

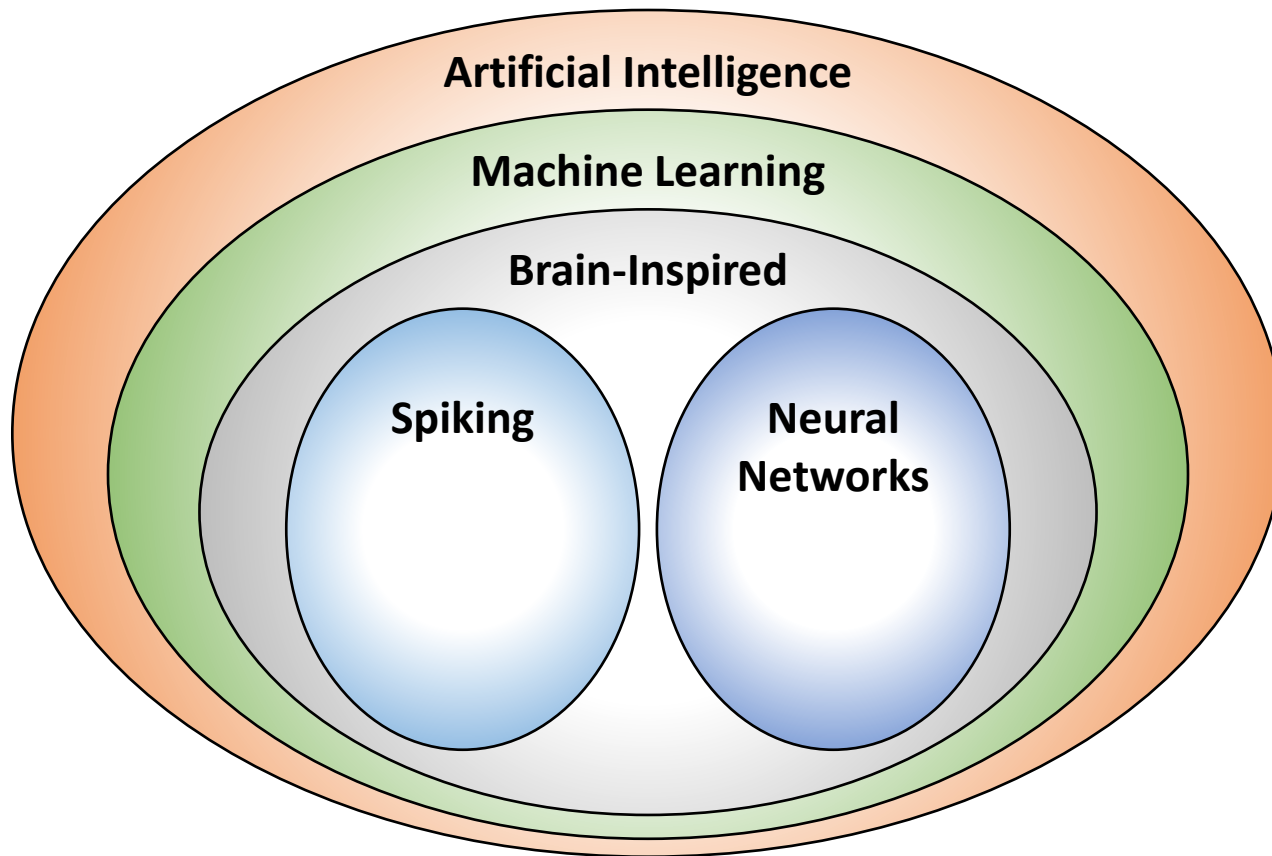
CS/ECE 752: Advanced Computer Architecture I



Source: XKCD

Professor Matthew D. Sinclair
ML (Mostly TPUs)

Machine Learning with Neural Networks



Slide Courtesy: Joel Emer and Vivienne Sze

NeurON: Weighted Sum

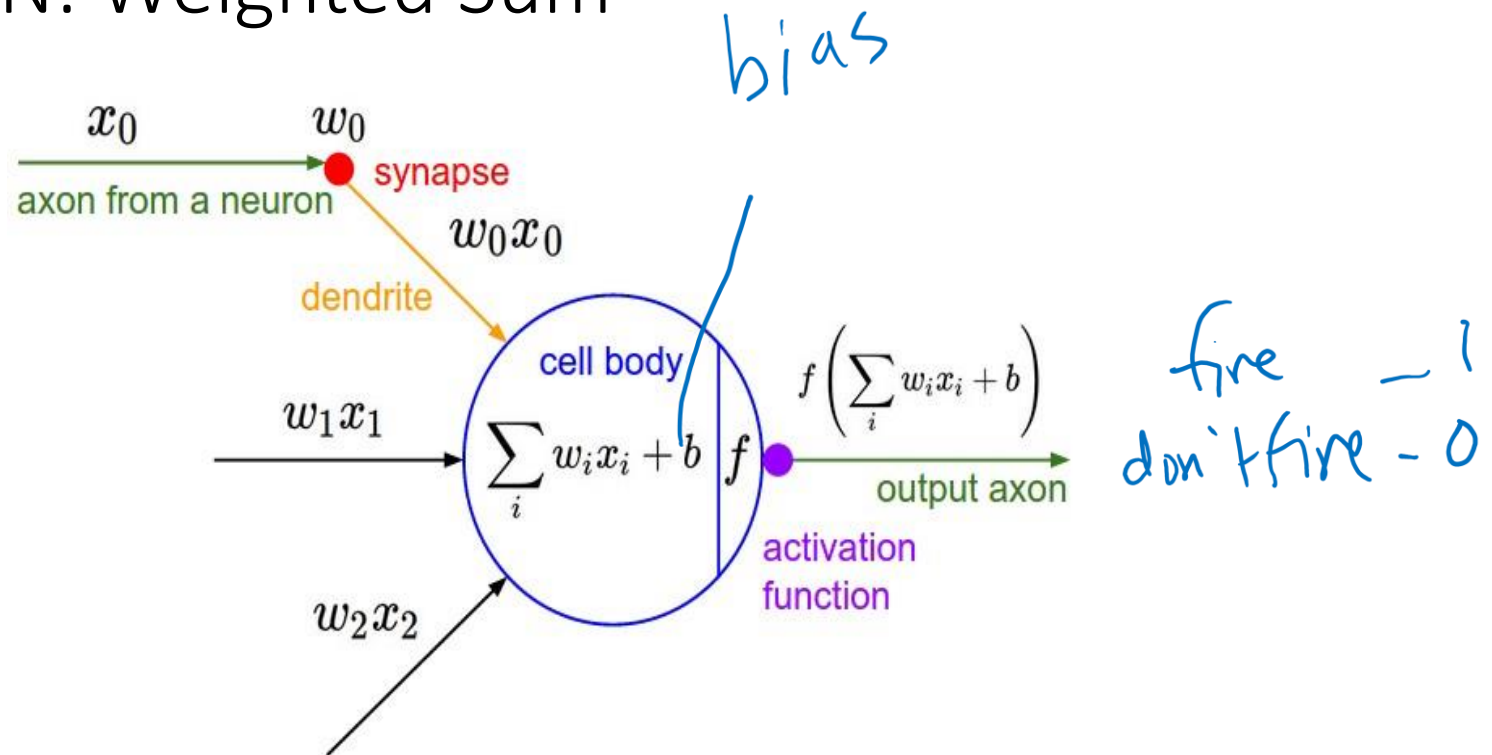
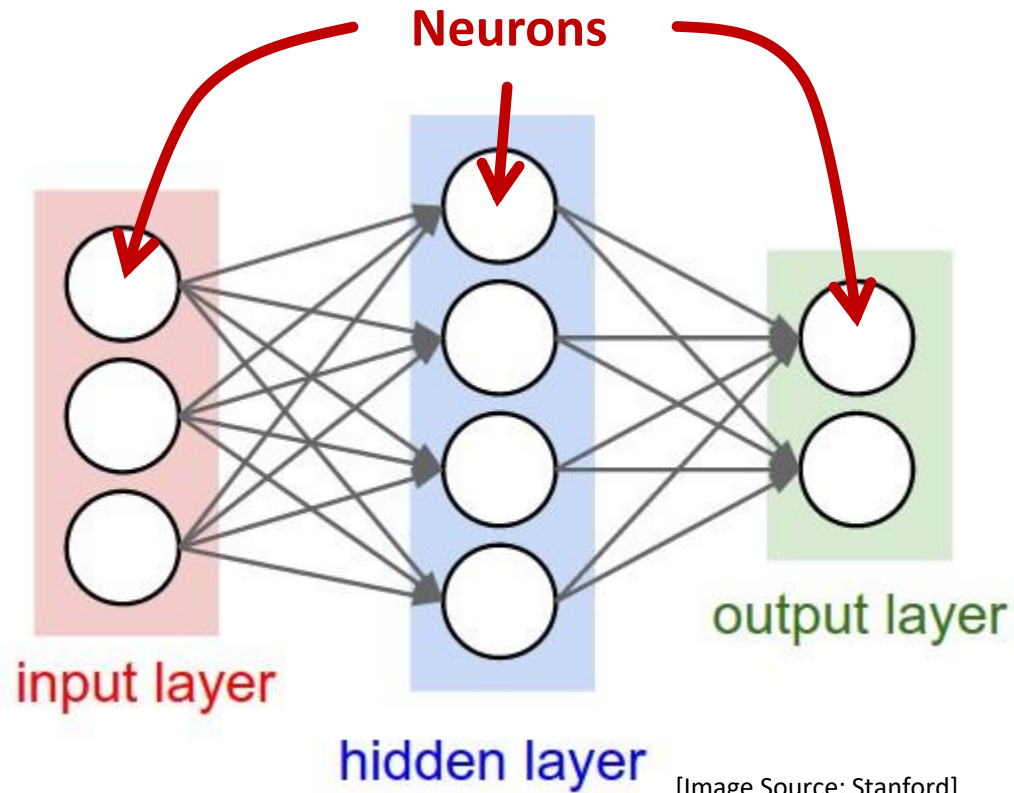


Image Source: Stanford

Slide Courtesy: Joel Emer and Vivienne Sze

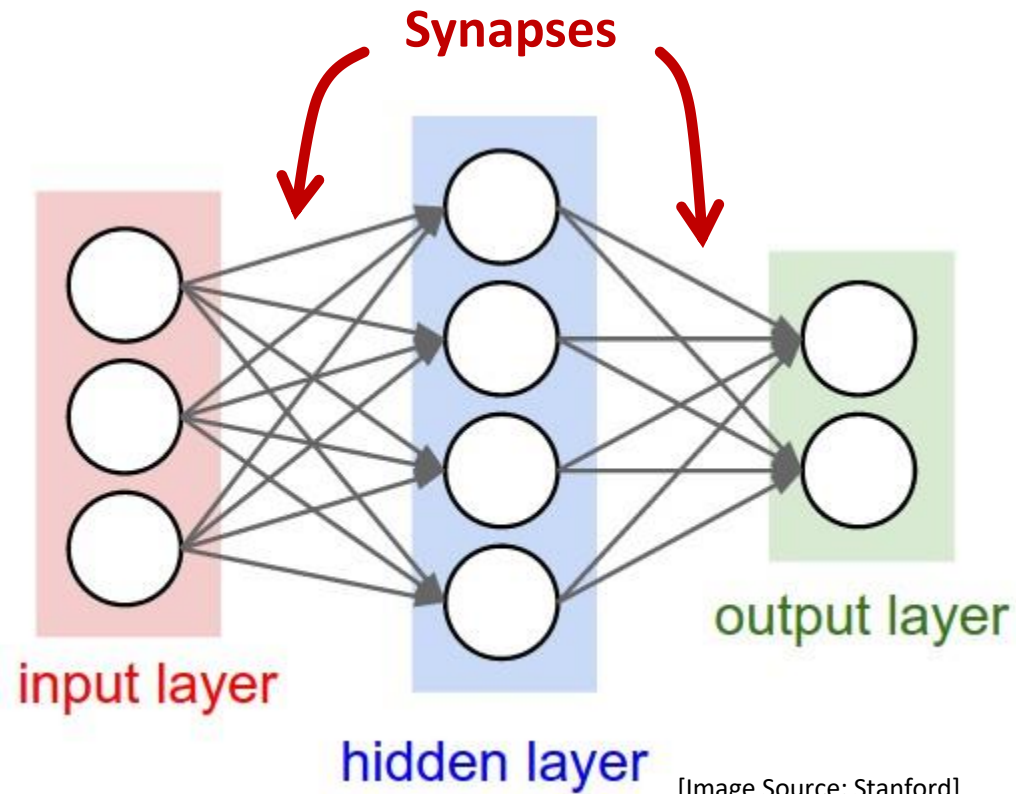
NEURAL NETWORK



[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK

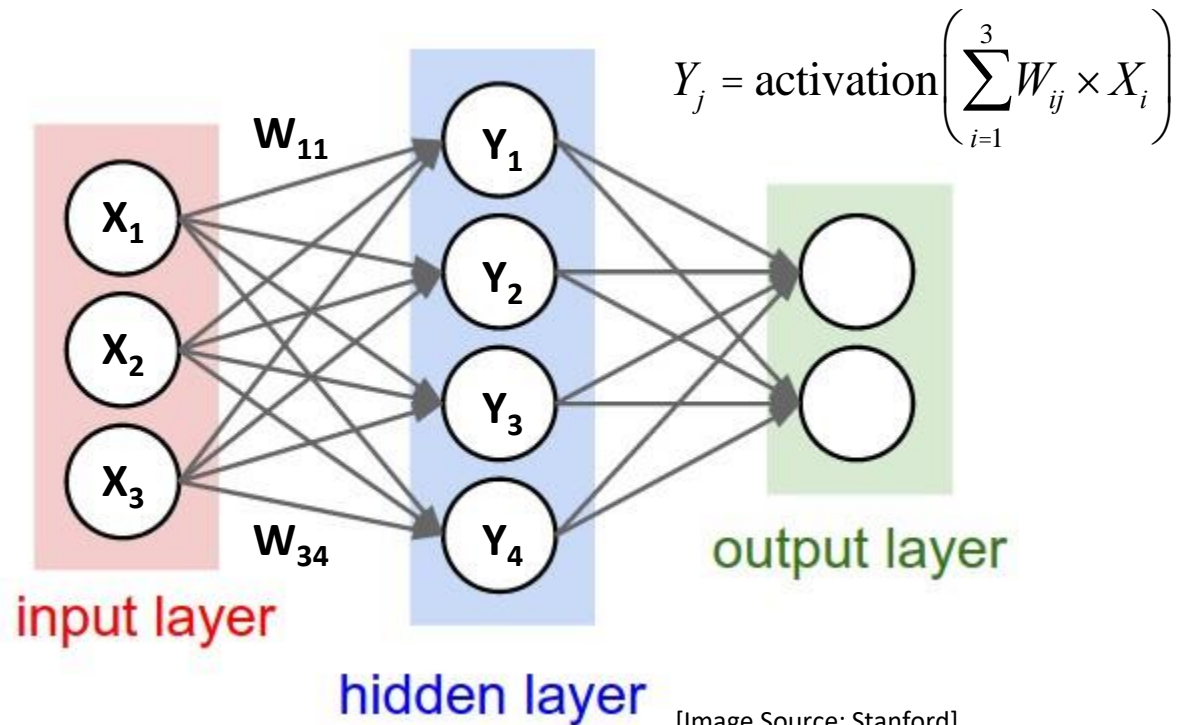


[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK: Multiple weighted sums

Each **synapse** has a **weight** for neuron **activation**

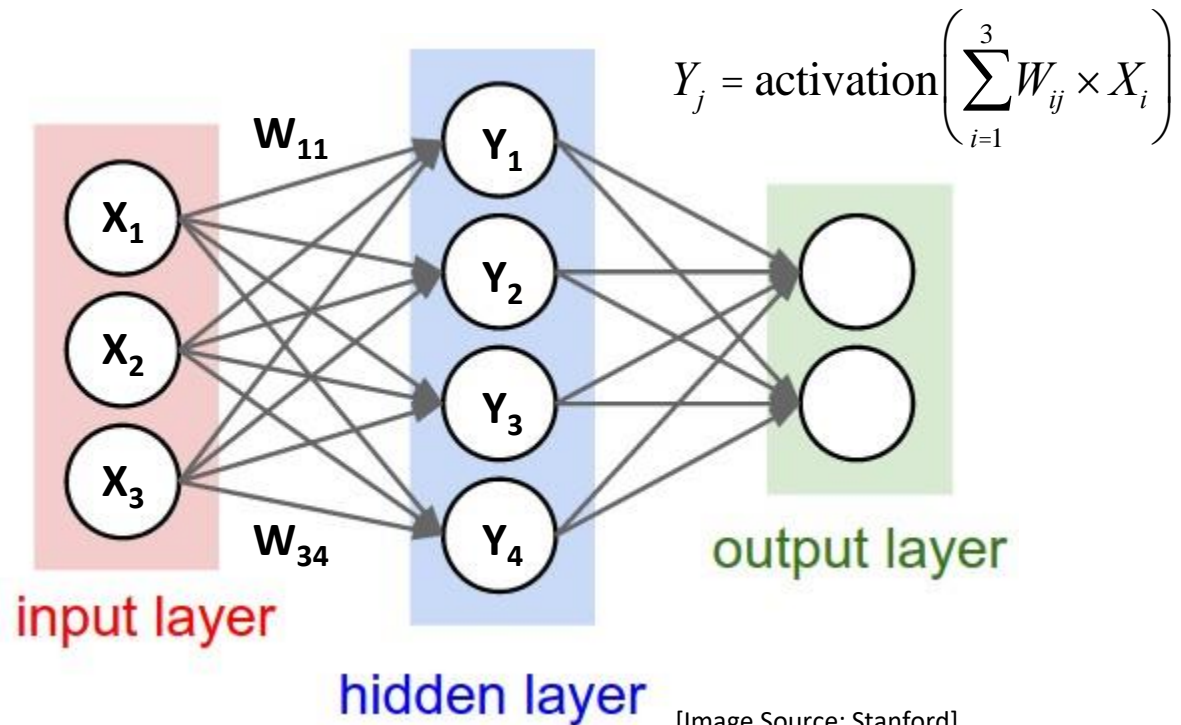


[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK: Multiple weighted sums

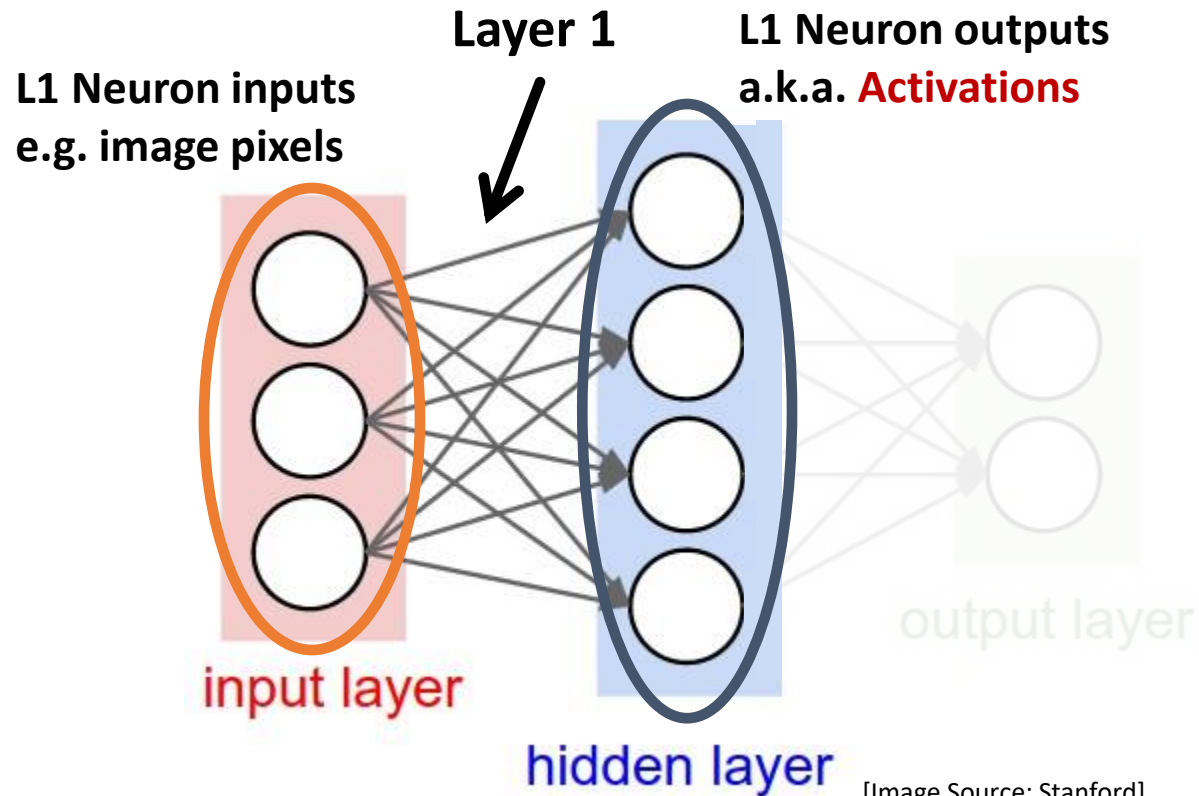
Weight Sharing: multiple synapses use the **same weight value**



[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

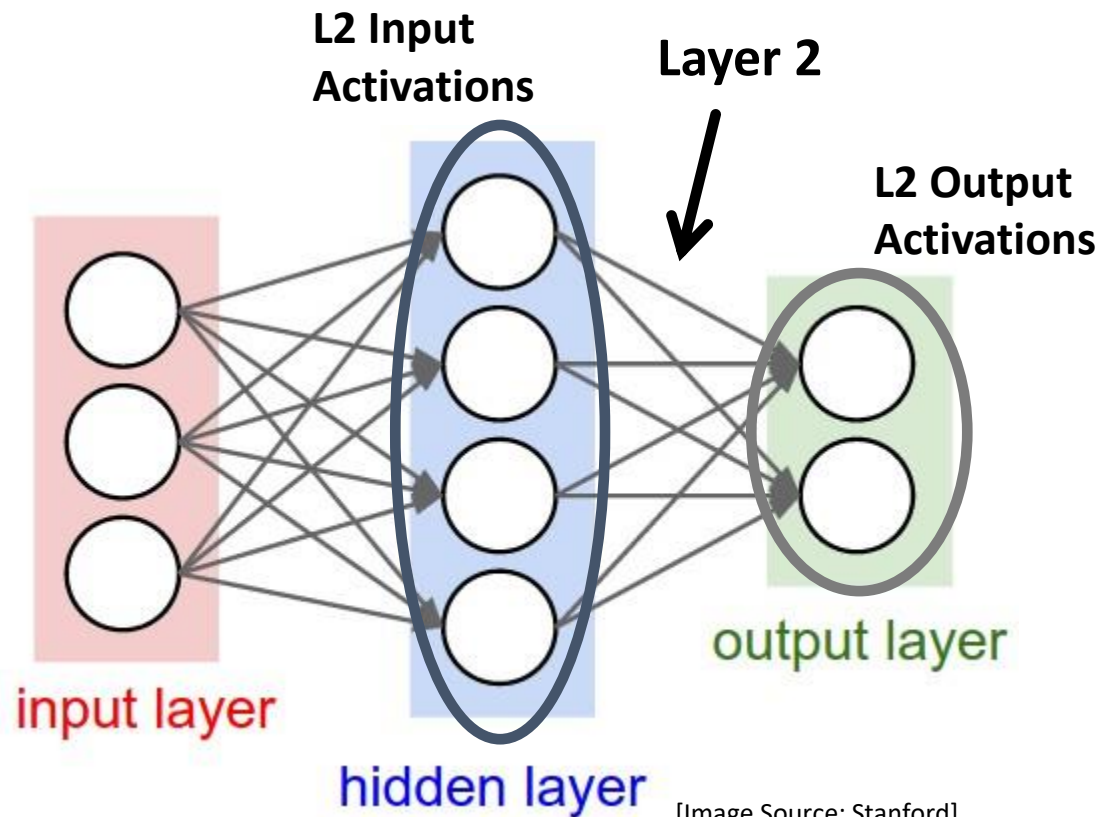
NEURAL NETWORK Terminology



[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK Terminology

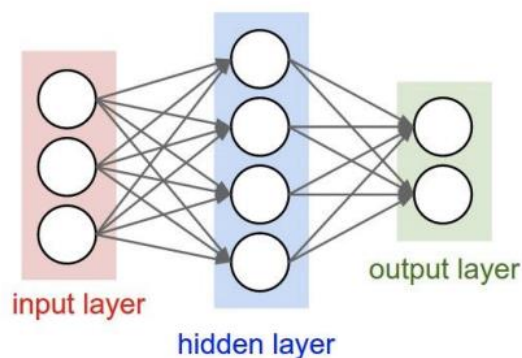


[Image Source: Stanford]

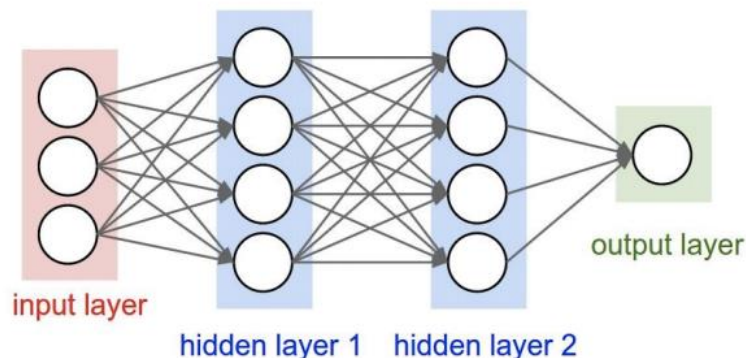
Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK Terminology

A **layer** can refer to a set activations or a set of weights.
In this class, we use **layer to refer to a set of weights**.



“2-layer Neural Net”, or
“1-hidden-layer Neural Net”



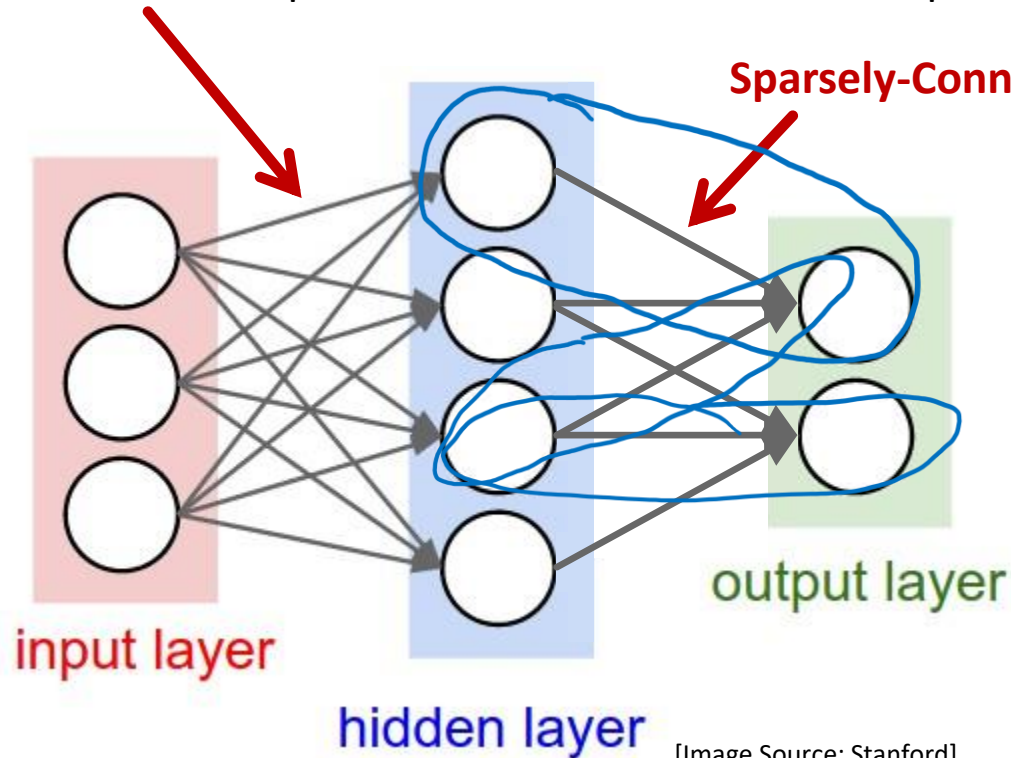
“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

NEURAL NETWORK Terminology

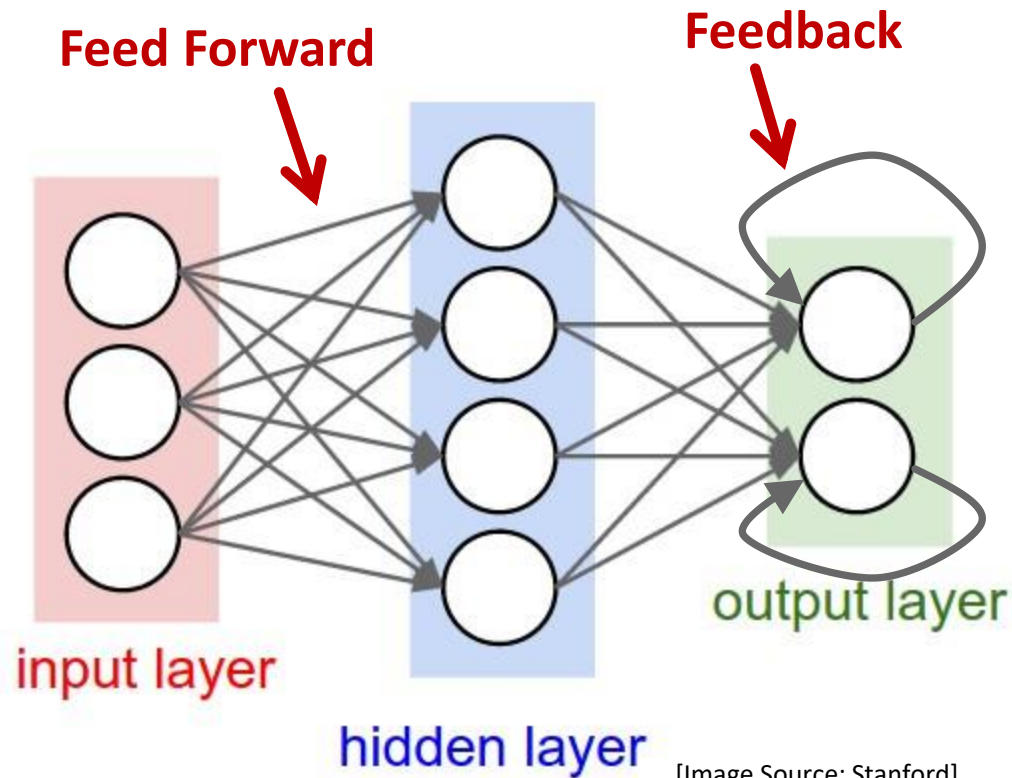
Fully-Connected: all input neurons connected to all output neurons



[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

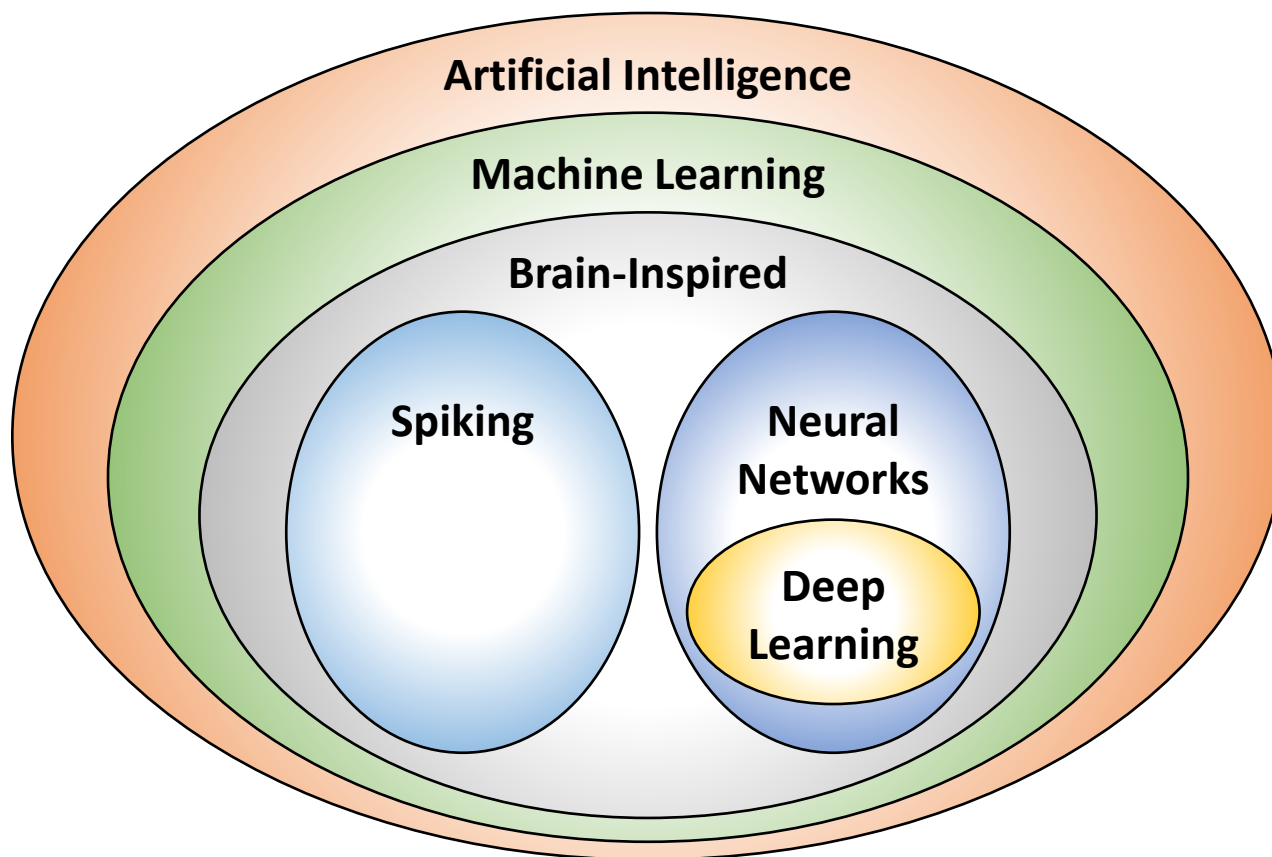
NEURAL NETWORK Terminology



[Image Source: Stanford]

Slide Courtesy: Joel Emer and Vivienne Sze

Deep Learning



Slide Courtesy: Joel Emer and Vivienne Sze

CONV Layer Tensor Computation

Output fmaps (O)

Input fmaps (I)

Biases (B)

Filter weights (W)

$$\underline{O[n][m][x][y]} = \text{Activation}(\underline{B[m]} + \sum_{i=0}^{R-1} \sum_{j=0}^{S-1} \sum_{k=0}^{C-1} \underline{I[n][k][Ux+i][Uy+j]} \times \underline{W[m][k][i][j]}),$$

$$0 \leq n < N, 0 \leq m < M, 0 \leq y < E, 0 \leq x < F,$$

$$E = (H - R + U)/U, F = (W - S + U)/U.$$

Shape Parameter	Description
N	fmap batch size
M	# of filters / # of output fmap channels
C	# of input fmap/filter channels
H/W	input fmap height/width
R/S	filter height/width
E/F	output fmap height/width
U	convolution stride

Google TPU (✓)

- Why ASIC chip?
- Specs
 - 256x256 = 64K 8-bit MAC
 - Peak throughput: 92 TOPS
 - Software Managed On-Chip Memory: 28 MB
- Highlights
 - 15-30X faster than K80 GPU for Inference
 - Bandwidth limited for 6 out of 8 workloads

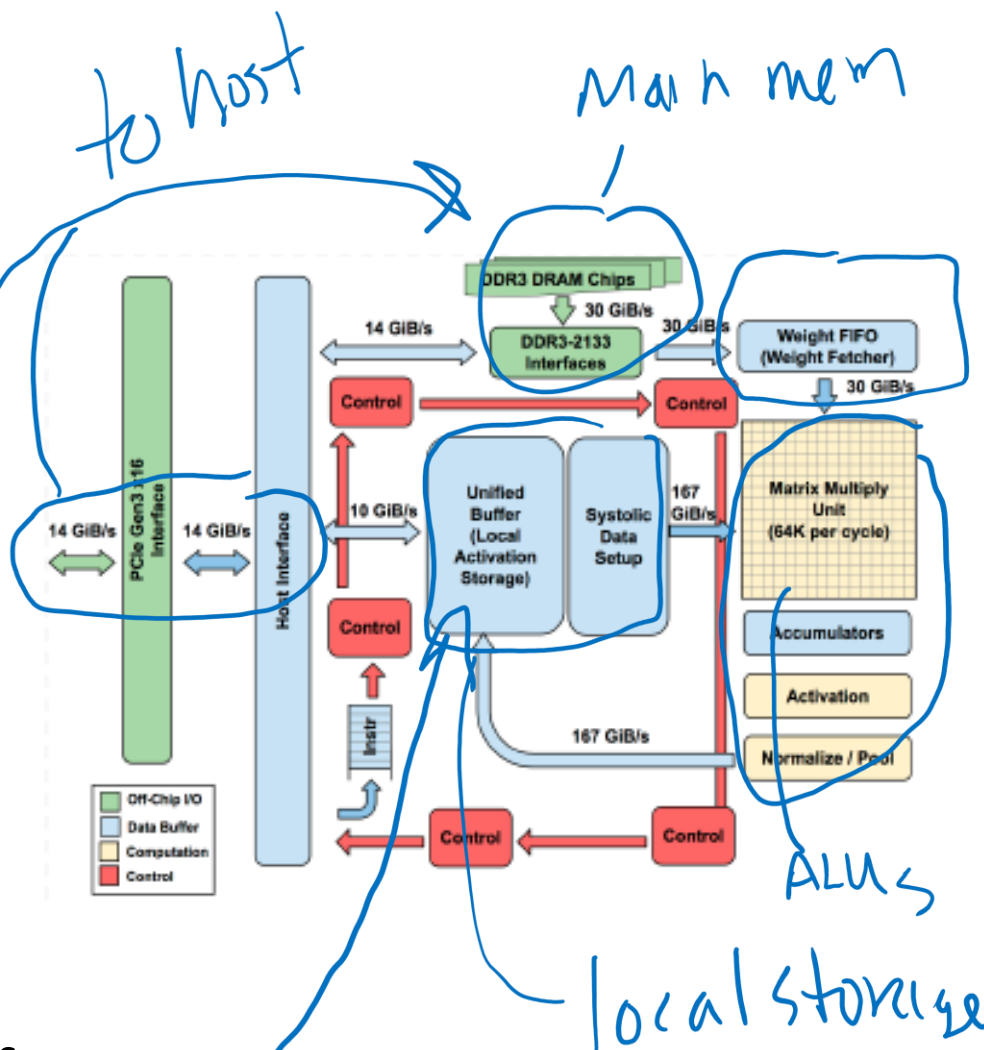
(scratch pads)

Name	LOC	Layers					Nonlinear function	Weights	TPU Ops / Weight Byte	TPU Batch Size	% of Deployed TPUs in July 2016
		FC	Conv	Vector	Pool	Total					
MLP0	100	5				5	ReLU	20M	200	200	61%
MLP1	1000	4				4	ReLU	5M	168	168	
LSTM0	1000	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1500	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1000		16			16	ReLU	8M	2888	8	5%
CNN1	1000	4	72		13	89	ReLU	100M	1750	32	

Table 1. Six NN applications (two per NN type) that represent 95% of the TPU's workload. The columns are the NN name; the number of lines of code; the types and number of layers in the NN (FC is fully connected, Conv is convolution, Vector is self-explanatory, Pool is pooling, which does nonlinear downsizing on the TPU; and TPU application popularity in July 2016. One DNN is RankBrain [Cla15]; one LSTM is a subset of GNM Translate [Wu16]; one CNN is Inception; and the other CNN is DeepMind AlphaGo [Sil16][Jou15].

Google TPU Summary

- Focus: **supervised** learning
- <10 CISC instructions!
- Lots and lots of MACs
 - Reduced precision (8/16 bits)
 - Common for ML workloads
- Large memories
 - Bandwidth is the key limiter
 - MACs only saturated with > 1000 batch size
 - Accumulators locally collect values



TPU Microarchitecture

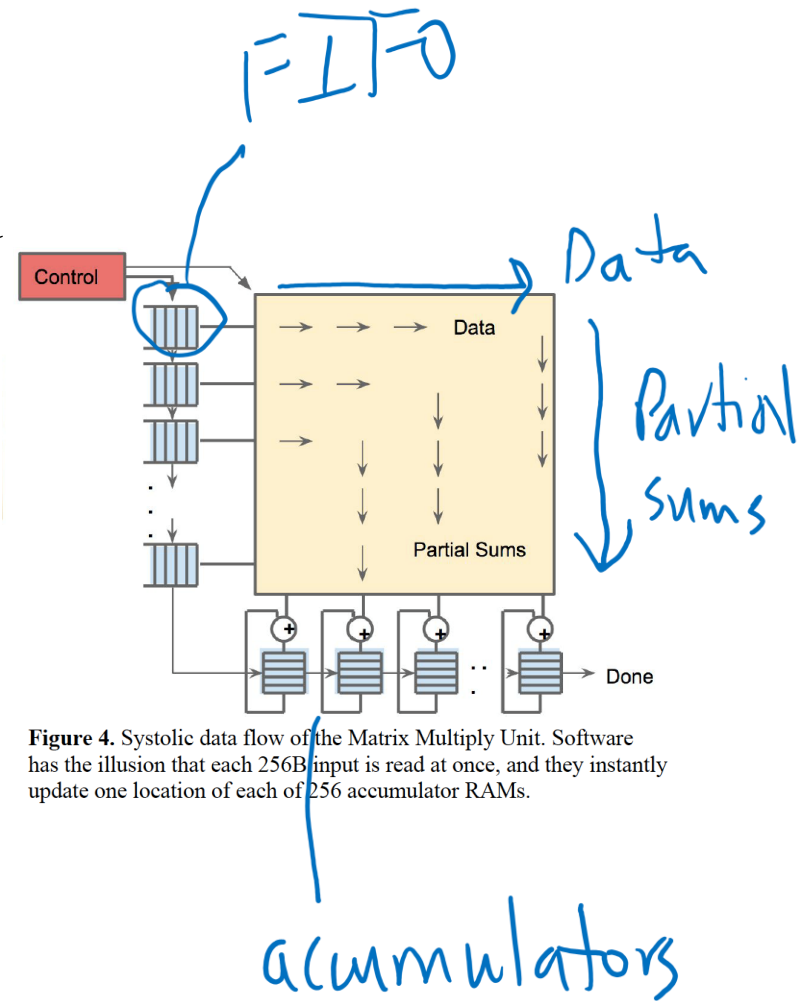
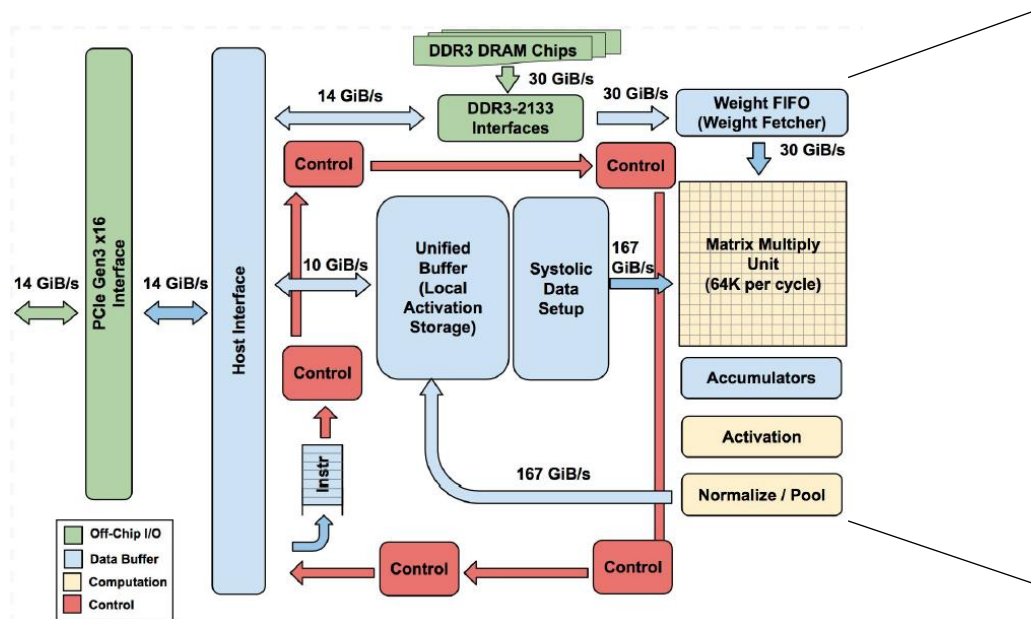


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Why systolic arrays?

- Compute vs I/O
- How many computations are required to compute each matrix element?

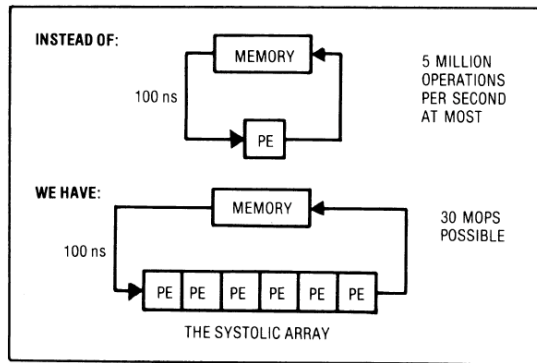
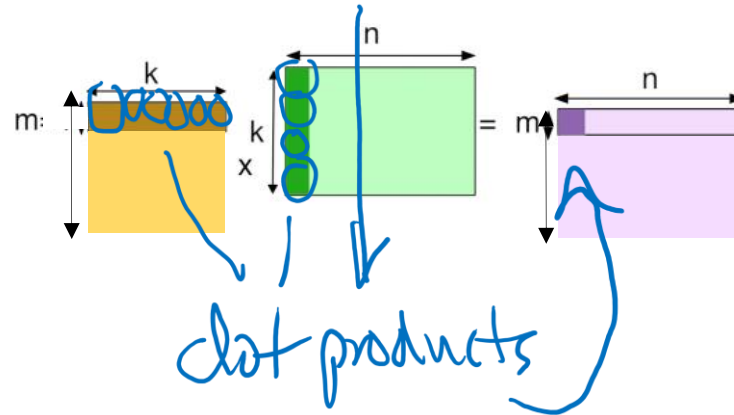
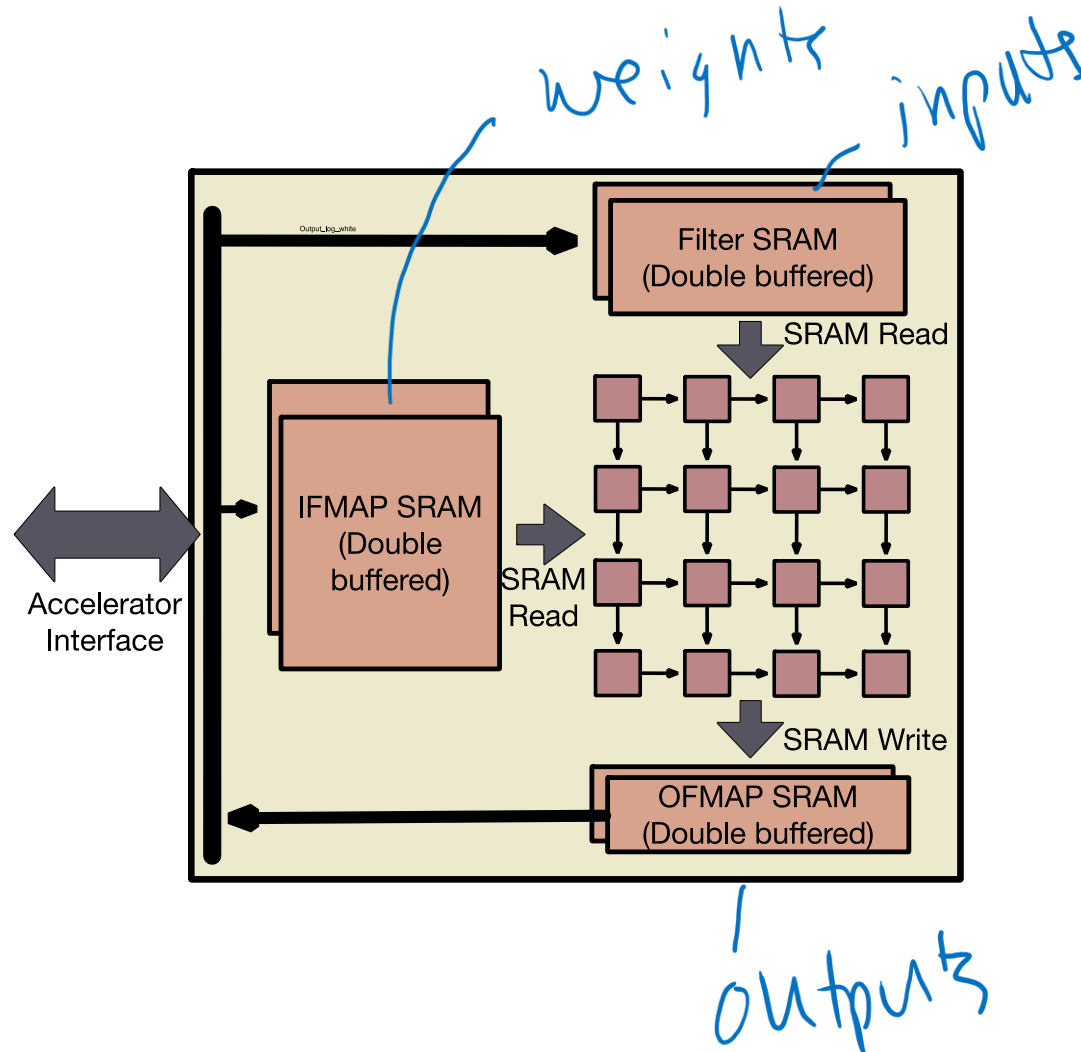


Figure 1. Basic principle of a systolic system.

Source: HT Kung, 1982

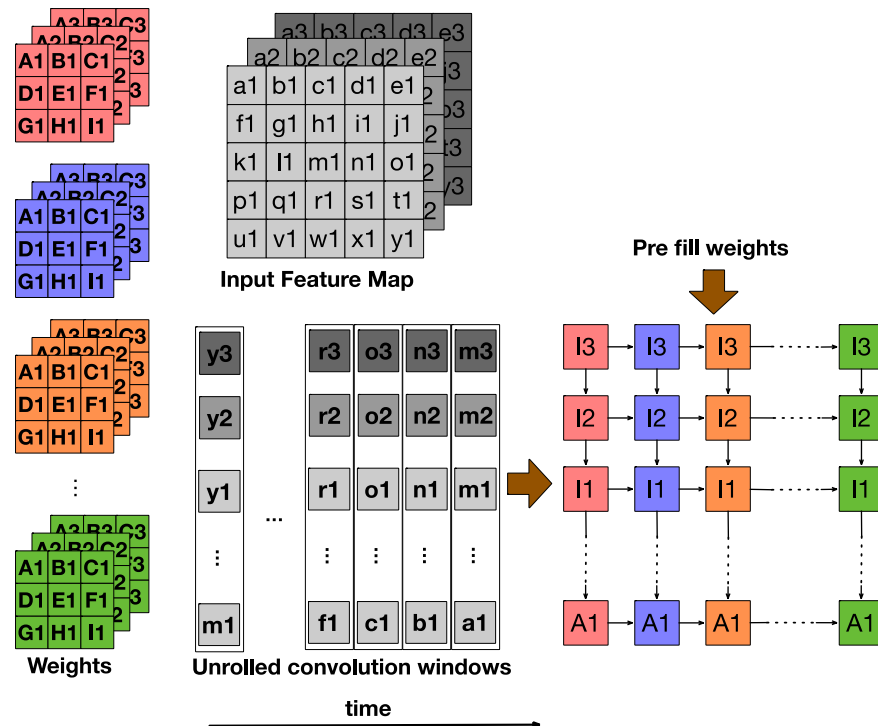


How does a systolic array work?



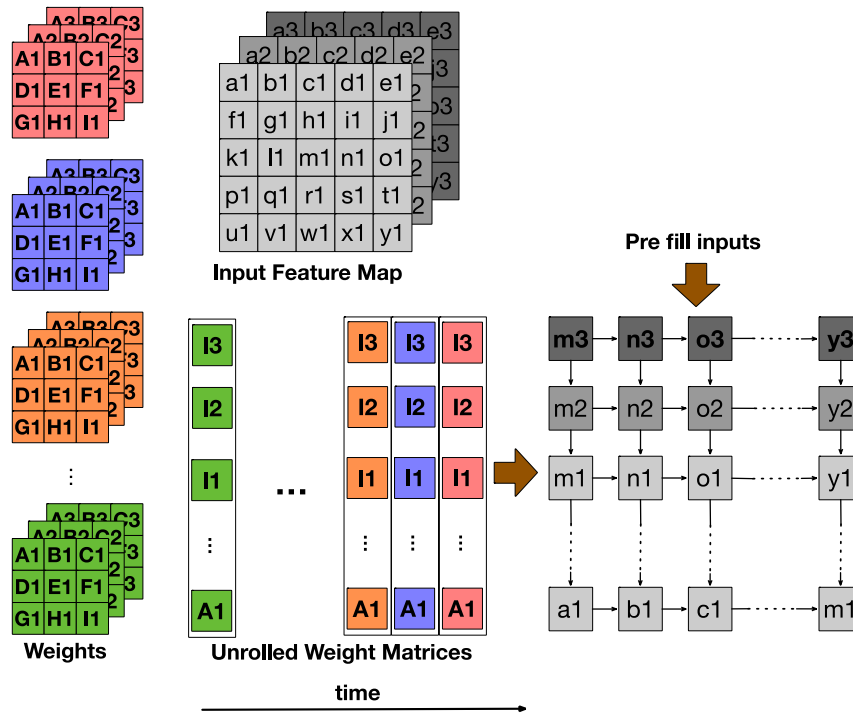
Design 1

- weights stationary, broadcast inputs, move results



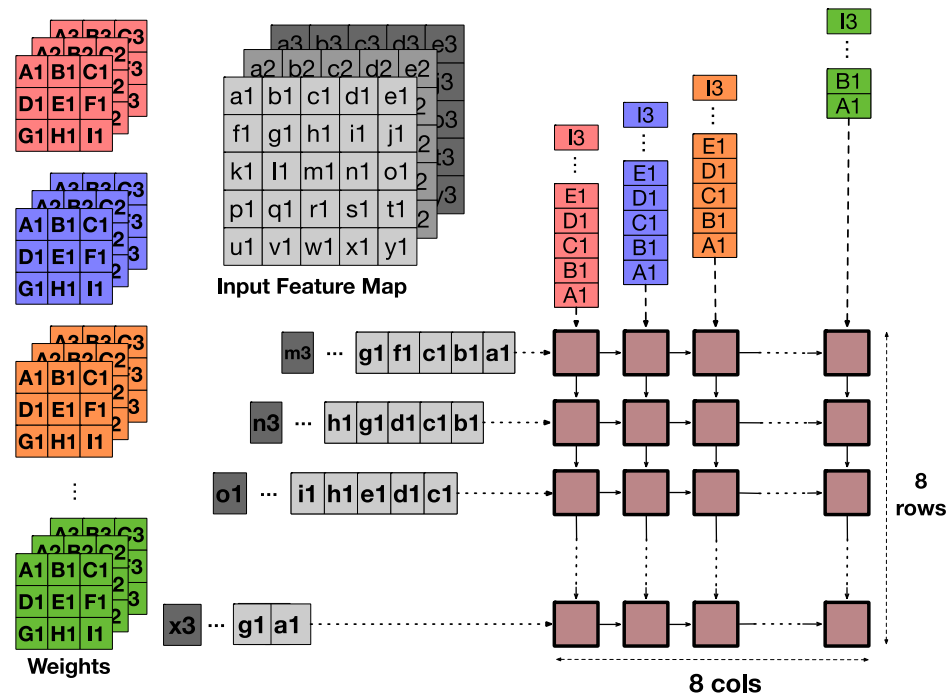
Design 2

- Inputs stationary, broadcast weights, move results

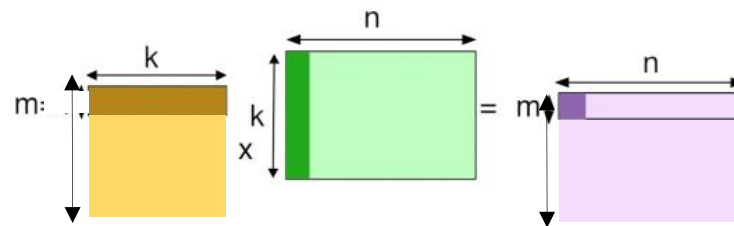


Design 3

- Outputs stationary, broadcast weights, broadcast inputs



What about MLP and LSTMs?



TPU Microarchitecture

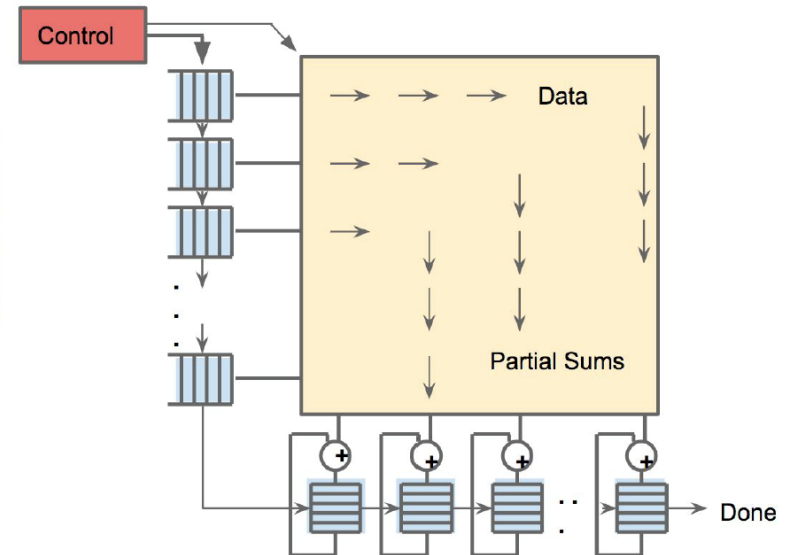
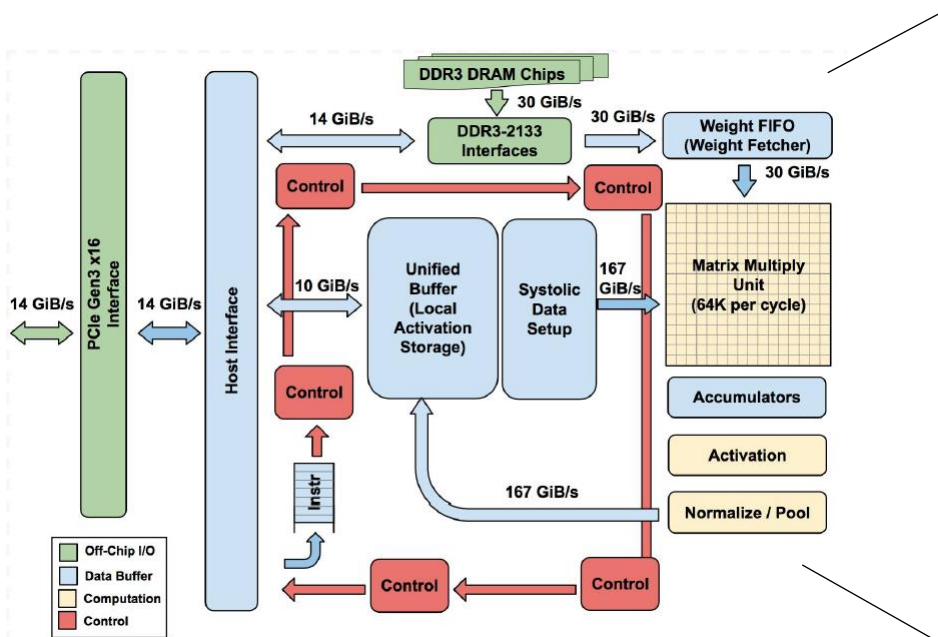
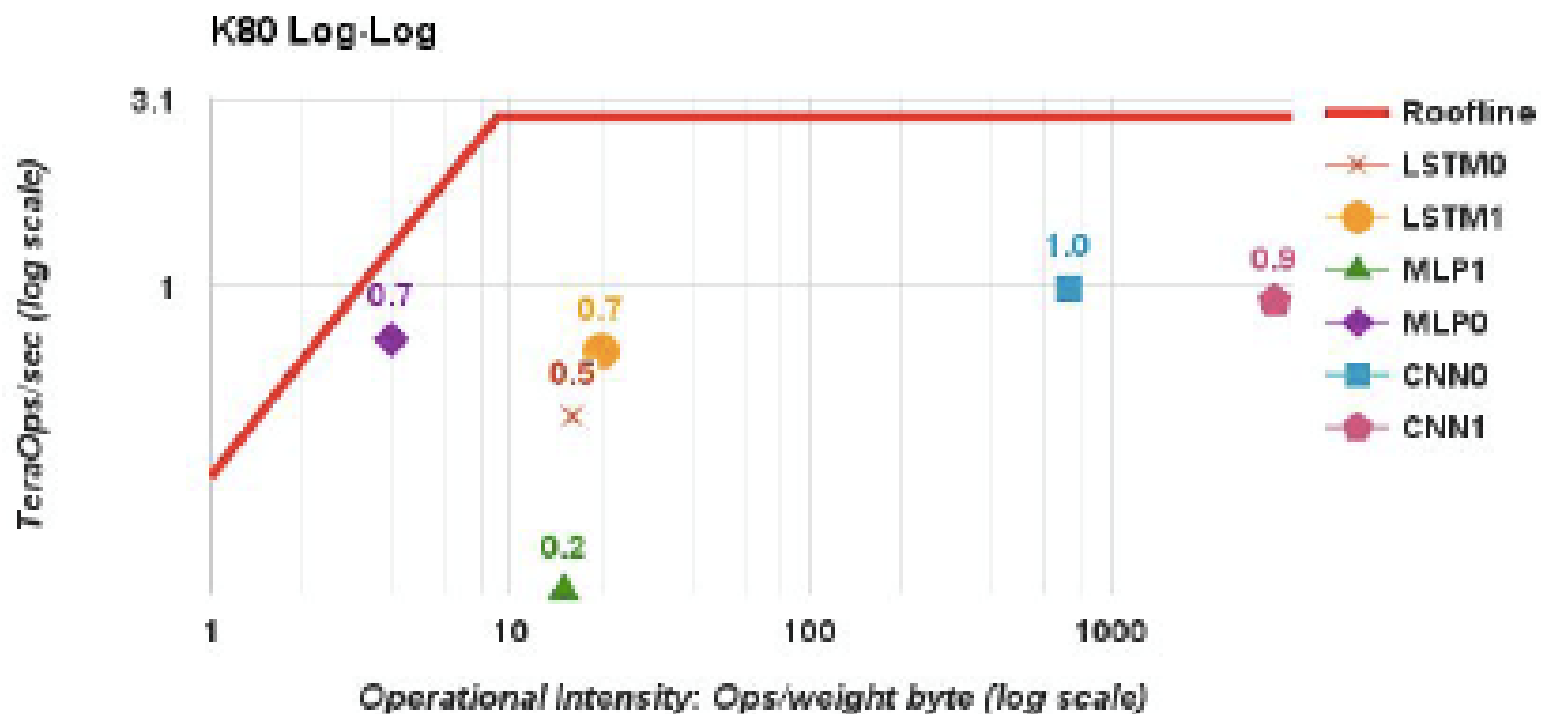


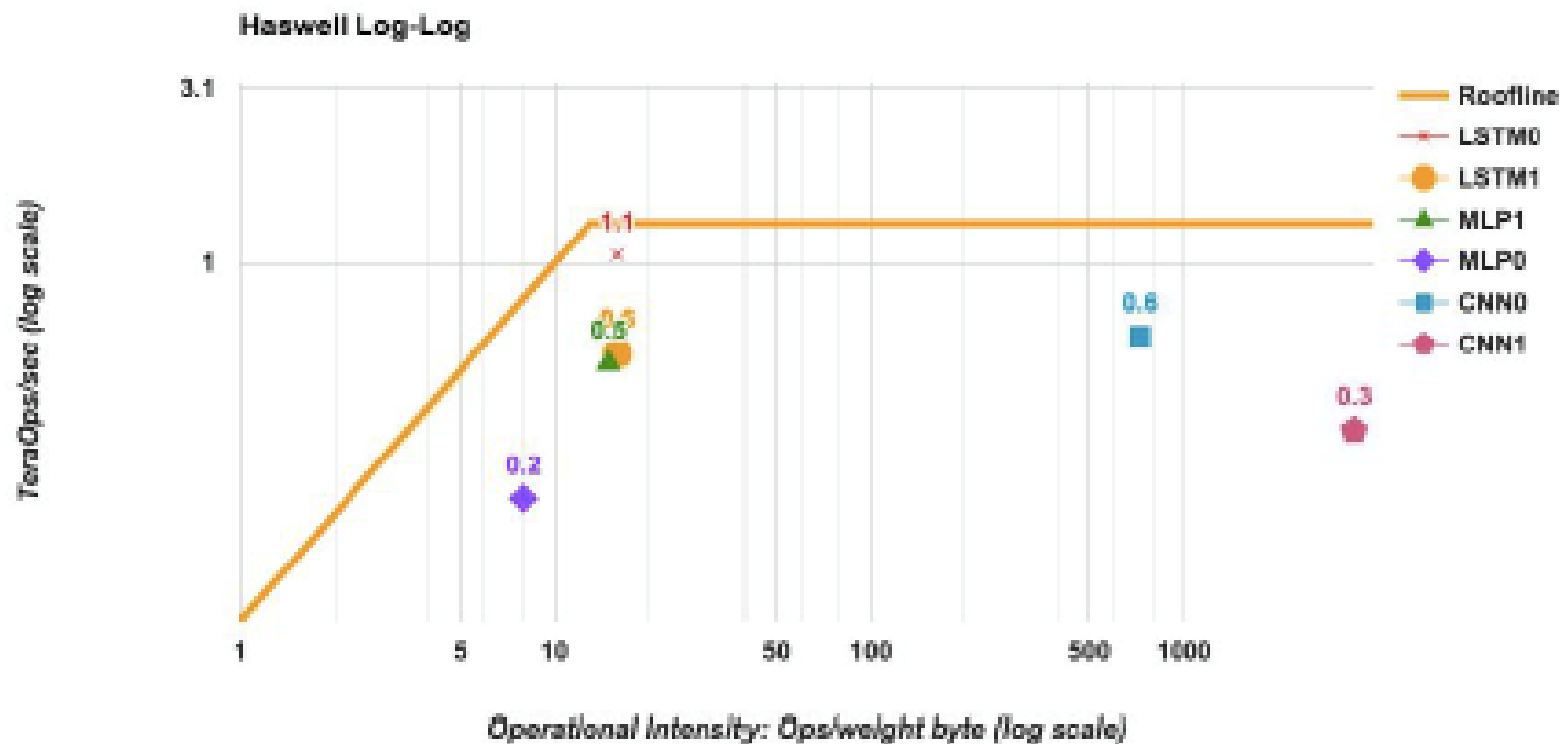
Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

TPU vs. K80 Performance



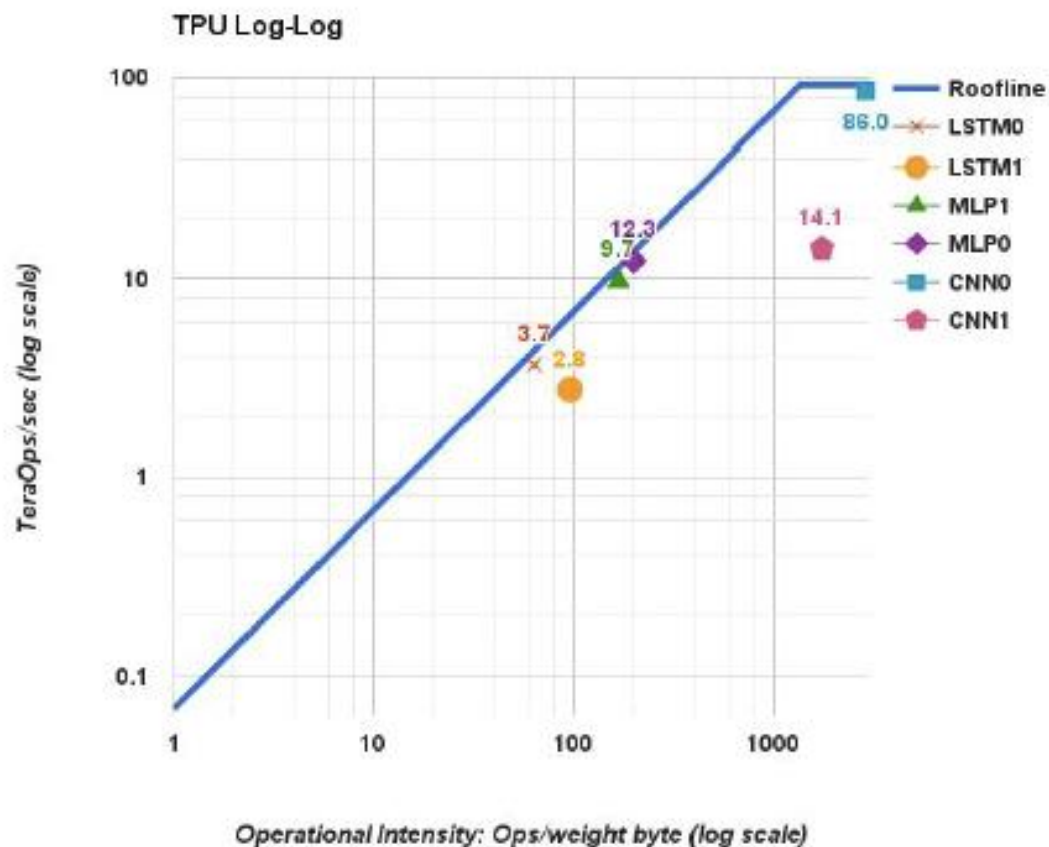
What are Roofline Plots?

TPU vs. CPU Performance



What are Roofline Plots?

TPU Performance



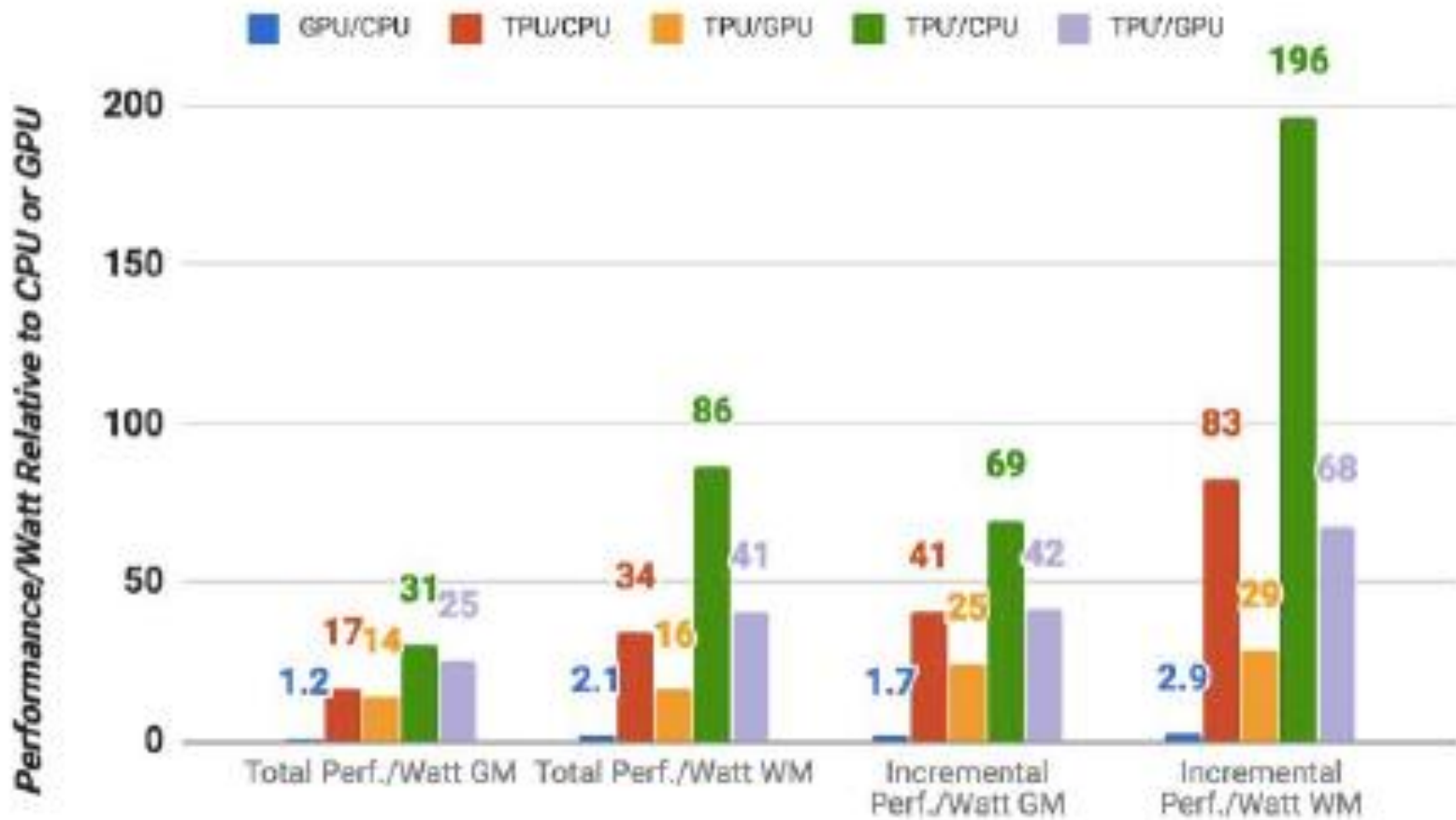
What are Roofline Plots?

TPU Performance

<i>Application</i>	<i>MLP0</i>	<i>MLP1</i>	<i>LSTM0</i>	<i>LSTM1</i>	<i>CNN0</i>	<i>CNN1</i>	<i>Mean</i>	<i>Row</i>
Array active cycles	12.7%	10.6%	8.2%	10.5%	78.2%	46.2%	28%	1
Useful MACs in 64K matrix (% peak)	12.5%	9.4%	8.2%	6.3%	78.2%	22.5%	23%	2
Unused MACs	0.3%	1.2%	0.0%	4.2%	0.0%	23.7%	5%	3
Weight stall cycles	53.9%	44.2%	58.1%	62.1%	0.0%	28.1%	43%	4
Weight shift cycles	15.9%	13.4%	15.8%	17.1%	0.0%	7.0%	12%	5
Non-matrix cycles	17.5%	31.9%	17.9%	10.3%	21.8%	18.7%	20%	6
RAW stalls	3.3%	8.4%	14.6%	10.6%	3.5%	22.8%	11%	7
Input data stalls	6.1%	8.8%	5.1%	2.4%	3.4%	0.6%	4%	8
TeraOps/sec (92 Peak)	12.3	9.7	3.7	2.8	86.0	14.1	21.4	9

Table 3. Factors limiting TPU performance of the NN workload based on hardware performance counters. Rows 1, 4, 5, and 6 total 100% and are based on measurements of activity of the matrix unit. Rows 2 and 3 further break down the fraction of 64K weights in the matrix unit that hold useful weights on active cycles. Our counters cannot exactly explain the time when the matrix unit is idle in row 6; rows 7 and 8 show counters for two possible reasons, including RAW pipeline hazards and PCIe input stalls. Row 9 (TOPS) is based on measurements of production code while the other rows are based on performance-counter measurements, so they are not perfectly consistent. Host server overhead is excluded here. The MLPs and LSTMs are memory-bandwidth limited but CNNs are not. CNN1 results are explained in the text.

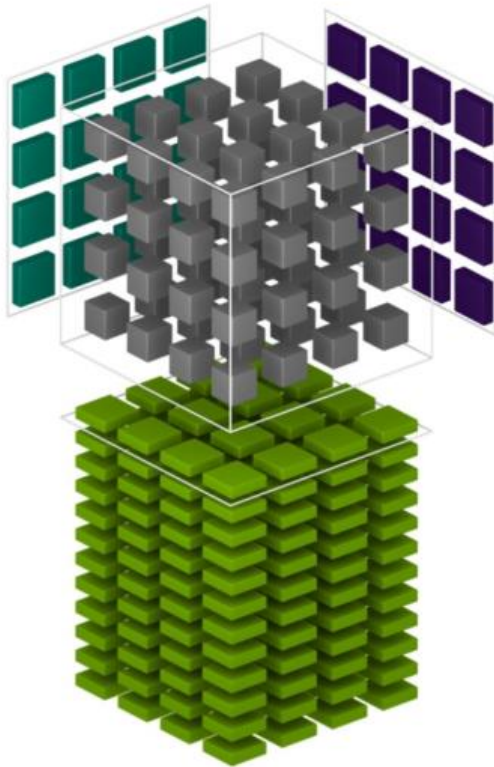
Performance/Watt



NVIDIA's response: TensorCores!

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	1/13X	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16 MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

NVIDIA Volta – TensorCores



$$D = \begin{matrix} \text{FP16 or FP32} & \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} & \text{FP16} & \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} & \text{FP16} & + & \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix} & \text{FP16 or FP32} \end{matrix}$$

Google response to NVIDIA's response!

- TPU v2

- Some apps needed more precision – 16 bits instead of 8 bits
- Added training
- Slightly more general purpose
- Can connect multiple TPUs together to form larger cluster
 - “pod” – supercomputer class machine
- HBM memory – 30X improvement

- TPU v3 – liquid cooled

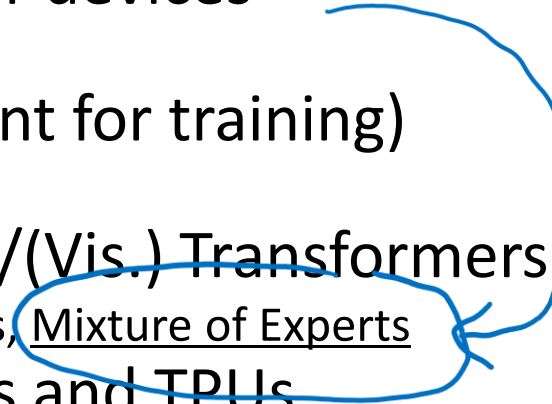
- Enables even more scaling
- Double MXUs/memory capacity, increase frequency, memory BW, etc.
- TPU “pods” (up to 1024 chips) – essentially a supercomputer of TPUs

- All in the name of running larger and larger ML workloads

TPU v4 [Jouppi ISCA'21, ISCA'23]

- Optimized design to essentially have a single design for training and inference
 - Inference: 1 “core”
 - Training: 2 “cores”
 - Both designed to connect together in same ecosystem
 - Back to air cooling as a result
- ALU changes
 - 4 systolic FP matrix units (vs. 1 for TPUv1) + parallel partial sums
 - 4D tensor DMA support
- Shared on-chip interconnect instead of point-to-point
- Uses an optical network (NVIDIA claims gives worse BW 😊)
 - Dynamically reconfigures network to fit application needs
- Embrace multi-tenancy – now multiple concurrent jobs → sharing/fighting over resources
 - Likely some sort of virtualization added (unclear)
- Compiler: changed VLIW width (322b → ~403b)
 - Reject backwards compatibility!

TPU Next Steps

- TPUv5 exists but less publicized...
 - Uses RL to help design it
 - TPUv6 (“preview”) announced earlier this year
 - More memory/BW, higher clock speed, more ALUs ...
 - Open research questions
 - Compress ML models to fit on a smaller devices
 - Current hot area of research
 - Find alternatives to backprop (important for training)
 - “Brains don’t do backprop”
 - Other kinds of ML beyond CNNs/RNNs/(Vis.) Transformers
 - Unsupervised learning, student-teacher networks, Mixture of Experts
 - Remove barriers (synch) between CPUs and TPUs
 - CPU-GPU redux?
- 

TPU Overview (as of 11/15/24)

Tensor Processing Unit products^{[15][16][17]}

	TPUv1	TPUv2	TPUv3	TPUv4 ^[16] ^[18]	TPUv5e ^[19]	TPUv5p ^[20] ^[21]	v6e (Trillium) ^{[22][23]}
Date introduced	2015	2017	2018	2021	2023	2023	2024
Process node	28 nm	16 nm	16 nm	7 nm	Unstated	Unstated	
Die size (mm ²)	331	< 625	< 700	< 400	300-350	Unstated	
On-chip memory (MiB)	28	32	32	32	48	112	
Clock speed (MHz)	700	700	940	1050	Unstated	1750	
Memory	8 GiB DDR3	16 GiB HBM	32 GiB HBM	32 GiB HBM	16 GB HBM	95 GB HBM	32 GB
Memory bandwidth	34 GB/s	600 GB/s	900 GB/s	1200 GB/s	819 GB/s	2765 GB/s	1640 GB/s
TDP (W)	75	280	220	170	Not Listed	Not Listed	
TOPS (Tera Operations Per Second)	23	45	123	275	197 (bf16) 393 (int8)	459 (bf16) 918 (int8)	918 (bf16) 1836 (int8)
TOPS/W	0.31	0.16	0.56	1.62	Not Listed	Not Listed	

Source: Wikipedia

Conclusion

- Moore's Law petering out (even for GPUs – Dark Silicon paper)
 - Can't scale a single GPU much/any further
 - Solution: multi core-like GPUs (and CPUs – analogy breaks down)!
- Lots of opportunities for interesting research!
 - Open questions: workload partitioning, imbalance, scheduling, coherence, consistency, synchronization
- Rely on data locality to avoid inter-chiplet communication
 - Redesign scheduler to avoid/exploit temporal locality across chiplets
 - Redesign page placement to avoid/exploit temporal locality per chiplet
 - ...