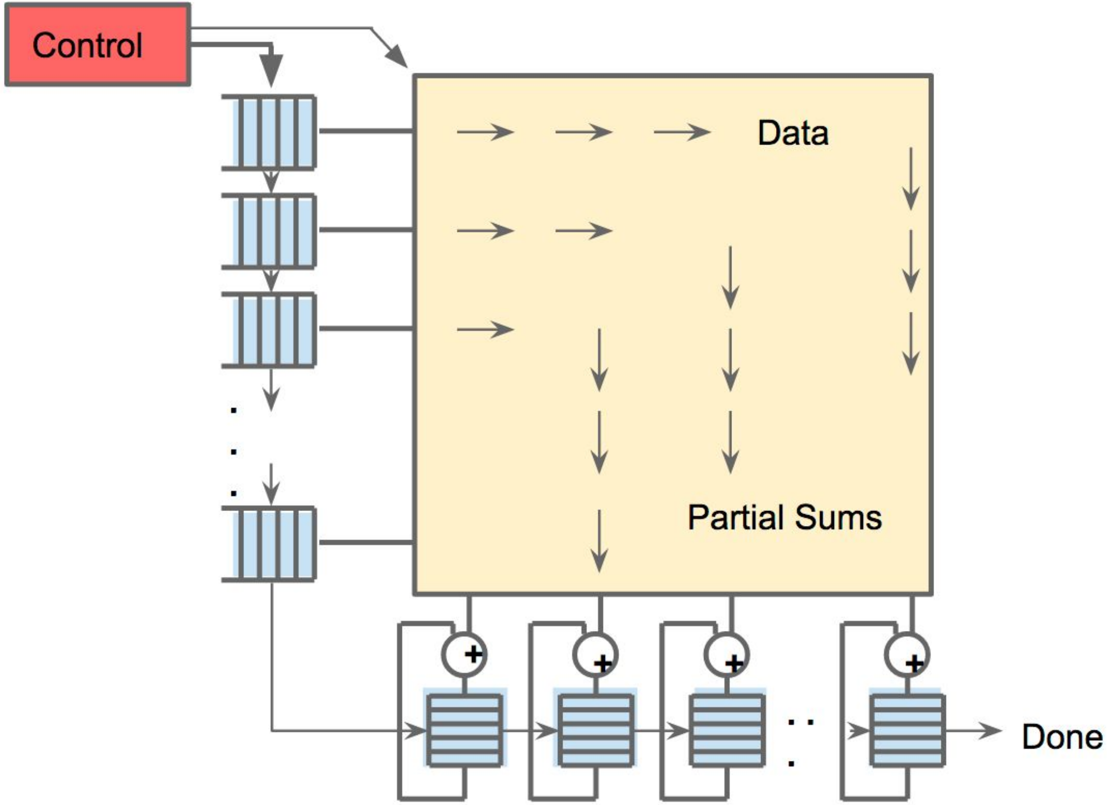
TPU Architecture, programmer's view

- 5 main (CISC) instructions
  `Read_Host_Memory`
  `Write_Host_Memory`
  `Read_Weights`
  `MatrixMultiply/Convolve`
  `Activate(ReLU,Sigmoid,Maxpool,LRN,…)`
- Average Clock cycles per instruction: >10
- 4-stage overlapped execution, 1 instruction type / stage
  - Execute other instructions while matrix multiplier busy
- Complexity in SW: No branches, in-order issue,
  SW controlled buffers, SW controlled pipeline synchronization

Systolic Execution in Matrix Array

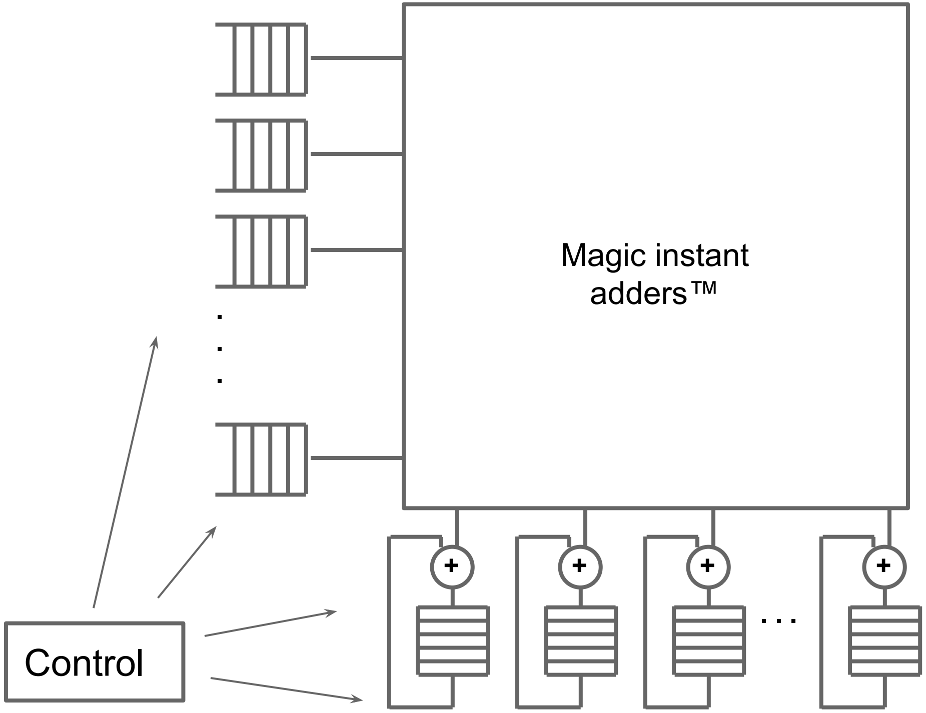
- Problem: energy/ time for repeated SRAM accesses of matrix multiply
- Solution: “Systolic Execution” to compute data on the fly in buffers by pipelining control and data
  - Relies on data from different directions arriving at cells in an array at regular intervals and being combined

Systolic Execution: Control and Data are pipelined



Can now ignore pipelining in matrix

Pretend each 256B input read at once, & they instantly update 1 location of each of 256 accumulator RAMs.



Relative Performance: 3 Contemporary Chips

Processor	*mm^2*	*Clock MHz*	*TDP Watts*	*Idle Watts*	*Memory GB/sec*	*Peak TOPS/chip*	
						8b int.	*32b FP*
CPU: Haswell (18 core)	662	2300	145	41	51	2.6	1.3
GPU: Nvidia K80 (2 / card)	561	560	150	25	160	--	2.8
TPU	<331*	700	75	28	34	91.8	--

*TPU is less than half die size of the Intel Haswell processor

K80 and TPU in 28 nm process; Haswell fabbed in Intel 22 nm process

These chips and platforms chosen for comparison because widely deployed in Google data centers

GPUs and TPUs added to CPU server

Relative Performance: 3 Platforms

Processor	*Chips/ Server*	*DRAM*	*TDP Watts*	Idle Watts	Observed Busy Watts in datacenter
CPU: Haswell (18 cores)	2	256 GB	504	159	455
NVIDIA K80 (13 cores) (2 die per card; 4 cards per server)	8	256 GB (host) + 12GB x 8	1838	357	991
TPU (1 core) (1 die per card; 4 cards per server)	4	256GB (host) + 8GB x 4	861	290	384

These chips and platforms chosen for comparison because widely deployed in Google datacenters

Roofline Visual Performance Model

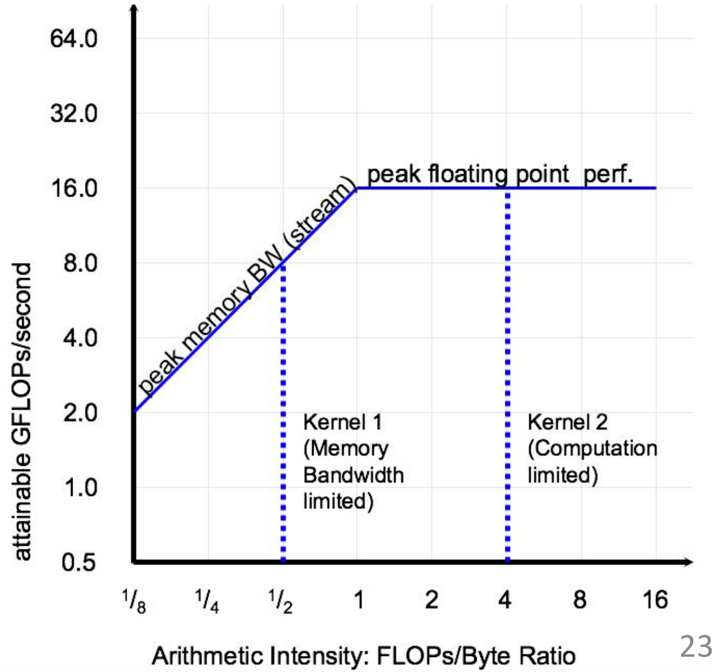
2 Limits to performance:

1. Peak Computation
2. Peak Memory Bandwidth (For apps with large data that don’t fit in cache)

Arithmetic Intensity (FLOP/byte or reuse) determines which limit

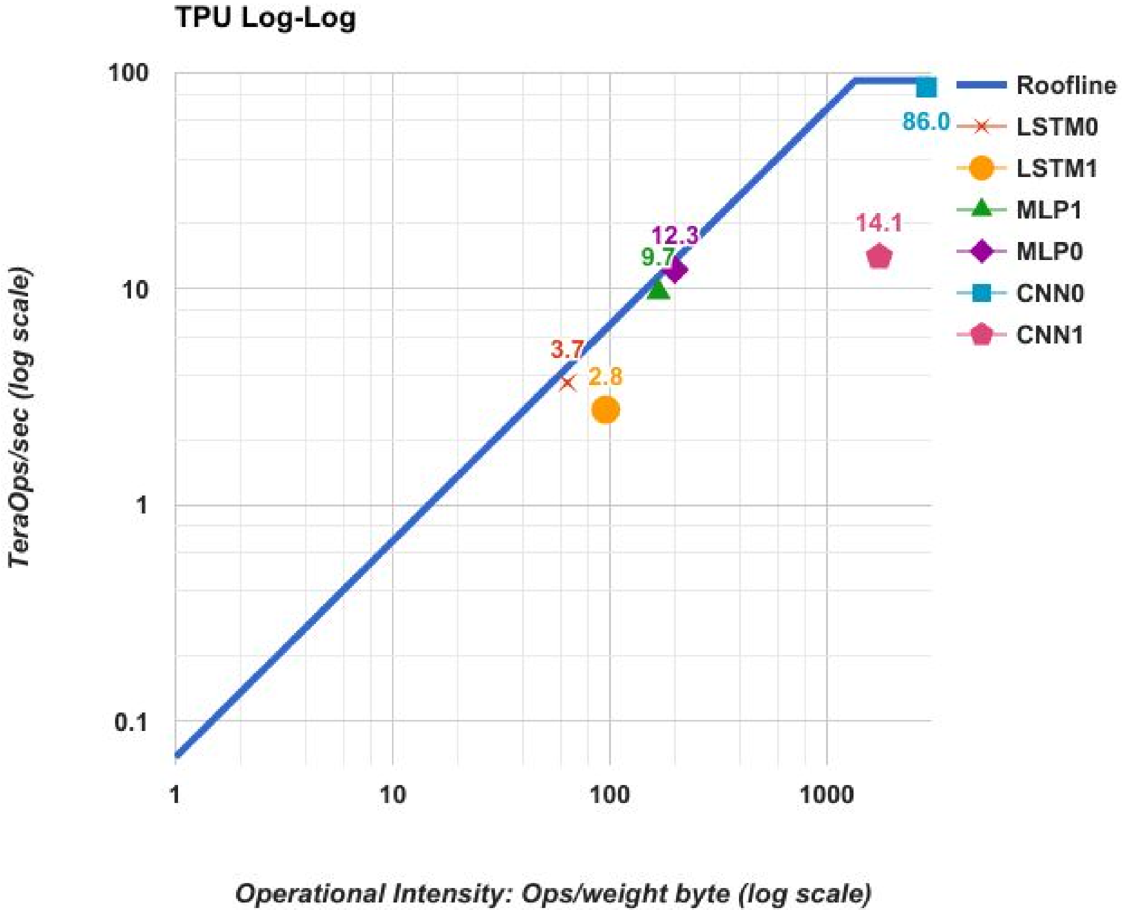
Weight-reuse = Arithmetic Intensity for DNN roofline

GFLOP/s = Min(Peak GFLOP/s, Peak GB/s x AI)

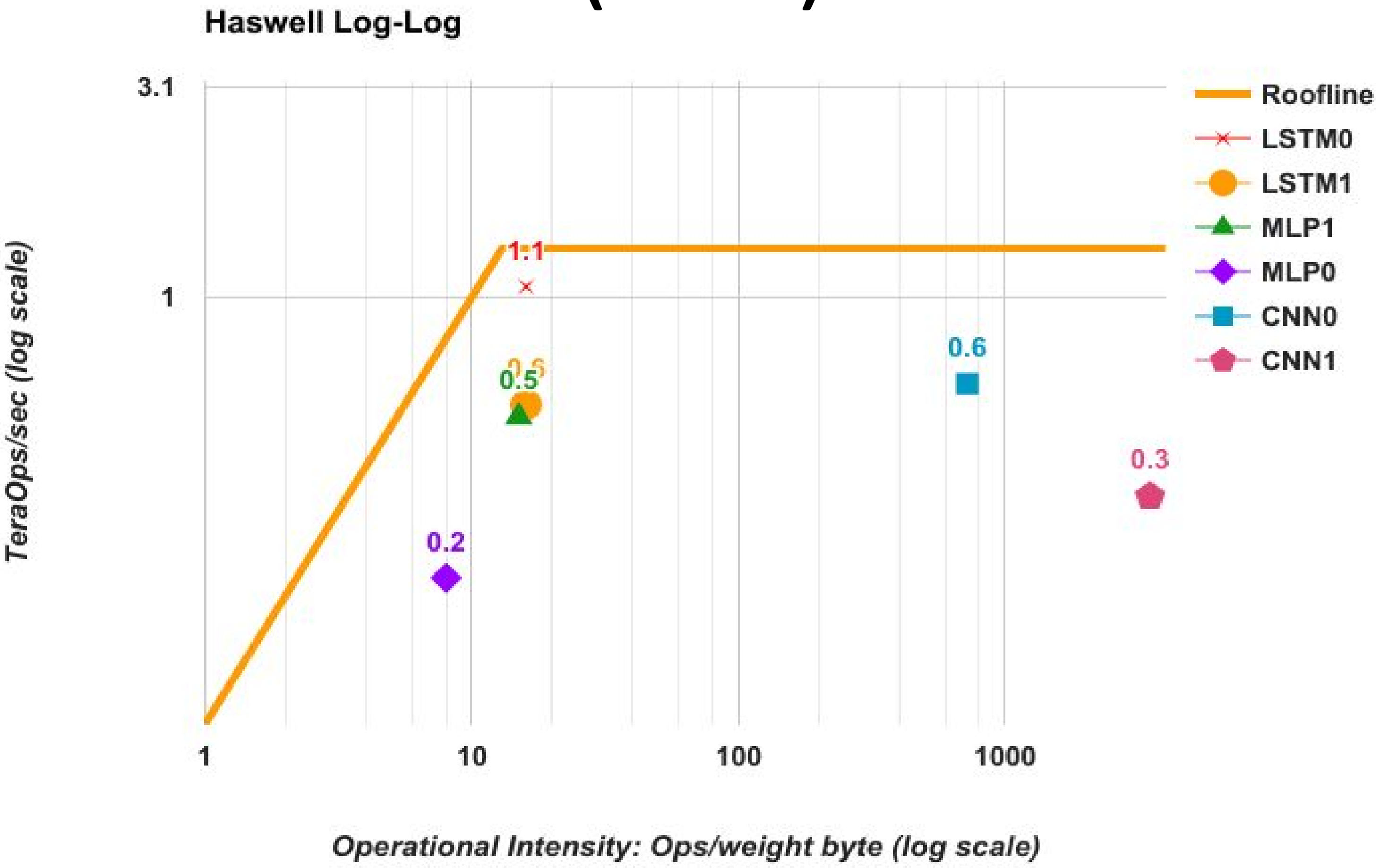


Samuel Williams, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures."*Communications of the ACM* 52.4 (2009): 65-76.

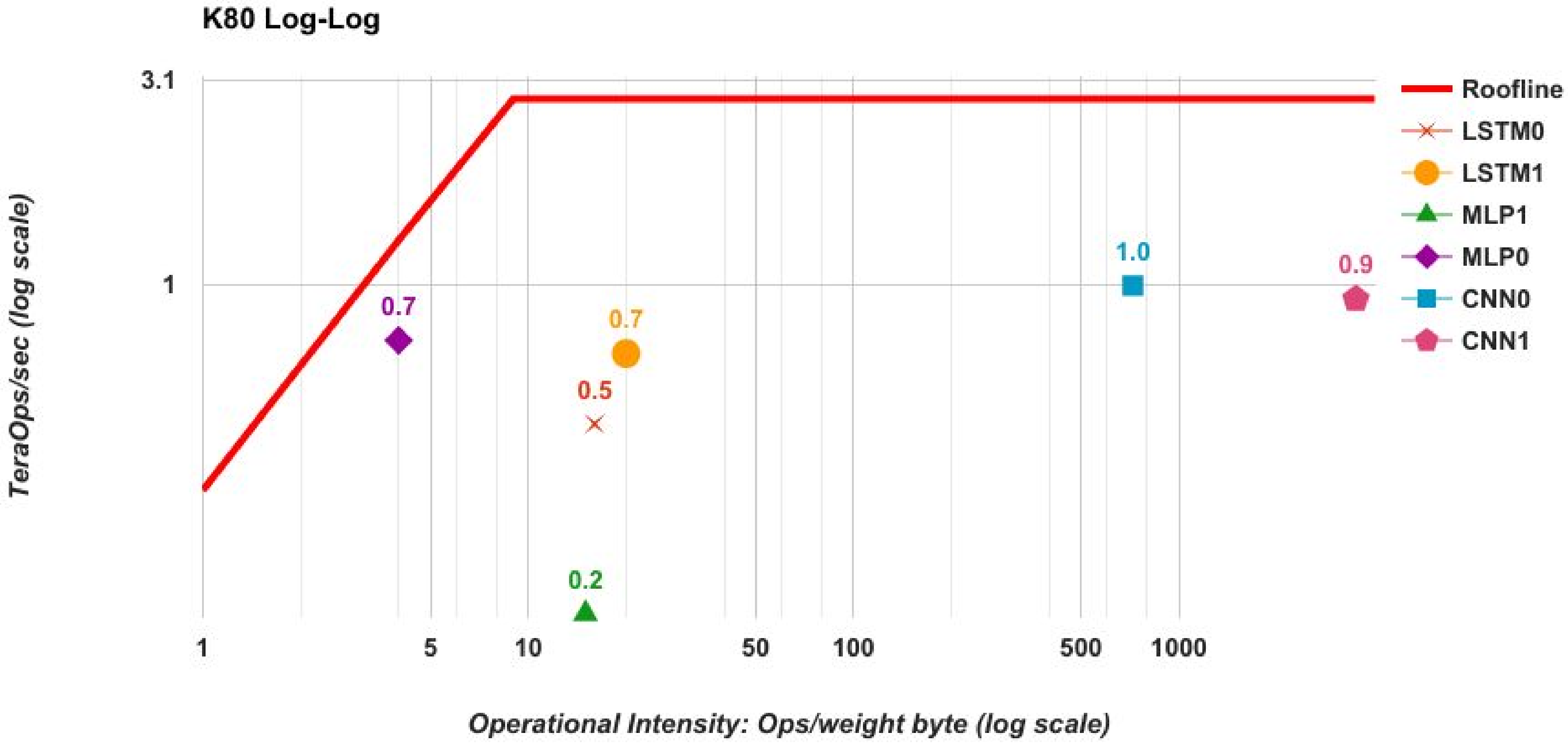
TPU Die Roofline



Haswell (CPU) Die Roofline



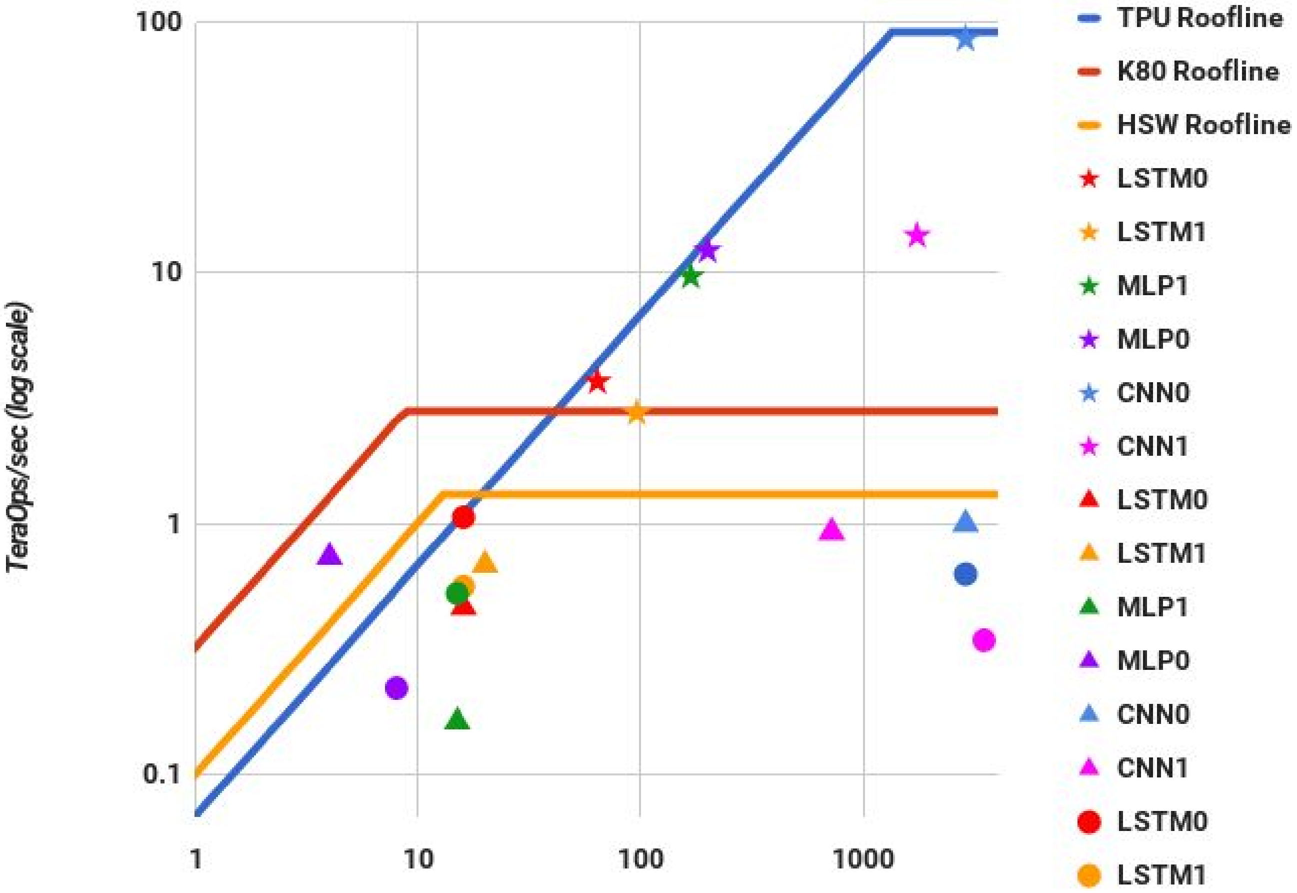
K80 (GPU) Die Roofline



Why so far below Rooflines? (MLP0)

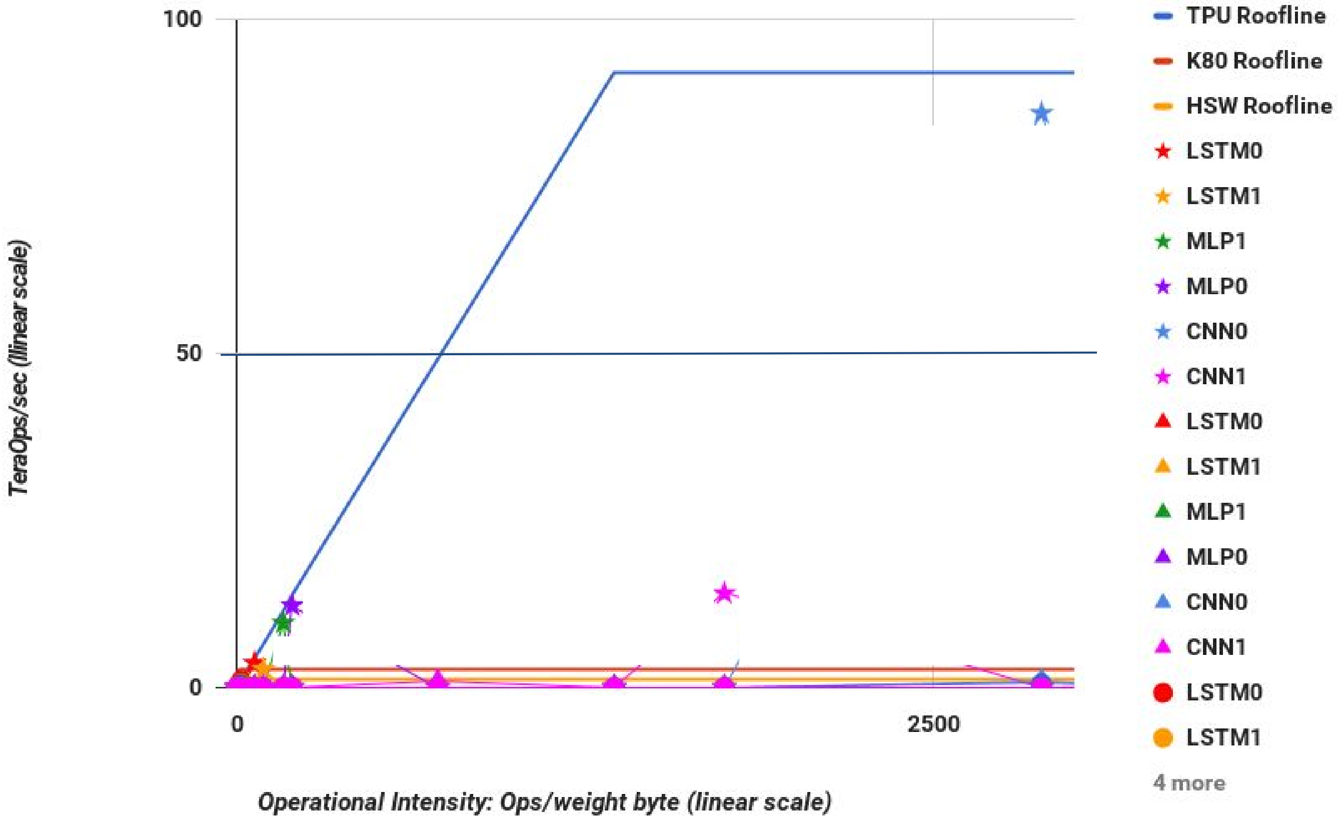
Type	*Batch*	*99th% Response*	*Inf/s (IPS)*	*% Max IPS*
CPU	16	7.2 ms	5,482	42%
CPU	64	21.3 ms	13,194	100%
GPU	16	6.7 ms	13,461	37%
GPU	64	8.3 ms	36,465	100%
TPU	200	7.0 ms	225,000	80%
TPU	250	10.0 ms	280,000	100%

Log Rooflines for CPU, GPU, TPU



☆ Star = TPU
△ Triangle = GPU
○ Circle = CPU

Linear Rooflines for CPU, GPU, TPU

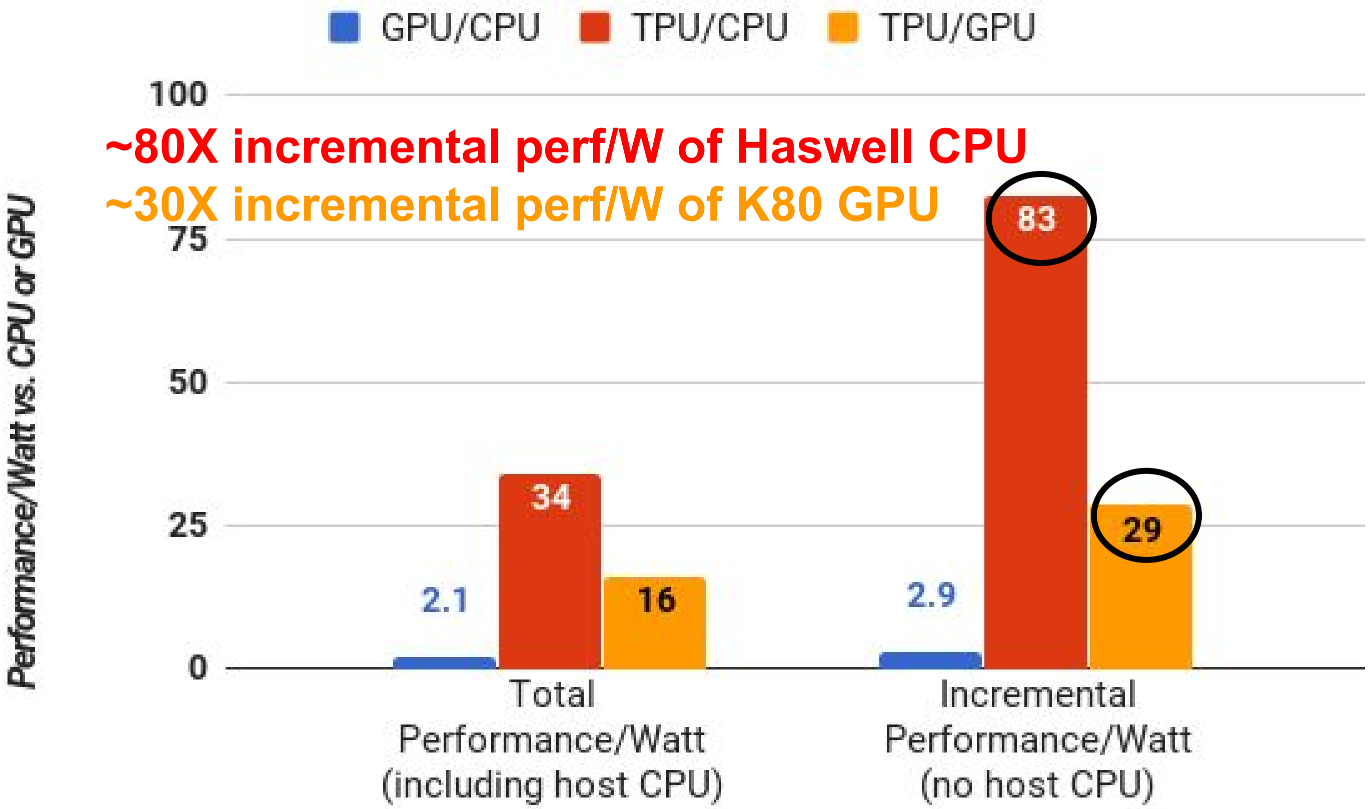


☆Star = TPU
△Triangle = GPU
○Circle = CPU

TPU & GPU Relative Performance to CPU

Type	*MLP*		*LSTM*		*CNN*		*Weighted Mean*
	0	*1*	*0*	*1*	*0*	*1*	
GPU	2.5	0.3	0.4	1.2	1.6	2.7	1.9
TPU	41.0	18.5	3.5	1.2	40.3	71.0	29.2
Ratio	16.7	60.0	8.0	1.0	25.4	26.3	15.3

Perf/Watt TPU vs CPU & GPU



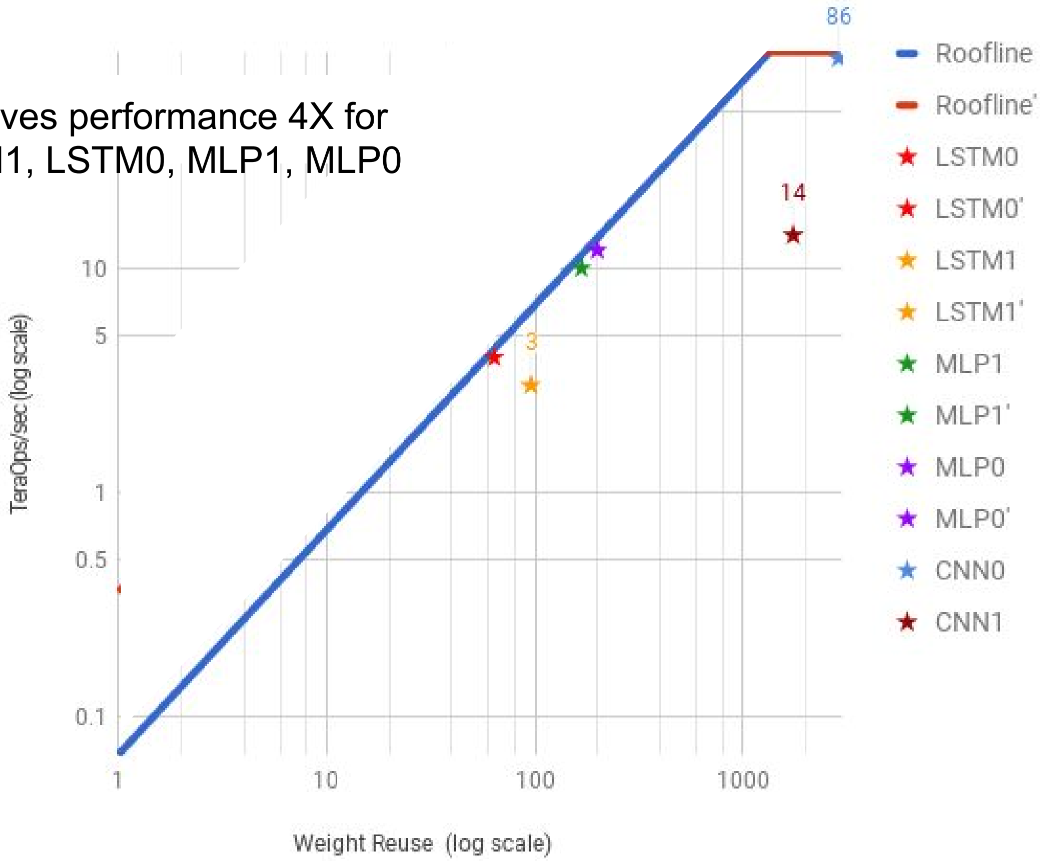
~80X incremental perf/W of Haswell CPU

~30X incremental perf/W of K80 GPU

Improving TPU: Move "Ridge Point" to the left

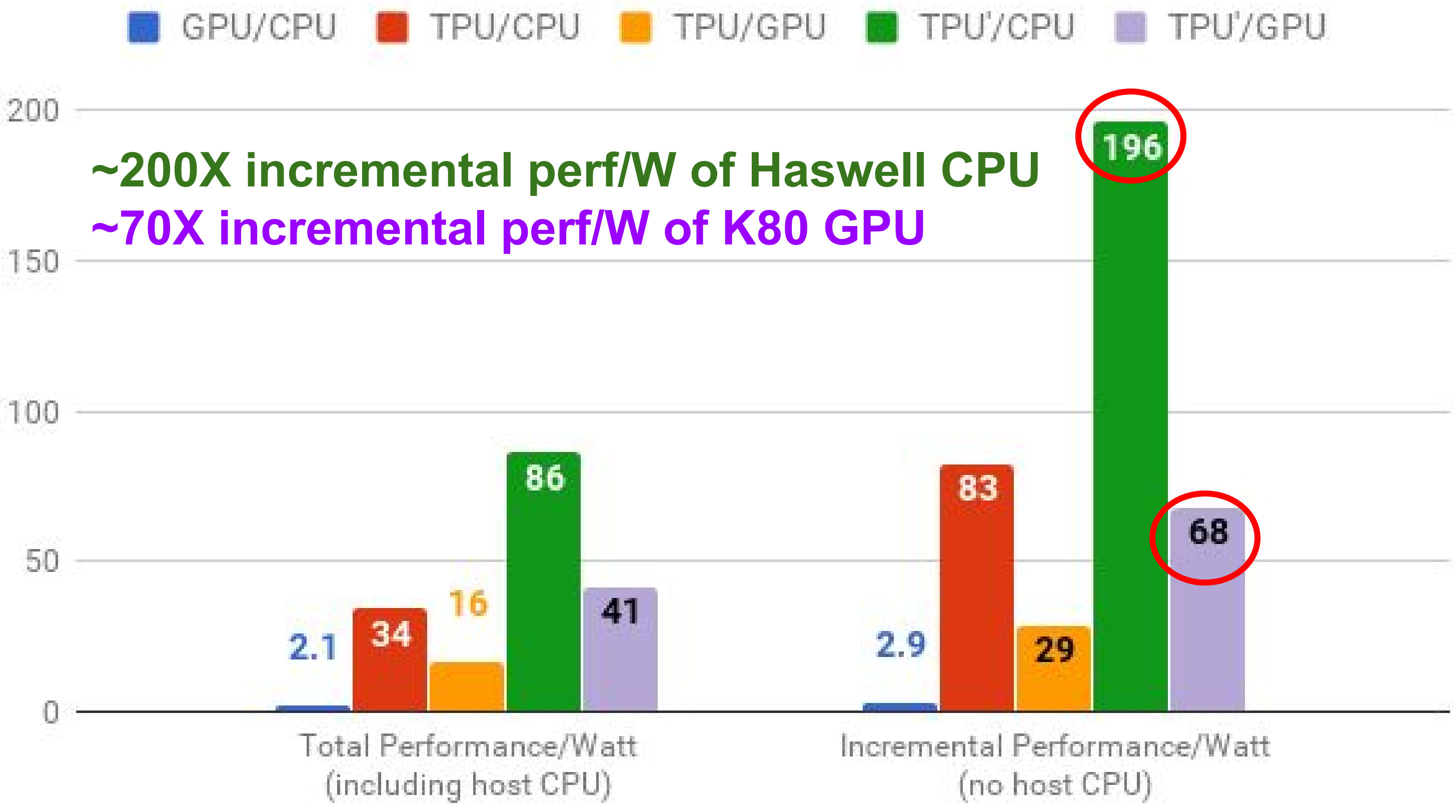
- Current DRAM
  - 2 DDR3 2133 ⇒ 34 GB/s
- Replace with GDDR5 like in K80 ⇒ 180 GB/s
  - Move Ridge Point from 1400 to 256

Revised TPU Raises Roofline



Improves performance 4X for LSTM1, LSTM0, MLP1, MLP0

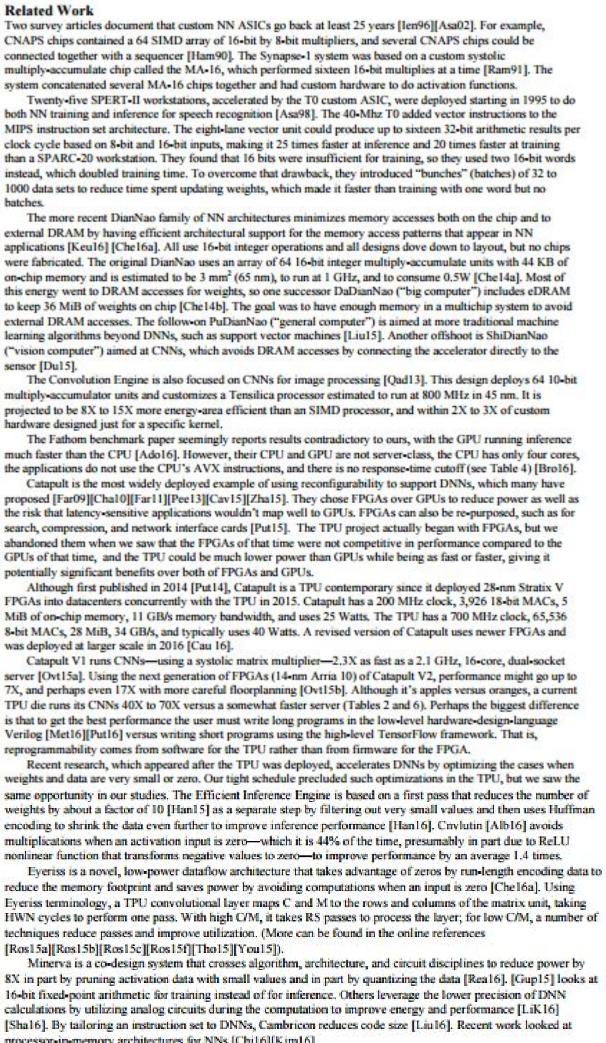
Perf/Watt Original & Revised TPU



~200X incremental perf/W of Haswell CPU

~70X incremental perf/W of K80 GPU

Related Work



Related Work

Two survey articles document that custom NN ASICs go back at least 25 years [Ien96][Asa02]. For example, CNAPS chips contained a 64 SIMD array of 16-bit by 8-bit multipliers, and several CNAPS chips could be connected together with a sequencer [Ham90]. The Synapse-1 system was based on a custom systolic multiply-accumulate chip called the MA-16, which performed sixteen 16-bit multiplies at a time [Ram91]. The system concatenated several MA-16 chips together and had custom hardware to do activation functions.

Twenty-five SPERT-II workstations, accelerated by the T0 custom ASIC, were deployed starting in 1995 to do both NN training and inference for speech recognition [Asa98]. The 40-Mhz T0 added vector instructions to the MIPS instruction set architecture. The eight-lane vector unit could produce up to sixteen 32-bit arithmetic results per clock cycle based on 8-bit and 16-bit inputs, making it 25 times faster at inference and 20 times faster at training than a SPARC-20 workstation. They found that 16 bits were insufficient for training, so they used two 16-bit words instead, which doubled training time. To overcome that drawback, they introduced "bunches" (batches) of 32 to 1000 data sets to reduce time spent updating weights, which made it faster than training with one word but no batches.

The more recent DianNao family of NN architectures minimizes memory accesses both on the chip and to external DRAM by having efficient architectural support for the memory access patterns that appear in NN applications [Keu16] [Che16a]. All use 16-bit integer operations and all designs dove down to layout, but no chips were fabricated. The original DianNao uses an array of 64 16-bit integer multiply-accumulate units with 44 KB of on-chip memory and is estimated to be 3 mm^2 (65 nm), to run at 1 GHz, and to consume 0.5W [Che14a]. Most of this energy went to DRAM accesses for weights, so one successor DaDianNao ("big computer") includes eDRAM to keep 36 MiB of weights on chip [Che14b]. The goal was to have enough memory in a multichip system to avoid external DRAM accesses. The follow-on PuDianNao ("general computer") is aimed at more traditional machine learning algorithms beyond DNNs, such as support vector machines [Liu15]. Another offshoot is ShiDianNao ("vision computer") aimed at CNNs, which avoids DRAM accesses by connecting the accelerator directly to the sensor [Du15].

The Convolution Engine is also focused on CNNs for image processing [Qad13]. This design deploys 64 10-bit multiply-accumulator units and customizes a Tensilica processor estimated to run at 800 MHz in 45 nm. It is projected to be 8X to 15X more energy-area efficient than an SIMD processor, and within 2X to 3X of custom hardware designed just for a specific kernel.

The Fathom benchmark paper seemingly reports results contradictory to ours, with the GPU running inference much faster than the CPU [Ado16]. However, their CPU and GPU are not server-class, the CPU has only four cores, the applications do not use the CPU's AVX instructions, and there is no response-time cutoff (see Table 4) [Bro16].

Catapult is the most widely deployed example of using reconfigurability to support DNNs, which many have proposed [Far09][Cha10][Far11][Pee13][Cav15][Zha15]. They chose FPGAs over GPUs to reduce power as well as the risk that latency-sensitive applications wouldn't map well to GPUs. FPGAs can also be re-purposed, such as for search, compression, and network interface cards [Put15]. The TPU project actually began with FPGAs, but we abandoned them when we saw that the FPGAs of that time were not competitive in performance compared to the GPUs of that time, and the TPU could be much lower power than GPUs while being as fast or faster, giving it potentially significant benefits over both of FPGAs and GPUs.

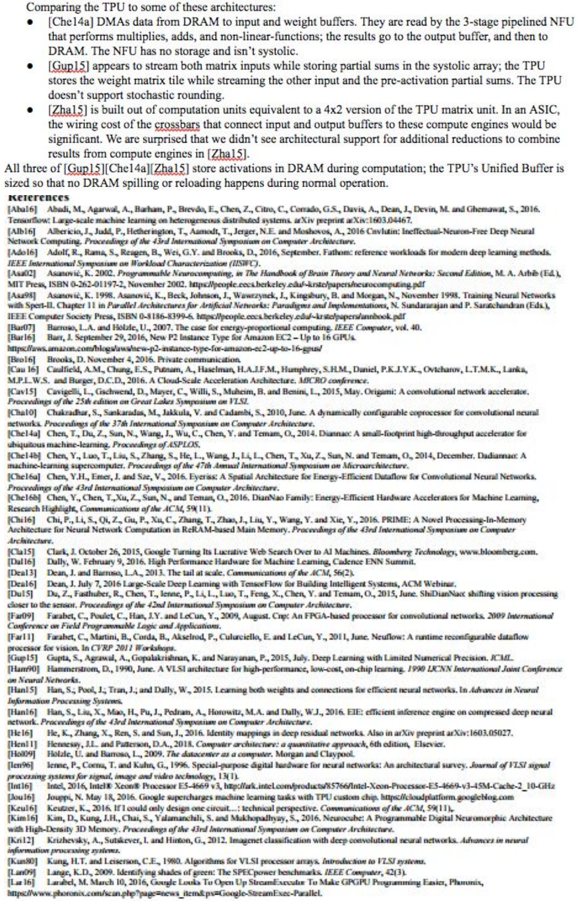
Although first published in 2014 [Put14], Catapult is a TPU contemporary since it deployed 28-nm Stratix V FPGAs into datacenters concurrently with the TPU in 2015. Catapult has a 200 MHz clock, 3,926 18-bit MACs, 5 MiB of on-chip memory, 11 GB/s memory bandwidth, and uses 25 Watts. The TPU has a 700 MHz clock, 65,536 8-bit MACs, 28 MiB, 34 GB/s, and typically uses 40 Watts. A revised version of Catapult uses newer FPGAs and was deployed at larger scale in 2016 [Cau 16].

Catapult V1 runs CNNs—using a systolic matrix multiplier—2.3X as fast as a 2.1 GHz, 16-core, dual-socket server [Ovt15a]. Using the next generation of FPGAs (14-nm Arria 10) of Catapult V2, performance might go up to 7X, and perhaps even 17X with more careful floorplanning [Ovt15b]. Although it's apples versus oranges, a current TPU die runs its CNNs 40X to 70X versus a somewhat faster server (Tables 2 and 6). Perhaps the biggest difference is that to get the best performance the user must write long programs in the low-level hardware-design-language Verilog [Met16][Put16] versus writing short programs using the high-level TensorFlow framework. That is, reprogrammability comes from software for the TPU rather than from firmware for the FPGA.

Recent research, which appeared after the TPU was deployed, accelerates DNNs by optimizing the cases when weights and data are very small or zero. Our tight schedule precluded such optimizations in the TPU, but we saw the same opportunity in our studies. The Efficient Inference Engine is based on a first pass that reduces the number of weights by about a factor of 10 [Han15] as a separate step by filtering out very small values and then uses Huffman encoding to shrink the data even further to improve inference performance [Han16]. Cnvlutin [Alb16] avoids multiplications when an activation input is zero—which it is 44% of the time, presumably in part due to ReLU nonlinear function that transforms negative values to zero—to improve performance by an average 1.4 times.

Eyeriss is a novel, low-power dataflow architecture that takes advantage of zeros by run-length encoding data to reduce the memory footprint and saves power by avoiding computations when an input is zero [Che16a]. Using Eyeriss terminology, a TPU convolutional layer maps C and M to the rows and columns of the matrix unit, taking HWN cycles to perform one pass. With high C/M, it takes RS passes to process the layer; for low C/M, a number of techniques reduce passes and improve utilization. (More can be found in the online references [Ros15a][Ros15b][Ros15c][Ros15f][Tho15][You15]).

Minerva is a co-design system that crosses algorithm, architecture, and circuit disciplines to reduce power by 8X in part by pruning activation data with small values and in part by quantizing the data [Rea16]. [Gup15] looks at 16-bit fixed-point arithmetic for training instead of for inference. Others leverage the lower precision of DNN calculations by utilizing analog circuits during the computation to improve energy and performance [LiK16] [Sha16]. By tailoring an instruction set to DNNs, Cambricon reduces code size [Liu16]. Recent work looked at processor-in-memory architectures for NNs [Chi16][Kim16].



Comparing the TPU to some of these architectures:

- [Che14a] DMAs data from DRAM to input and weight buffers. They are read by the 3-stage pipelined NFU that performs multiplies, adds, and non-linear-functions; the results go to the output buffer, and then to DRAM. The NFU has no storage and isn't systolic.
- [Gup15] appears to stream both matrix inputs while storing partial sums in the systolic array; the TPU stores the weight matrix tile while streaming the other input and the pre-activation partial sums. The TPU doesn't support stochastic rounding.
- [Zha15] is built out of computation units equivalent to a 4x2 version of the TPU matrix unit. In an ASIC, the wiring cost of the crossbars that connect input and output buffers to these compute engines would be significant. We are surprised that we didn't see architectural support for additional reductions to combine results from compute engines in [Zha15].

All three of [Gup15][Che14a][Zha15] store activations in DRAM during computation; the TPU's Unified Buffer is sized so that no DRAM spilling or reloading happens during normal operation.

References

[Aba16] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[Alb16] Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N.E. and Moshovos, A., 2016 Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Ado16] Adolf, R., Rama, S., Reagen, B., Wei, G.Y. and Brooks, D., 2016, September. Fathom: reference workloads for modern deep learning methods. *IEEE International Symposium on Workload Characterization (IISWC).*

[Asa02] Asanović, K. 2002. *Programmable Neurocomputing, in The Handbook of Brain Theory and Neural Networks: Second Edition*, M. A. Arbib (Ed.), MIT Press, ISBN 0-262-01197-2, November 2002. https://people.eecs.berkeley.edu/~krste/papers/neurocomputing.pdf

[Asa98] Asanović, K. 1998. Asanović, K., Beck, Johnson, J., Wawrzynek, J., Kingsbury, B. and Morgan, N., November 1998. Training Neural Networks with Spert-II. Chapter 11 in *Parallel Architectures for Artificial Networks: Paradigms and Implementations*, N. Sundararajan and P. Saratchandran (Eds.), IEEE Computer Society Press, ISBN 0-8186-8399-6. https://people.eecs.berkeley.edu/~krste/papers/annbook.pdf

[Bar07] Barroso, L.A. and Hölzle, U., 2007. The case for energy-proportional computing. *IEEE Computer*, vol. 40.

[Bar16] Barr, J. September 29, 2016, New P2 Instance Type for Amazon EC2 – Up to 16 GPUs. https://aws.amazon.com/blogs/aws/new-p2-instance-type-for-amazon-ec2-up-to-16-gpus/

[Bro16] Brooks, D. November 4, 2016. Private communication.

[Cau 16] Caulfield, A.M., Chung, E.S., Putnam, A., Haselman, H.A.J.F.M., Humphrey, S.H.M., Daniel, P.K.J.Y.K., Ovtcharov, L.T.M.K., Lanka, M.P.L.W.S. and Burger, D.C.D., 2016. A Cloud-Scale Acceleration Architecture. *MICRO conference.*

[Cav15] Cavigelli, L., Gschwend, D., Mayer, C., Willi, S., Muheim, B. and Benini, L., 2015, May. Origami: A convolutional network accelerator. *Proceedings of the 25th edition on Great Lakes Symposium on VLSI.*

[Cha10] Chakradhar, S., Sankaradas, M., Jakkula, V. and Cadambi, S., 2010, June. A dynamically configurable coprocessor for convolutional neural networks. *Proceedings of the 37th International Symposium on Computer Architecture.*

[Che14a] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y. and Temam, O., 2014. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *Proceedings of ASPLOS.*

[Che14b] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O., 2014, December. Dadiannao: A machine-learning supercomputer. *Proceedings of the 47th Annual International Symposium on Microarchitecture.*

[Che16a] Chen, Y.H., Emer, J. and Sze, V., 2016. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Che16b] Chen, Y., Chen, T.,Xu, Z., Sun, N., and Teman, O., 2016. DianNao Family: Energy-Efficient Hardware Accelerators for Machine Learning, Research Highlight, *Communications of the ACM*, 59(11).

[Chi16] Chi, P., Li, S., Qi, Z., Gu, P., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y. and Xie, Y., 2016. PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation in ReRAM-based Main Memory. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Cla15] Clark, J. October 26, 2015, Google Turning Its Lucrative Web Search Over to AI Machines. *Bloomberg Technology*, www.bloomberg.com.

[Dal16] Dally, W. February 9, 2016. High Performance Hardware for Machine Learning, Cadence ENN Summit.

[Dea13] Dean, J. and Barroso, L.A., 2013. The tail at scale. *Communications of the ACM*, 56(2).

[Dea16] Dean, J. July 7, 2016 Large-Scale Deep Learning with TensorFlow for Building Intelligent Systems, ACM Webinar.

[Du15] Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y. and Temam, O., 2015, June. ShiDianNao: shifting vision processing closer to the sensor. *Proceedings of the 42nd International Symposium on Computer Architecture.*

[Far09] Farabet, C., Poulet, C., Han, J.Y. and LeCun, Y., 2009, August. Cnp: An FPGA-based processor for convolutional networks. *2009 International Conference on Field Programmable Logic and Applications.*

[Far11] Farabet, C., Martini, B., Corda, B., Akselrod, P., Culurciello, E. and LeCun, Y., 2011, June. Neuflow: A runtime reconfigurable dataflow processor for vision. In *CVRP 2011 Workshops.*

[Gup15] Gupta, S., Agrawal, A., Gopalakrishnan, K. and Narayanan, P., 2015, July. Deep Learning with Limited Numerical Precision. *ICML.*

[Ham90] Hammerstrom, D., 1990, June. A VLSI architecture for high-performance, low-cost, on-chip learning. *1990 IJCNN International Joint Conference on Neural Networks.*

[Han15] Han, S.; Pool, J.; Tran, J.; and Dally, W., 2015. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems.*

[Han16] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M.A. and Dally, W.J., 2016. EIE: efficient inference engine on compressed deep neural network. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[He16] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Identity mappings in deep residual networks. Also in arXiv preprint arXiv:1603.05027.

[Hen11] Hennessy, J.L. and Patterson, D.A., 2018. *Computer architecture: a quantitative approach*, 6th edition, Elsevier.

[Hol09] Hölzle, U. and Barroso, L., 2009. *The datacenter as a computer.* Morgan and Claypool.

[Ien96] Ienne, P., Cornu, T. and Kuhn, G., 1996. Special-purpose digital hardware for neural networks: An architectural survey. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(1).

[Int16] Intel, 2016, Intel® Xeon® Processor E5-4669 v3, http://ark.intel.com/products/85766/Intel-Xeon-Processor-E5-4669-v3-45M-Cache-2_10-GHz

[Jou16] Jouppi, N. May 18, 2016. Google supercharges machine learning tasks with TPU custom chip. https://cloudplatform.googleblog.com

[Keu16] Keutzer, K., 2016. If I could only design one circuit...: technical perspective. *Communications of the ACM*, 59(11).

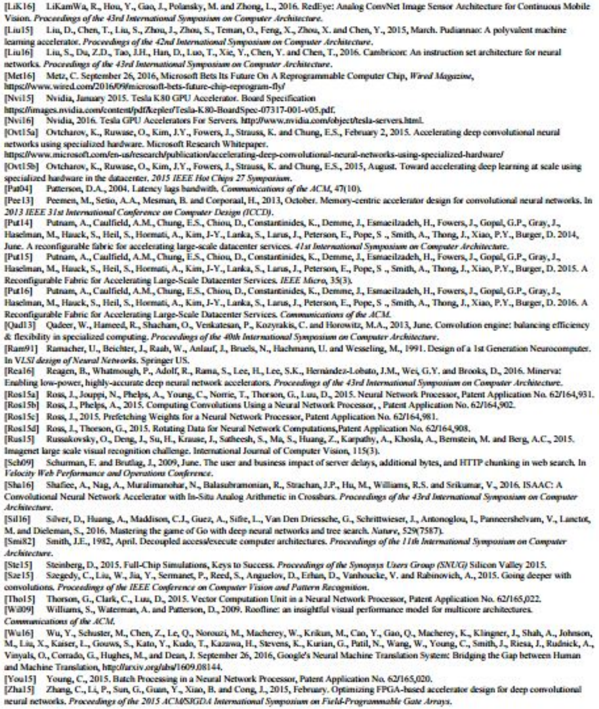
[Kim16] Kim, D., Kung, J.H., Chai, S., Yalamanchili, S. and Mukhopadhyay, S., 2016. Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Kri12] Krizhevsky, A., Sutskever, I. and Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems.*

[Kun80] Kung, H.T. and Leiserson, C.E., 1980. Algorithms for VLSI processor arrays. *Introduction to VLSI systems.*

[Lan09] Lange, K.D., 2009. Identifying shades of green: The SPECpower benchmarks. *IEEE Computer*, 42(3).

[Lar16] Larabel, M. March 10, 2016, Google Looks To Open Up StreamExecutor To Make GPGPU Programming Easier, Phoronix, https://www.phoronix.com/scan.php?page=news_item&px=Google-StreamExec-Parallel.



[LiK16] LiKamWa, R., Hou, Y., Gao, J., Polansky, M. and Zhong, L., 2016. RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Liu15] Liu, D., Chen, T., Liu, S., Zhou, J., Zhou, S., Teman, O., Feng, X., Zhou, X. and Chen, Y., 2015, March. Pudiannao: A polyvalent machine learning accelerator. *Proceedings of the 42nd International Symposium on Computer Architecture.*

[Liu16] Liu, S., Du, Z.D., Tao, J.H., Han, D., Luo, T., Xie, Y., Chen, Y. and Chen, T., 2016. Cambricon: An instruction set architecture for neural networks. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Met16] Metz, C. September 26, 2016, Microsoft Bets Its Future On A Reprogrammable Computer Chip, *Wired Magazine*, https://www.wired.com/2016/09/microsoft-bets-future-chip-reprogram-fly/

[Nvi15] Nvidia, January 2015. Tesla K80 GPU Accelerator. Board Specification https://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf.

[Nvi16] Nvidia, 2016. Tesla GPU Accelerators For Servers. http://www.nvidia.com/object/tesla-servers.html.

[Ovt15a] Ovtcharov, K., Ruwase, O., Kim, J.Y., Fowers, J., Strauss, K. and Chung, E.S., February 2, 2015. Accelerating deep convolutional neural networks using specialized hardware. Microsoft Research Whitepaper. https://www.microsoft.com/en-us/research/publication/accelerating-deep-convolutional-neural-networks-using-specialized-hardware/

[Ovt15b] Ovtcharov, K., Ruwase, O., Kim, J.Y., Fowers, J., Strauss, K. and Chung, E.S., 2015, August. Toward accelerating deep learning at scale using specialized hardware in the datacenter. *2015 IEEE Hot Chips 27 Symposium.*

[Pat04] Patterson, D.A., 2004. Latency lags bandwith. *Communications of the ACM*, 47(10).

[Pee13] Peemen, M., Setio, A.A., Mesman, B. and Corporaal, H., 2013, October. Memory-centric accelerator design for convolutional neural networks. In *2013 IEEE 31st International Conference on Computer Design (ICCD).*

[Put14] Putnam, A., Caulfield, A.M., Chung, E.S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G.P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J-Y., Lanka, S., Larus, J., Peterson, E., Pope, S ., Smith, A., Thong, J., Xiao, P.Y., Burger, D. 2014, June. A reconfigurable fabric for accelerating large-scale datacenter services. *41st International Symposium on Computer Architecture.*

[Put15] Putnam, A., Caulfield, A.M., Chung, E.S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G.P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J-Y., Lanka, S., Larus, J., Peterson, E., Pope, S ., Smith, A., Thong, J., Xiao, P.Y., Burger, D. 2015. A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. *IEEE Micro*, 35(3).

[Put16] Putnam, A., Caulfield, A.M., Chung, E.S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G.P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J-Y., Lanka, S., Larus, J., Peterson, E., Pope, S ., Smith, A., Thong, J., Xiao, P.Y., Burger, D. 2016. A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. *Communications of the ACM.*

[Qad13] Qadeer, W., Hameed, R., Shacham, O., Venkatesan, P., Kozyrakis, C. and Horowitz, M.A., 2013, June. Convolution engine: balancing efficiency & flexibility in specialized computing. *Proceedings of the 40th International Symposium on Computer Architecture.*

[Ram91] Ramacher, U., Beichter, J., Raab, W., Anlauf, J., Bruels, N., Hachmann, U. and Wesseling, M., 1991. Design of a 1st Generation Neurocomputer. In *VLSI design of Neural Networks.* Springer US.

[Rea16] Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S.K., Hernández-Lobato, J.M., Wei, G.Y. and Brooks, D., 2016. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Ros15a] Ross, J., Jouppi, N., Phelps, A., Young, C., Norrie, T., Thorson, G., Luu, D., 2015. Neural Network Processor, Patent Application No. 62/164,931.

[Ros15b] Ross, J., Phelps, A., 2015. Computing Convolutions Using a Neural Network Processor, , Patent Application No. 62/164,902.

[Ros15c] Ross, J., 2015. Prefetching Weights for a Neural Network Processor, Patent Application No. 62/164,981.

[Ros15d] Ross, J., Thorson, G., 2015. Rotating Data for Neural Network Computations,Patent Application No. 62/164,908.

[Rus15] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3).

[Sch09] Schurman, E. and Brutlag, J., 2009, June. The user and business impact of server delays, additional bytes, and HTTP chunking in web search. In *Velocity Web Performance and Operations Conference.*

[Sha16] Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J.P., Hu, M., Williams, R.S. and Srikumar, V., 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. *Proceedings of the 43rd International Symposium on Computer Architecture.*

[Sil16] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587).

[Smi82] Smith, J.E., 1982, April. Decoupled access/execute computer architectures. *Proceedings of the 11th International Symposium on Computer Architecture.*

[Ste15] Steinberg, D., 2015. Full-Chip Simulations, Keys to Success. *Proceedings of the Synopsys Users Group (SNUG)* Silicon Valley 2015.

[Sze15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[Tho15] Thorson, G., Clark, C., Luu, D., 2015. Vector Computation Unit in a Neural Network Processor, Patent Application No. 62/165,022.

[Wil09] Williams, S., Waterman, A. and Patterson, D., 2009. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM.*

[Wu16] Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. September 26, 2016, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, http://arxiv.org/abs/1609.08144.

[You15] Young, C., 2015. Batch Processing in a Neural Network Processor, Patent Application No. 62/165,020.

[Zha15] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B. and Cong, J., 2015, February. Optimizing FPGA-based accelerator design for deep convolutional neural networks. *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.*

Conclusions (1/2)

TPU succeeded because of

- Large matrix multiply unit
- Substantial software-controlled on-chip memory
- Run whole inference models to reduce host CPU
- Single-threaded, deterministic execution model good match to 99th-percentile response time
- Enough flexibility to match NNs of 2017 vs. 2013
- Omission of GP features ⇒ small, low power die
- Use of 8-bit integers in the quantized apps
- Apps in TensorFlow, so easy to port at speed

Conclusions (2/2)

- Inference prefers latency over throughput
- K80 GPU relatively poor at inference (vs. training)
- Small redesign improves TPU at low cost
- 15-month design & live on I/O bus yet TPU 15X-30X faster Haswell CPU, K80 GPU (inference), <½ die size, ½ Watts
  - 65,536 (8-bit) TPU MACs cheaper, lower energy, & faster 576 (32-bit) CPU MACs, 2496 GPU (32-bit) MACs
- 10X difference in computer products are rare

Questions?

*4/5/17 Google published a blog on the TPU. A 17-page technical paper with same title will be on arXiv.org. (Paper will also appear at the *International Symposium on Computer Architecture* on June 26, 2017.)

https://cloudplatform.googleblog.com/2017/04/quantifying-the-performance-of-the-TPU-our-first-machine-learning-chip.html