http://inst.eecs.berkeley.edu/~ee290-2/sp21/

Hardware for Machine Learning

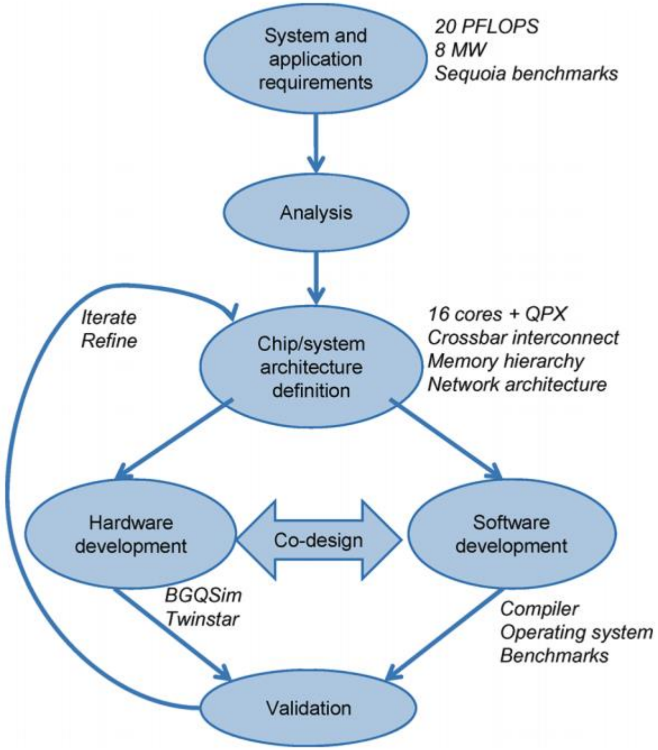
Lecture 14: Codesign

Sophia Shao

HW/SW Co-Design

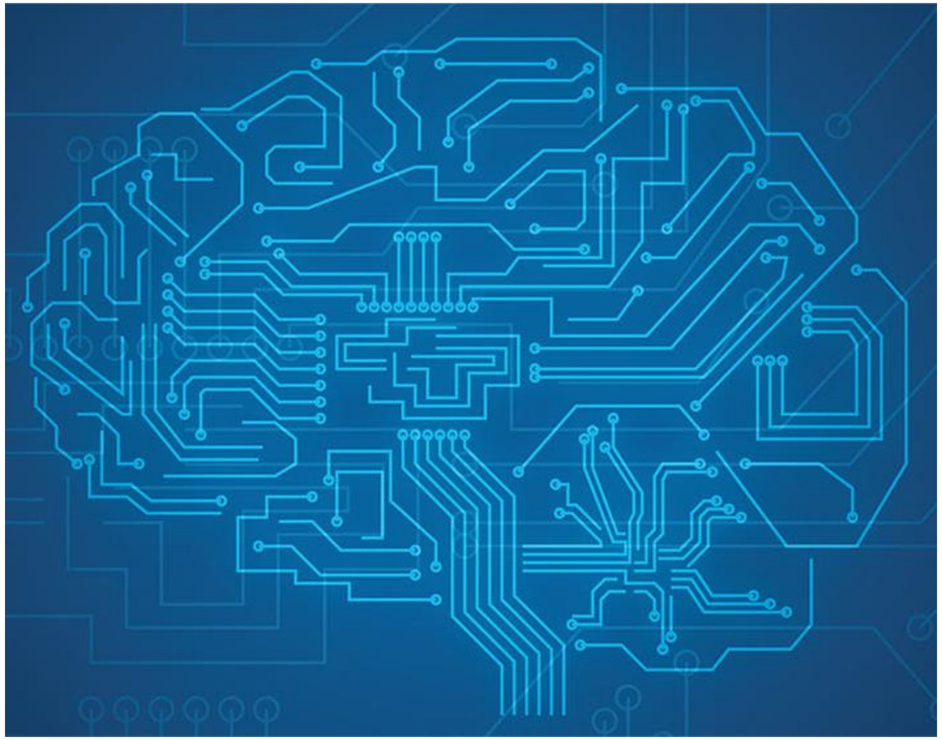
"In the context of building complex computer systems, the term **co-design** refers to the **concurrent design and optimization** of several aspects of the system, including **hardware** and **software**, to achieve a set target for a given system metric, such as **throughput, latency, power, size, or a combination thereof**."

Modeling, validation, and co-design of IBM Blue Gene/Q: Tools and examples



Review

- Core computation in DNN
- Execution order of the core computation
- Hardware realization of the core computation
- Mapping DNNs to hardware
- Data transfer mechanisms across storage hierarchy
- Last Lecture: sparsity in DNNs
  - Source of sparsity
  - Sparsity in storage:
    - Compression formats
  - Sparsity in compute:
    - Indirection and intersection



Codesign

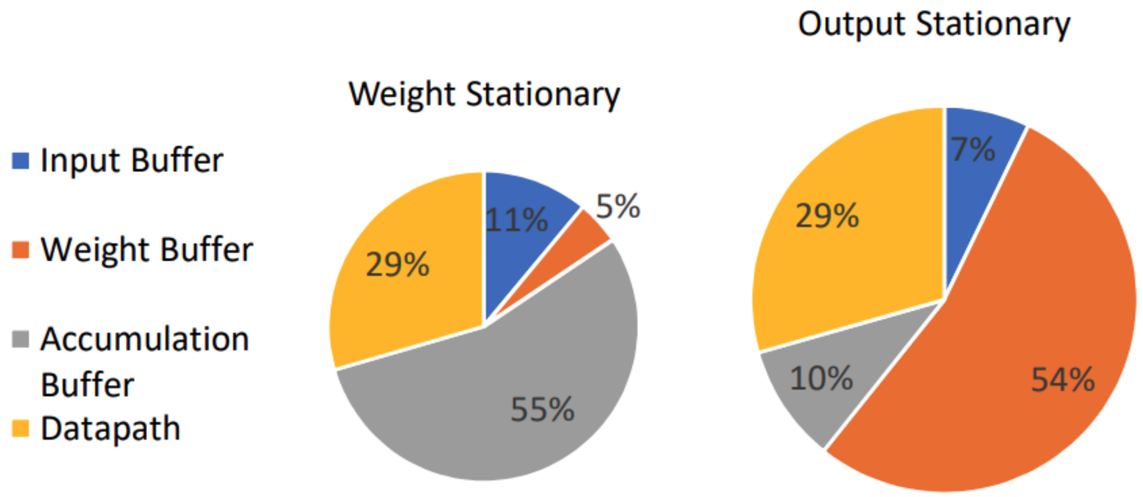
- Implications on HW
- HW-Aware Design
  - Pruning
  - Compact Network
  - Neural Arch. Search

Domain-Specific Hardware

- To understand the requirements of applications or a domain of applications in the hardware design process.
- So far we have discussed specialization in:
  - Control
  - Compute
  - Data orchestration
  - Data format (compression)
- At the same time, even within a domain, different applications or optimizations of different applications have different implications.

Implications on HW: Dataflow

- Stationary: which operand is being reused at which level of memory hierarchy
  - Weight stationary: TPU, NVDLA
  - Output stationary: ShiDianNao
  - Row stationary: Eyeriss



MAGNet, ICCAD'2019

Implications on HW: Hybrid Dataflows

- Reuse more operands across multiple levels of register files.

```
for k1=[0:K1):
 for r=[0:R):
  for s=[0:S):
   for c1=[0:C1):
    // WS
    for p1=[0:P1):
     for q1=[0:Q1):


        Vector MACs
```

(a) WS dataflow

```
for k1=[0:K1):
 for p1=[0:P1):
  for q1=[0:Q1):
   // OS
   for r=[0:R):
    for s=[0:S):
     for c1=[0:C1):


        Vector MACs
```

(b) OS dataflow

```
for h1=[0:H1):
 for w1=[0:W1):
  for c1=[0:C1):
   // IS
   for k1=[0:K1):
    for r=[0:R):
     for s=[0:S):


        Vector MACs
```

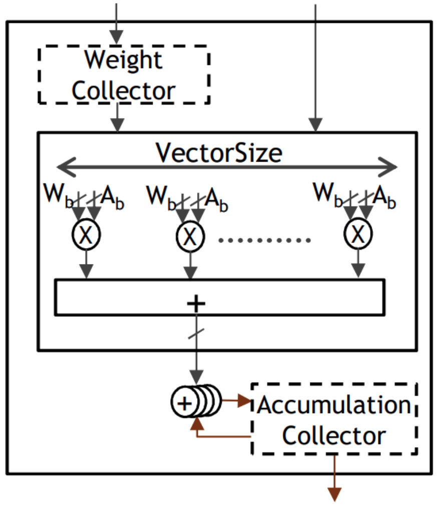
(c) IS dataflow

```
for k1=[0:K1):
 for r=[0:R):
  for s=[0:S):
   for c1=[0:C1):
    // WS
    for p1=[0:P1):
     for q1=[0:Q1):
      // LOS
      for c0=[0:C0):
        Vector MACs
```

(d) WS-LOS dataflow

```
for k1=[0:K1):
 for p1=[0:P1):
  for q1=[0:Q1):
   // OS
   for r=[0:R):
    for s=[0:S):
     for c1=[0:C1):
      // LWS
      for q0=[0:Q0):
        Vector MACs
```

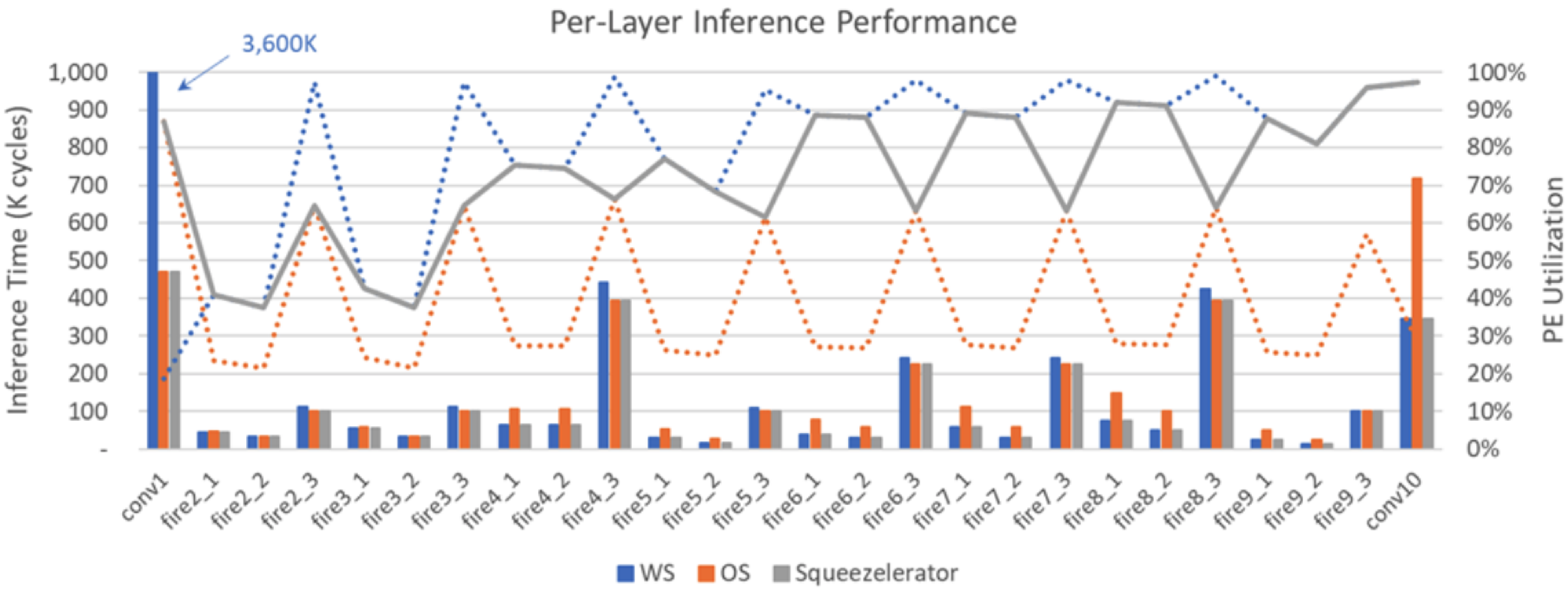
(e) OS-LWS dataflow



Dataflow	Weight Reuse	Input Reuse	Output Reuse
WS	P1×Q1	0	0
OS	0	0	R×S×C1
IS	0	R×S×K1	0
WS-LOS	P1×Q1	0	C0
OS-LWS	Q0	0	R×S×C1

MAGNet, ICCAD'2019

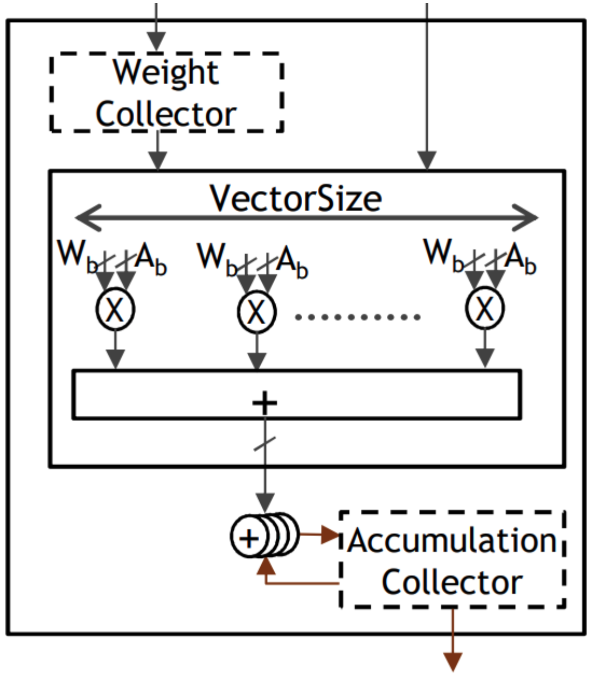
Implications on HW: Dataflow

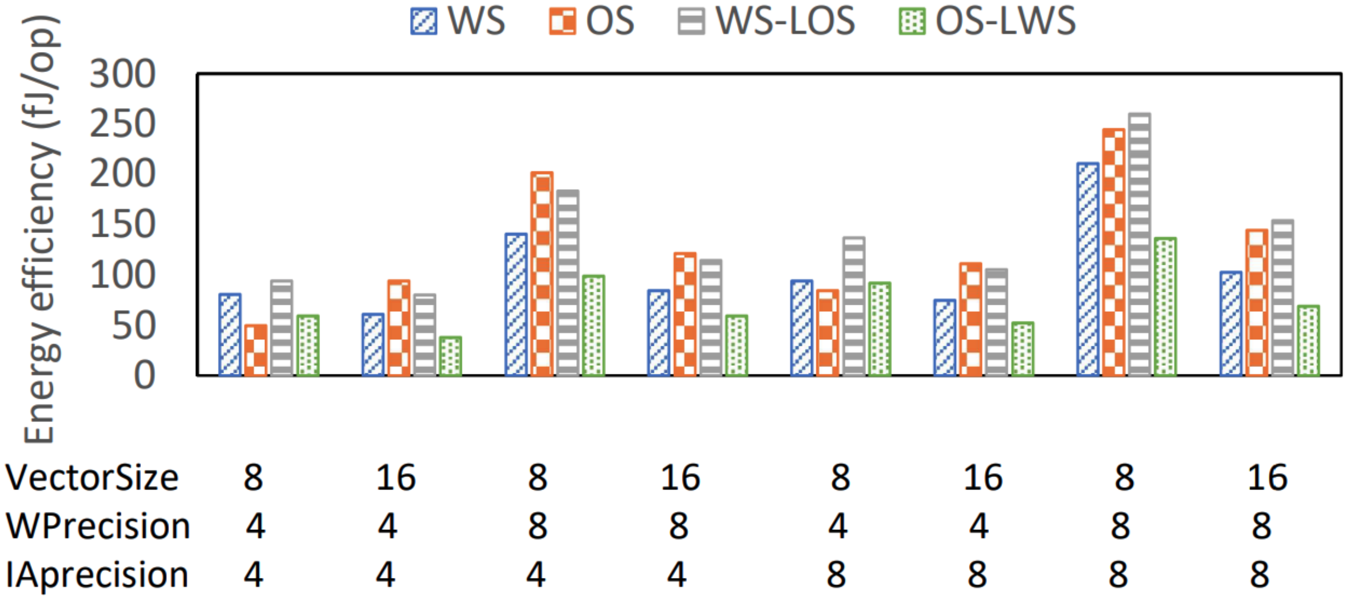


Co-Design of Deep Neural Nets and Neural Net Accelerators for Embedded Vision Applications, DAC'2018

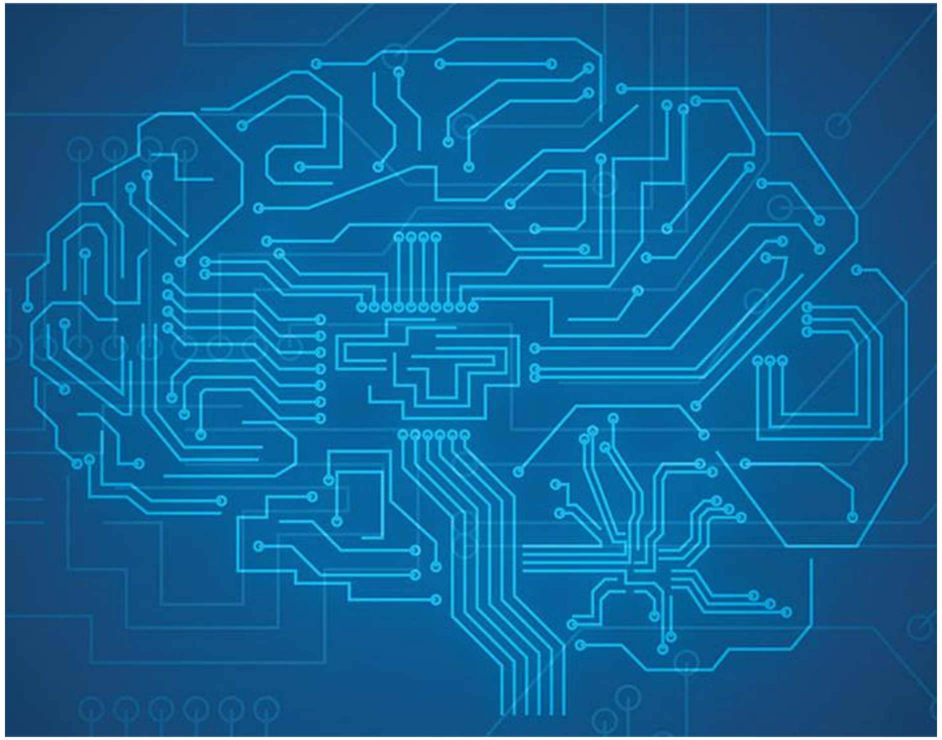
Implications on HW: Hybrid Dataflows

- A diverse set of HW parameters leads to codesign opportunities.
  - Dataflow
  - Vector size
  - Precisions





MAGNet, ICCAD'2019

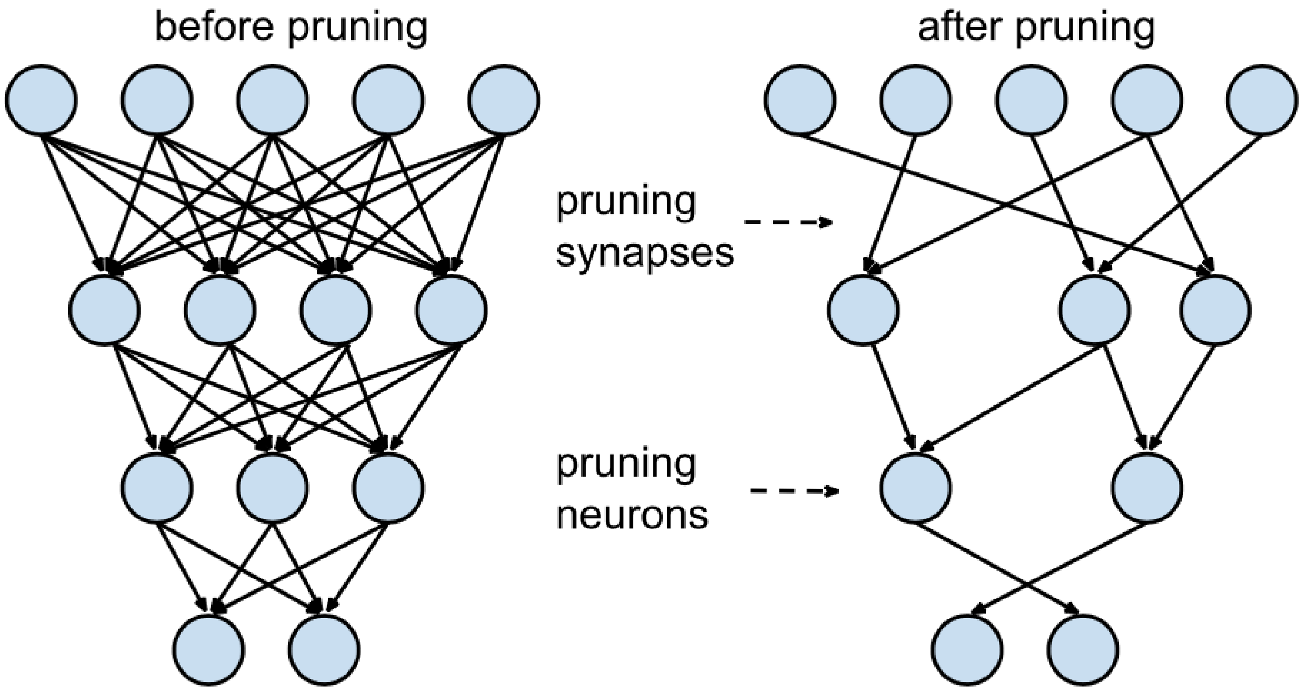


Codesign

- Implications on HW
- HW-Aware Design
  - Pruning
  - Compact Network
  - Neural Arch. Search

Pruning Neural Networks

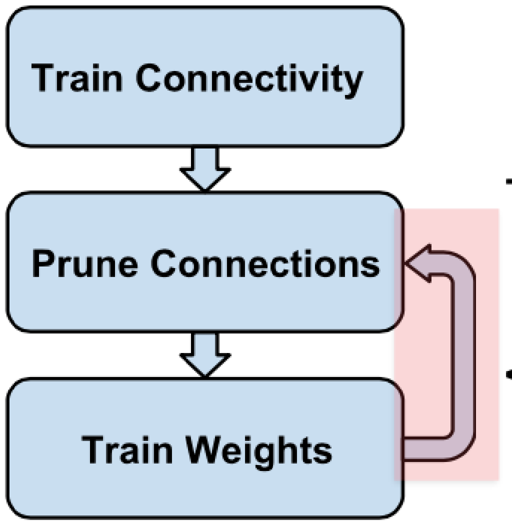
- Turn weights/input activations to zero.

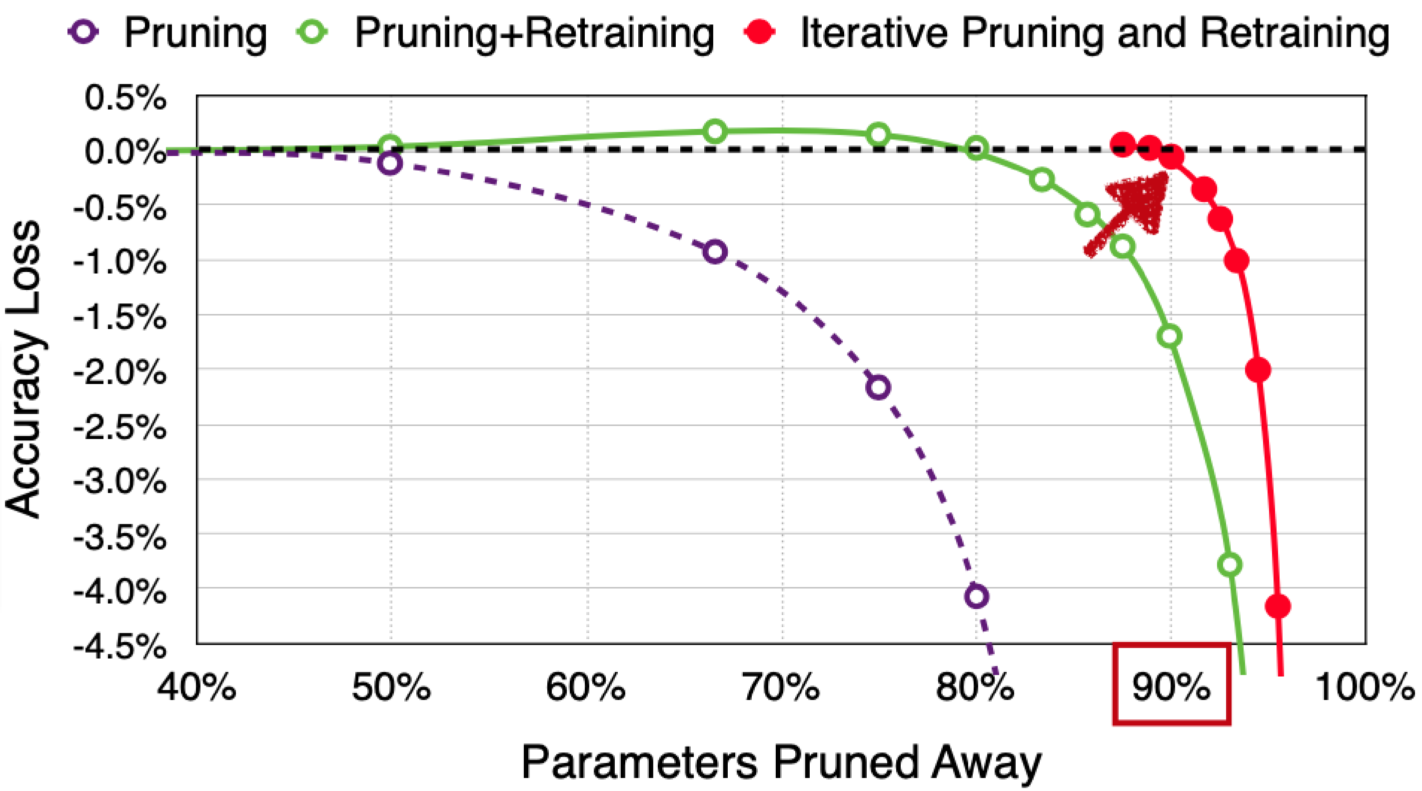


NIPS'89, NIPS'15

Pruning Neural Networks

- Turn weights/input activations to zero.





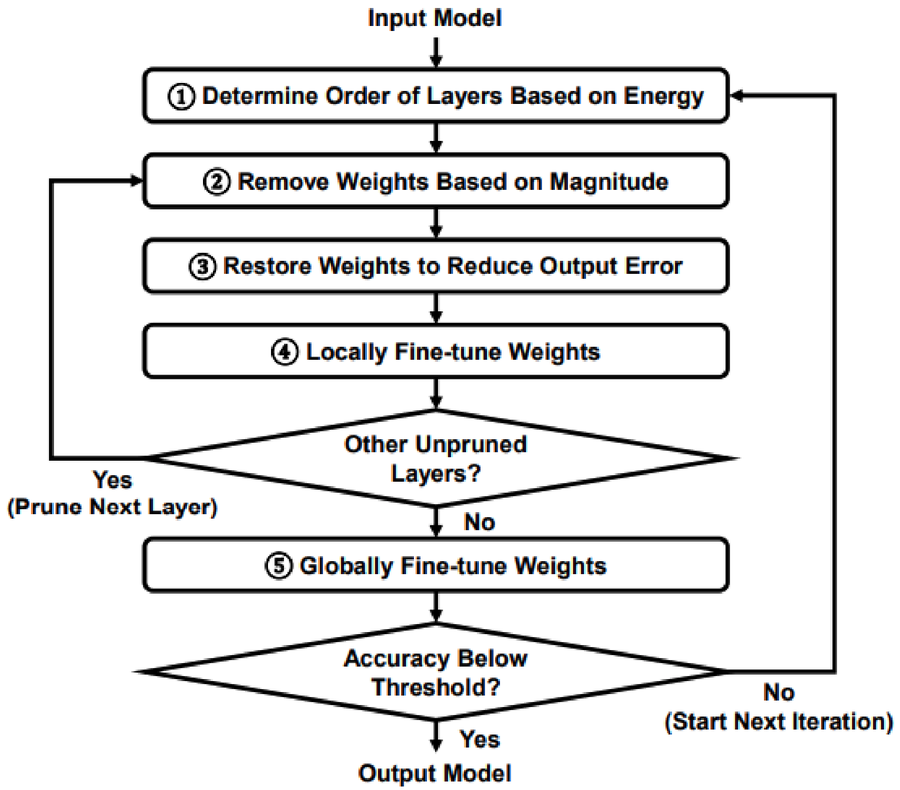
Han et al., NIPS'15

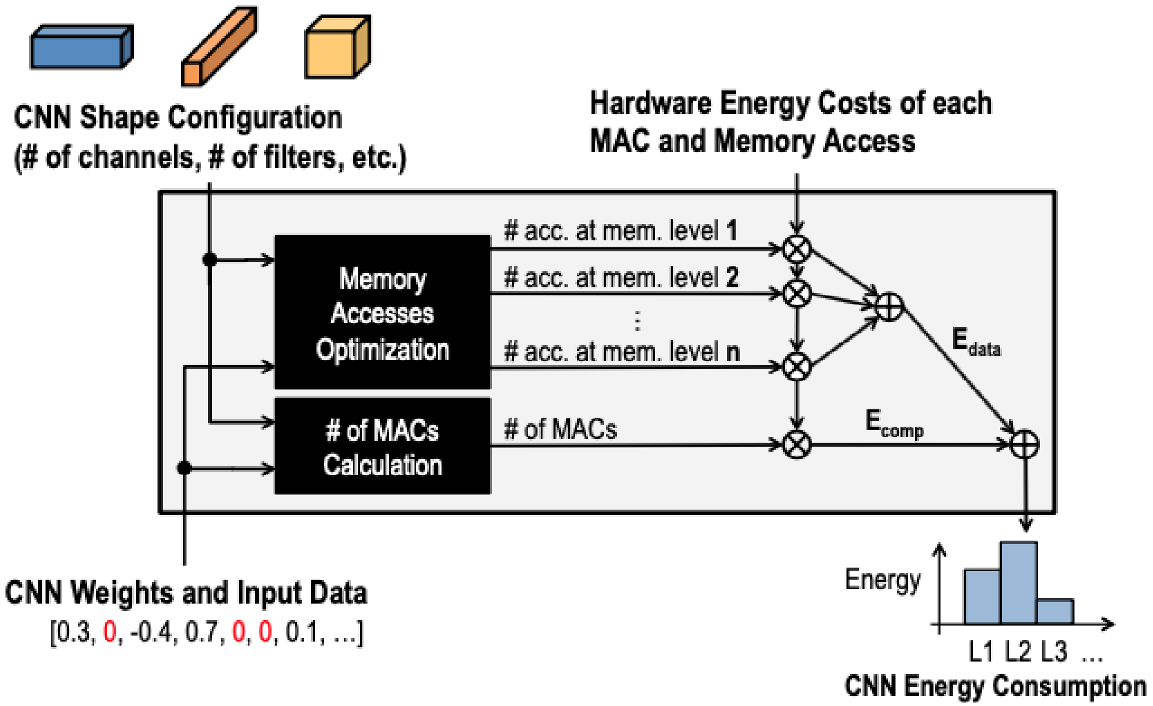
Is # of parameters pruned the right metric?

- Indirect metrics:
  - # of parameters
  - # of multiply-accumulate operations
- Direct metrics:
  - Latency
  - Energy consumption
- Direct metrics are hardware dependent.

Energy-aware Pruning

- Direct prune the network based on energy estimations.

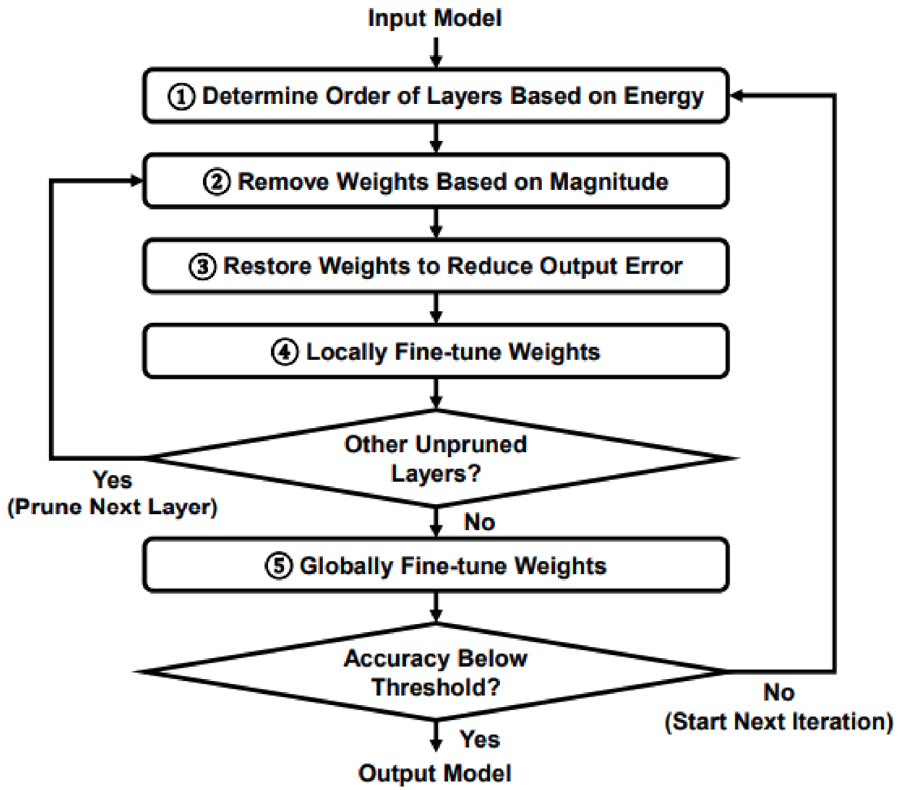


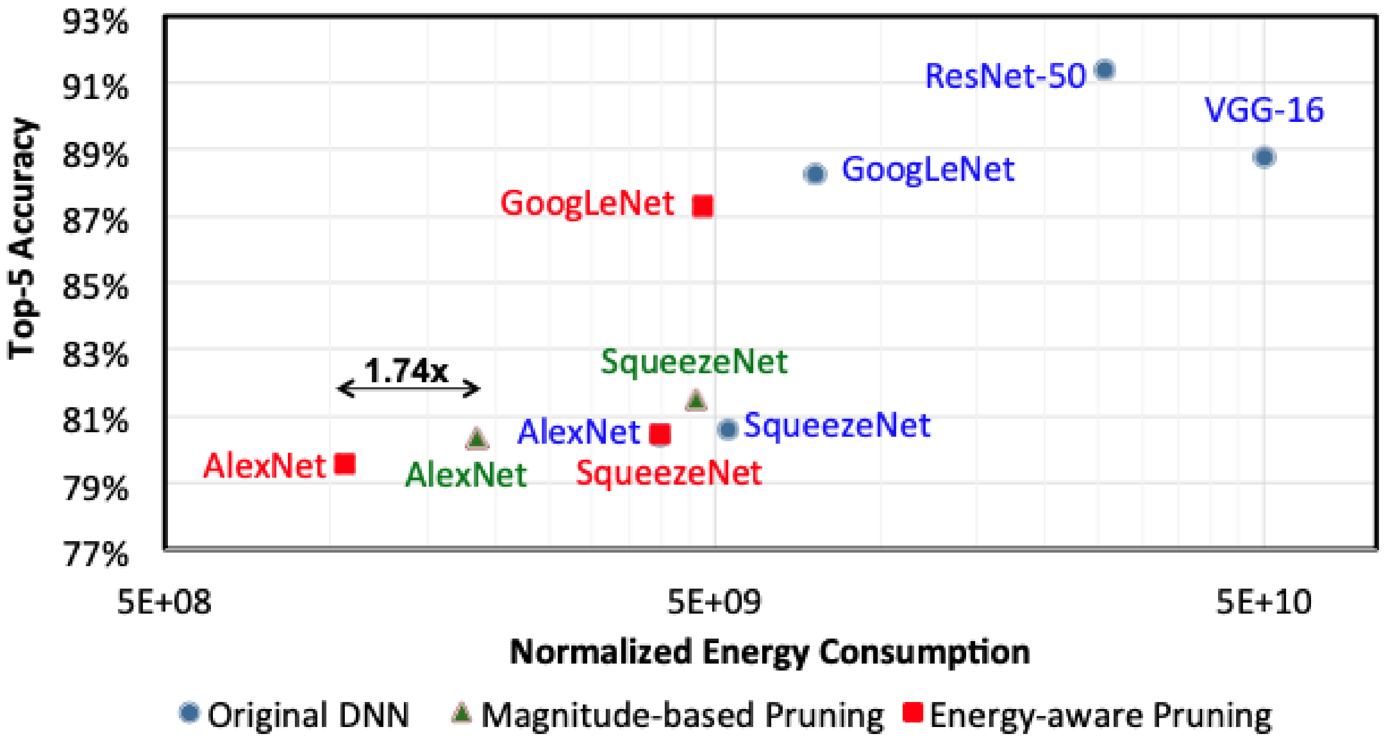


Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning, CVPR'2017

Energy-aware Pruning

- Direct prune the network based on energy estimations.

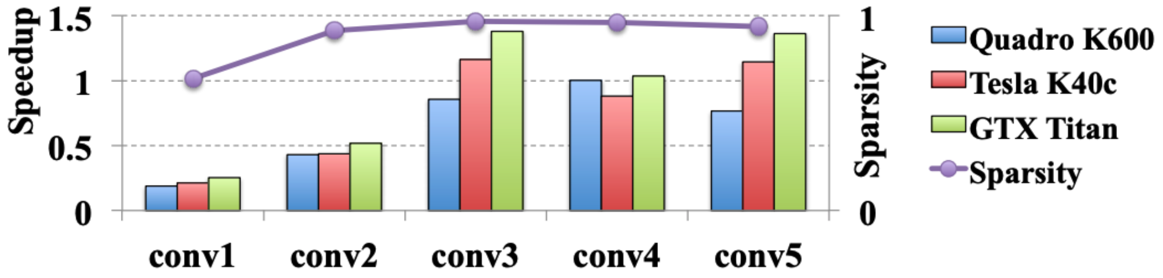




Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning, CVPR'2017

Structured Sparsity

- Unstructured sparsity does not directly translate to speedup.



- Structured sparsity.

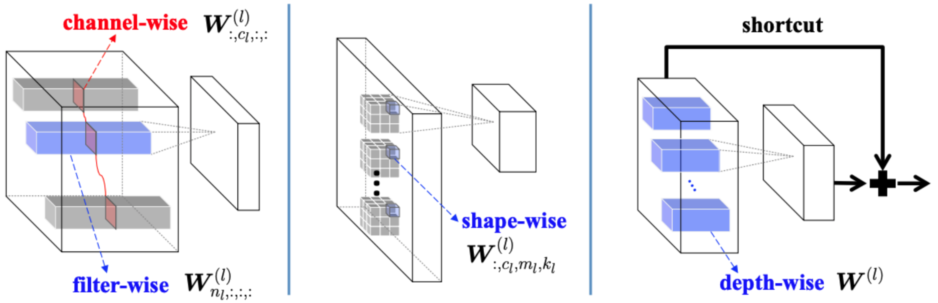


Figure 2: The proposed *Structured Sparsity Learning* (SSL) for DNNs. The weights in filters are split into multiple groups. Through group Lasso regularization, a more compact DNN is obtained by removing some groups. The figure illustrates the filter-wise, channel-wise, shape-wise, and depth-wise structured sparsity that are explored in the work.

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda \cdot R(\boldsymbol{W}) + \lambda_g \cdot \sum_{l=1}^{L} R_g\left(\boldsymbol{W}^{(l)}\right).$$

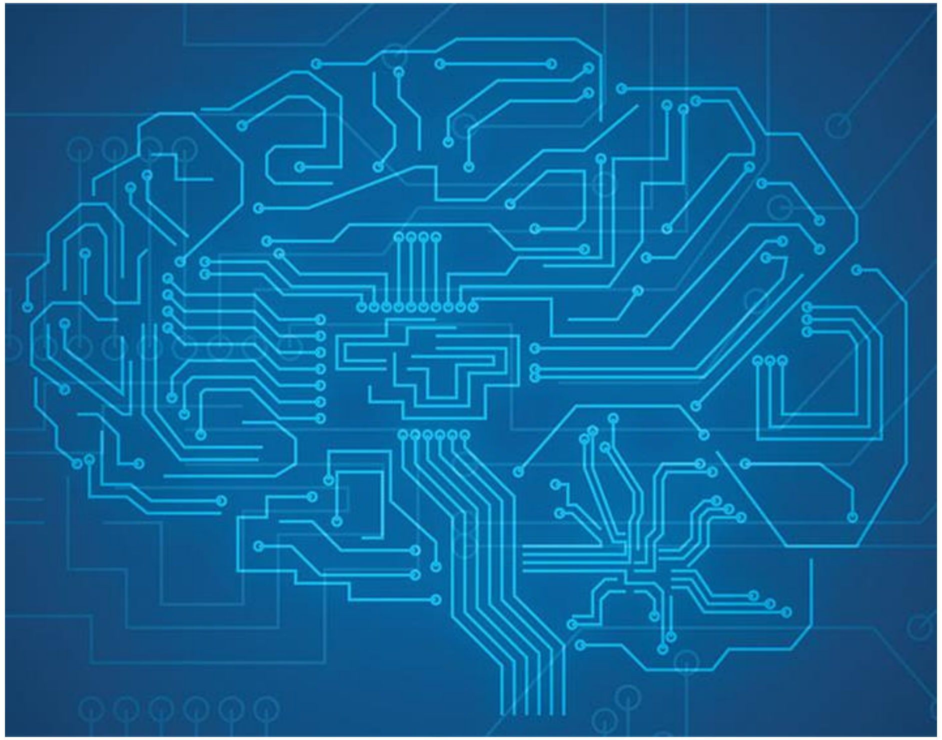
Learning Structured Sparsity in Deep Neural Networks, NIPS'2016

Administrivia

- Project proposal due next Friday.
  - Sample projects posted here:
    - https://docs.google.com/spreadsheets/d/1xoyiparn5G-2_QCfyeEj_kKI0XKLWwyVCxH9fn0oJEM/edit?usp=sharing

Final Project

- Project Proposal (due 3/19, before Spring Break)
  - Find 1-2 relevant research papers of your topic.
  - Write a summary of that research paper.
  - Describe how you hope to see or adapt ideas from it and how you plan to extend or improve it in your final project.
  - Project plan: describe milestones to achieve every two weeks:
    - Checkpoint 1 (early April)
    - Checkpoint 2 (late April)
    - Final report (early May)

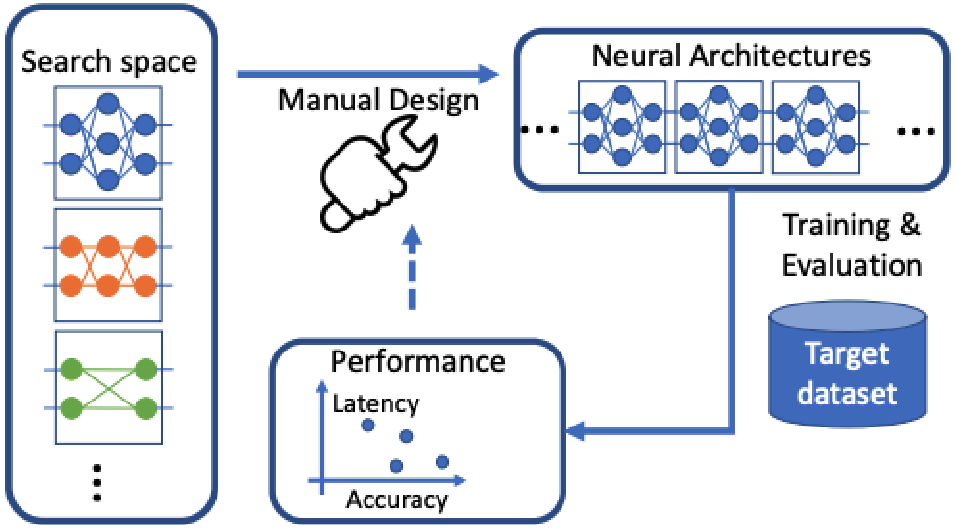


Codesign

- Implications on HW
- HW-Aware Design
  - Pruning
  - Compact Network
  - Neural Arch. Search

Compact Networks

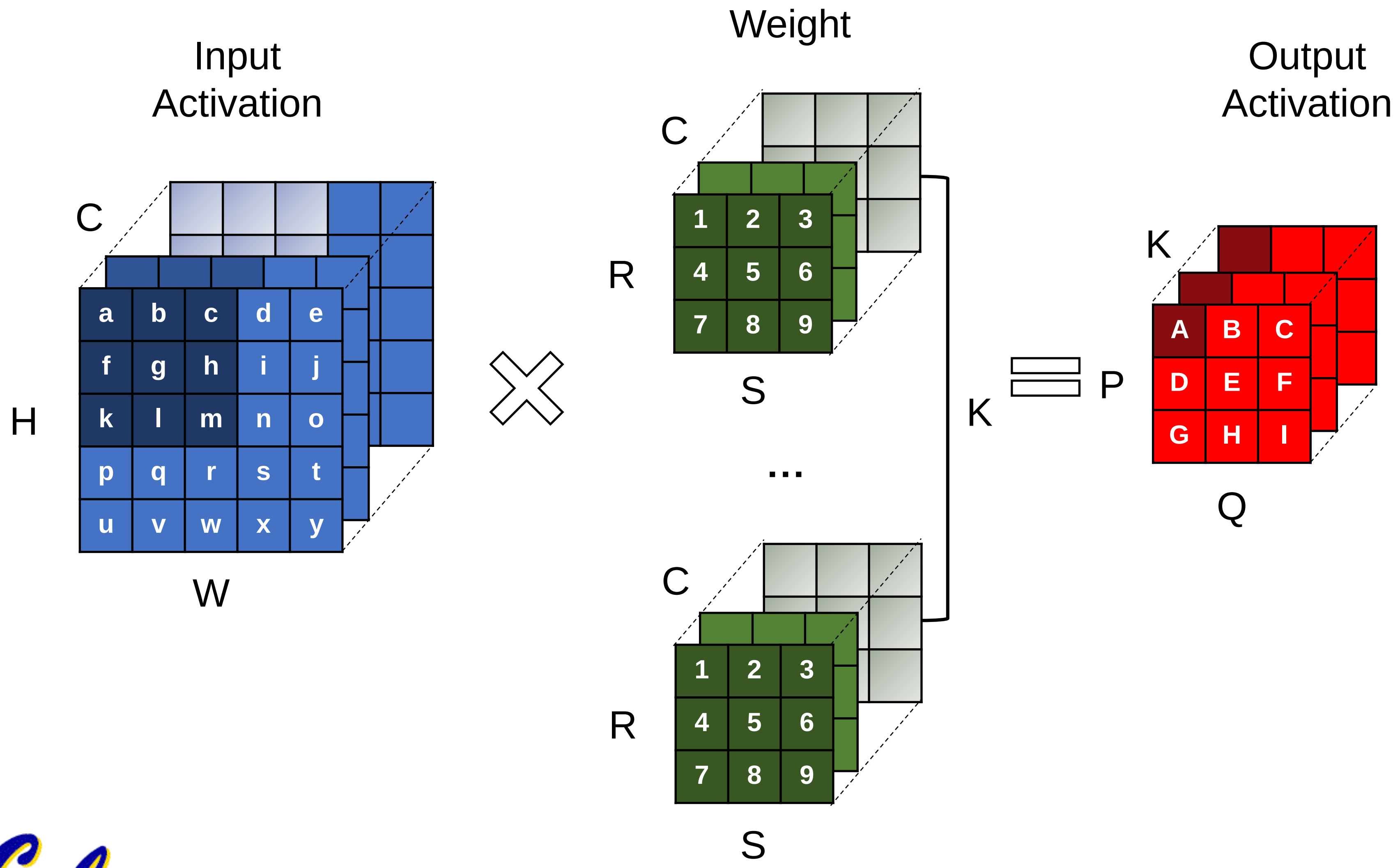
- Instead of pruning networks after training, directly design networks that are “efficient”.
  - Depth-wise convolution
    - E.g., MobileNet
  - Tensor Decomposition
    - CP decomposition
    - Tucker decomposition



(a) A typical flow of manual ConvNet design.

FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

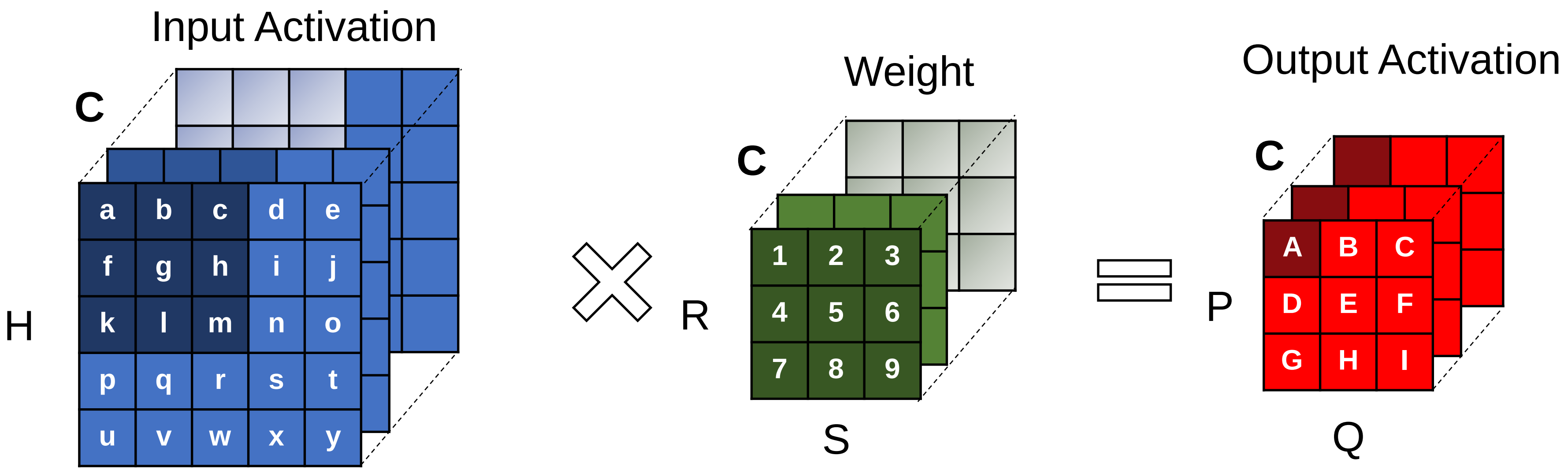
3-D Convolution



H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation
stride: # of rows/columns traversed per step
padding: # of zero rows/columns added

C: # of Input Channels
K: # of Output Channels

Depth-wise Convolution



H: Height of Input Activation
W: Width of Input Activation
R: Height of Weight
S: Width of Weight
P: Height of Output Activation
Q: Width of Output Activation
stride: # of rows/columns traversed per step
padding: # of zero rows/columns added

C: # of Input Channels
K: Not applicable
N: Batch size

- Reduce weight size (KRSC -> RSC)
- Reduce # of Ops (RSC mul./out -> RS mul./out)

MobileNet Architecture

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

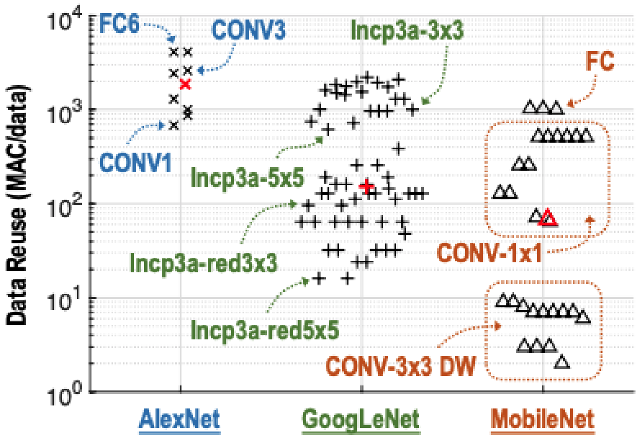
Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

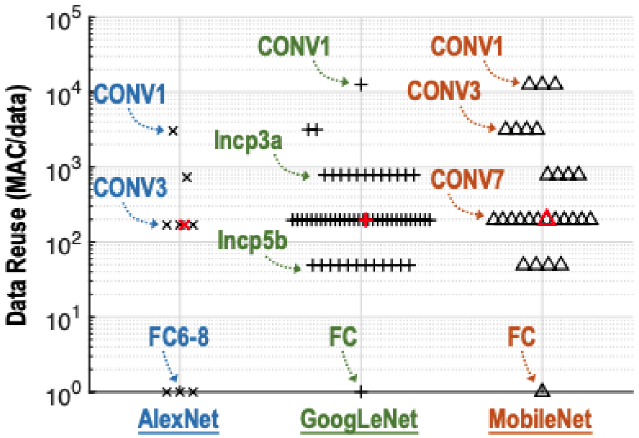
MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

MobileNet Implications

- Depth-wise convolution
  - reduces the amount of data reuse in HW.
  - Leads to low utilization



(a) Input activations (iacts)

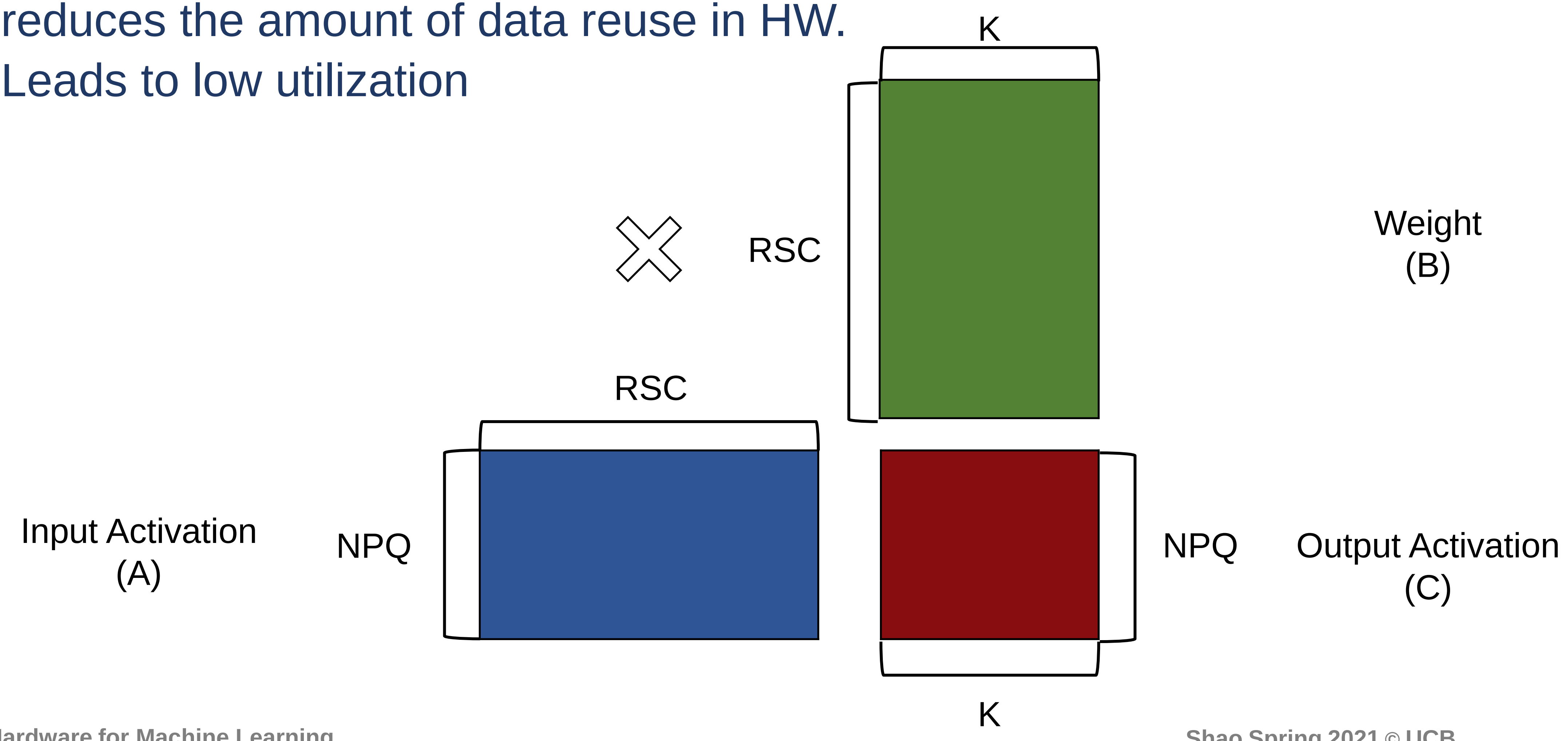


(b) Weights (batch size = 1)

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices

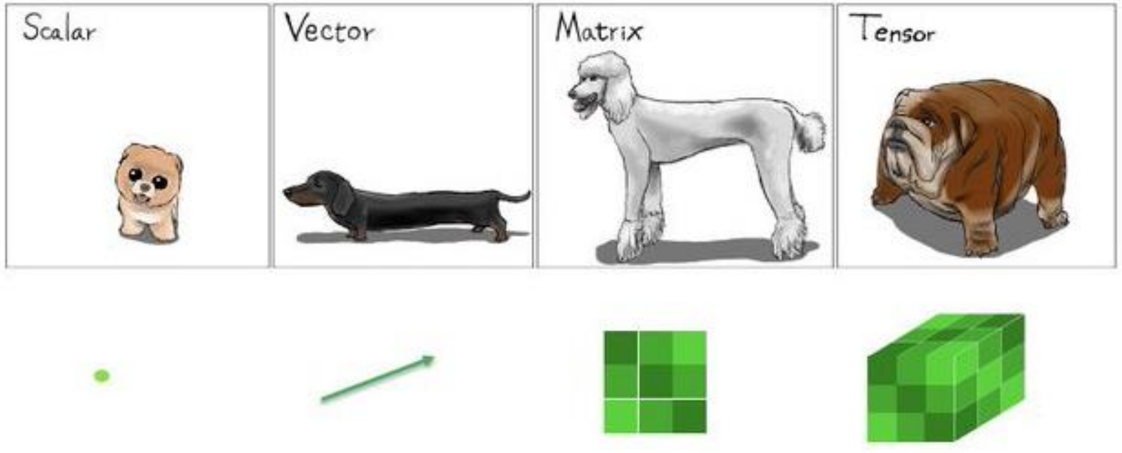
MobileNet Implications

- Depth-wise convolution
  - reduces the amount of data reuse in HW.
  - Leads to low utilization



Tensor Decomposition

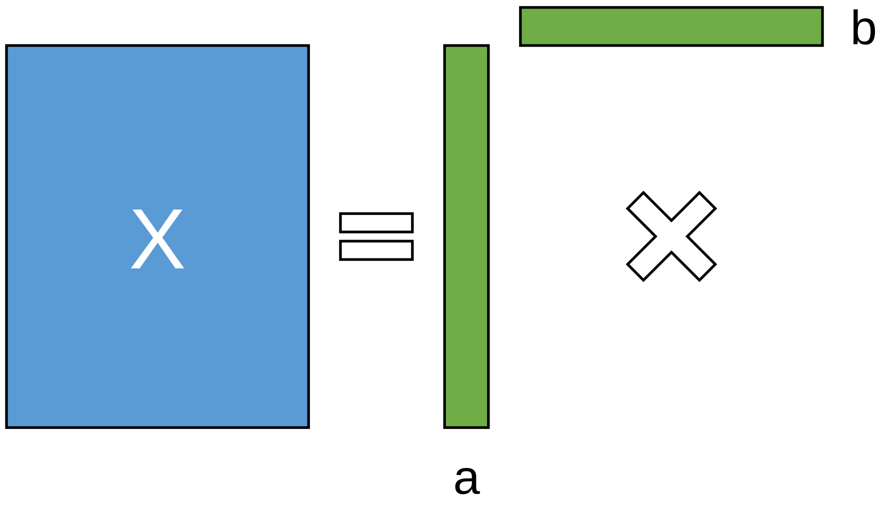
- Tensor is a multi-dimensional array.
- Tensor decomposition:
  - Express a tensor as a sequence of elementary operations acting on each out, often simpler tensors.
  - Benefits:
    - Provide a structured way to lower the complexity of weight tensors
  - Originally in scientific computing
  - Start seeing more in machine learning



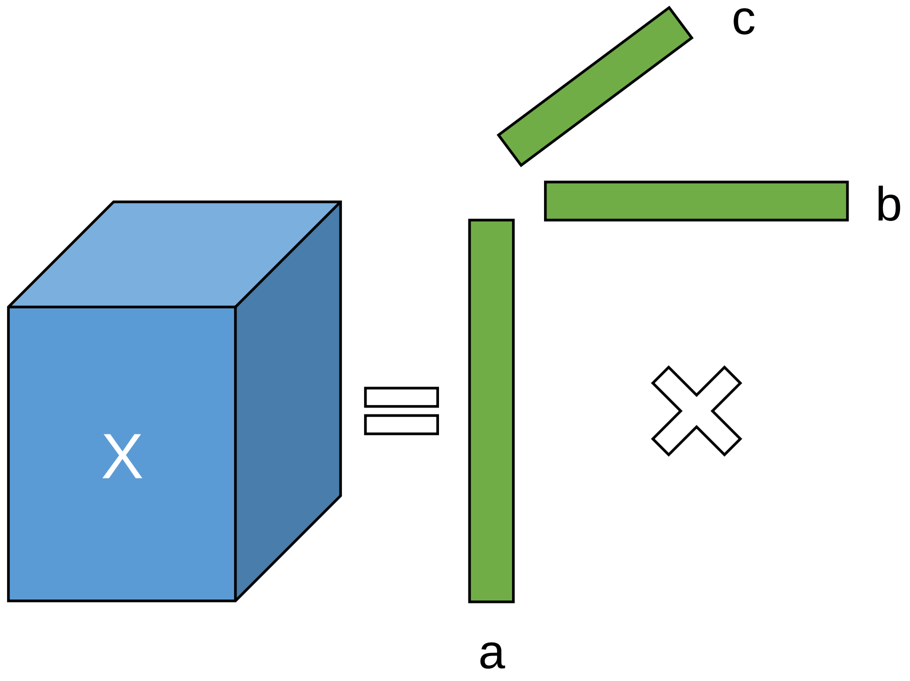
Anima Anandkumar, ScaledML'2018

Tensor Decomposition

- Building block for decomposition:
  - Rank-1 tensors, i.e., vectors
- Matrix version:
  - $X = a \times b$ (outer product)

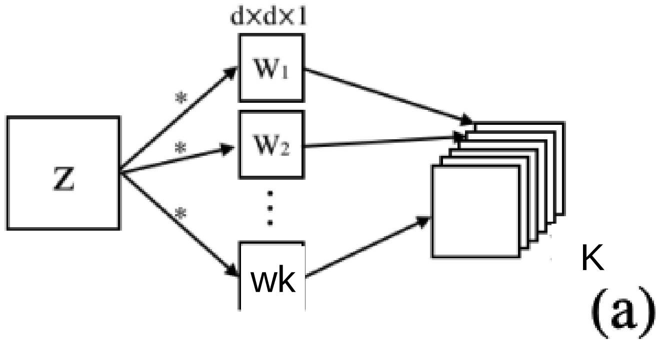


- Tensor version:
  - $X = a \times b \times c$ (outer product)



Tensor Decomposition in DNN (1):

- Approximate the original convolution with a linear combination of a set of basic filters.



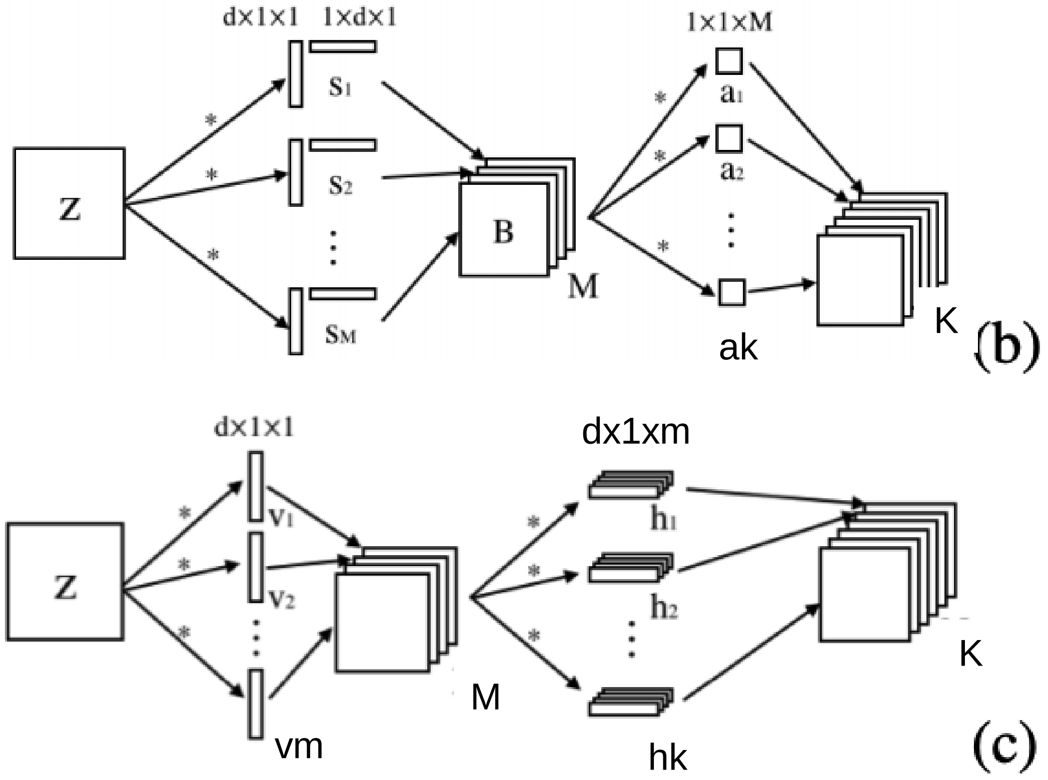
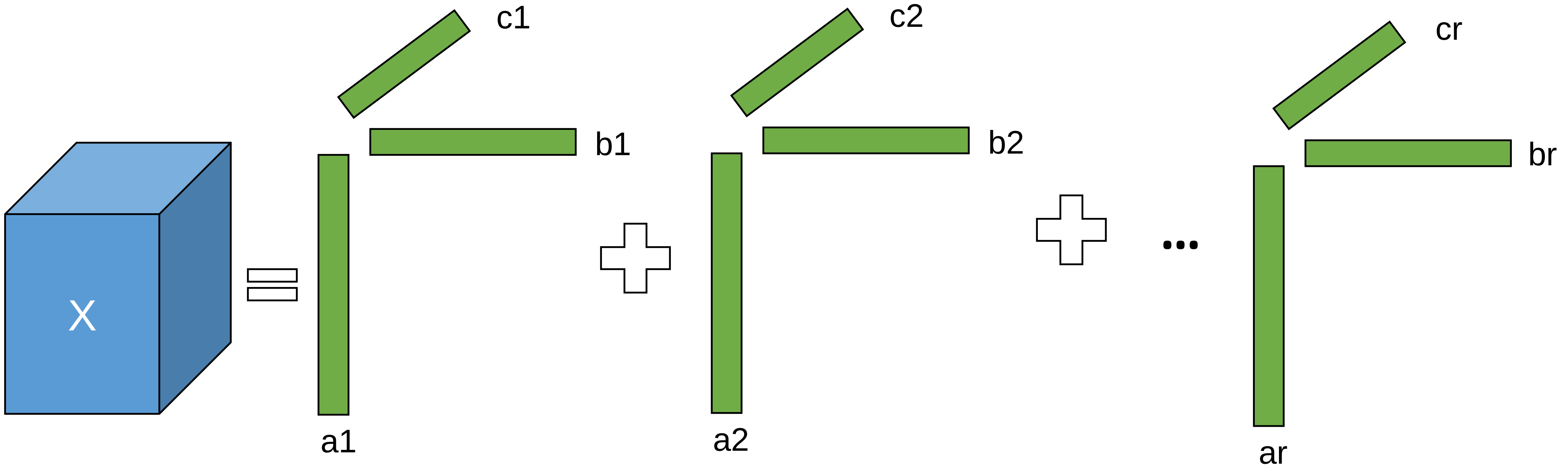


Figure 1: (a) The original convolutional layer acting on a single-channel input *i.e.* C=1. (b) The approximation to that layer using the method of Scheme 1. (c) The approximation to that layer using the method of Scheme 2. Individual filter dimensions are given above the filter layers.

Speeding up Convolutional Neural Networks with Low Rank Expansions

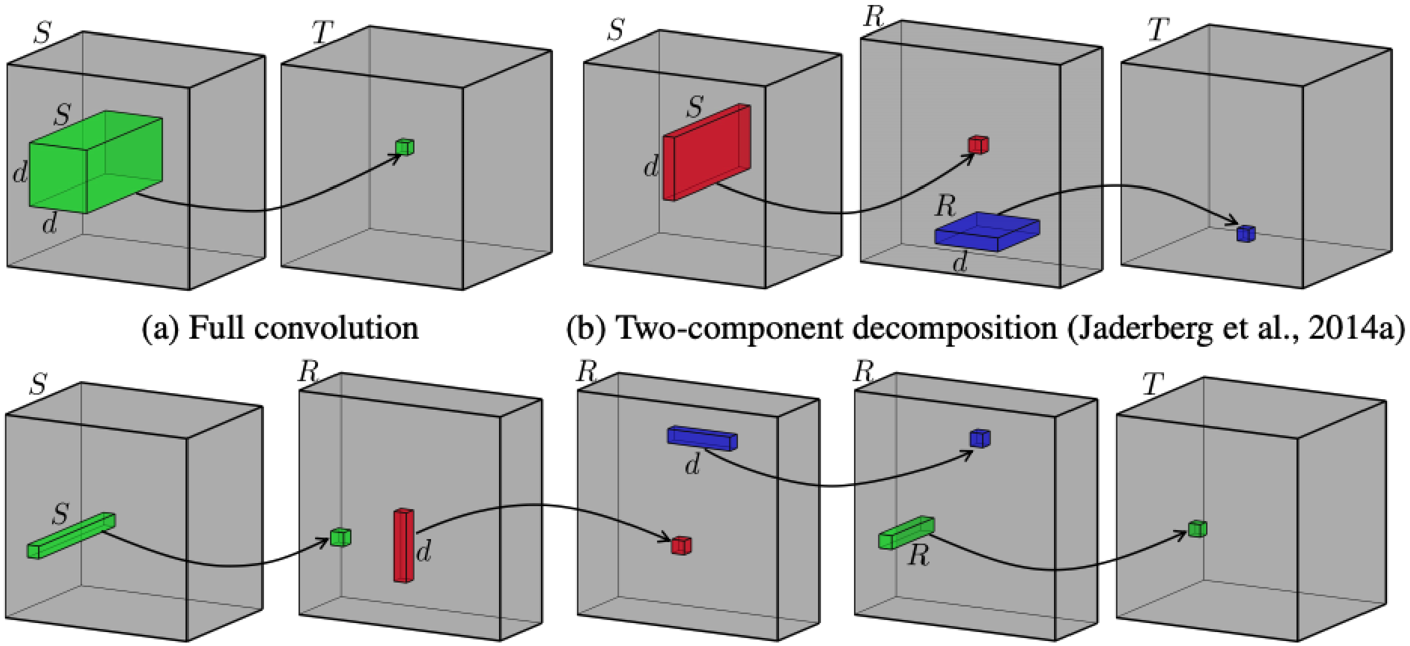
Tensor Decomposition in DNN (2):

- CP decomposition: factorizes a tensor into a sum of outer products of vectors.



Tensor Decomposition in DNN (2):

- CP decomposition: factorizes a tensor into a sum of outer products of vectors.



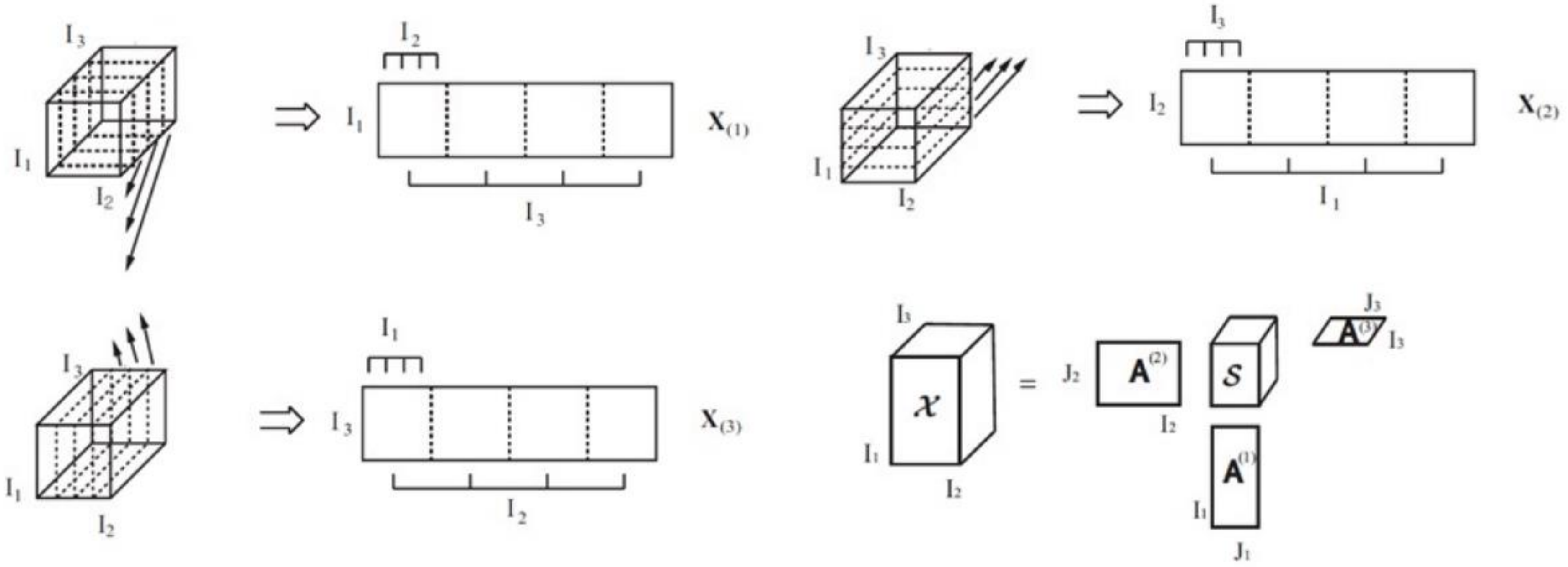
(a) Full convolution (b) Two-component decomposition (Jaderberg et al., 2014a)

(c) CP-decomposition

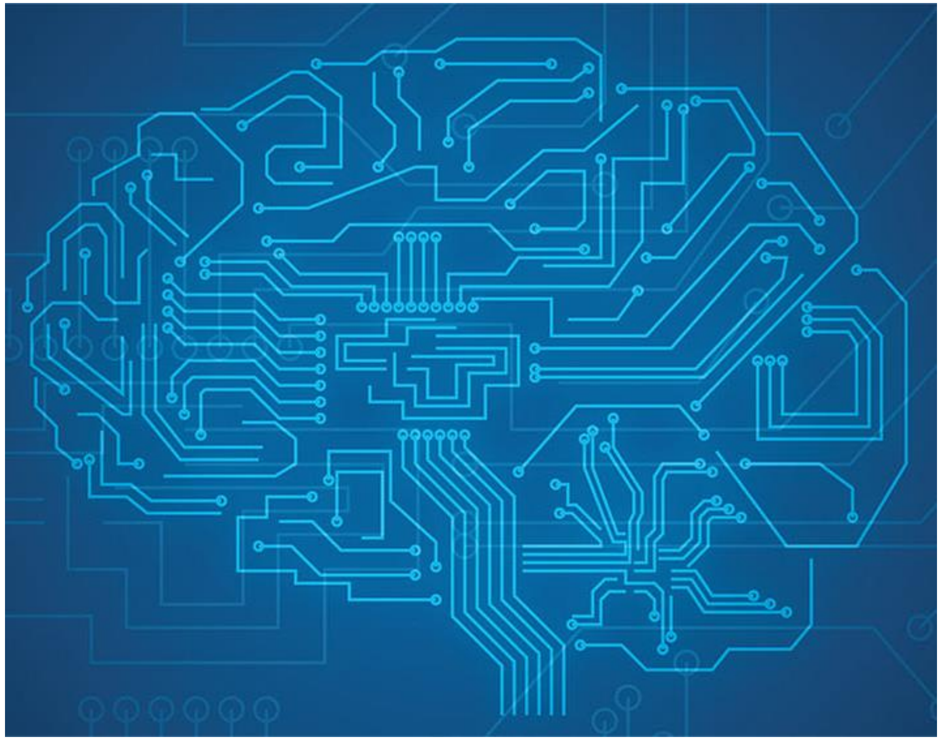
SPEEDING-UP CONVOLUTIONAL NEURAL NETWORKS USING FINE-TUNED CP-DECOMPOSITION

Tensor Decomposition in DNN (3):

- Tucker decomposition: decomposes a tensor into a core tensor multiplied by a matrix along each mode.



COMPRESSION OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR FAST AND LOW POWER MOBILE APPLICATIONS

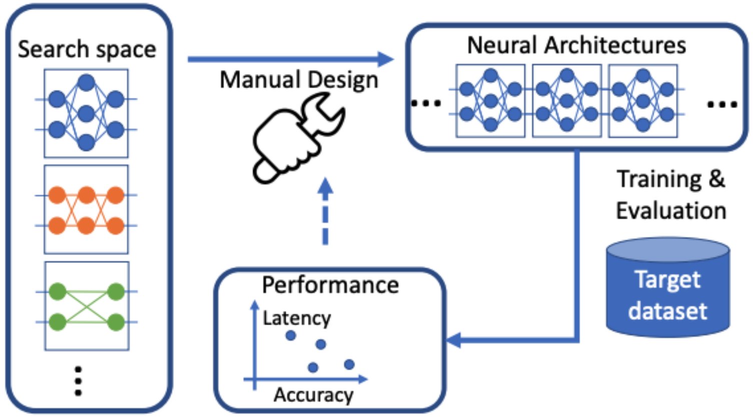


Codesign

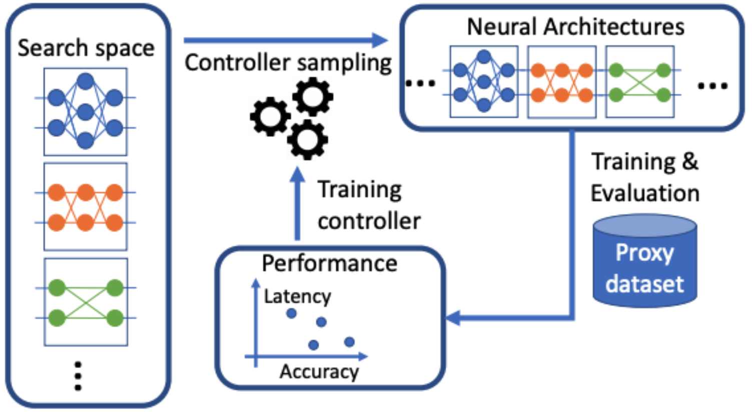
- Implications on HW
- HW-Aware Design
  - Pruning
  - Compact Network
  - Neural Arch. Search

Neural Architecture Search (NAS)

- Manually determine the network architecture is a tedious process.
    - Large number of hyperparameters, e.g., # of layers, connections between layers, and types of layers
- NAS is proposed to automatically search for the optimal architecture at the cost of more computation.



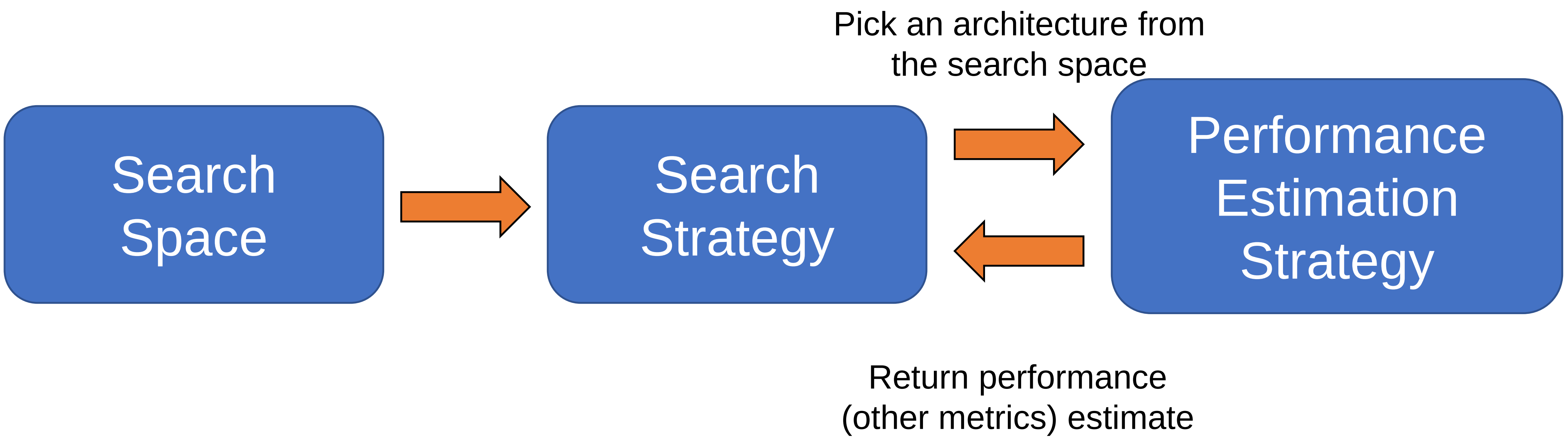
(a) A typical flow of manual ConvNet design.



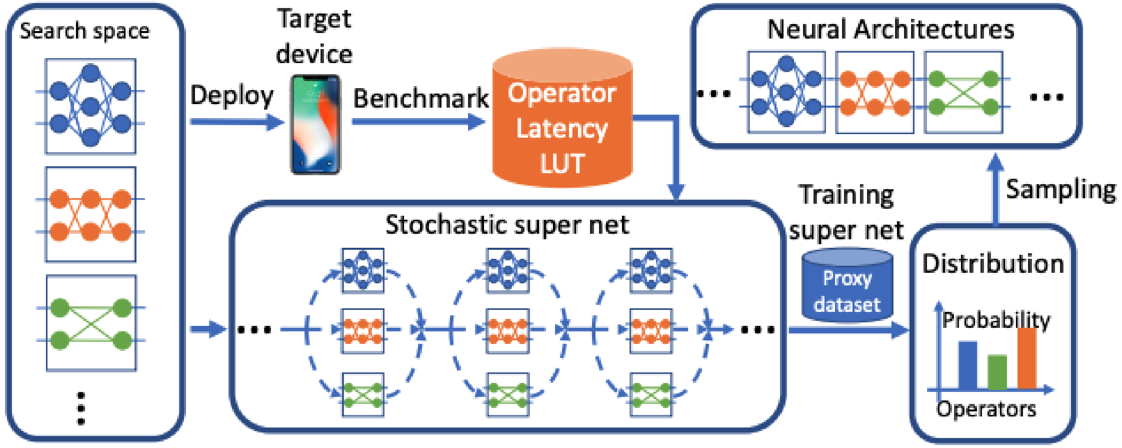
(b) A typical flow of reinforcement learning based neural architecture search.

FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

Neural Architecture Search (NAS)



Neural Architecture Search (NAS)



Model	#Parameters	#FLOPs	Latency on iPhone X	Latency on Samsung S8	Top-1 acc (%)
FBNet-iPhoneX	4.47M	322M	19.84 ms (target)	23.33 ms	73.20
FBNet-S8	4.43M	293M	27.53 ms	22.12 ms (target)	73.27

Table 5. FBNets searched for different devices.

FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search

Review

- Core computation in DNN
- Execution order of the core computation
- Hardware realization of the core computation
- Mapping DNNs to hardware
- Data transfer mechanisms across storage hierarchy
- Sparsity in DNNs
- This Lecture: Codesign example
  - Implications on HW
  - HW-aware design:
    - pruning, compact network design, neural architecture search