Improving Efficiency in Neural Network Accelerator using Operands Hamming Distance Optimization

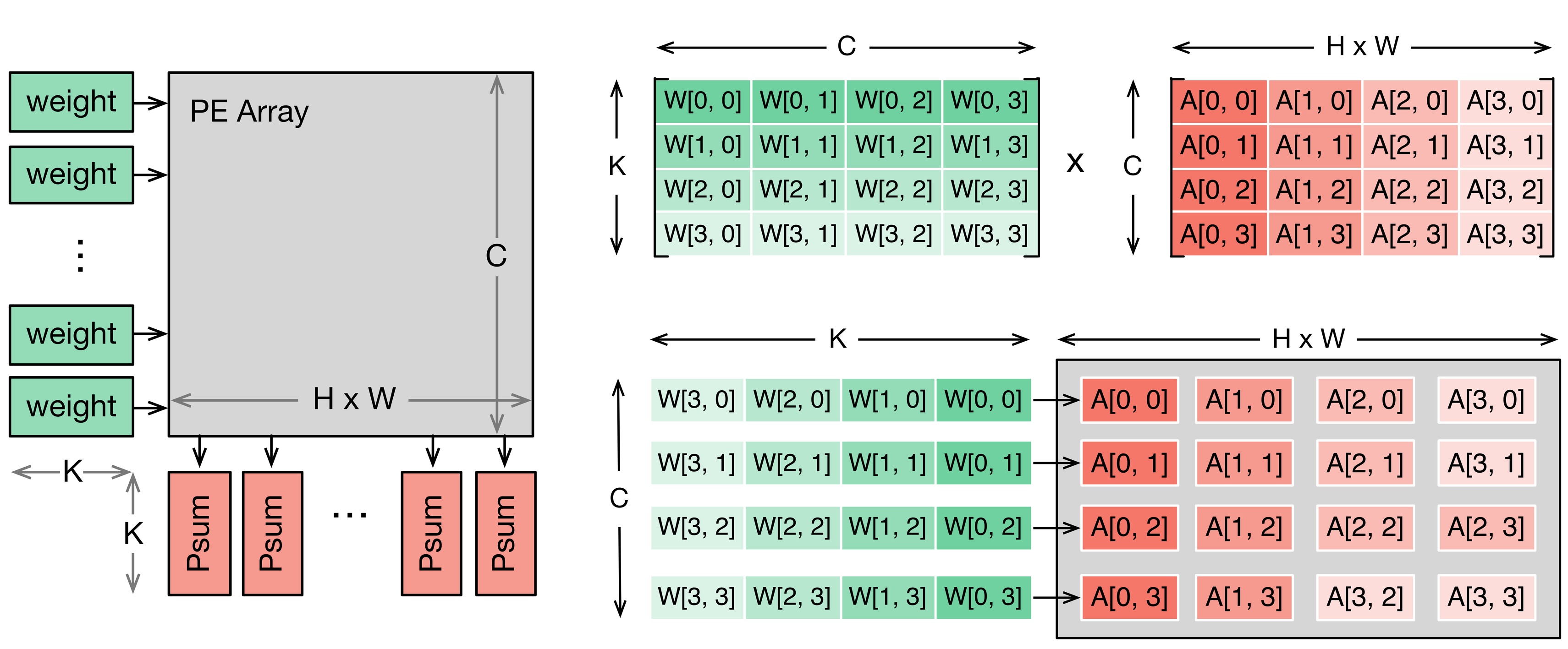
Meng Li*, Yilei Li*, Pierce Chuang, Liangzhen Lai, and Vikas Chandra
Facebook Silicon AI Research

Motivation: Datapath Energy Determined by Hamming Distance of Operand Streaming

Dataflow processing is widely exploited in NN accelerators

- Enable data reuse among processing elements (PE)
- Amortize memory access energy

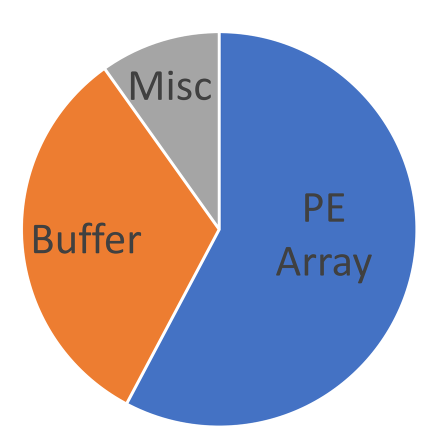
A common example is systolic array with input/output stationary dataflow



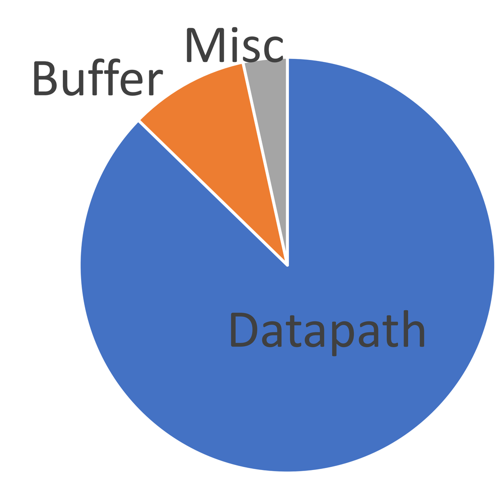
Datapath energy is important for dataflow accelerators

- Consist of compute energy in process elements (PEs) and data transfer energy among PEs
- Datapath energy is determined by the total bit flips induced by operand streaming

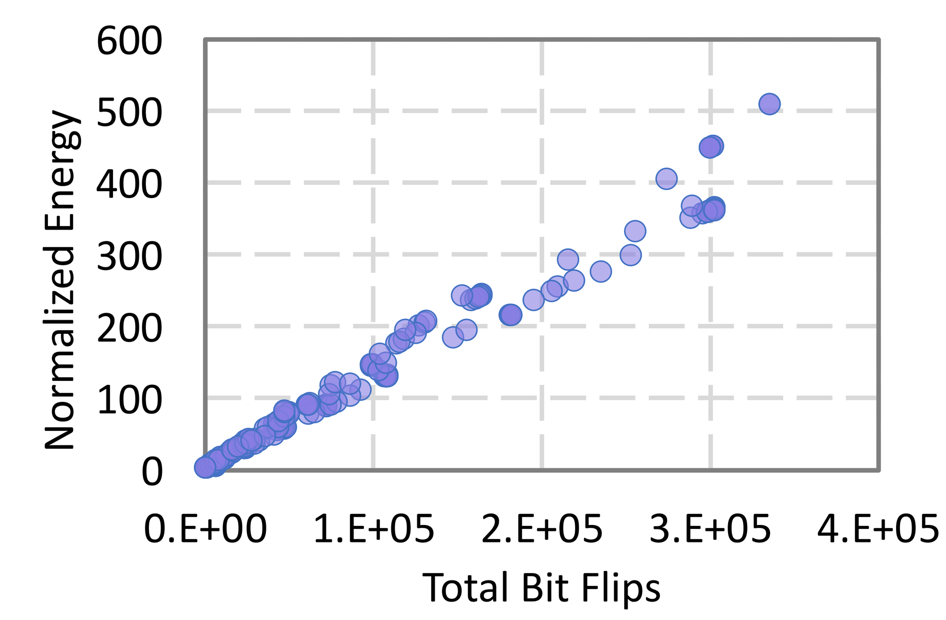
Target: propose both post-training and training aware techniques to reduce bit flips and thus, datapath energy



Thinker [Yin+, JSSC'18]



ShiDianNao [Du+, ISCA'15]



Hamming Distance Reduction through Post-Training and Training-Aware Techniques

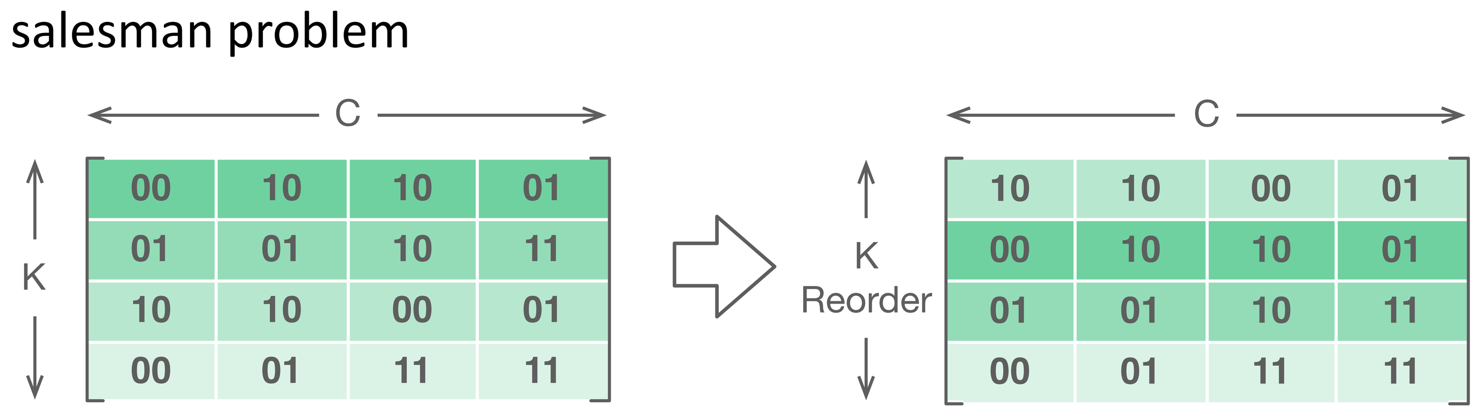
Bit flips of streaming weight can be captured by hamming distance (HD):

$$HD(W) = \sum_{j=1}^{K-1} \sum_{i=1}^{C} HD(W[j,i], W[j+1,i])$$

- W, K, C denotes the weight matrix, output channel, input channel
- HD denotes the hamming distance between two weight elements

To reduce HD, most straightforward method is output channel reordering

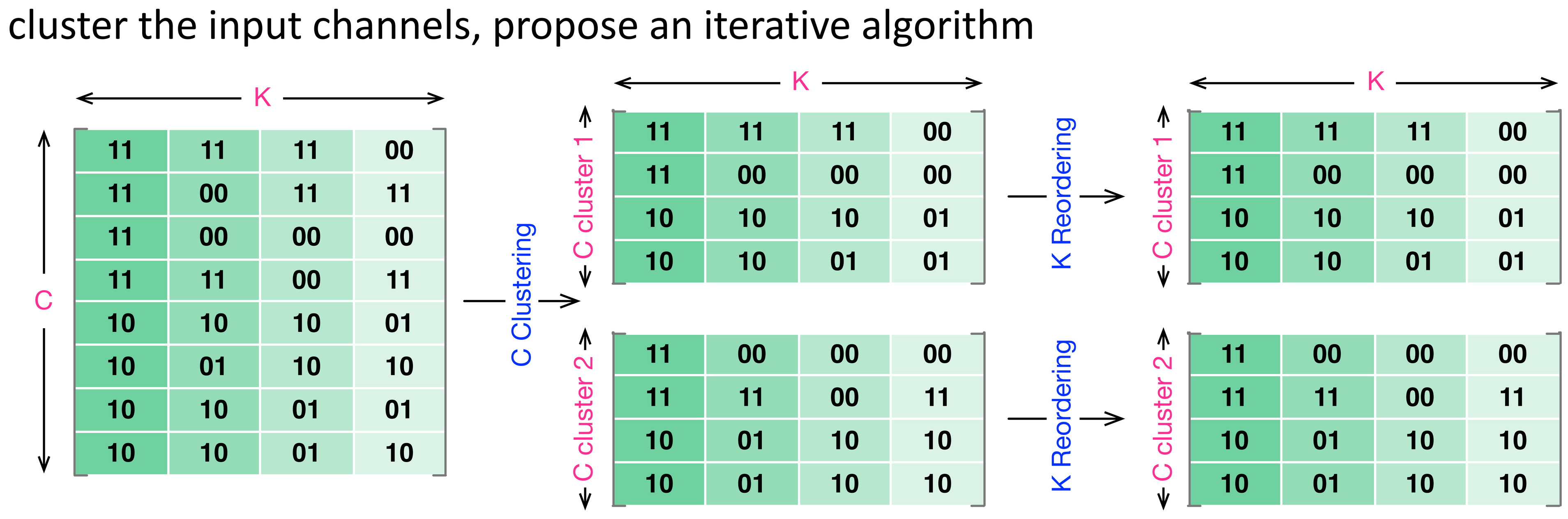
- To determine optimal channel order is equivalent to solving the traveling salesman problem



To further reduce HD, we segment the weight matrix before reordering

- For many networks, input channel dim is larger than the PE array size
- Each weight sub-matrix can use different output channel orders
- Cluster different input channels before the segment

To cluster the input channels, propose an iterative algorithm

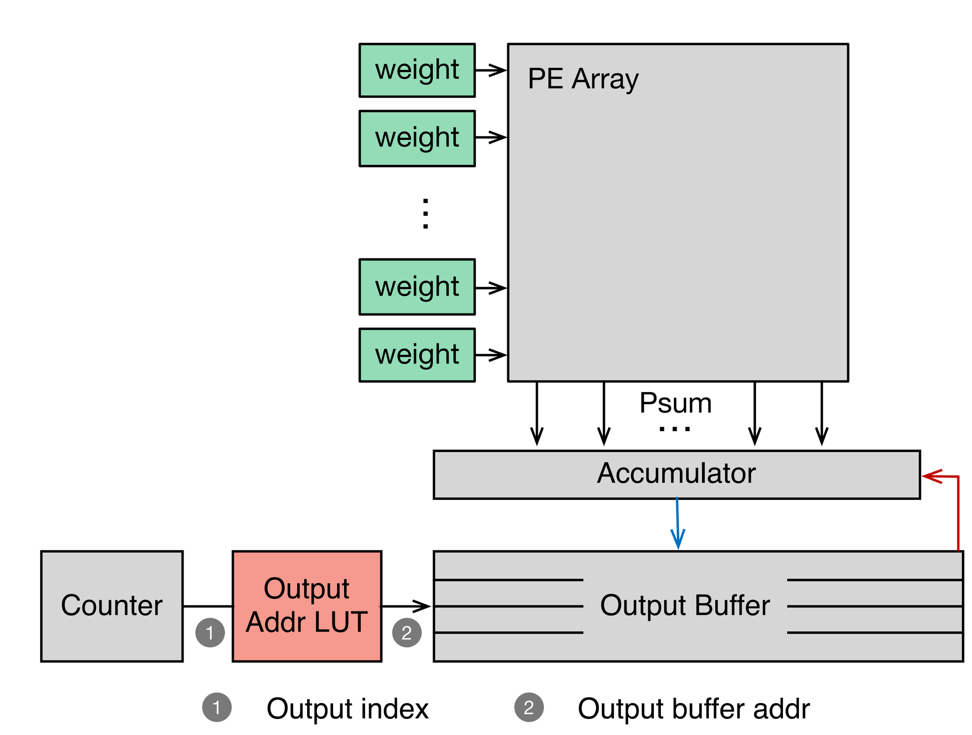


Input: weight matrix $W \in \mathbb{R}^{K \times C}$, number of iterations N, number of clusters t
Output: cluster of input channels $\{T_1^*, \ldots, T_t^*\}$ and the optimal sequence of output channels $\{S_1^*, \ldots, S_t^*\}$

```
{S_1^(0), ..., S_t^(0)}, n ← RANDOM_INITIALIZE(), 0
while n ≤ N do
    // Assignment step
    for i = 1 : C do
        l ← argmin_k HD_{S_k^(n)}(W[:, i])
        T_l^(n) ← T_l^(n) ∪ {i}
    end
    // Update step
    for i = 1 : t do
        S_i^(n+1) ← argmin_S HD_S(W_{T_i})
    end
    n += 1
end
{S_1^*, ..., S_t^*} ← {S_1^(N), ..., S_t^(N)}
{T_1^*, ..., T_t^*} ← {T_1^(N), ..., T_t^(N)}
```

Hardware support for cluster-then-reorder algorithm

- Only a small LUT is needed → small overhead

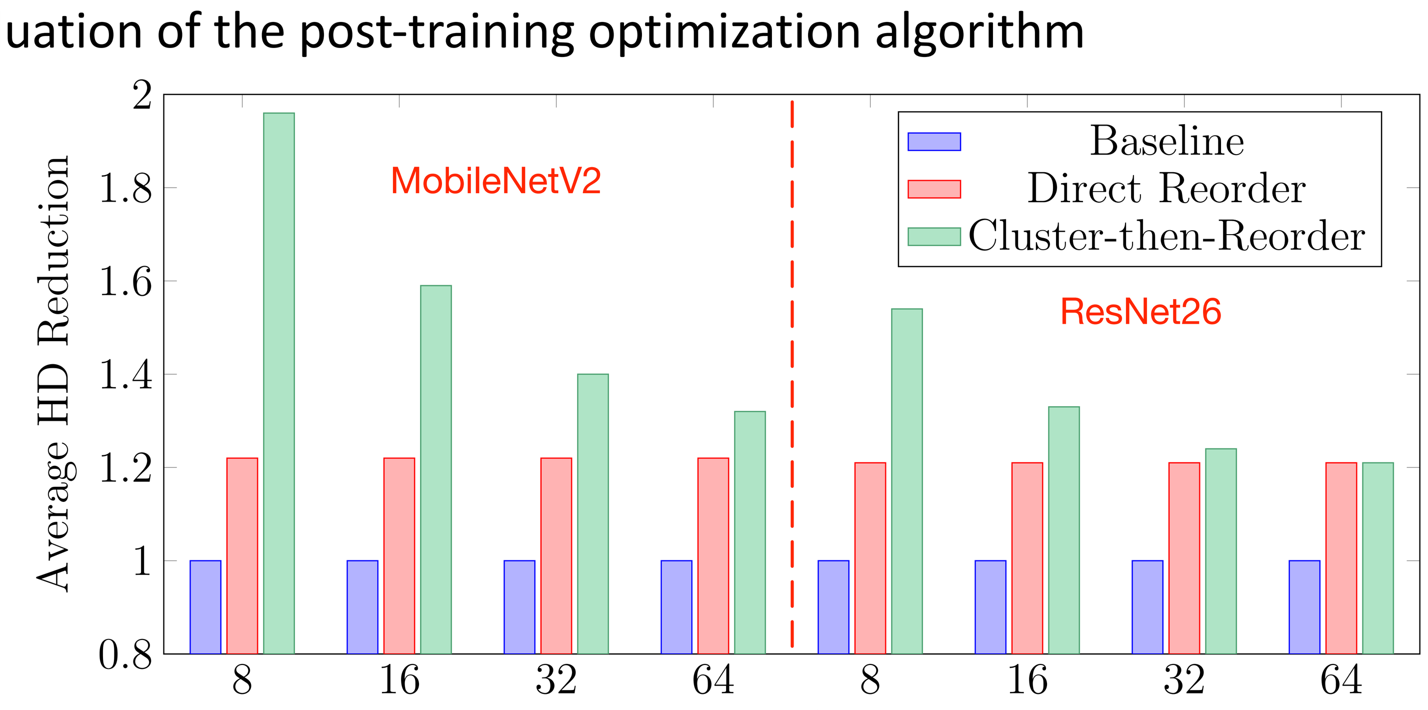


Experimental Results

Use MobileNetV2 and ResNet26 trained on Cifar10/Cifar100 for evaluation

- Select 1x1 Conv in MobileNetV2 and 3x3 Conv in ResNet26

Evaluation of the post-training optimization algorithm

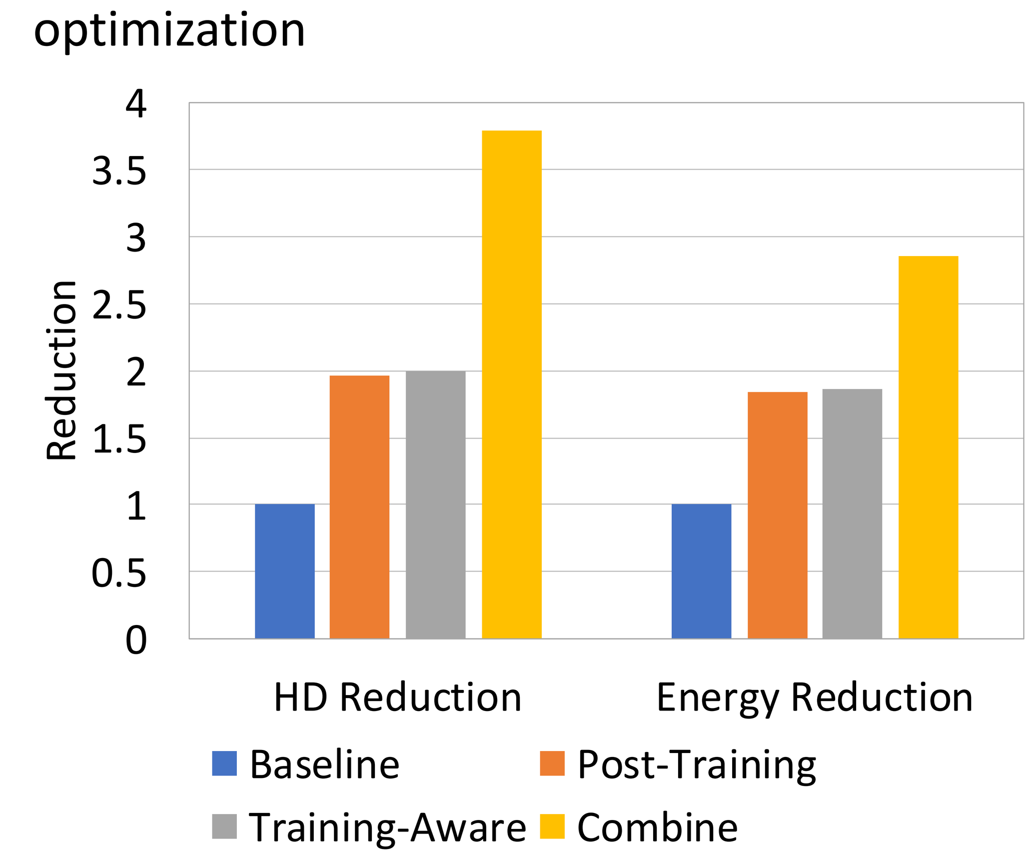


Training-aware optimization:

- Add HD regularization to the training

Dataset	λ	Top-1 Acc	Average HD Reduction
Cifar10	0.0	94.38	1.0×
	1×10^{-4}	94.22	7.55×
Cifar100	0.0	78.21	1.0×
	1×10^{-5}	77.98	1.24×
	3×10^{-5}	77.47	1.50×
	5×10^{-5}	77.29	1.76×
	7×10^{-5}	77.62	2.00×

Combine post-training and training-aware optimization



The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019