Georgia Tech School of Electrical and Computer Engineering
College of Engineering

SYNERGY

http://synergy.ece.gatech.edu

MAERI-FPGA: Enabling HW Design Space Exploration on Real FPGA Hardware Platform

Tushar Krishna
Associate Professor
School of ECE
Georgia Institute of Technology

Email: tushar@ece.gatech.edu

ICS 2022
Tutorial

June 27, 2022

Presenters

Tushar Krishna

Associate Professor
Georgia Tech

Jianming Tong

PhD Student
Georgia Tech

Other Contributors

- Yangyu Chen
- Yue Pan
- Abhimanyu Bambhaniya
- Taekyung Heo
- Hyoukjun Kwon

Acknowledgment: Some of the work done as part of ARIAA Co-Design Center (Georgia Tech, PNNL, Sandia National Labs)

Schedule (EST)

Time slot	Topic	
14:00 to 14:30	Introduction to DNN Accelerators	Tushar
14:30 – 14:40	Break	
14:40: 15:10	MAERI2.0 Architecture and Tool Flow	Jianming
15:10 to 15:30	Demo on FPGA	Jianming

Brief Q/A at the end of each talk.

Please feel free to interrupt and ask questions or use chat

Attention: Tutorial is being recorded!

https://maeri-project.github.io/tutorials/ics-2022

Deep Learning Applications

“AI is the new electricity” – Andrew Ng

Object Detection

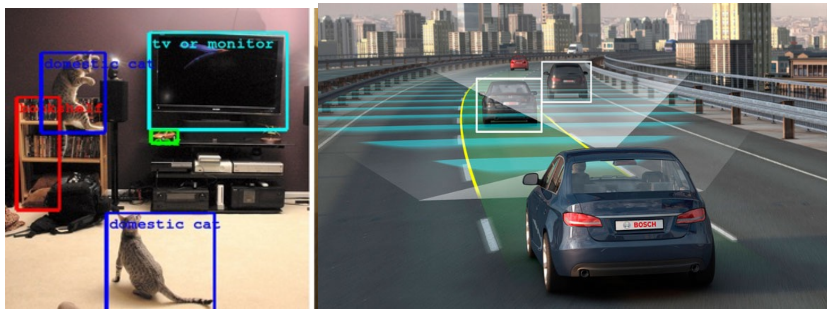
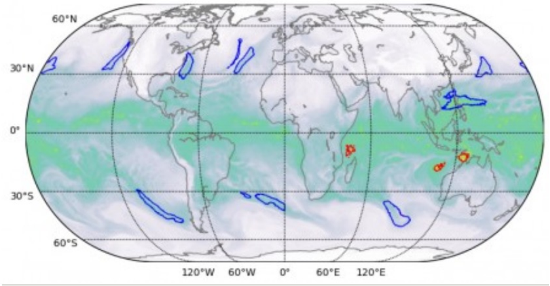
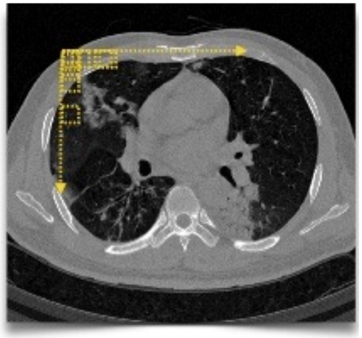


Image Segmentation

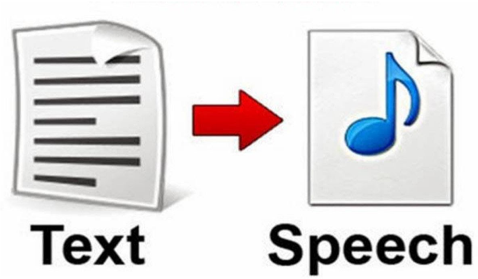


Medical Imaging



Speech Recognition

Text to Speech

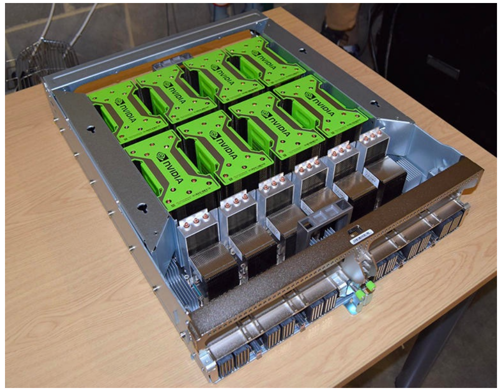


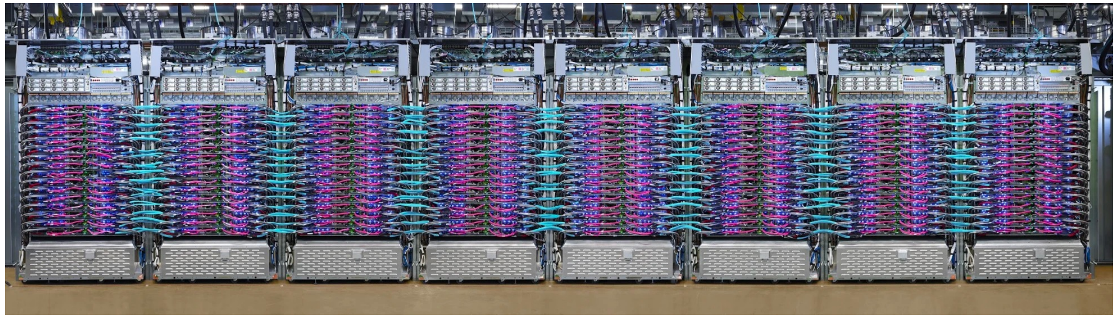
Recommendations

Games

Computation Platforms in Deep Learning

Training → Inference





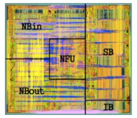
CPU / GPU / TPU clusters

ARM Trillum

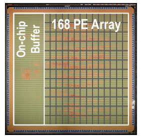
NVDLA

Apple Neural Engine

CambriconX



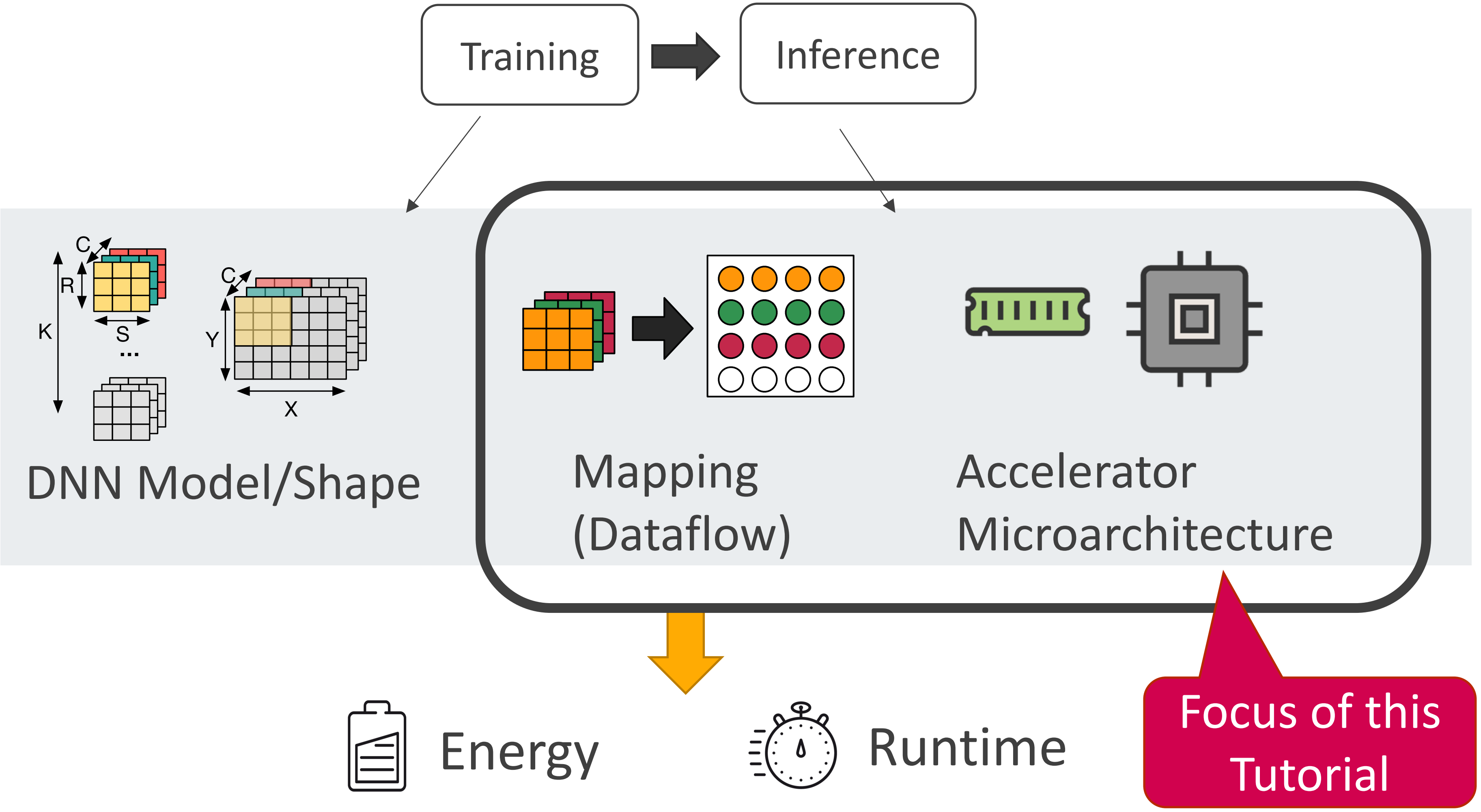
ShiDianNao



Eyeriss

Inference Accelerators

Challenges in Design and Deployment



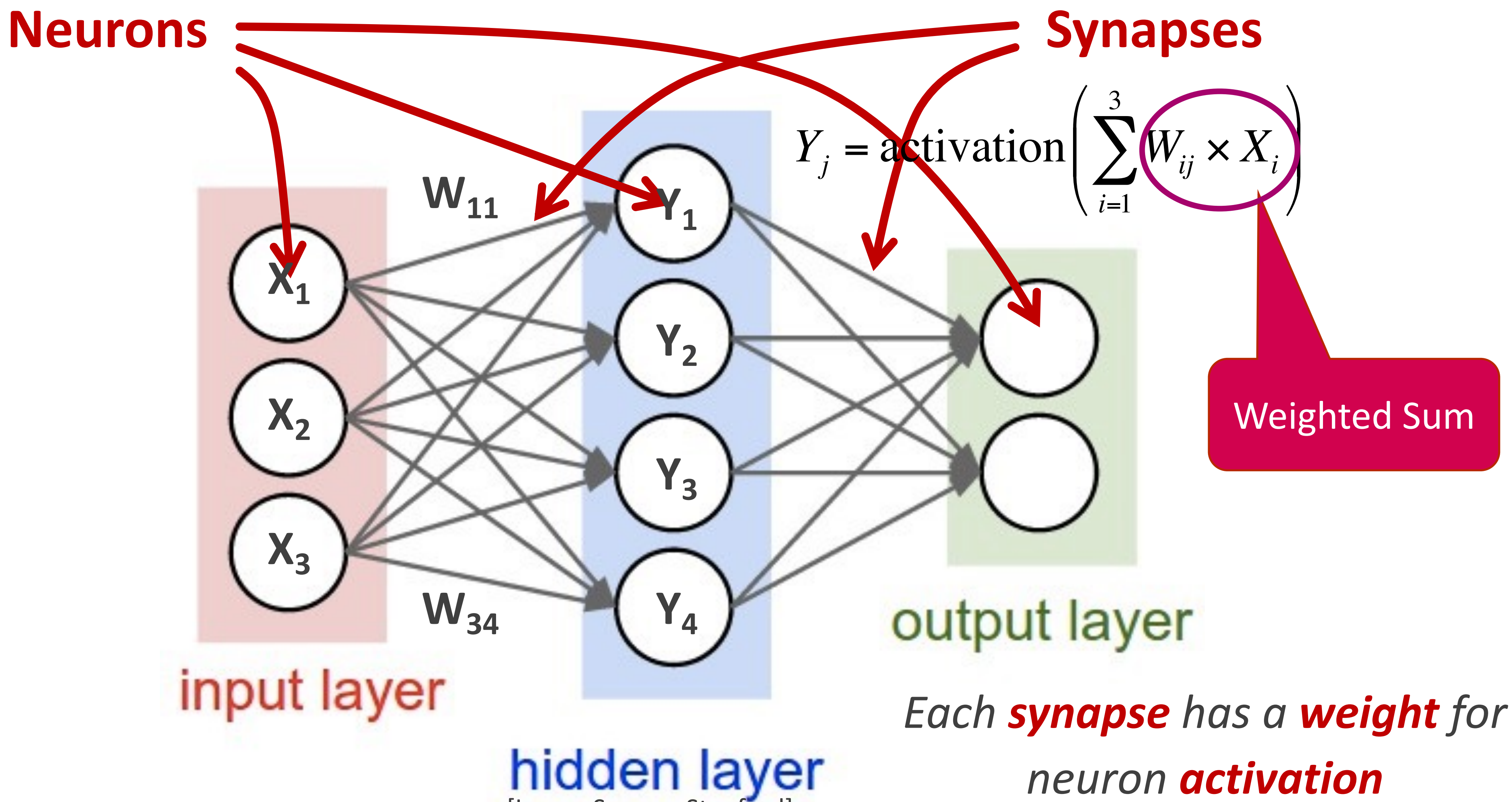
Outline

- Background on DNNs
- DNN Accelerators
- Dataflow and Mapping
- Flexibility

Outline

- Background on DNNs
- DNN Accelerators
- Dataflow and Mapping
- Flexibility

What is a Deep Neural Network?



$$Y_j = \text{activation}\left(\sum_{i=1}^{3} W_{ij} \times X_i\right)$$

[Image Source: Stanford]

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

Modern Deep Learning Landscape

Model Creation ⇄ Training → Inference

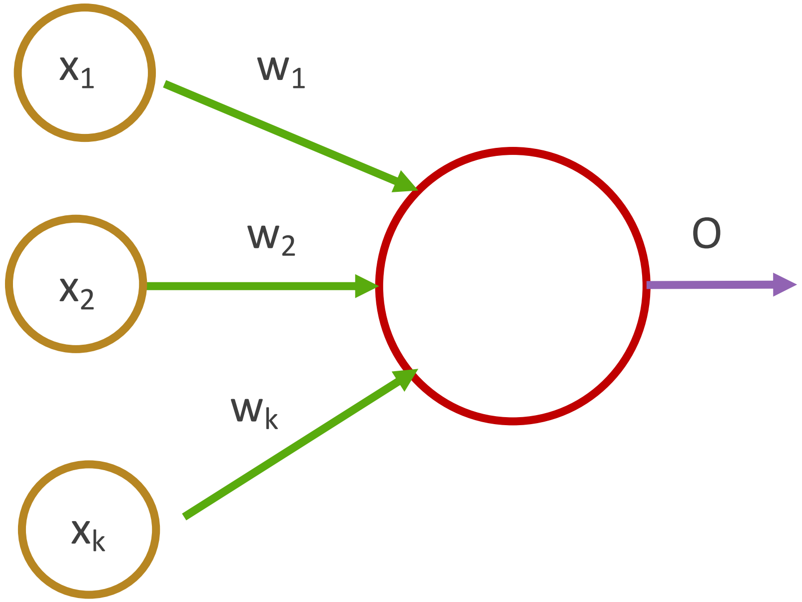
Convolutional Layers (Feature Extraction)

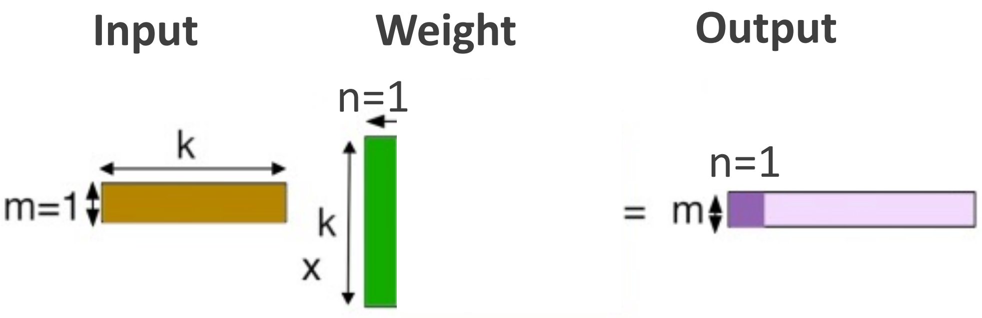
Summarize features

Conv. Layer → Conv. Layer → … → Conv. Layer → Pool. Layer → FC Layer → **"Georgia Institute of Technology, Atlanta"**

Intermediate features

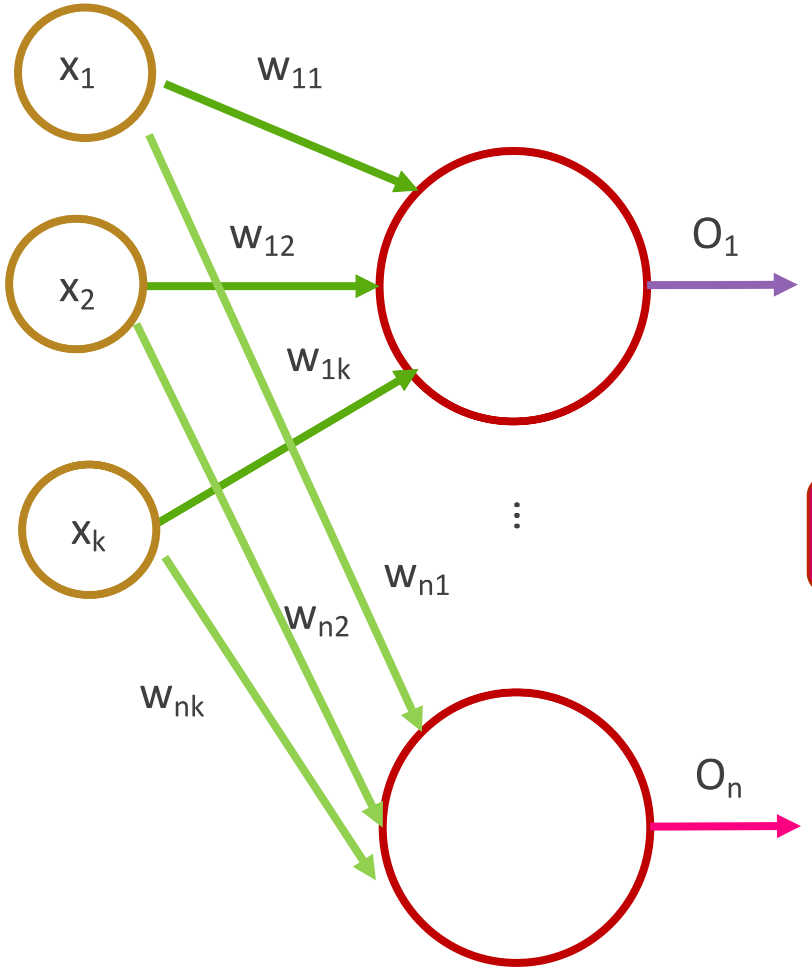
Computations in a DNN → Linear Algebra

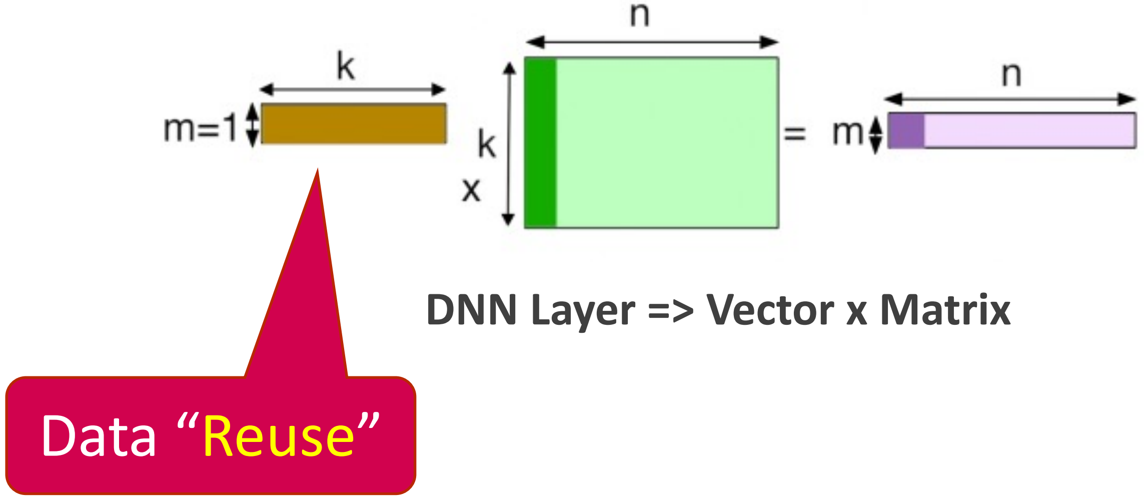




Neuron => Vector x Vector

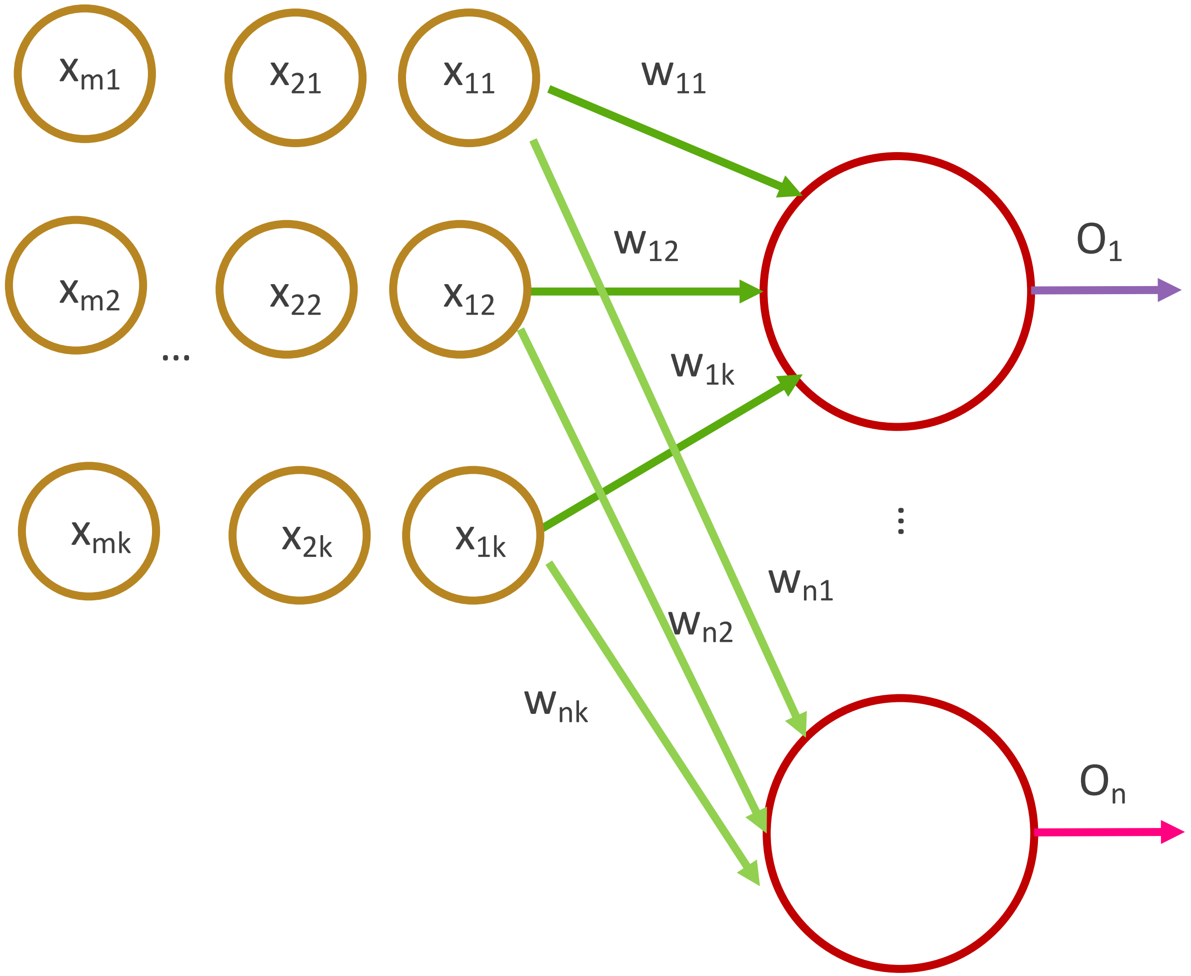
Computations in a DNN → Linear Algebra

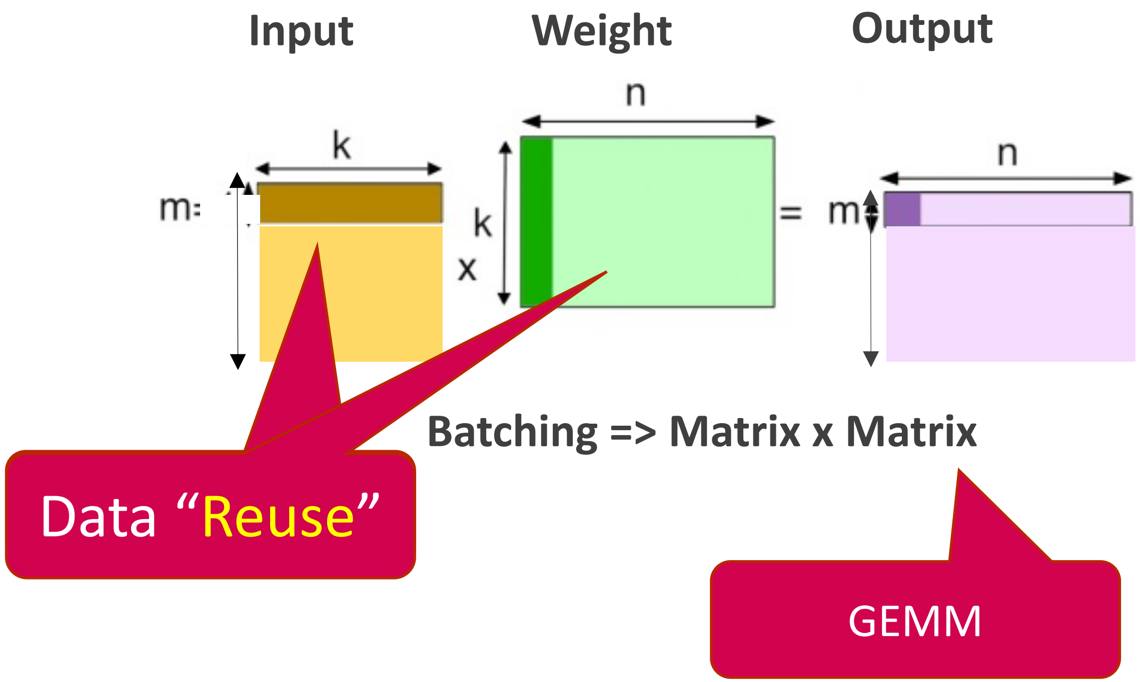




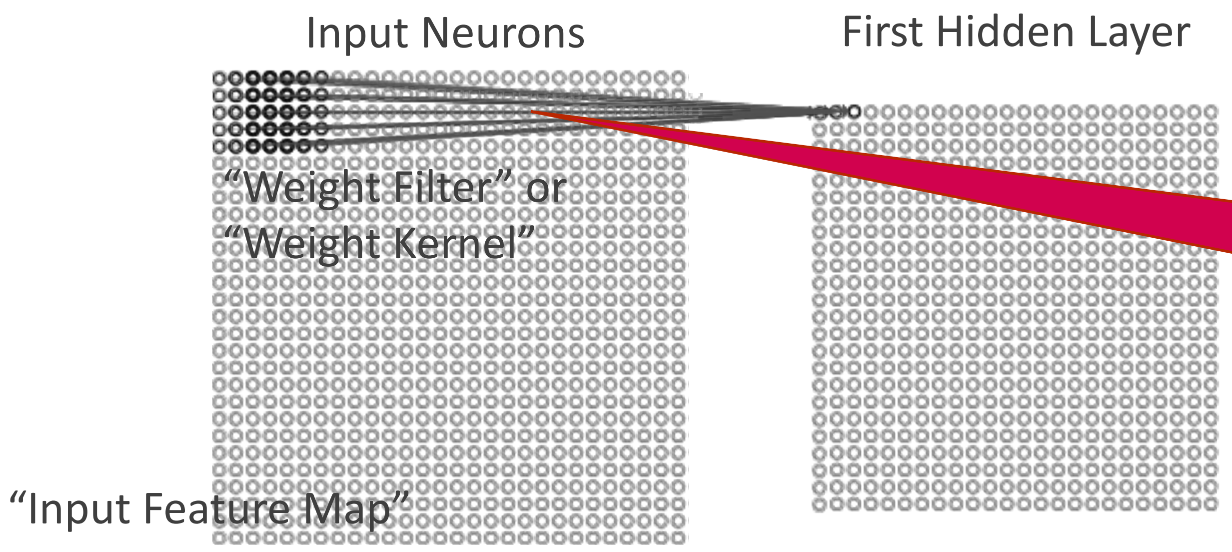
DNN Layer => Vector x Matrix

Computations in a DNN → Linear Algebra





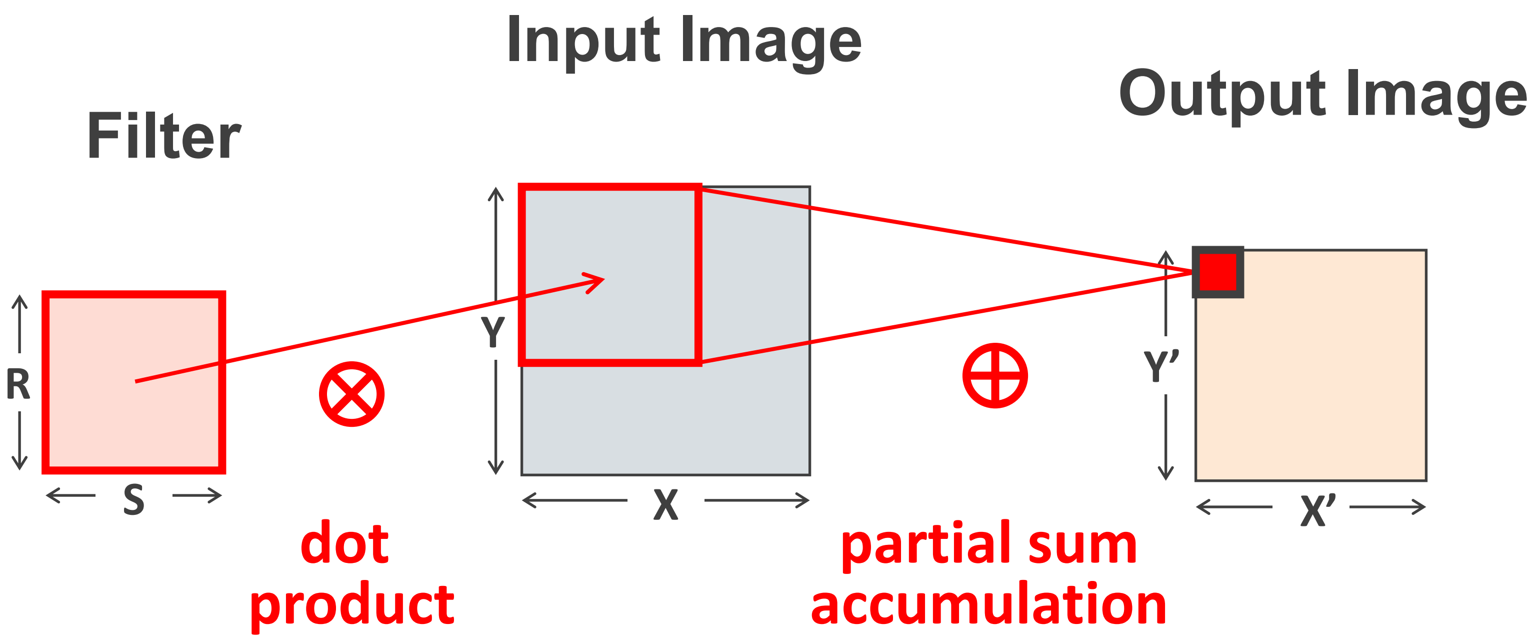
Convolutional Neural Networks



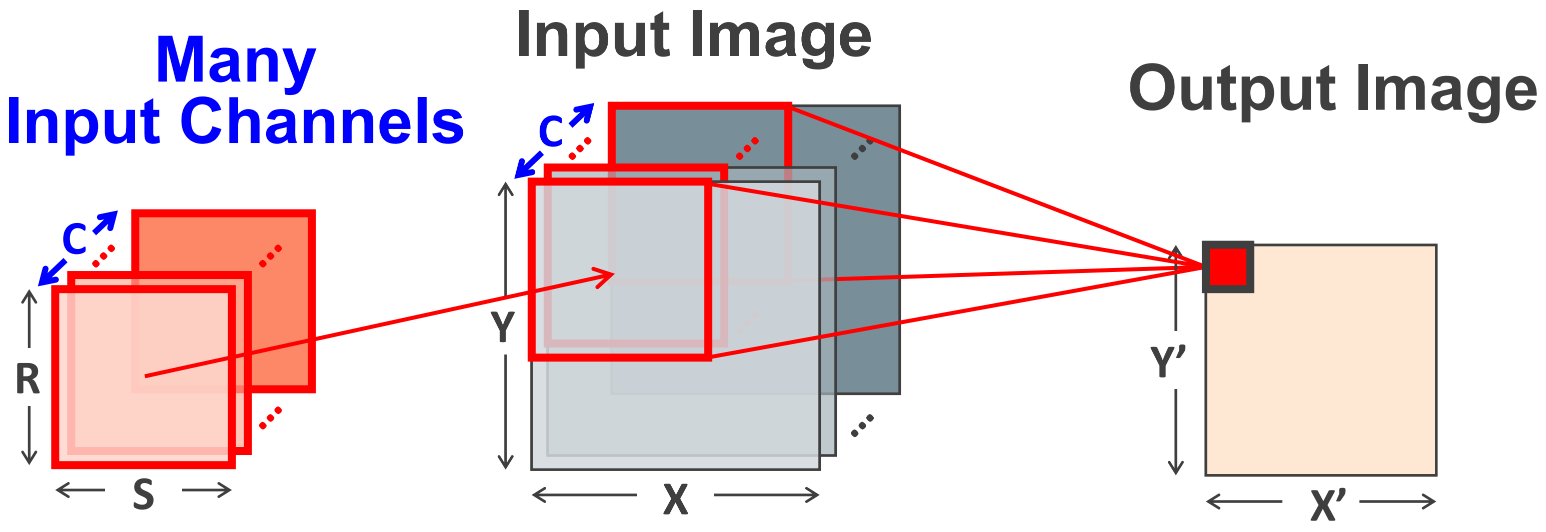
Even more
“Reuse”

Shared Weights:
All neurons use the *same* filter weights

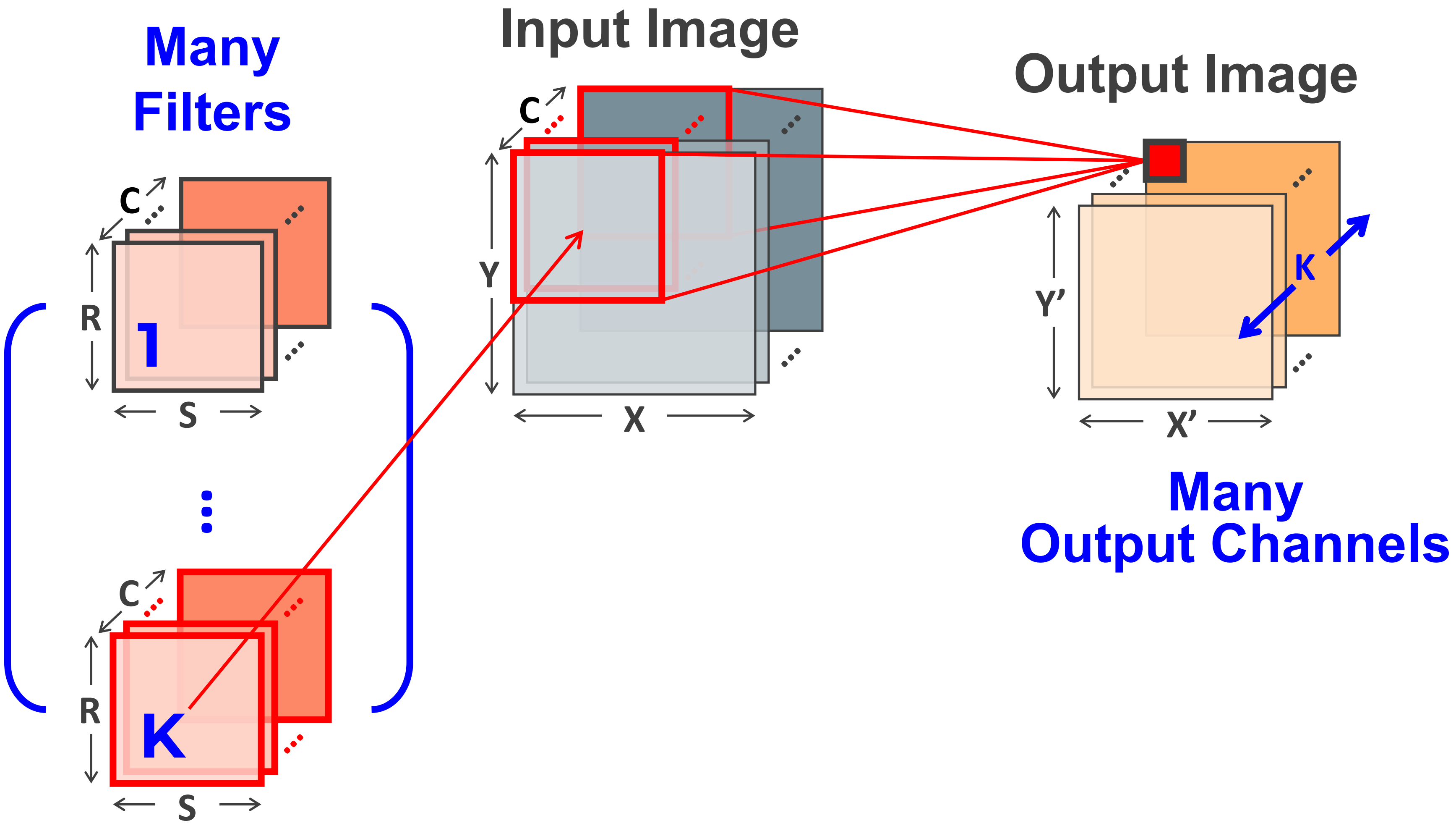
Convolution in CNN



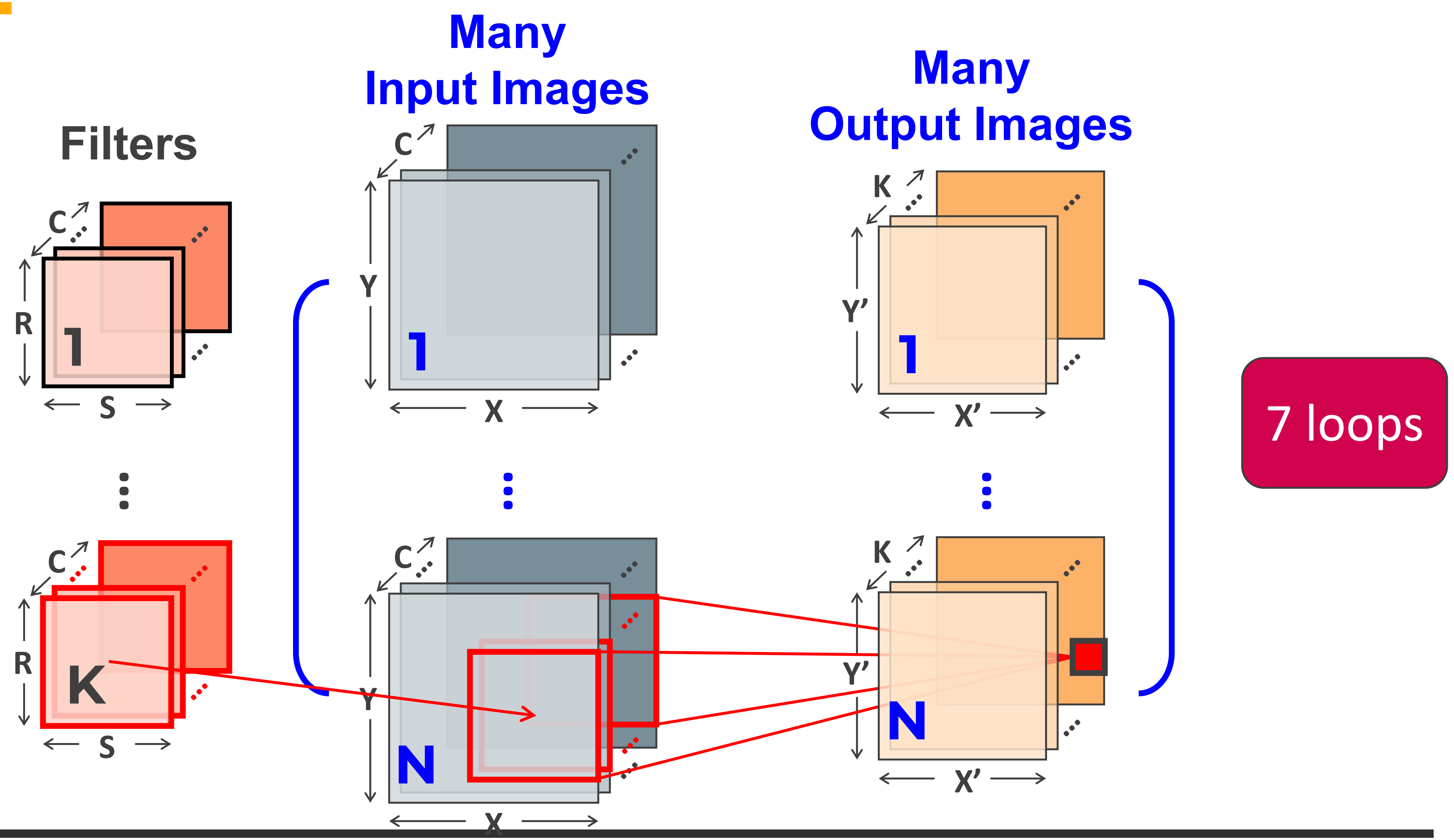
Convolution in CNN



Convolution in CNN



Convolution in CNN



Loop Nest Representation

7th (outermost) loop used during training

```
for(n=0; n<N; n++) { // Input feature maps (IFMaps)
 for(m=0; m<M; m++) { // Weight Filters
  for(c=0; c<C; c++) { // IFMap/Weight Channels
   for(y=0; y<H; y++) {  // Input feature map row
    for(x=0; x<H; x++) {  // Input feature map column
     for(j=0; j<R; j++) {  // Weight filter row
      for(i=0; i<R; i++) {  // Weight filter column
        O[n][m][x][y] += W[m][c][i][j] * I[n][c][y][x]}}}}}}}
```

Challenges with DNN Computations

- **Millions of Parameters (i.e., weights)**
  - Billions of computations → **Need lots of parallel compute**

This makes CPUs inefficient

DNN Topology	Number of Weights
AlexNet (2012)	3.98M
VGGnet-16 (2014)	28.25M
GoogleNet (2015)	6.77M
Resnet-50 (2016)	23M
DLRM (2019)	540M
Megatron (2019)	8.3B

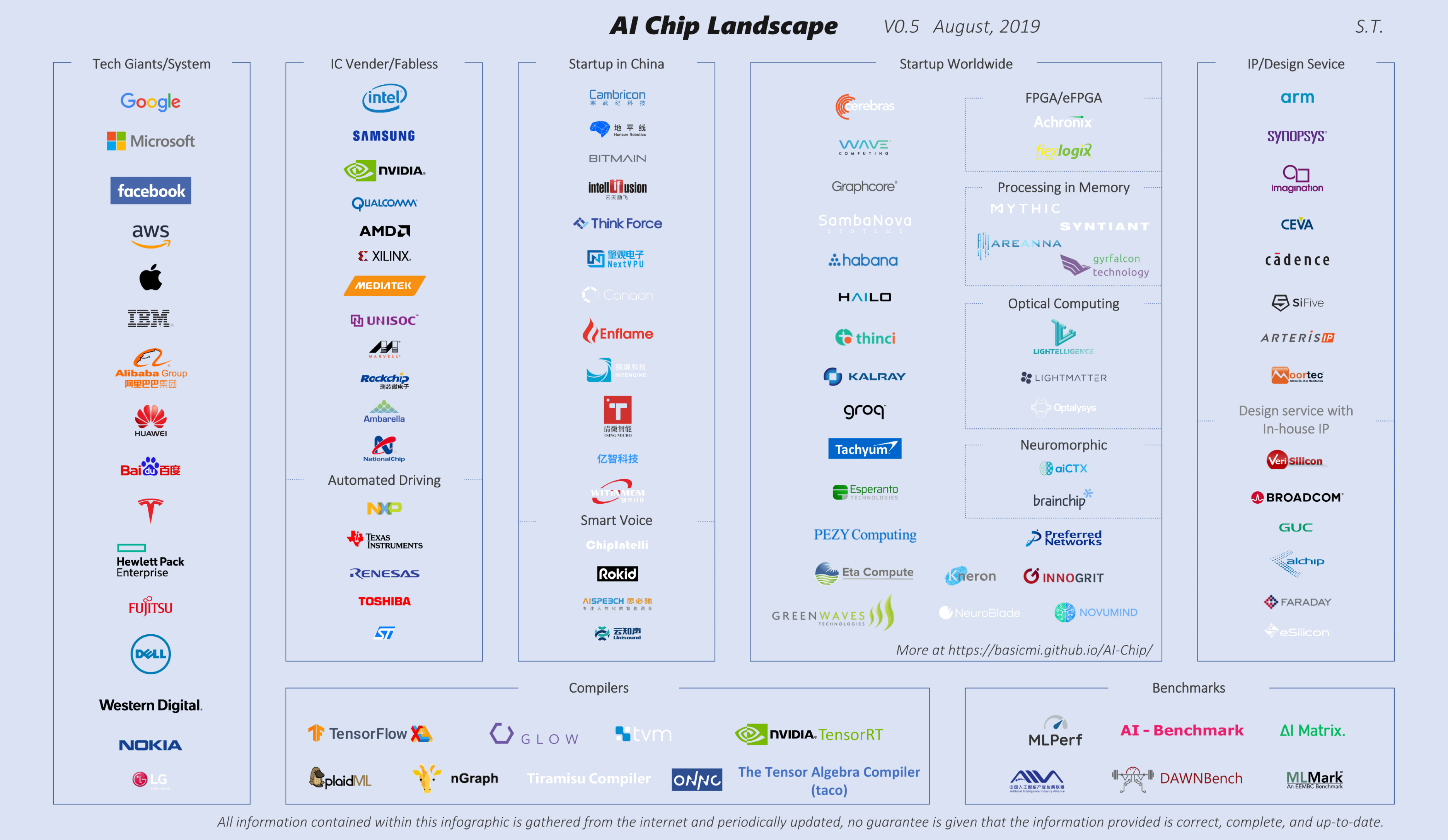
  - Heavy data movement → **Need to reduce energy**

This makes GPUs inefficient

Outline

- Background on DNNs
- DNN Accelerators
- Dataflow and Mapping
- Flexibility

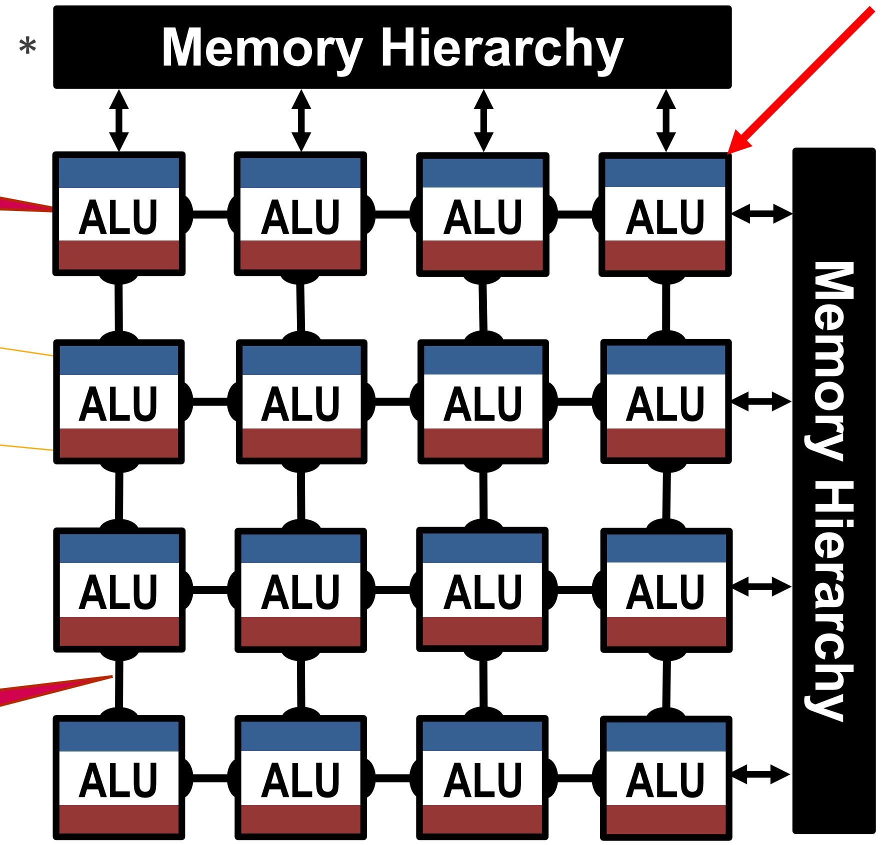
The DL Inference Accelerator Zoo



Spatial (or Dataflow) Accelerators

- **Millions of Parameters (i.e., weights)**
  - Billions of computations

Spread computations across hundreds of ALUs



  - Heavy data movement

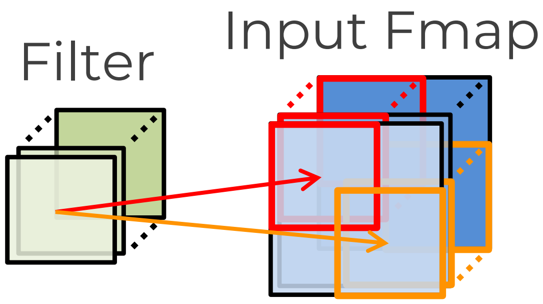
Reuse data within the array via local memories and direct communication

*Y. Chen et. al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," ISCA, 2016.

Types of Algorithmic Data Reuse in DNNs

Convolutional Reuse

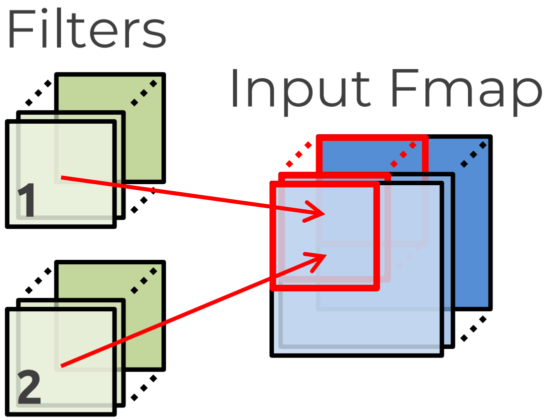
CONV layers only
(sliding window)



Reuse: Activations
Filter weights

Fmap Reuse

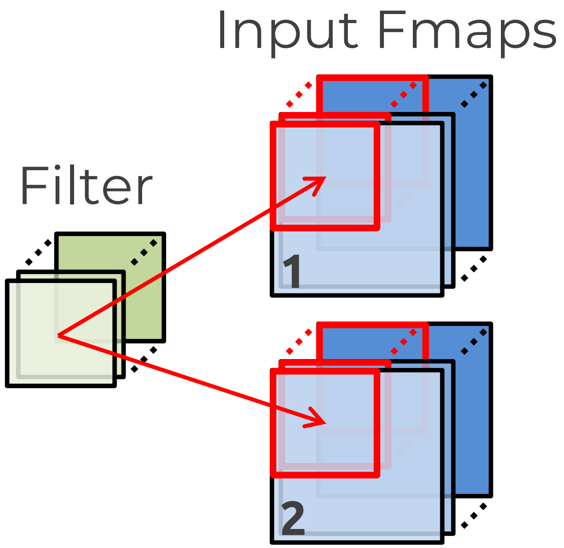
CONV and FC layers



Reuse: Activations

Filter Reuse

CONV and FC layers
(batch size > 1)



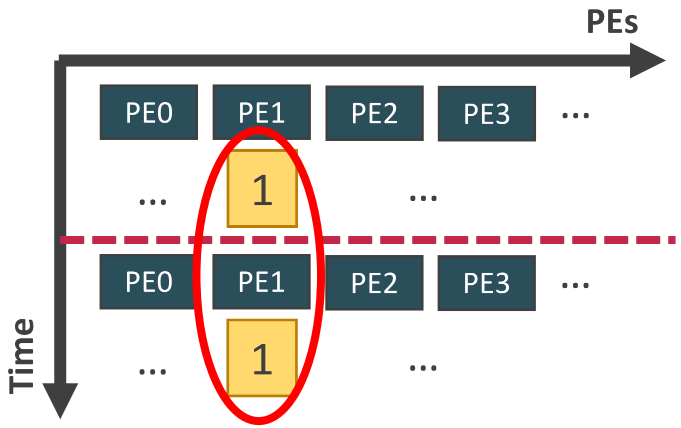
Reuse: Filter weights

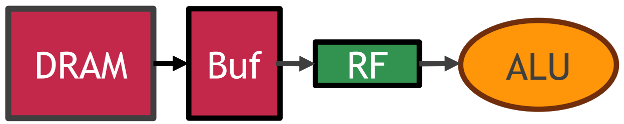
How to exploit reuse?

Slide Acknowledgment: Yu-Hsin Chen, Vivenne Sze, Joel Emer (MIT)

Hardware structures to exploit reuse

Temporal Reuse

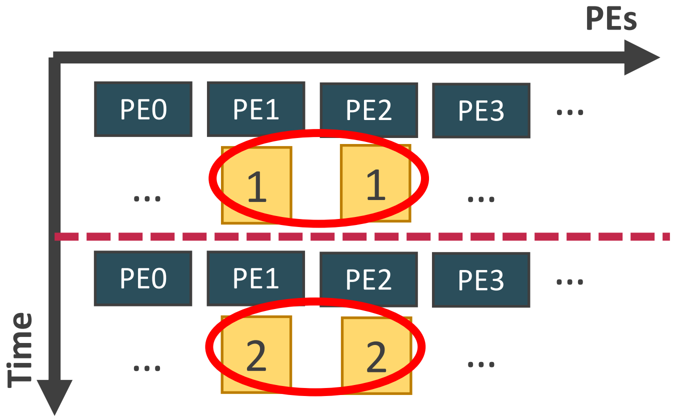


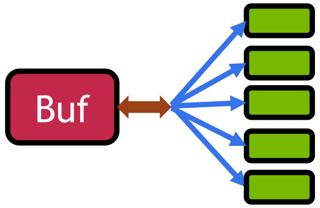


Memory Hierarchy / Staging Buffers

E.g., Custom memory hierarchies in accelerators.

Spatial Reuse

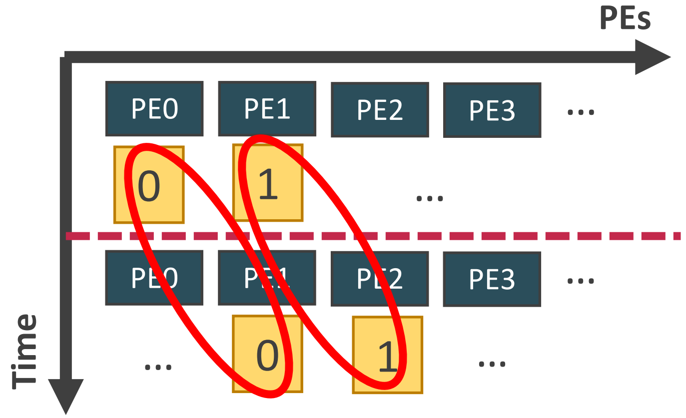


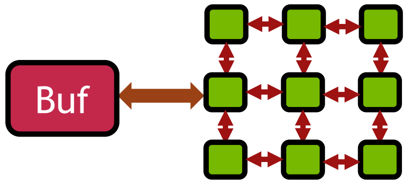


Multicasting-support NoCs

E.g., Hierarchical Bus in Eyeriss (ISCA 2016), Tree in MAERI (ASPLOS 2018)

Spatio-Temporal Reuse



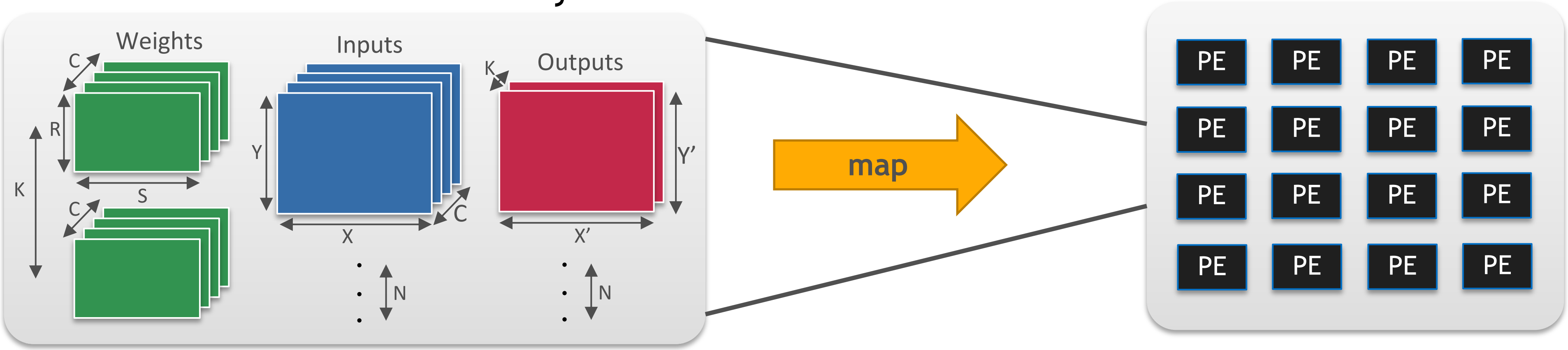


Neighbor-to-Neighbor Connections

E.g., TPU (ISCA 2017), local network in Eyeriss (ISCA 2016)

Mapping and Dataflow

7-dimensional network layer



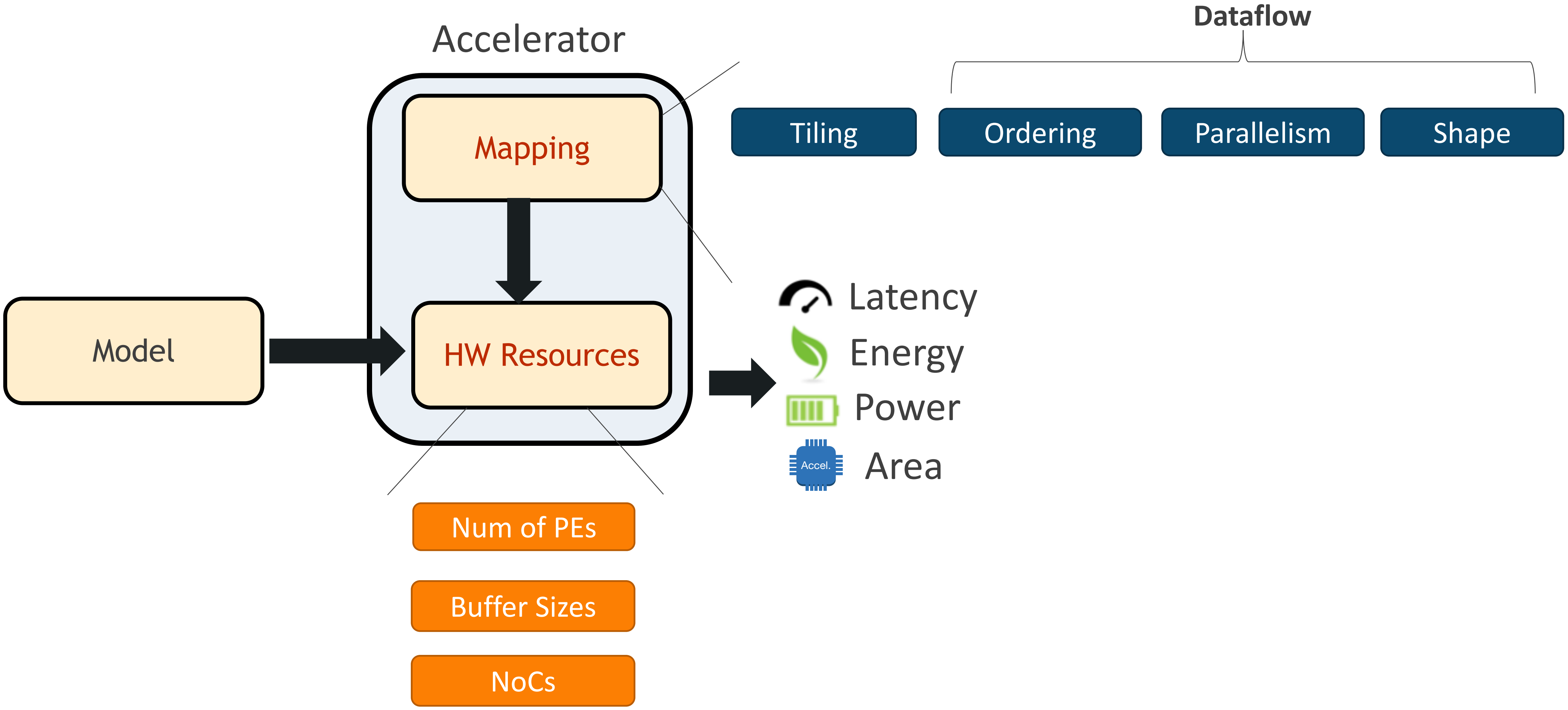
- Number of PEs
- Memory Hierarchy
- Interconnect Bandwidth
- …

7D Computation Space: R * S * X * Y * C * K * N

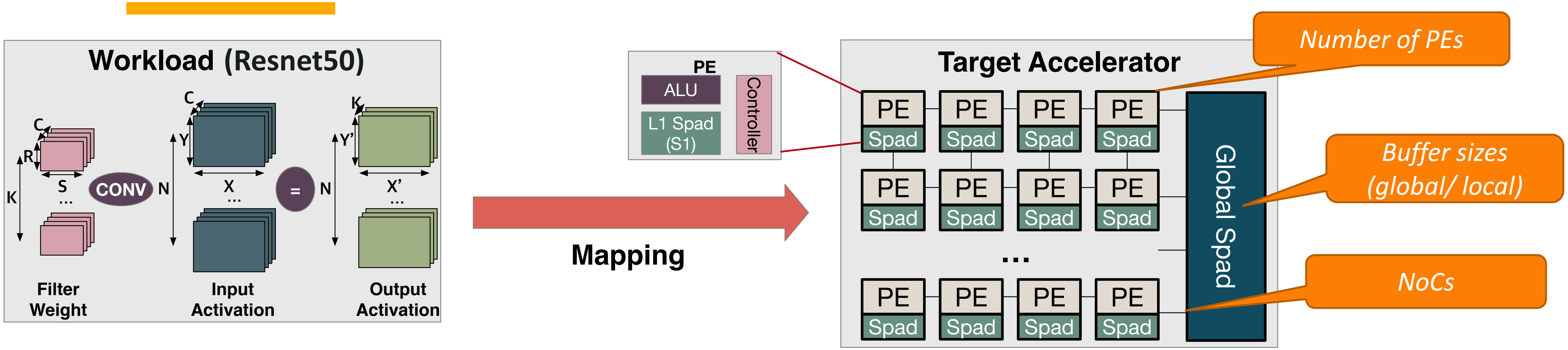
- **Goal of Mapping**: *translate algorithmic data reuse to HW data reuse*
- **Precise Definition of Mapping:** Fine-grained schedule of computations within DNN accelerators
  - **Computation Order** *(slowest tensor dimension often called "stationary")*
  - **Parallelization Strategy** *(which loops to unroll spatially)*
  - **Tiling Strategy** *(number of levels of memory hierarchy)*
  - **Tile Sizes**

Dataflow

Architectural Components of a DNN Accelerator

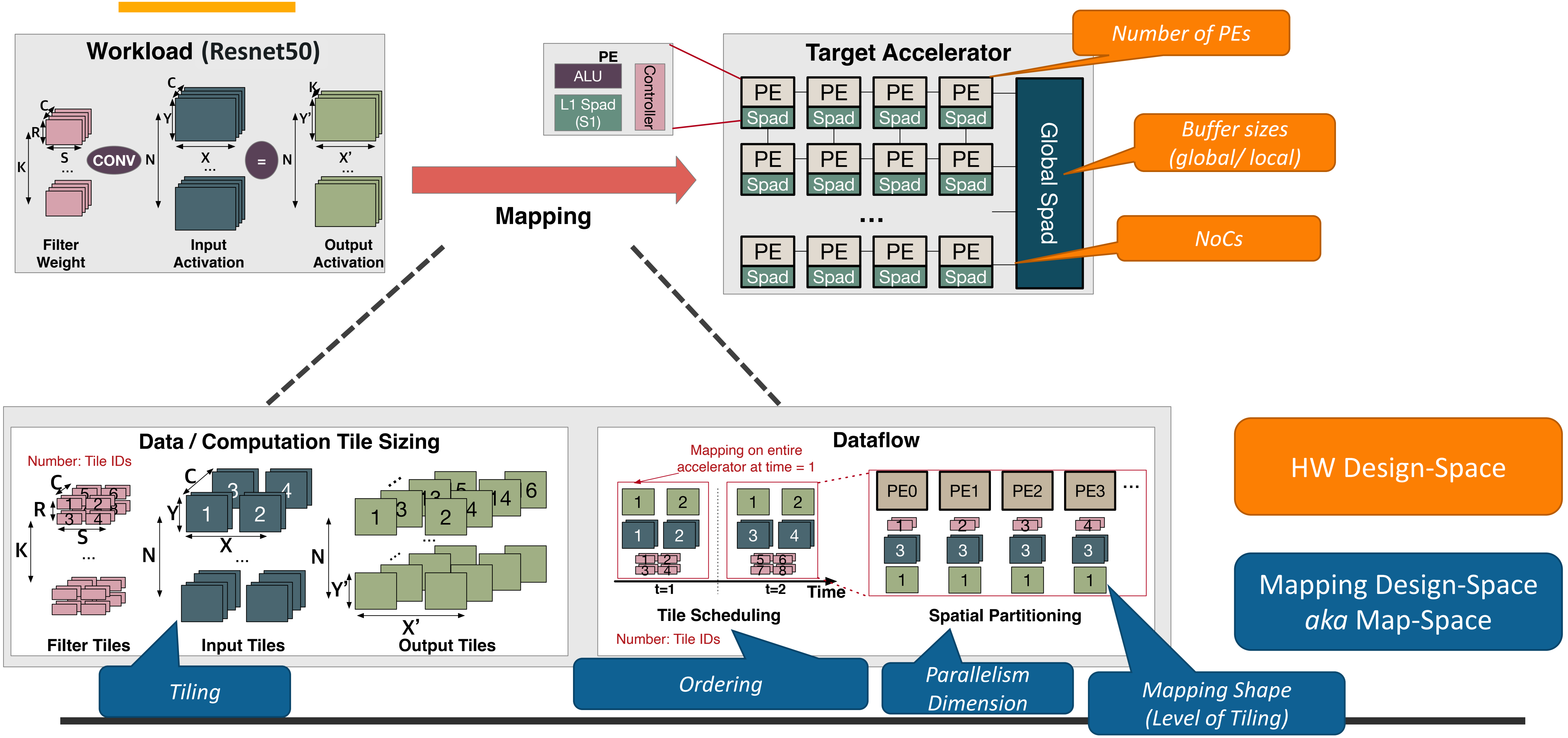


Architectural Components of a DNN Accelerator



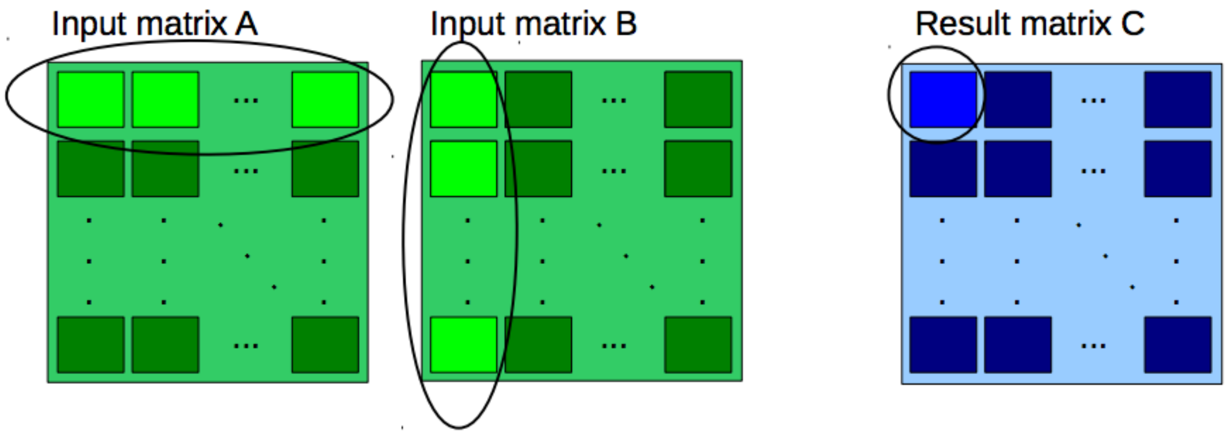
HW Design-Space

Architectural Components of a DNN Accelerator



GEMM vs CONV2D Accelerators

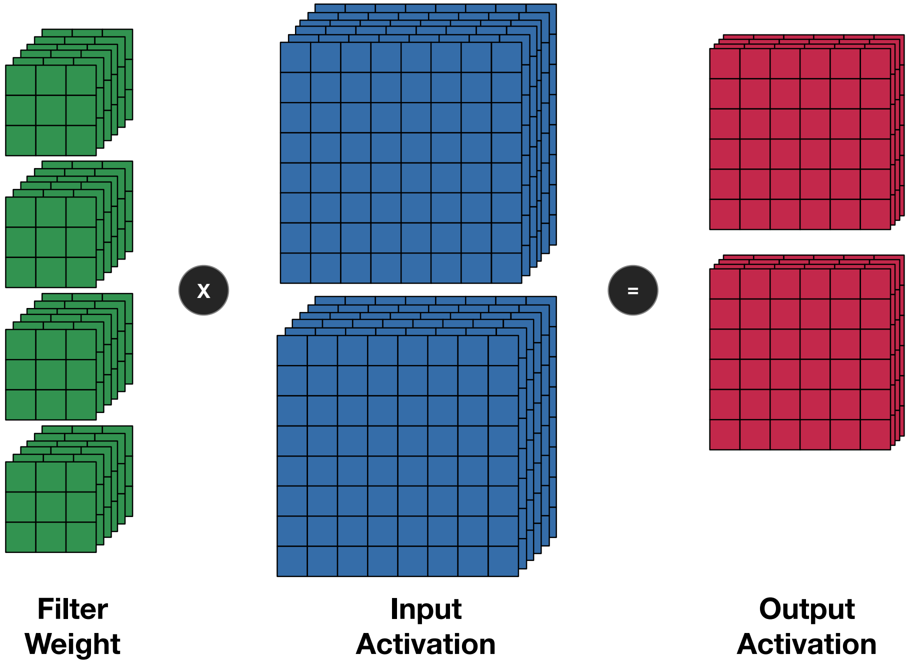
GEMM Operation



3 Loops

- *Less Opportunities for Reuse*
- *More general: any DNN layer (including convolutions) can be lowered to GEMM (e.g, Im2Col)*
- *E.g., NVIDIA Tensor Core, Google TPU*

CONV2D Operation



7 Loops

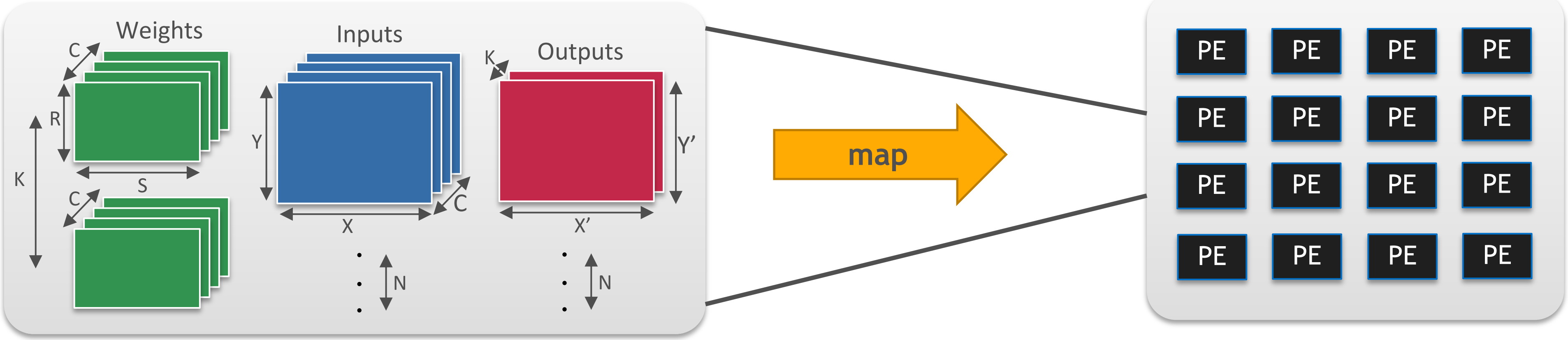
- *More Opportunities for Reuse*
- *Only applicable for convolution layers*
- *E.g., NVDLA, MAERI (this work)*

Outline

- Background on DNNs
- DNN Accelerators
- Dataflow and Mapping
- Flexibility

Dataflow and Mapping

7-dimensional network layer



- **Number of PEs**
- **Memory Hierarchy**
- **Interconnect Bandwidth**
- **…**

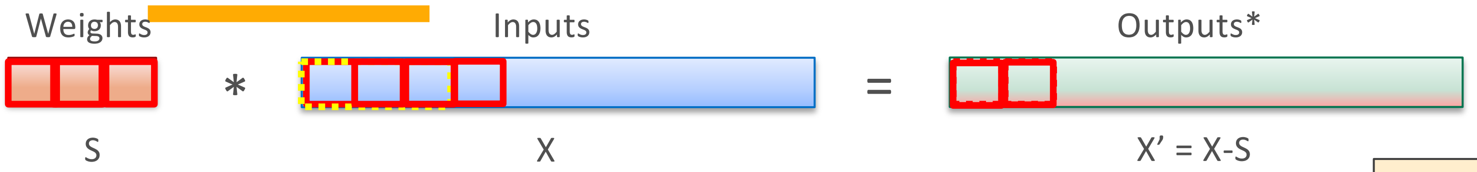
7D Computation Space: R * S * X * Y * C * K * N

- **Goal of Mapping**: *translate algorithmic data reuse to HW data reuse*
- **Precise Definition of Mapping:** Fine-grained schedule of computations within DNN accelerators
  - **Computation Order** *(slowest tensor dimension often called "stationary")*
  - **Parallelization Strategy** *(which loops to unroll spatially)*
  - **Tiling Strategy** *(number of levels of memory hierarchy)*
  - **Tile Sizes**

Dataflow

Impact of Computation Order

"Output Stationary" Dataflow



CONV1D

Computation

```
for(int x = 0; x < X'; x++)
  for(int s = 0; s < S; s++)
Output[x] += Weight[s] * Input[x+s]
```

Data

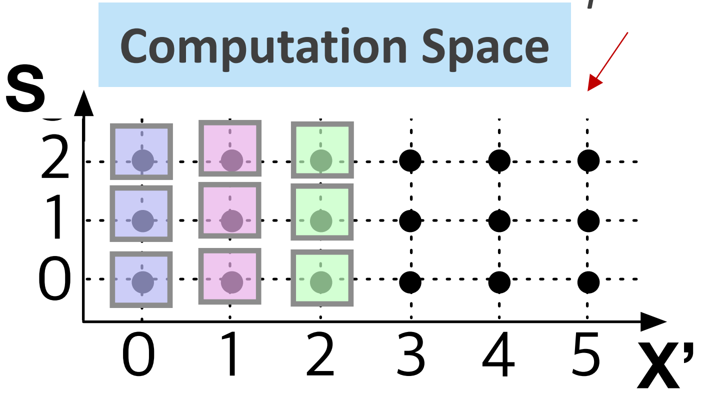
PartialSum[X'][S] needs to access:

- Weight[s]
- Output[x']
- Input[x'+s]

Time = 0

*Suppose we **map** this computation over three PEs*

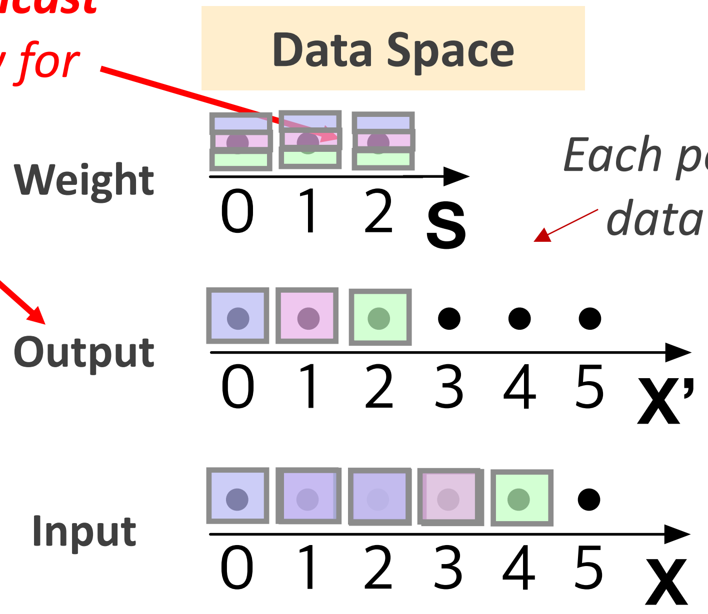
PE2
PE1
PE0



Each point is a partial sum

***Spatial multicast** opportunity for weights*

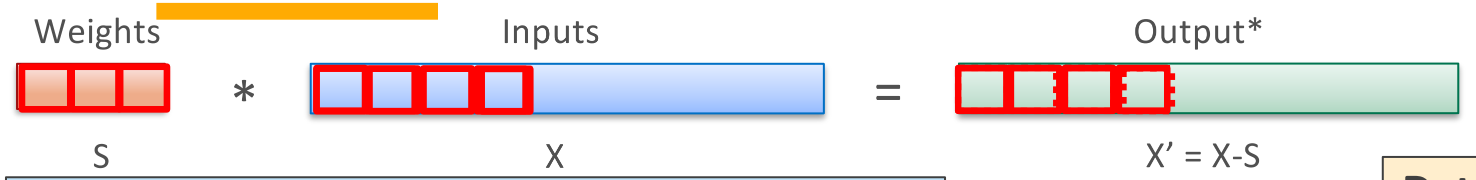
*Output does not change over time => **Temporal reuse** opportunity*



Each point is a data point

Impact of Computation Order

“Weight Stationary” Dataflow



CONV1D

Computation

```
for(int s = 0; s < S; s++)
  for(int x = 0; x < X'; x++)
Output[x] += Weight[s] * Input[x+s]
```

Data

PartialSum[X'][S] needs to access:

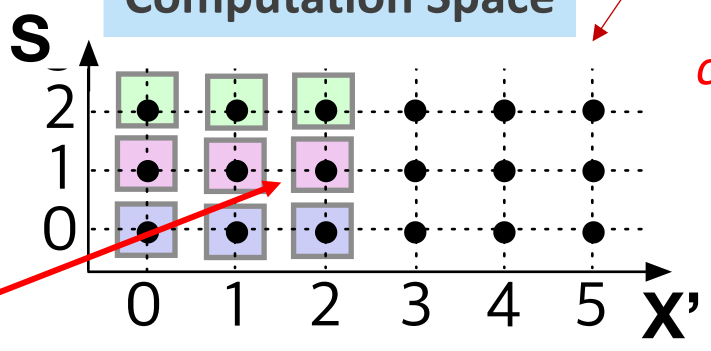
- Weight[s]
- Output[x']
- Input[x'+s]

Time = 0

*Suppose we **map** this computation over three PEs*

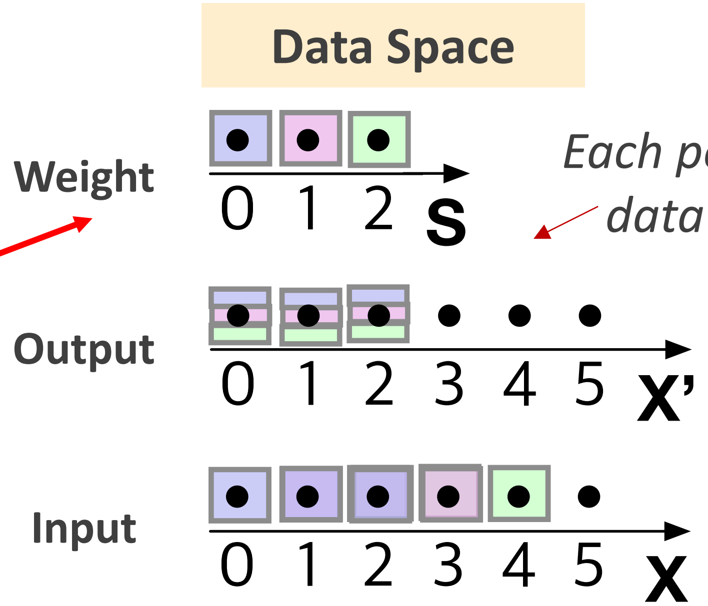
- PE2
- PE1
- PE0

*Need **Spatial reduction** for output*



Each point is a partial sum

*Weight does not change over time => **Temporal reuse** opportunity*



Each point is a data point

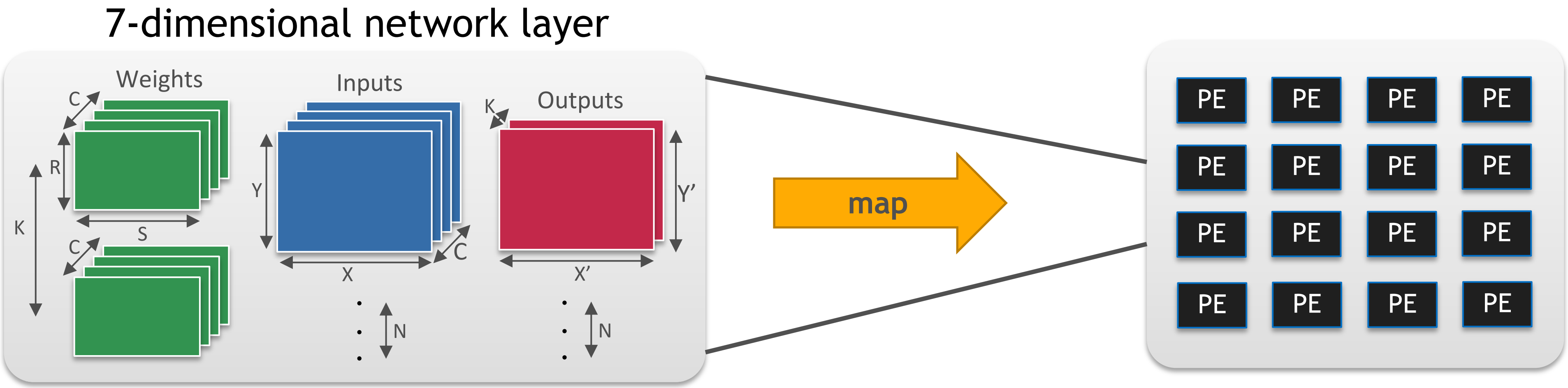
Takeaways: Data Reuse + Hardware Support

- Dataflow exposes data reuse opportunities
- Hardware support is needed to leverage reuse opportunity

Hardware Structure	Per Data Type	Weight Stationary Dataflow Implication	Output Stationary Dataflow Implication
Bandwidth to MAC	Weight Fetch Rate	Every S Cycles	Every Cycle
	Input Fetch Rate	Every Cycle	Every Cycle
	Output Fetch Rate	Every Cycle	Every S Cycles
Local Buffer Sizes for Temporal Reuse	Weight Buffer Size	1	3
	Input Buffer Size	3	3
	Output Buffer Size	3	1
Network-on-Chip for Spatial Reuse	Weight Distribution	Unicast	Spatial Multicast
	Input Distribution	Spatial Multicast	Unicast
	Output Collection	Spatial Reduction	Temporal Reduction

Note: *for full 6D conv,* trillions of *valid dataflow choices → Huge Design Space*

Dataflow and Mapping



- **Number of PEs**
- **Memory Hierarchy**
- **Interconnect Bandwidth**
- **…**

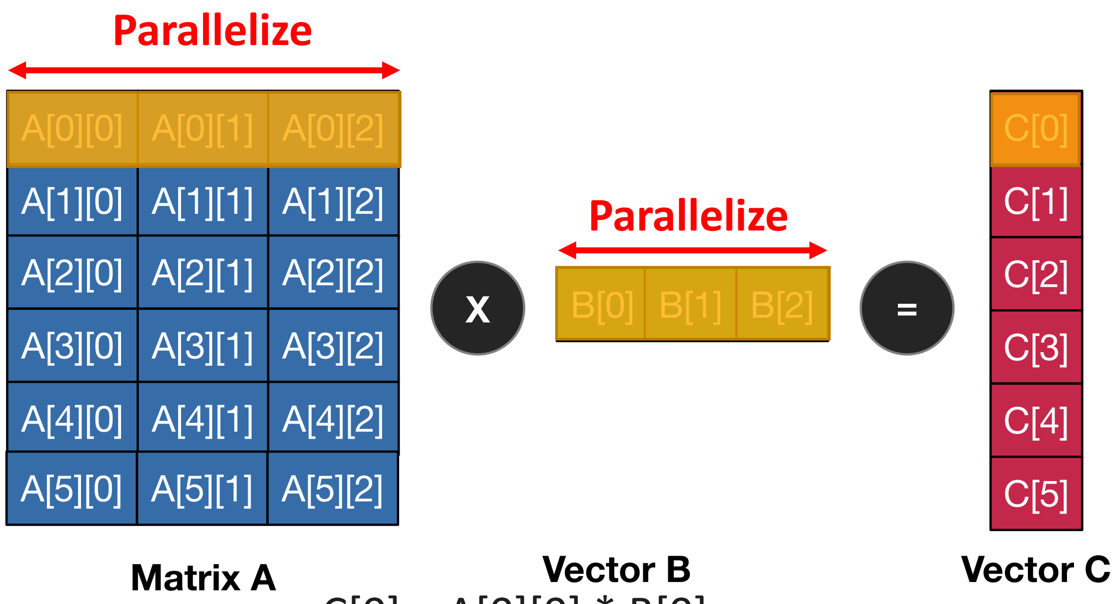
7D Computation Space: R * S * X * Y * C * K * N

- **Goal of Mapping**: *translate algorithmic data reuse to HW data reuse*
- **Precise Definition of Mapping:** Fine-grained schedule of computations within DNN accelerators
  - **Computation Order** *(slowest tensor dimension often called "stationary")*
  - **Parallelization Strategy** *(which loops to unroll spatially)*
  - **Tiling Strategy** *(number of levels of memory hierarchy)*
  - **Tile Sizes**

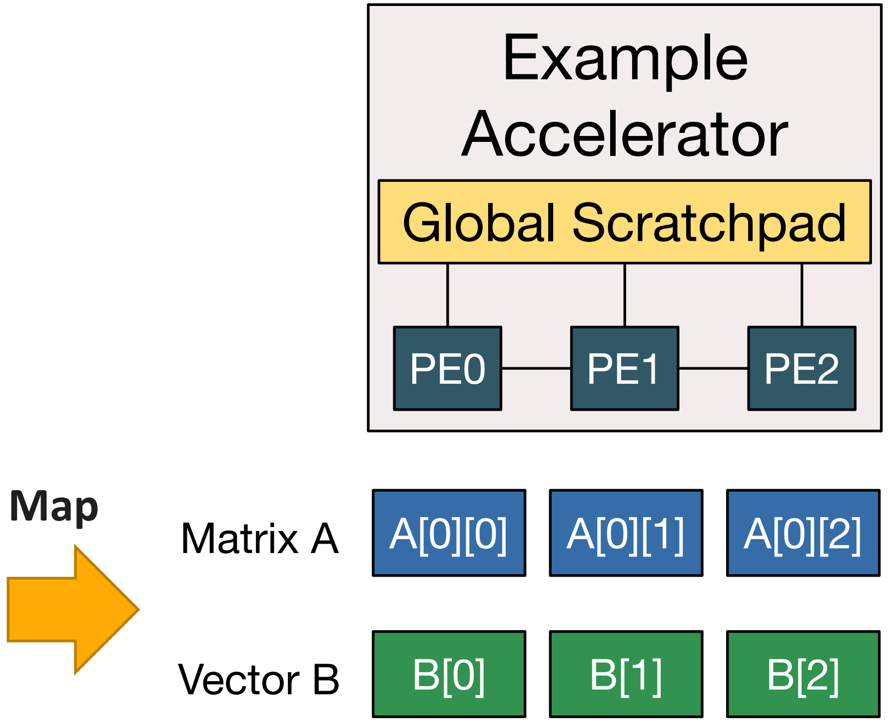
Dataflow

Impact of Parallelization

Example Model A: Matrix-Vector Multiplication
(i.e., Simplified Fully-connected layer)



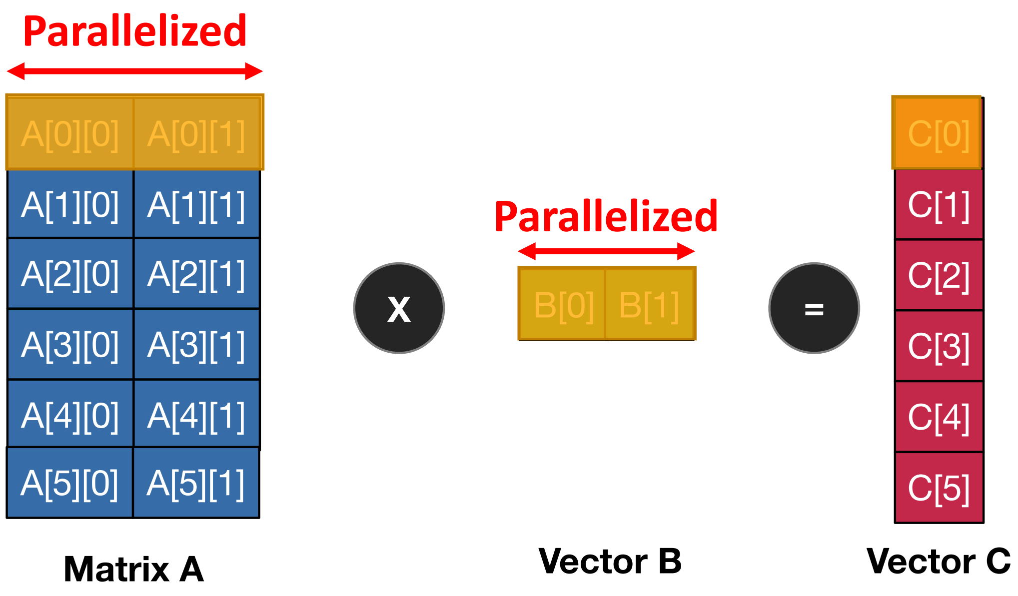
C[0] = A[0][0] * B[0]
+ A[0][1] * B[1]
+ A[0][2] * B[2]



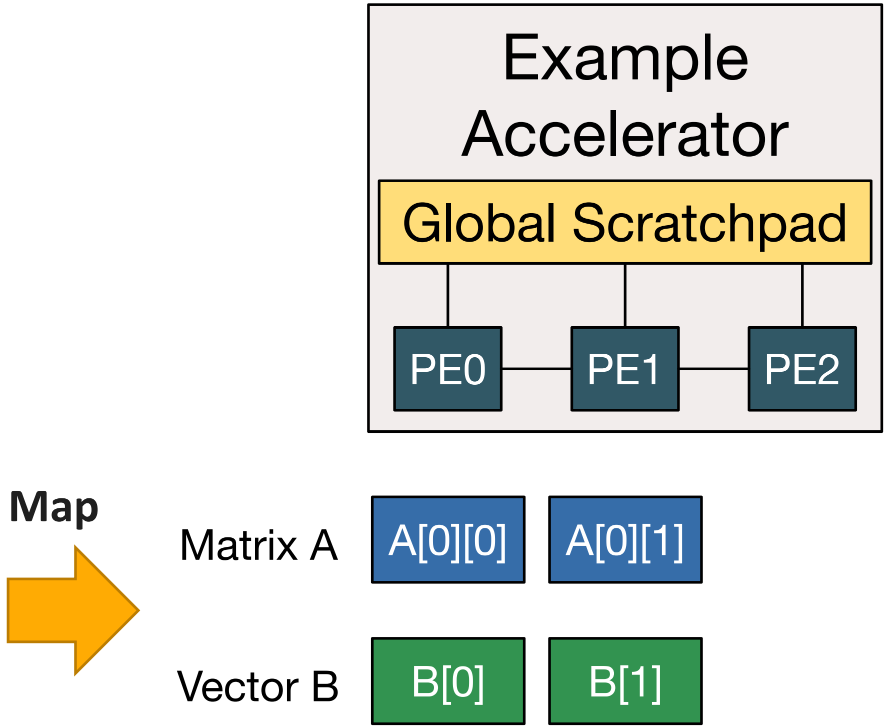
Avg. Utilization: 100%

Impact of Parallelization

Example Model B: Matrix-Vector Multiplication
(i.e., Simplified Fully-connected layer)



C[0] = A[0][0] * B[0]
+ A[0][1] * B[1]

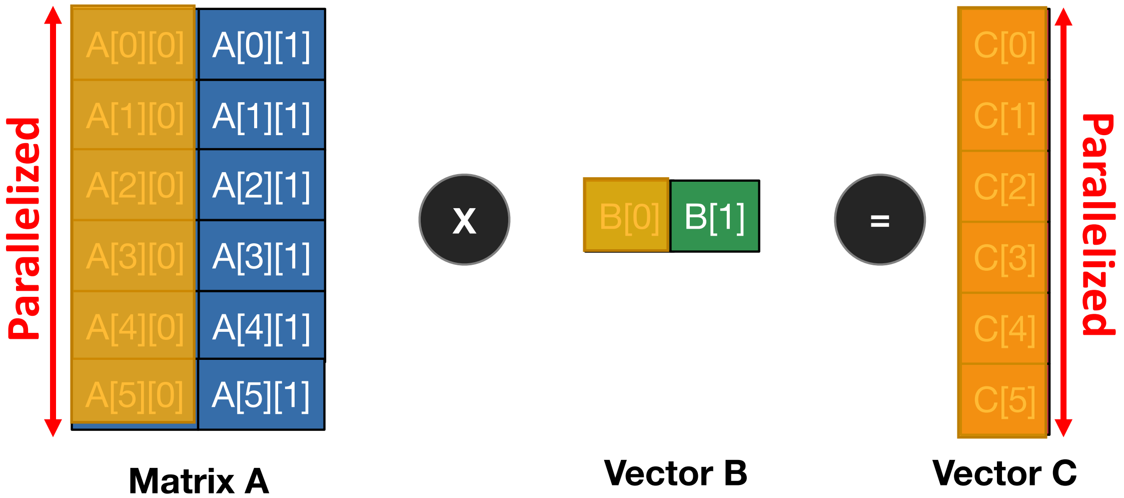


Avg. Utilization: 66%

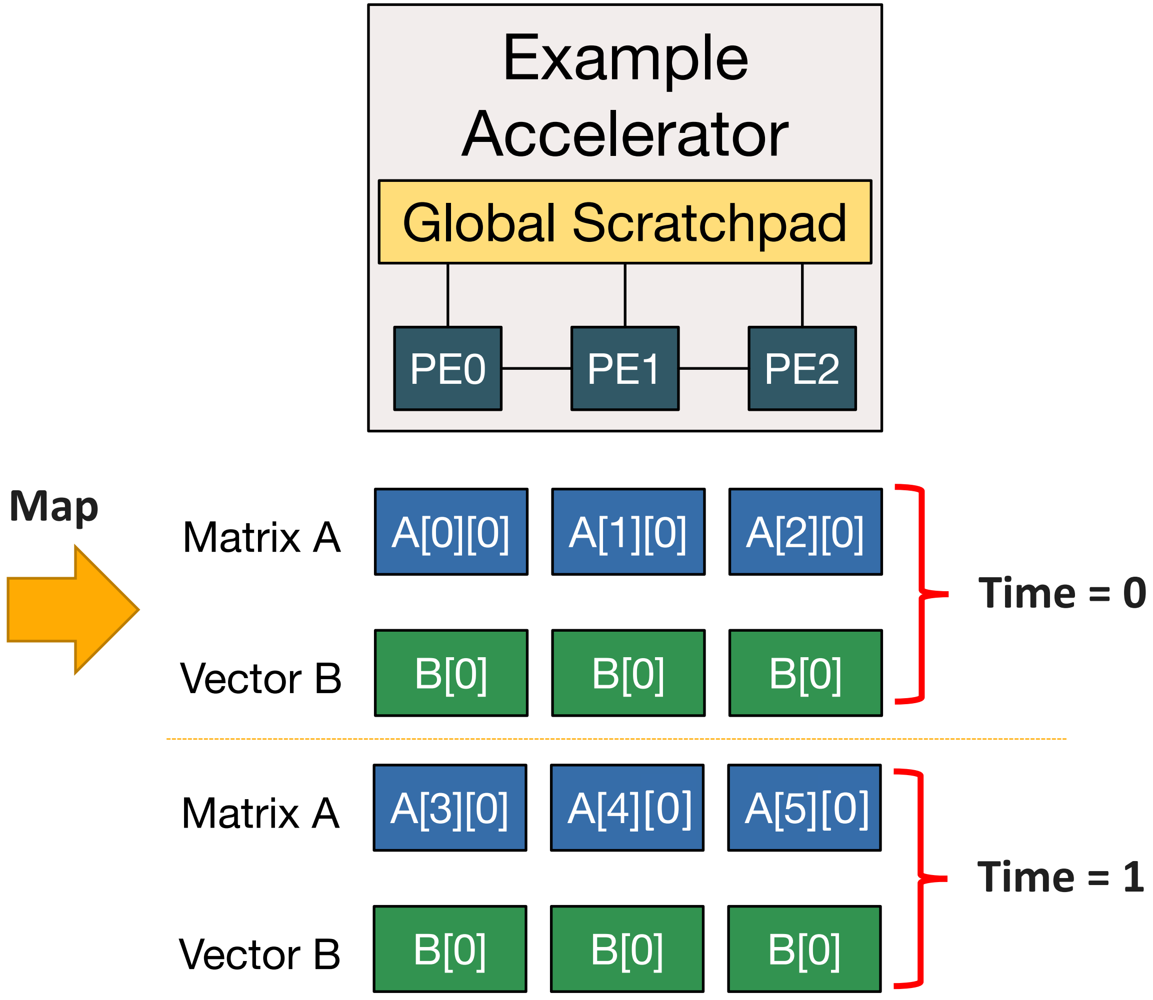
Can we map it in a better way?

Impact of Parallelization

Example Model B: Matrix-Vector Multiplication
(i.e., Simplified Fully-connected layer)



C[0] = A[0][0] * B[0]
+ A[0][1] * B[1]
+ A[0][2] * B[2]



Avg. Utilization: 100%

The more dimensions, the more optimization opportunities

Outline

- Background on DNNs
- DNN Accelerators
- Dataflow and Mapping
- Flexibility

Why do we need *flexible* DNN accelerators?

- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types

<Number of new ML papers in Arxiv>

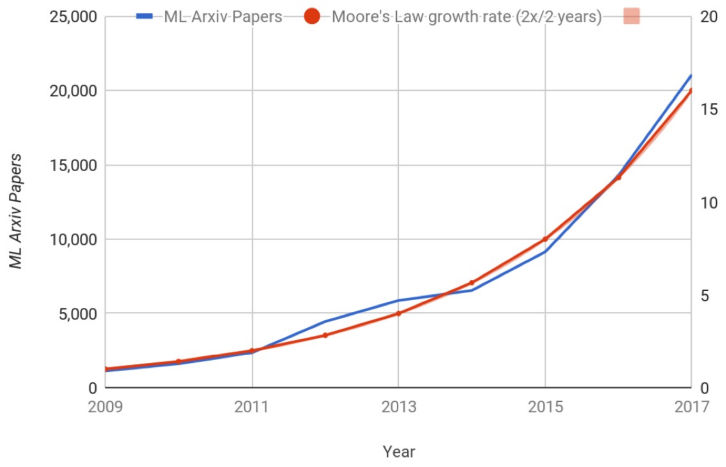


Image Processing

Speech Recognition

Translation

...

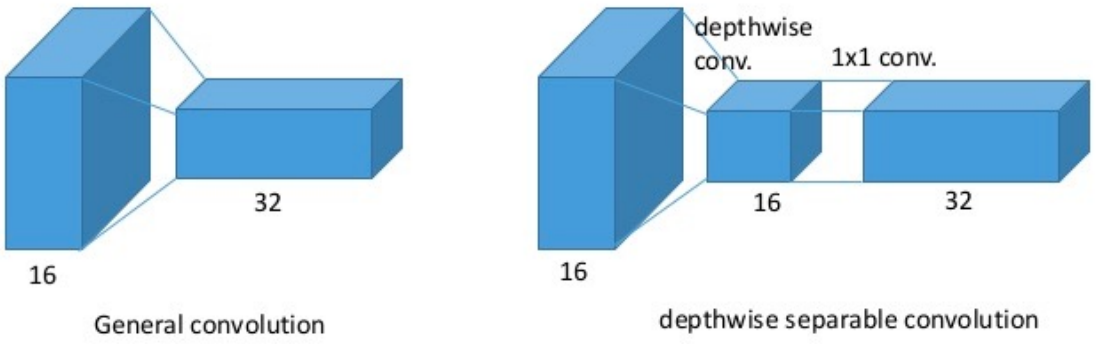
Evolution of DNN Applications

AlexNet 2012 | VGGNet 2014 | ResNet 2015 | 2016 | Transformer 2017 | MobileNetv2 2018 | EfficientNet 2019

Evolution of DNN Models

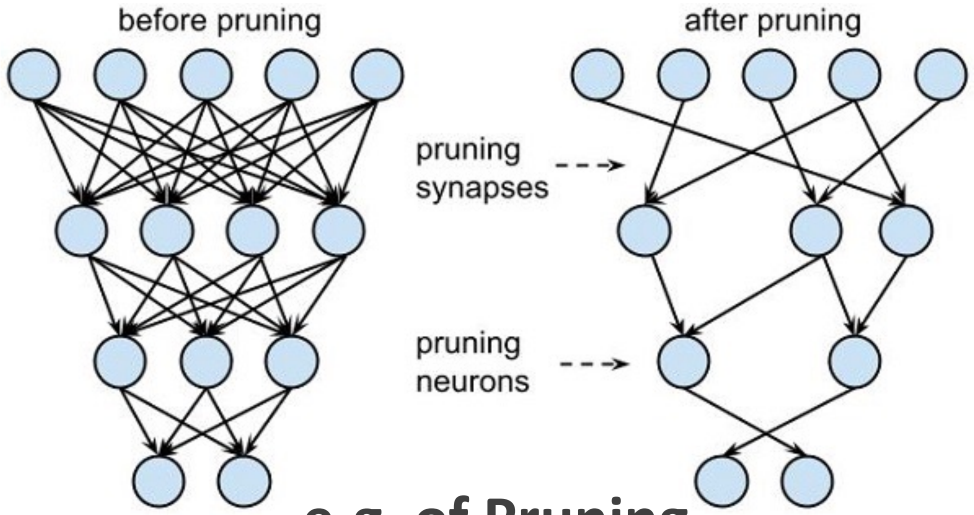
Why do we need *flexible* DNN accelerators?

- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types



e.g. of Depth-wise Separable CONV

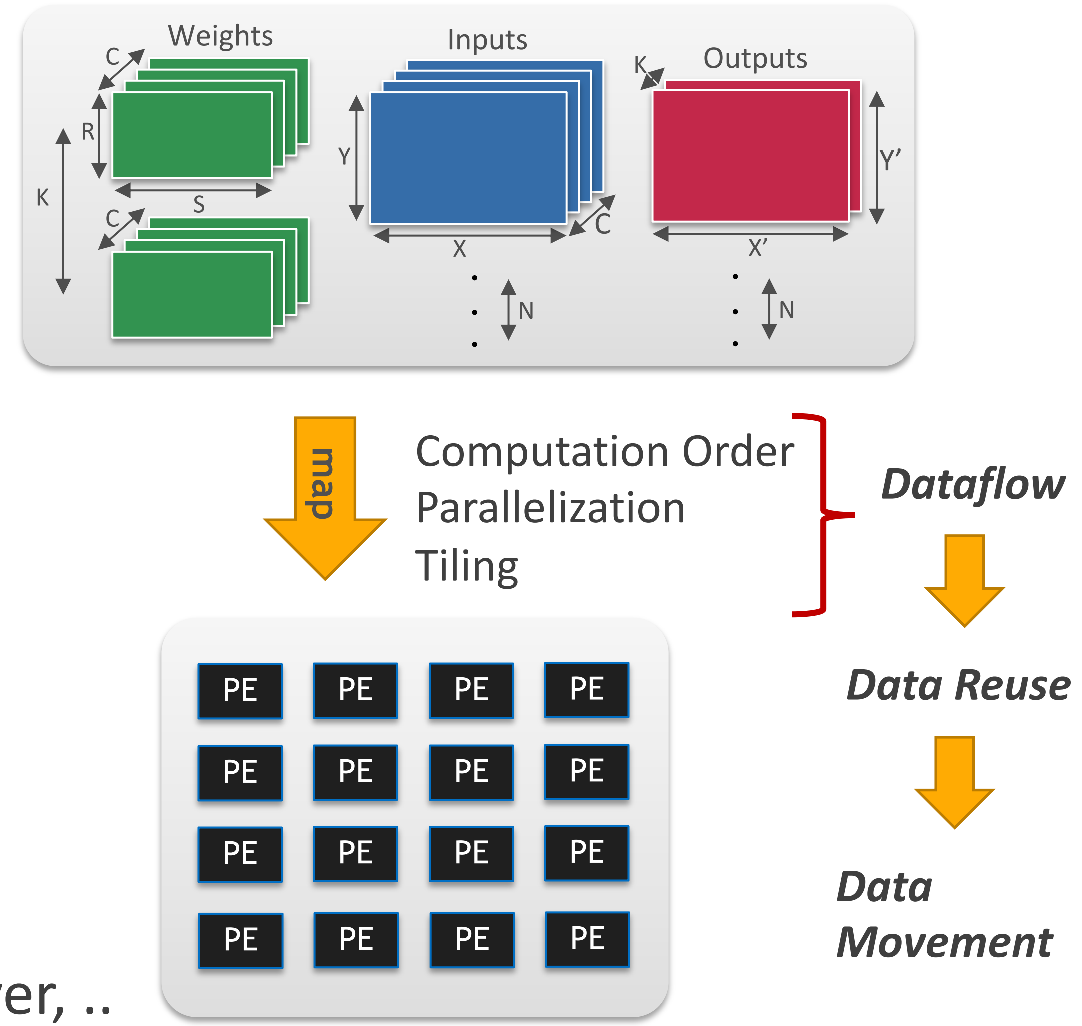
- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity



e.g. of Pruning

Why do we need *flexible* DNN accelerators?

- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types
- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity
- **Trend 3: Diversity in Mapping/Dataflow**
  - Loop Transformations ("Dataflow")
    - Order, Parallelization, Tiling
    - "Weight Stationary", "Row Stationary"
  - Partitioning Strategies – Per Layer, Cross Layer, ..



Why do we need *flexible* DNN accelerators?

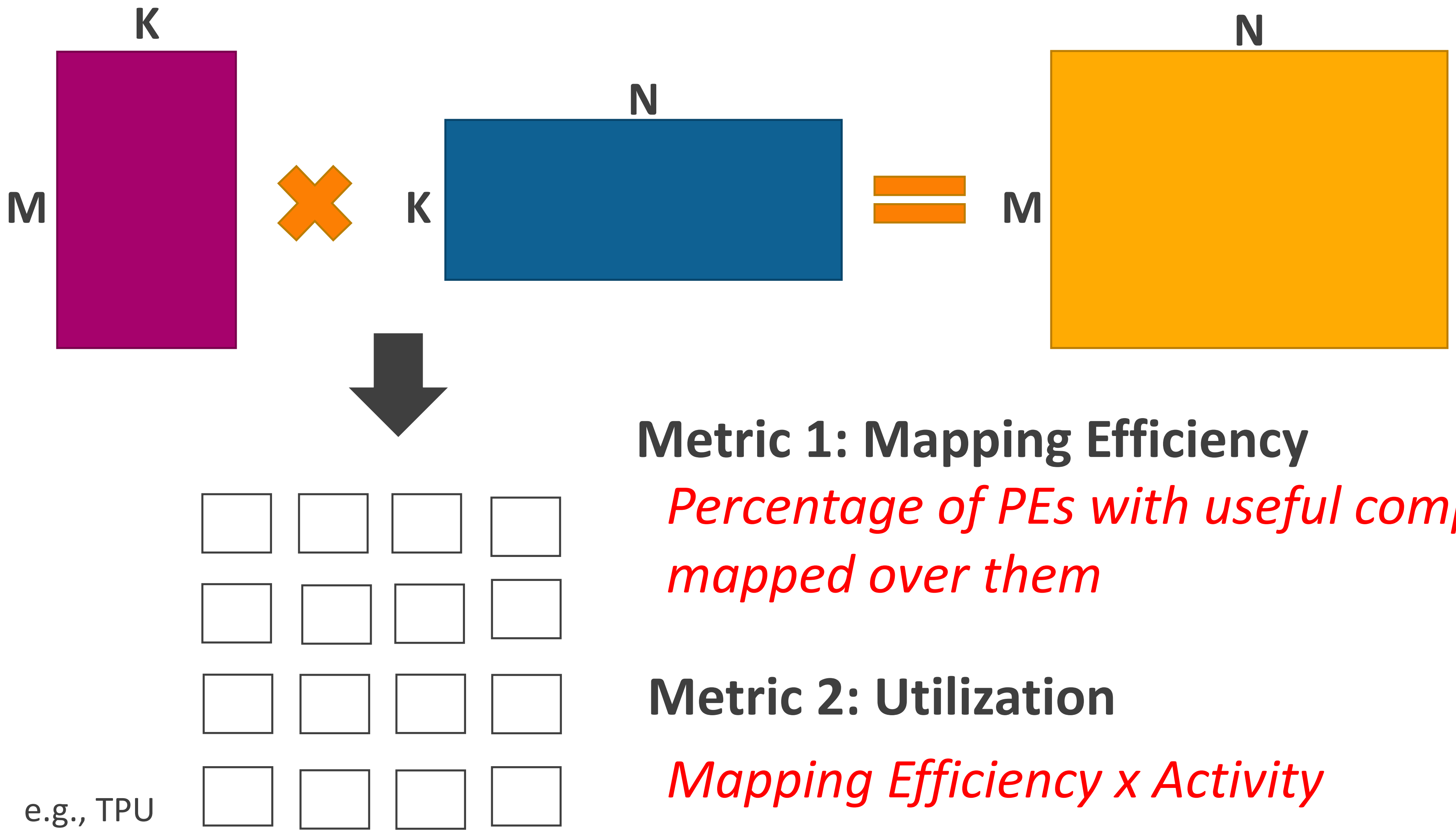
- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types
- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity
- **Trend 3: Diversity in Mapping/Dataflow**
  - Loop Transformations (“Dataflow”)
    - Order, Parallelization, Tiling
    - “Weight Stationary”, “Row Stationary”
  - Partitioning Strategies – Per Layer, Cross Layer, ..

Myriad “irregular” shapes, sizes, accesses

Challenge:
Getting high-utilization from accelerator for all cases.

Why?
Aren’t DNNs essentially Matrix-Matrix multiplications?

Example of GEMM Operation



Metric 1: Mapping Efficiency

Percentage of PEs with useful computations mapped over them

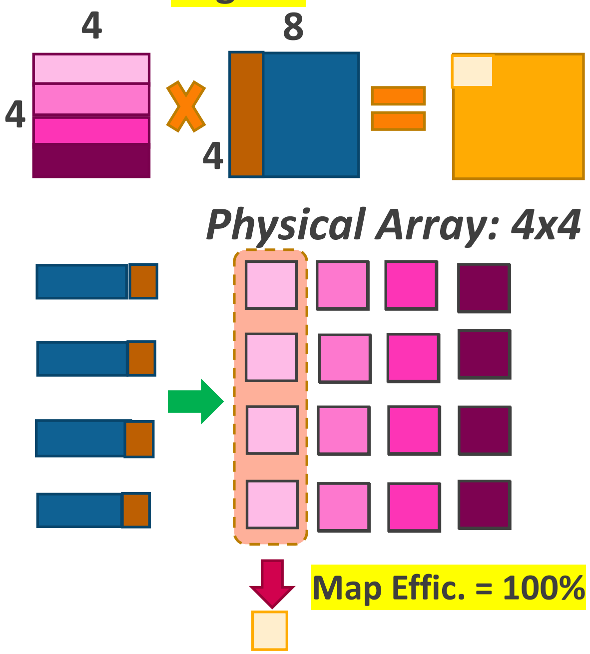
Metric 2: Utilization

Mapping Efficiency x Activity

Mapping Examples

Communication
Distribute
Collect

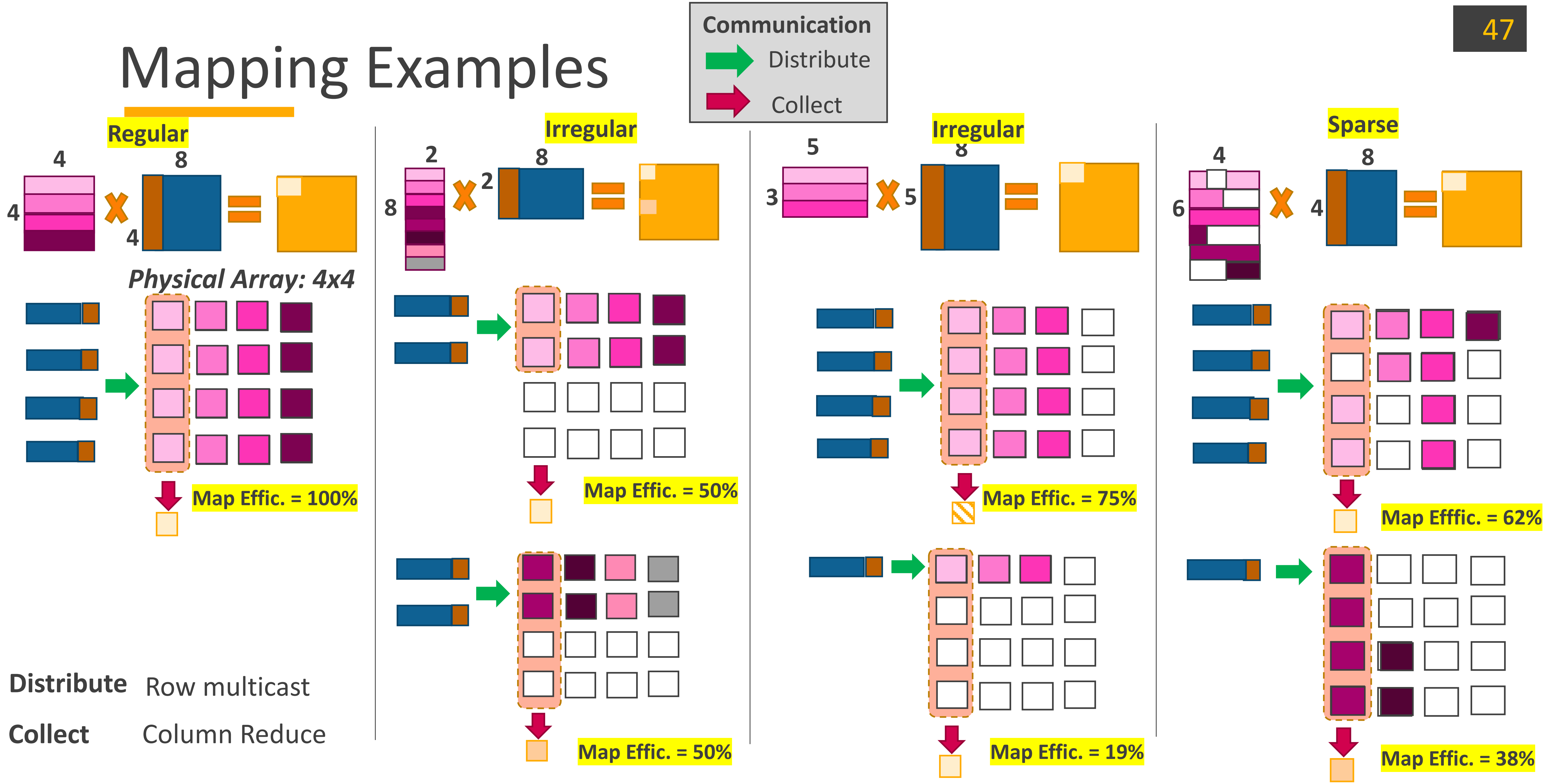
Regular



Distribute Row multicast

Collect Column Reduce

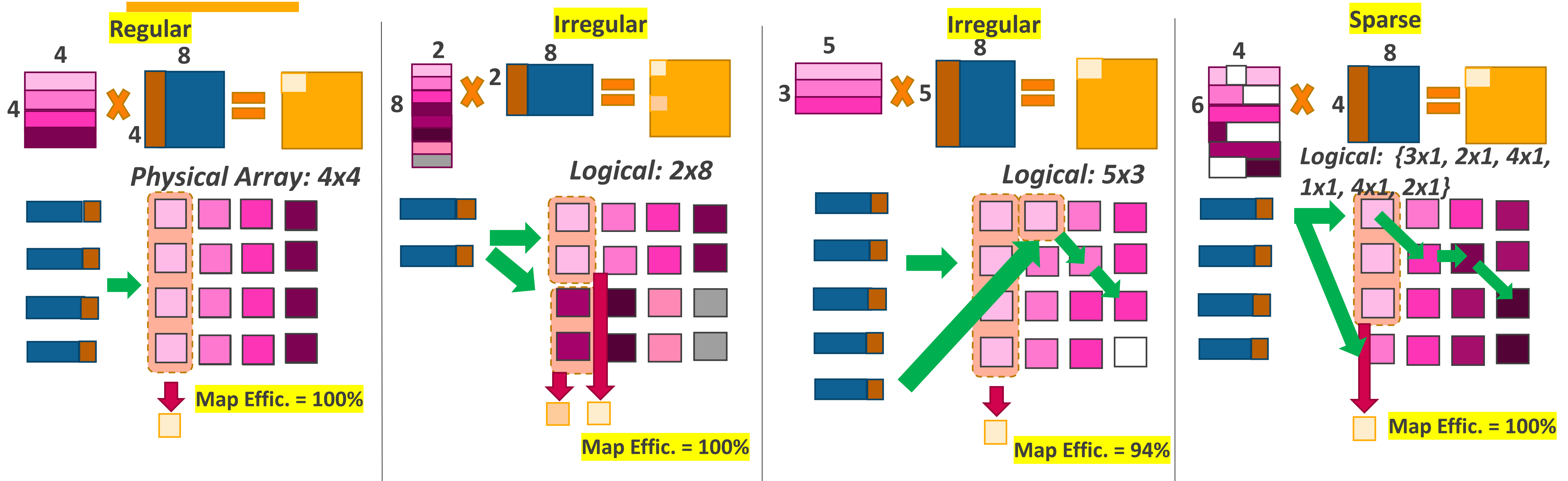
Mapping Examples



Distribute Row multicast

Collect Column Reduce

Mapping Efficiency needs Mapping Flexibility

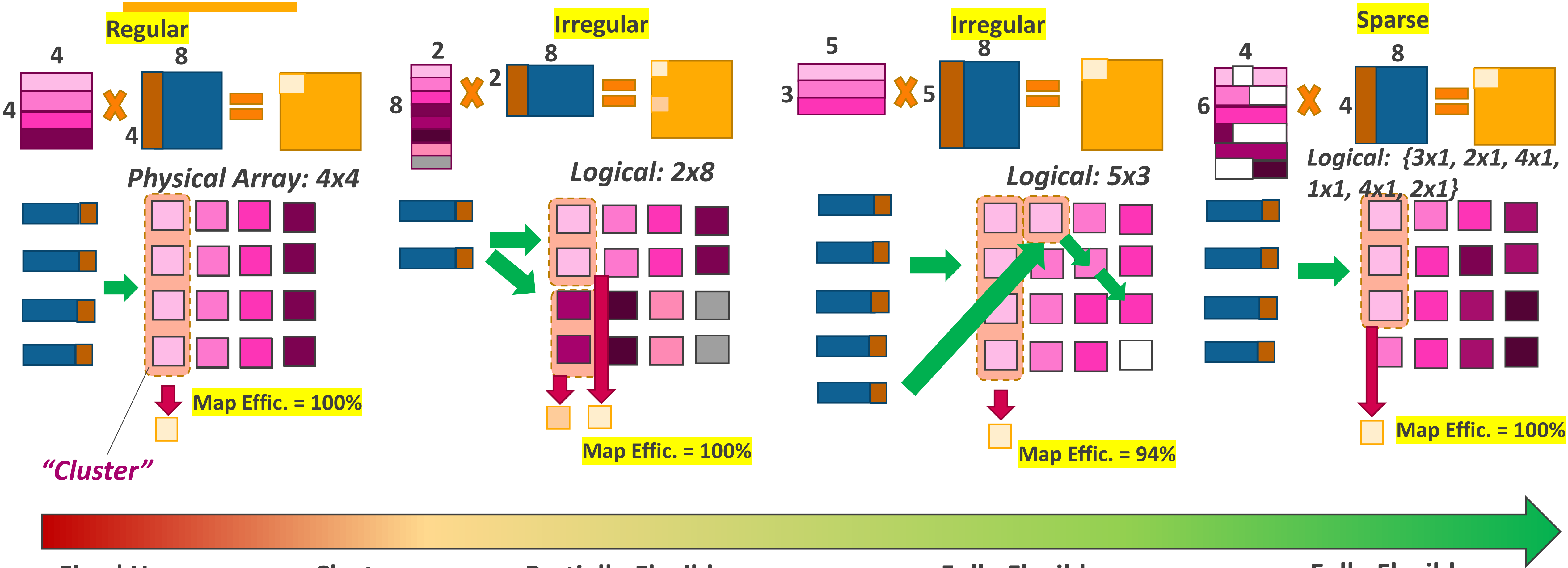


How to support Mapping Flexibility?

Distribute	Row multicast	Spatial Multicast	Multicast to non-neighbors	Only send non-zeros
Collect	Column Reduce	Multiple Parallel	Variable Length	Variable Non-Uniform Length

Flexible data distribution and reduction

Levels of Flexibility



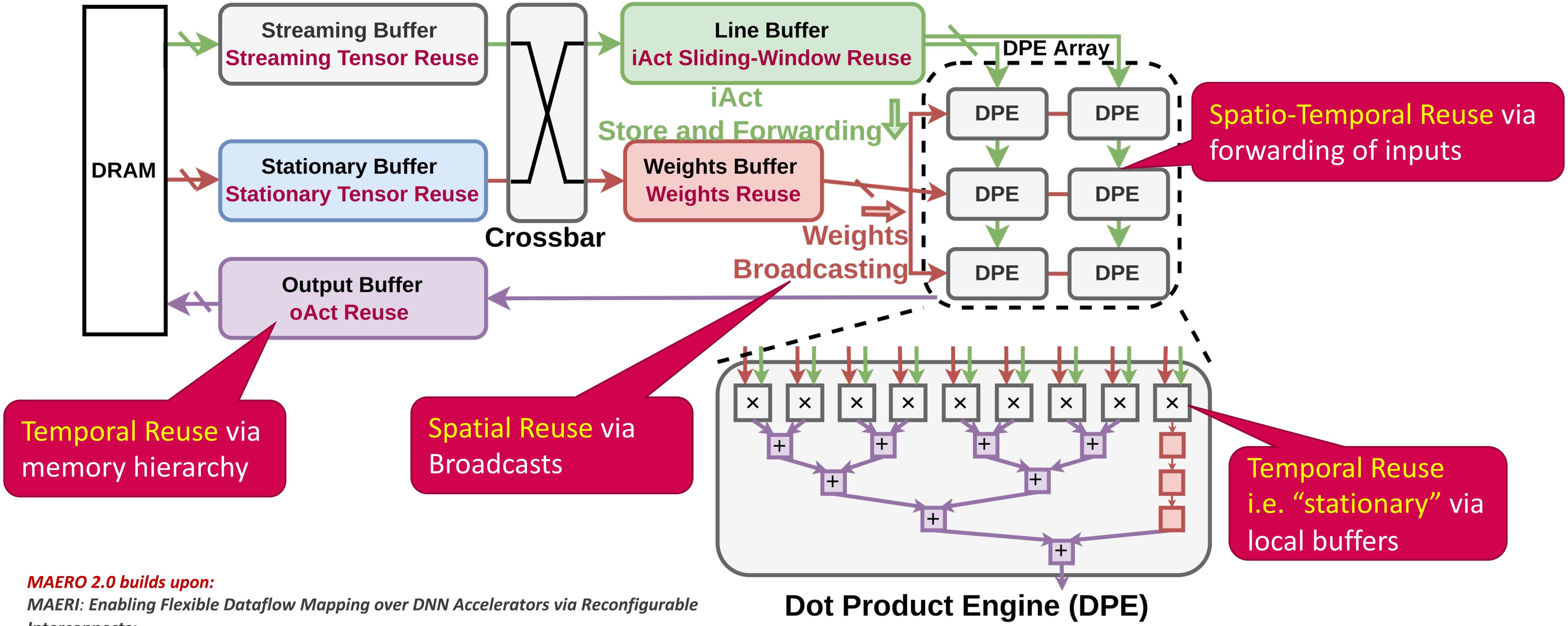
Fixed Homogeneous Clusters
(i.e., fixed cluster size => fixed aspect ratio)

Partially-Flexible Homogeneous Clusters
(configurable (limited choices) number of PEs per cluster)

Fully-Flexible Homogeneous Clusters
(configurable (any choice) number of PEs per cluster)

Fully-Flexible Heterogeneous Clusters
(configurable (any choice) unequal sized clusters)

Introducing MAERI2.0 – A Flexible DNN Accelerator



MAERO 2.0 builds upon:

MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects:

Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna

ASPLOS 2018, IEEE Micro Top Picks 2019 Honorable Mention

Focus of Today's Tutorial

- Supported Neural Network Model
- Quantization Flow
- Memory Layout
- Heterogeneous Scheduling
- MAERI 2.0 Microarchitecture
- FPGA DEMO

Future Work:
- Support for Sparsity
- Support for Multi-layer Mapping
- Compiler support

Schedule (EST)

Time slot	Topic	
14:00 to 14:30	Introduction to DNN Accelerators	Tushar
14:30 – 14:40	Break	
14:40: 15:10	MAERI2.0 Architecture and Tool Flow	Jianming
15:10 to 15:30	Demo on FPGA	Jianming

Brief Q/A at the end of each talk.

Please feel free to interrupt and ask questions or use chat

Attention: Tutorial is being recorded!

https://maeri-project.github.io/tutorials/ics-2022