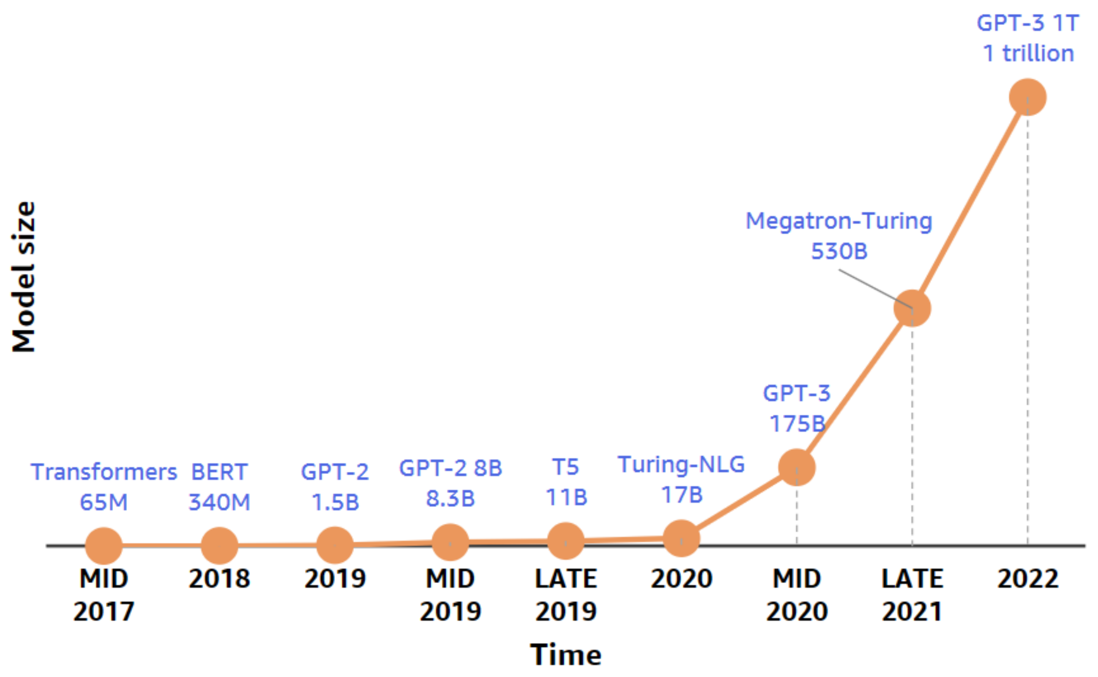
Lecture 3: Data Reuse

Puneet Gupta

Observation 1: Scaling of DNN Models



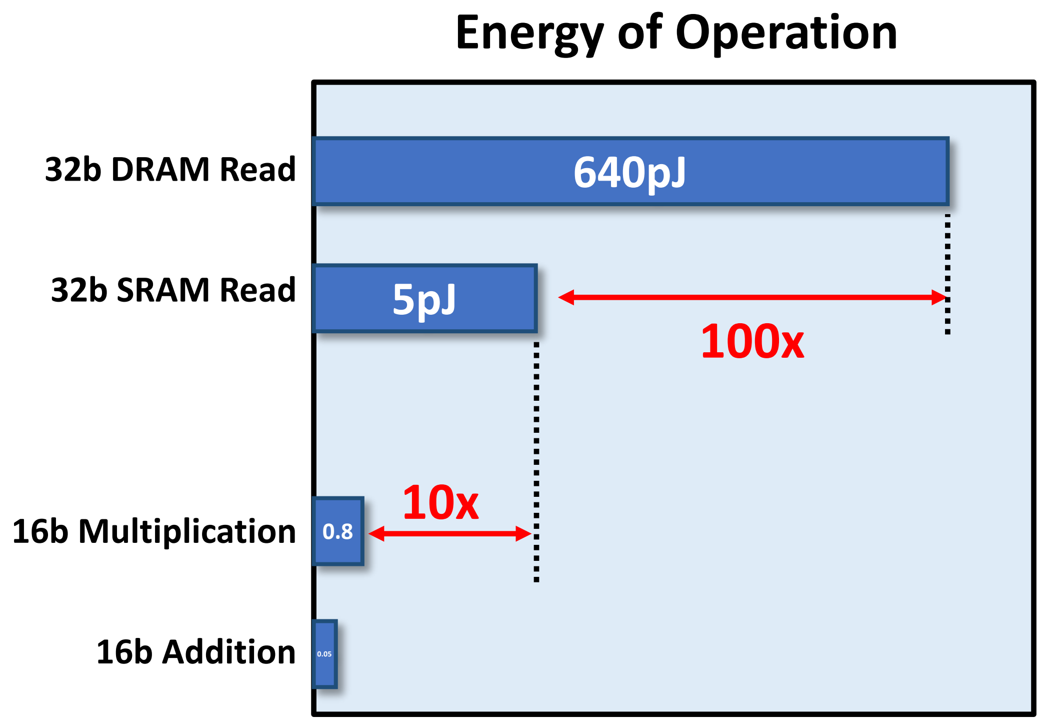
Metrics	LeNet-5 [1998]	AlexNet [2012]	VGG-16 [2015]	ResNet-50 [2016]
Top-5 error (ImageNet)	n/a	16.4	7.4	5.3
Input Size	28x28	227x227	224x224	224x224
# of CONV Layers	**2**	**5**	**16**	**49**
# of Weights	2.6k	2.3M	14.7M	23.5M
# of MACs	283k	666M	15.3G	3.86G
# of FC layers	**2**	**3**	**3**	**1**
# of Weights	58k	58.6M	124M	2M
# of MACs	58k	58.6M	124M	2M
Total Weights	**60k**	**61M**	**138M**	**25.5M**
Total MACs	**341k**	**724M**	**15.5G**	**3.9G**

- Compute and memory requirements have both been increasing at a rate much faster than technology scaling

Observation 2: Where does energy and time go ?

- Every MAC operation requires reading three operands → cost of reading 10X-100X higher than cost of compute
  - Latency follows similar trends
  - Minimizing data movement is critical

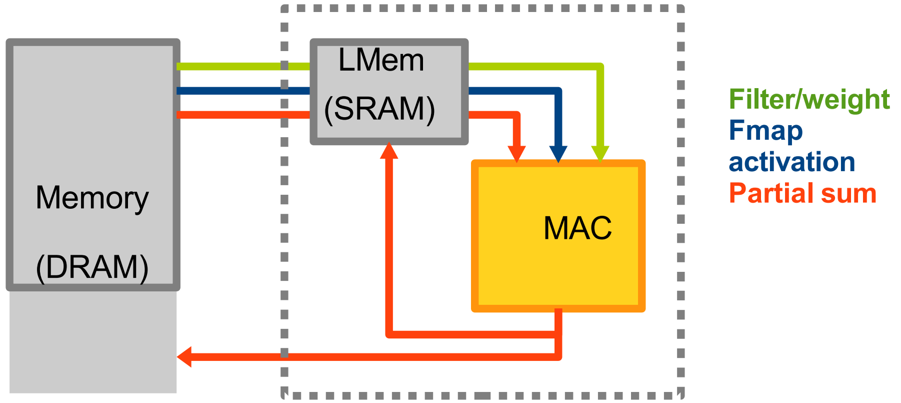
  → reuse data as much as you can once you have read it
- Reuse from where ?
  - From local buffer memory
  - From a distant larger memory
  - From even further away DRAM

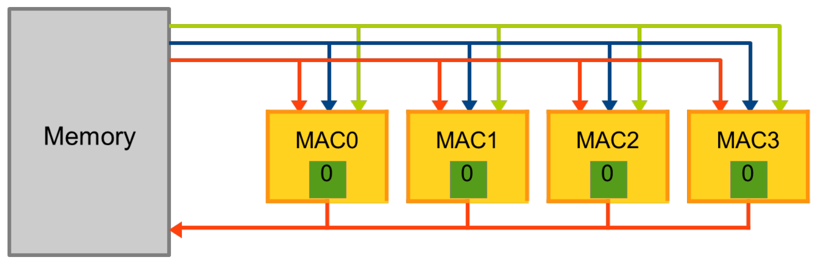


Basic Principles of Reuse

•

- Data read once from a large expensive memory
- **Temporal Reuse**
  - Store data to a small cheap memory and reuse data several times
- **Spatial Reuse**
  - Send the same data to multiple PEs and reuse data at distinct PEs

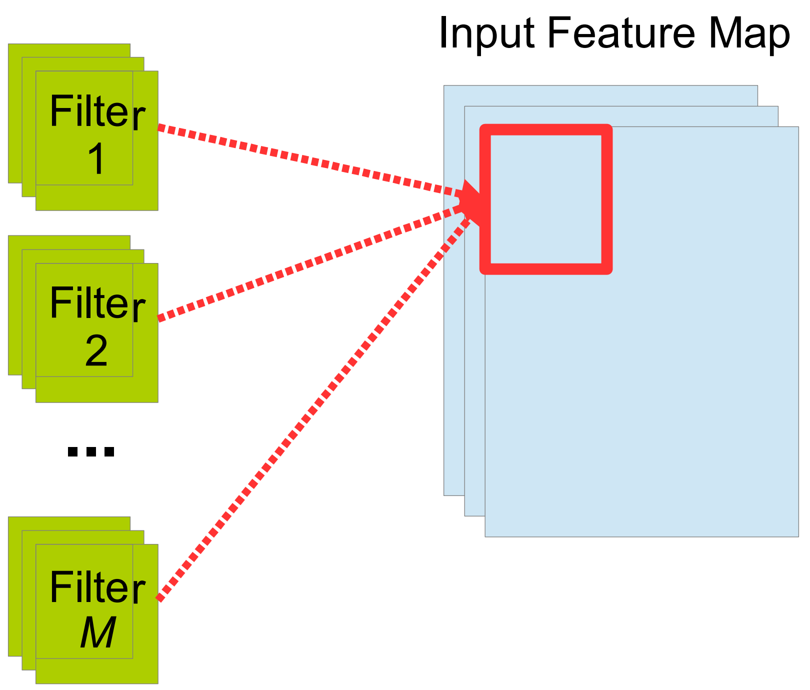




[IITG_JJTV_SPARC2]

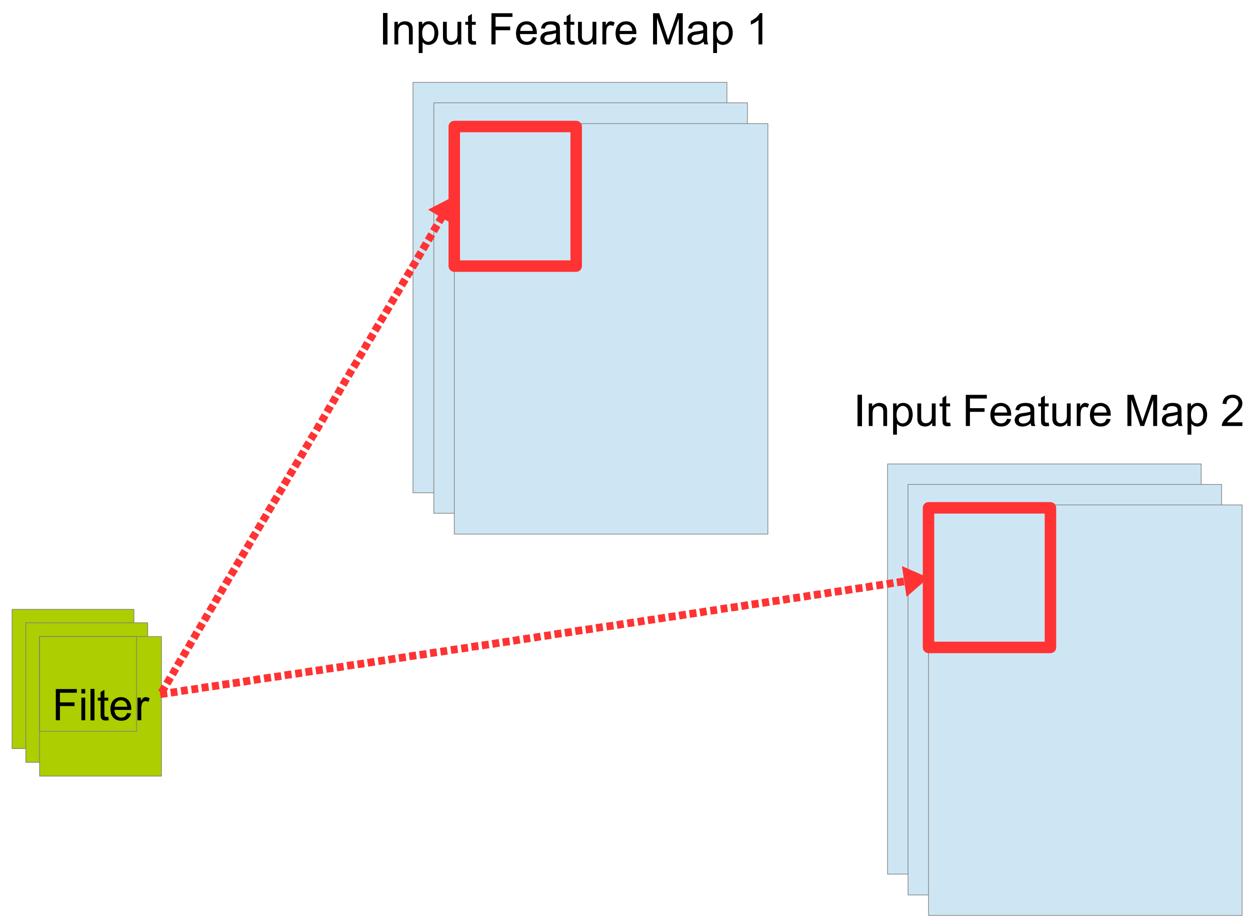
Input Reuse

- The same ifmap is used by several filters
  - Different filters applied to the same ifmap
  - Each input activation is reused *M* times



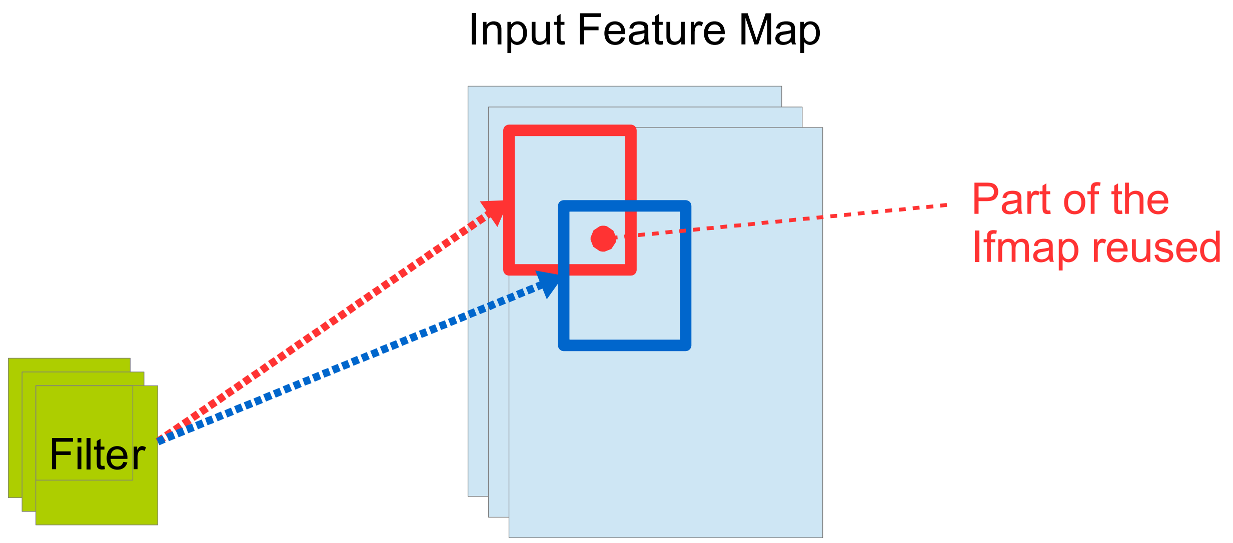
Weight Reuse

- The same filter is applied to different ifmap (batch size > 1)



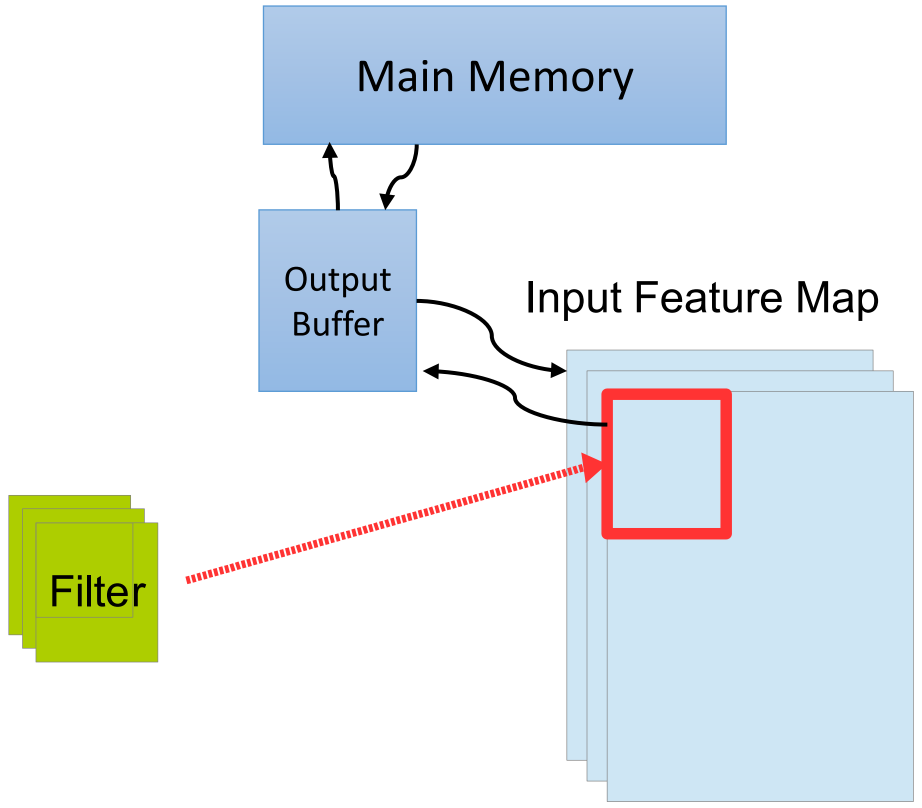
Convolutional Reuse

- The same filter is applied to different parts of the ifmap
  - The filter is reused
  - Part of the ifmap is reused

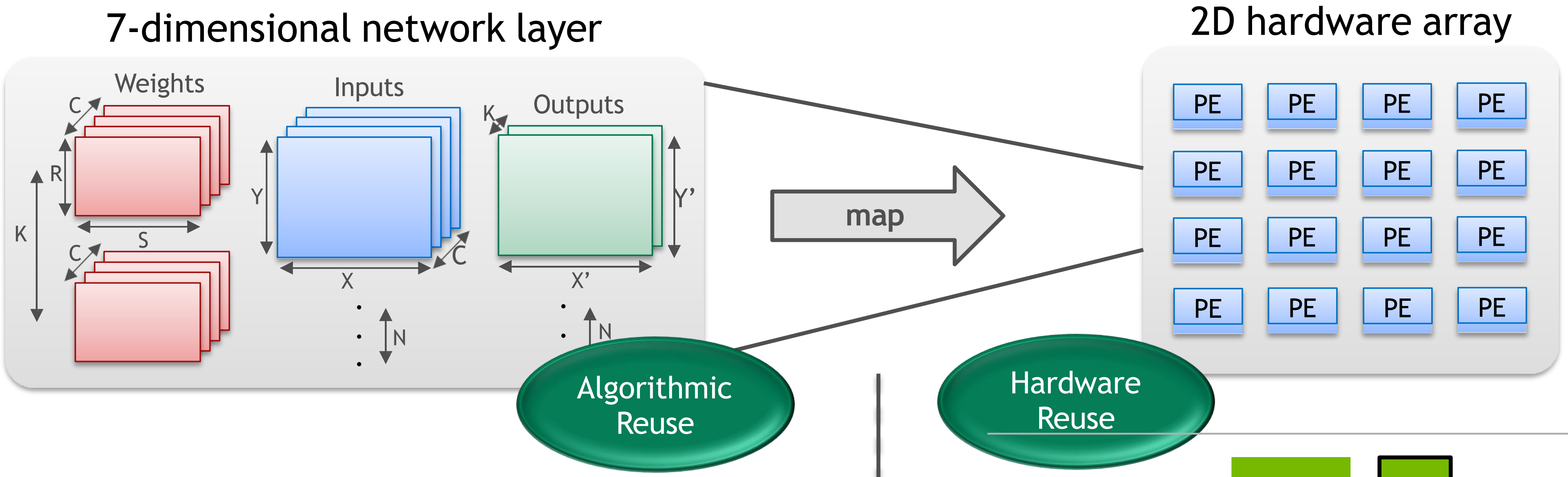


Output (Partial Sum) Reuse

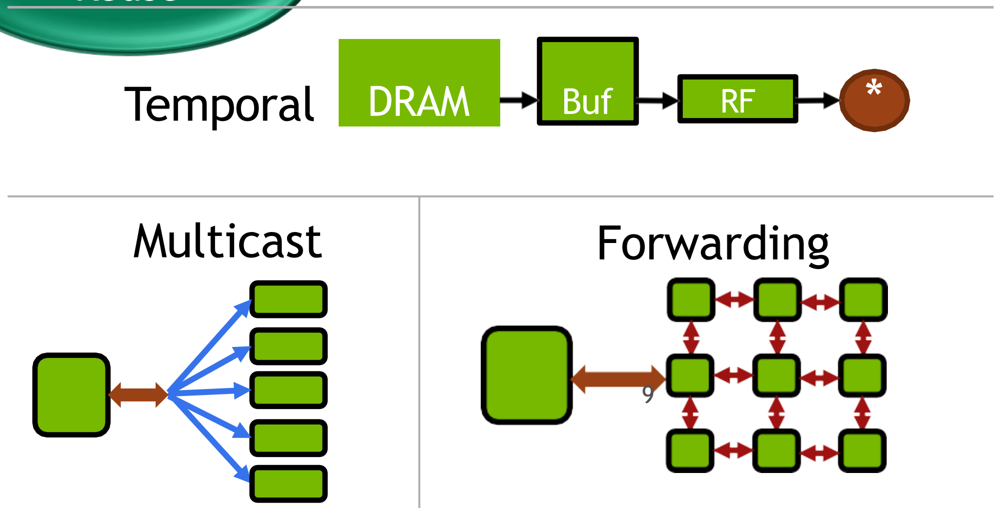
- Dot product sizes in NN layers are large (64-2048); #MACs is smaller → partial sums need to be stored
  - In memory → many costly main memory accesses
  - From buffer → cheaper but extra hardware
- E.g., 128x(3x3x128) CONV layer; 16-sized MAC in hardware;
  - Dot product size:
    - 1152
  - Number of partial sum read/writes:
    - 1152/16=72
  - How many main memory read/writes ?
    - Depends on output buffer size; dataflow
    - Buffer size = 1
      - OS: 0
      - WS: 72*128



MAPPING REUSE TO HARDWARE



- **7D Computation Space**
  - R * S * X * Y * C * K * N
- **4D Operand / Result Spaces -**
  - Weights - R * S * C * K
  - Inputs - X * Y * C * N
  - Outputs - X' * Y' * K * N



Millions of non-trivial mappings possible

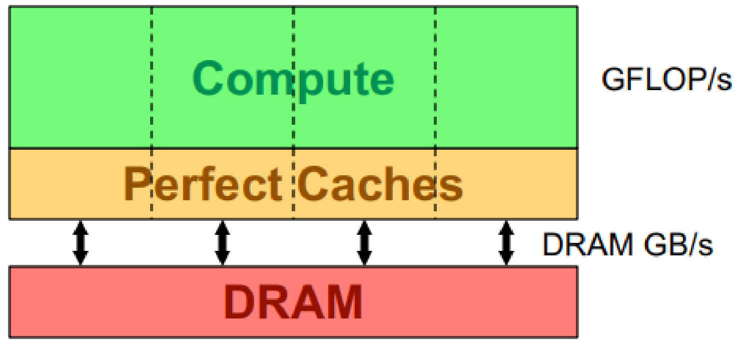
[Krishna, nvidia]

What limits performance of DNNs?

- Two primary categories: **Compute-bound and Memory-bound**
  - Compute-bound: Performance is bottlenecked by the computation (limited by FLOPs of the hardware)
  - Memory-bound: Performance is bottlenecked by memory bandwidth (e.g., limited by DRAM bandwidth in GB/s)
- How to determine whether a DNN operator is compute-bound or memory-bound?
  - Calculate the arithmetic intensity (FLOPs/B): number of floating-point operations (FLOP) performed per byte of memory transfer
    - Arithmetic intensity depends on which memory is considered (DRAM or SRAM)
    - For Neural networks, typically DRAM should be considered first
    - Higher arithmetic intensity → more compute heavy
  - Compare the arithmetic intensity with memory bandwidth
    - Ops/byte ratio: $\frac{BW_{math}(FLOPS)}{BW_{mem}(Bytes/s)}$ - ratio of compute and memory throughput
    - AI > Ops/byte -> compute-bound
    - AI < Ops/byte -> memory-bound
- Hardware and software should focus optimizations on the right problem!

Roofline model

- **Roofline model** is a simple throughput-oriented performance model
- DRAM roofline assumes no bandwidth/latency bottlenecks on L1/L2 caches
- Any given code loop will perform:
  - Computation (FLOPs)
  - Communication (moving data to/from DRAM)
- With perfect overlap of communication and computation
  - Run time is determined by which ever is greater
  - Overall attainable throughput is determined by:
    - AI (Arithmetic Intensity) = FLOPs / Bytes (considering DRAM)



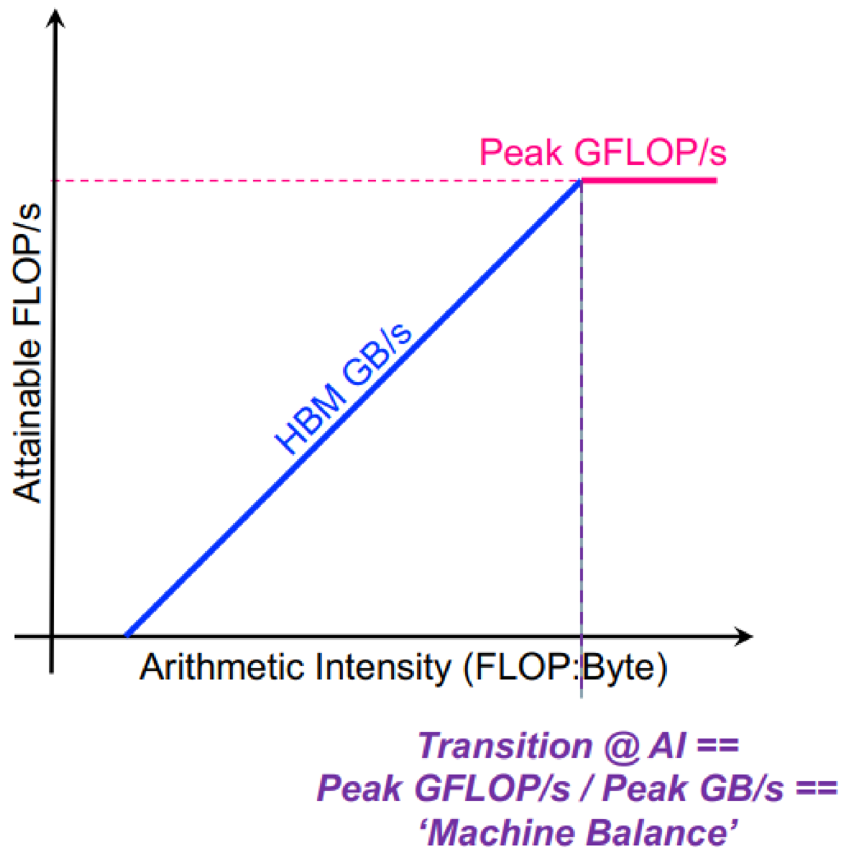
$$\text{Time} = \max\begin{cases}\text{\#FLOPs / Peak GFLOP/s}\\ \text{\#Bytes / Peak GB/s}\end{cases}$$

$$\textbf{GFLOP/s} = \min\begin{cases}\textbf{Peak GFLOP/s}\\ \textbf{AI * Peak GB/s}\end{cases}$$

DRAM Roofline

$$\text{GFLOP/s} = \min\begin{cases}\text{Peak GFLOP/s}\\ \text{AI} * \text{Peak GB/s}\end{cases}$$

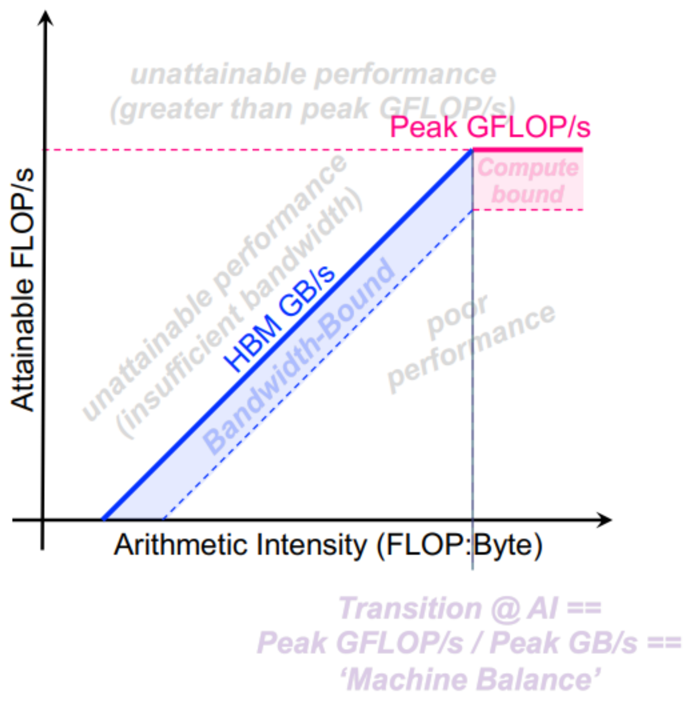
- Plot roofline bound using Arithmetic Intensity as the x-axis
- Log-log scale makes it easy to doodle, extrapolate performance, etc.



DRAM Roofline

$$\text{GFLOP/s} = \min\begin{cases}\text{Peak GFLOP/s}\\ \text{AI} * \text{Peak GB/s}\end{cases}$$

- Roofline model categories the performance into 5 regions
  - Unattainable performance due to compute
  - Unattainable performance due to memory
  - Memory-bound
  - Compute-bound
  - Poor performance (can potentially be optimized)
- Improving Arithmetic Intensity
  - Increase cache/SRAM size
  - Increase reuse
    - E,g,, increase batch size



Some examples

Examples of neural network operations with their arithmetic intensities. Limiters assume FP16 data and a NVIDIA Tesla V100 GPU

Operation	Arithmetic Intensity	Usually limited by
Linear (FC) layer (4096 outputs, 2014 inputs, batch size 512)	315 FLOPS/B	Arithmetic
Linear (FC) layer (4096 outputs, 2014 inputs, batch size 1)	1 FLOPS/B	Memory
Max pooling with 3x3 window and unit stride	2.25 FLOPS/B	Memory
ReLU activation	0.25 FLOPS/B	Memory
Layer normalization	< 10 FLOPS/B	Memory

Review papers for Week 3

- Lecture 5
    1. In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246
    2. Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks. In Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16). IEEE Press, 367–379. https://doi.org/10.1109/ISCA.2016.40
    3. Jeon, W., Ko, G., Lee, J., Lee, H., Ha, D., & Ro, W. W. (2021). Deep learning with GPUs. Advances in Computers, 122, 167-215. https://doi.org/10.1016/bs.adcom.2020.11.003

- Lecture 6
    1. Samajdar, Ananda, et al. "Scale-sim: Systolic cnn accelerator simulator." *arXiv preprint arXiv:1811.02883* (2018).
    2. J. Lee, C. Kim, S. Kang, D. Shin, S. Kim and H. -J. Yoo, "UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173-185, Jan. 2019, doi: 10.1109/JSSC.2018.2865489
    3. Y. Chen *et al.*, "DaDianNao: A Machine-Learning Supercomputer," *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, Cambridge, UK, 2014, pp. 609-622, doi: 10.1109/MICRO.2014.58.
- Speaker assignments posted on Piazza.

Logistics

- For paper presentations
  - Upload PDF or PPT slides on Gradescope. Grading will be done there
  - Stick to your time limit (points deducted for too long or too short a presentation)
  - If you do not understand some of the background for the paper, review it!
  - Do NOT just summarize the paper, critique it.
- Remainder of the Pytorch + Colab tutorial on Wednesday.
- Expect Project 1 announced end of this week or early next week.