Improving Efficiency in Neural Network Accelerator using Operands Hamming Distance Optimization

Meng Li[*], Yilei Li[*], Pierce Chuang, Liangzhen Lai, Vikas Chandra
{meng.li, yileil, pichuang, liangzhen, vchandra}@fb.com
Facebook Inc.

Abstract

Neural network accelerator is a key enabler for the on-device AI inference, for which energy efficiency is an important metric. The datapath energy, including the computation energy and the data movement energy among the arithmetic units, claims a significant part of the total accelerator energy. We find that the datapath energy is highly correlated with the bit flips when streaming the input operands into the arithmetic units, defined as the hamming distance of the input operand matrices. Based on the insight, we propose a post-training optimization algorithm and a hamming-distance-aware training algorithm to co-design and co-optimize the accelerator and the network synergistically. The experimental results based on post-layout simulation with MobileNetV2 demonstrate on average 2.85$\times$ datapath energy reduction and up to $8.51\times$ datapath energy reduction for certain layers on an input stationary compute array.

1 Introduction

Deep neural networks (DNNs) have revolutionized different applications ranging from computer vision to speech and natural language processing (LeCun et al., 2015), and are now widely deployed in data centers (Jouppi et al., 2017) and on edge devices (Wu et al., 2019). As modern DNNs usually require significant computation, neural network accelerators are extensively studied in recent years to enable energy-efficient processing (Sze et al., 2017).

Datapath of the neural network accelerator, including the arithmetic compute units and the data bus among the units, lies at the heart of neural network accelerators. The datapath energy consumption of a neural network accelerator can be estimated as $E_{datapath} = \lambda \cdot OPs \cdot Energy/OP$, where OPs denotes the total number of operations of the neural network, $Energy/OP$ is the datapath energy consumption of one operation and λ is a correction term that depends on the network parameters and the underlying hardware design. While previous researches focus on leveraging network architecture search, pruning, or quantization to reduce OPs and $Energy/OP$ (Sze et al., 2017), reducing λ receives less attention.

In this work, we explore a new dimension to reduce the datapath energy by reducing λ. We show that as most accelerators leverage spatial data reuse (Chen et al., 2016) and stream input operands into the compute array, the sequence of the input operands significantly impacts the datapath energy. Specifically, we find that the datapath energy is strongly correlated to the bit flips when streaming the input operands. In this paper, we leverage the concept of hamming distance to formalize the bit flip analysis. A series of post-training and training-aware techniques are proposed to co-design and co-optimize the accelerator and the network to reduce the hamming distance of the input operand sequence. Experimental results based on the post-layout simulation demonstrates on average 3.6$\times$ datapath energy reduction and up to $8.51\times$ energy reduction for certain layers.

[*]Equal Contribution

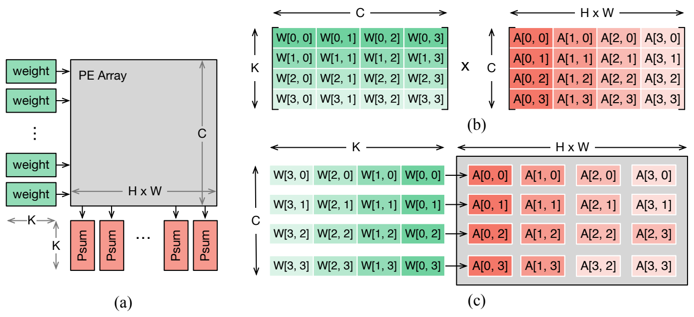
weight
PE Array
C
H x W
K
Psum
(a)
C
K
W[0, 0] W[0, 1] W[0, 2] W[0, 3]
W[1, 0] W[1, 1] W[1, 2] W[1, 3]
W[2, 0] W[2, 1] W[2, 2] W[2, 3]
W[3, 0] W[3, 1] W[3, 2] W[3, 3]
x
H x W
A[0, 0] A[1, 0] A[2, 0] A[3, 0]
A[0, 1] A[1, 1] A[2, 1] A[3, 1]
A[0, 2] A[1, 2] A[2, 2] A[3, 2]
A[0, 3] A[1, 3] A[2, 3] A[3, 3]
(b)
(c)

Figure 1: Mapping 1-by-1 convolution to the input stationary array (H, W, K, C denotes the output height, width, output channel, and input channel dimension, respectively).

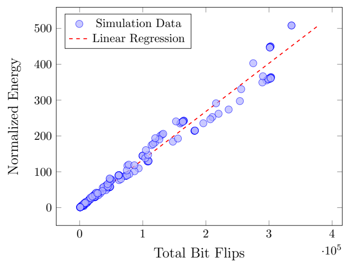

Figure 2: Total bit flips of the weight sequence and the datapath energy demonstrate strong correlation.

2 Background and Motivation

Modern DNN accelerators leverage specialized processing dataflows to encourage data reuse and reduce memory access (Sze et al., 2017). Figure 1(a) shows a widely used input stationary design. Consider the example of a 1-by-1 convolution in Figure 1(b). The weights are streamed into the array and can be reused horizontally, while the partial sums are accumulated spatially across the column.

The energy consumption of the accelerator is composed of the datapath energy (including the arithmetic computation energy and the data propagation energy among compute units), the memory access energy and the control energy. With the trend of aggressive operand quantization (< 8 bit) and specially designed dataflows for data reuse, datapath energy can consume 40-70% of the total energy in many state-of-the-art accelerator designs (Andri et al., 2016; Gao et al., 2017; Park et al., 2018).

The datapath energy mainly comprises of the switching energy and glitch energy, both of which are caused by the circuit nodes switching from 0 to 1 or from 1 to 0, denoted as bit flips. For the datapath, the bit flips are determined by the value and the streaming pattern of the input operands, i.e., weights, input activations, and partial sums. Because the activation and partial sums are input dependent, we focus on analyzing the impact of weight matrices. Specifically, we use the post-layout simulation to evaluate the relation between the bit flips of streaming the weight matrices and the datapath energy. As shown in Figure 2, the total bit flips and datapath energy demonstrate a strong correlation. Hence, to reduce the datapath energy, in this paper, we propose post-training and training-aware techniques to reduce the bit flips of streaming the weight matrices.

3 Hamming Distance Optimization

In this section, we formalize the concept of bit flips and propose post-training and training-aware techniques to minimize the bit flips of the streaming weights. We use the input stationary compute array (e.g., Figure 1(a)) as an example throughout the analysis but the definition, analysis, and conclusion can be easily applied to other dataflow schemes once the weights are streamed into the compute array.

Problem Formulation We define the hamming distance between two B-bit numbers a and b as

$$\mathrm{HD}(a, b) = \sum_{i=1}^{B} \mathrm{Bit}_i(a) \oplus \mathrm{Bit}_i(b),$$

where $\oplus$ denotes the XOR operation and $\mathrm{Bit}_i(\cdot)$ is the function that extracts the i-th bit of the number.

Consider a weight matrix $W \in \mathbb{R}^{K \times C}$[2]. As the input stationary dataflow unrolls the input channel dimension (C) along the compute array column direction and stream the weights along different

[2]We assume $F_x = F_y = 1$ for the weight matrix in this case, but the definition and analysis can be easily extended to cases where F_x and F_y are larger than 1.

output channels (K) in temporal sequence, we define the hamming distance of streaming W as

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

Output Channel Reordering A straightforward technique to minimize $\mathrm{HD}(W)$ is to reorder the sequence of W streaming into the compute array. Let S denote the sequence of output channels and $\mathrm{HD}_S(W)$ denote the hamming distance of streaming W following the sequence S, then, we have

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

The output channel reordering problem is defined as follows.

Problem 1 *(Output Channel Reordering) Given a weight matrix $W \in \mathbb{R}^{K\times C}$, find S^* such that $\mathrm{HD}_{S^*}(W)$ is minimized, i.e.,*

$$S^* = \mathrm{argmin}_S \mathrm{HD}_S(W).$$

The reordering problem can be mapped to the Traveling Salesman Problem (TSP), the complexity of which scales exponentially with the number of output channels. We propose a greedy reordering algorithm to compute S^*: we initialize the sequence S^* by assigning the first output channel to the starting position of S^*. Then, the channel with the smallest hamming distance compared to the previous channel is added to S^*. The complexity of the algorithm scales quadratically with K.

Input Channel Clustering and Segmentation For the convolution layer with large C, limited by the compute array size, not all the input channels can be streamed into the array simultaneously. Hence, the weight matrices need to be segmented first. Instead of directly segmenting the matrices, we propose to cluster the input channels first and then, compute the optimal sequence S^* for each sub-matrix separately.

Let $\{T_1, \ldots, T_t\}$ denote the t clusters of the input channel. The input channel clustering problem is then defined as follows.

Problem 2 *(Input Channel Clustering) Given a weight matrix $W \in \mathbb{R}^{K\times C}$, find t clusters $T_1, \ldots, T_t$ such that the total hamming distance of streaming each sub-matrix W_{T_i} is minimized, i.e.,*

$$\begin{aligned} \min_{T_1,\ldots,T_t} \quad & \sum_{i=1}^{t} \mathrm{HD}_{S_i^*}(W_{T_i}) \\ \text{s.t.} \quad & S_i^* = \mathrm{argmin}_S \mathrm{HD}_S(W_{T_i}) \\ & T_i \cap T_j = \emptyset \quad \forall i \neq j \\ & \cup_{i=1}^{t} T_i = \{1, \ldots, C\} \end{aligned}$$

We propose the cluster-then-reorder algorithm, which is a greedy iterative method to solve the nested optimization problem: the input channels are first randomly assigned to t clusters, then, the algorithm alternates between the following two steps:

- Update step: for each cluster $i \in \{1, \ldots, t\}$, compute the optimal sequence S_i.
- Assignment step: for each channel $l \in \{1, \ldots, C\}$, evaluate the hamming distance following the optimal sequence of each cluster and assign l to the cluster i with the smallest hamming distance.

Hamming Distance-Aware Training We also propose a hamming distance-aware training procedure to further reduce the hamming distance when streaming W. The basic idea is to incorporate the hamming distance into the loss function and encourage the hamming distance reduction:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{HD}(W),$$

where $\mathcal{L}_{CE}$ represents the original cross-entropy loss. λ is used to control the trade-off between the accuracy and the hamming distance reduction.

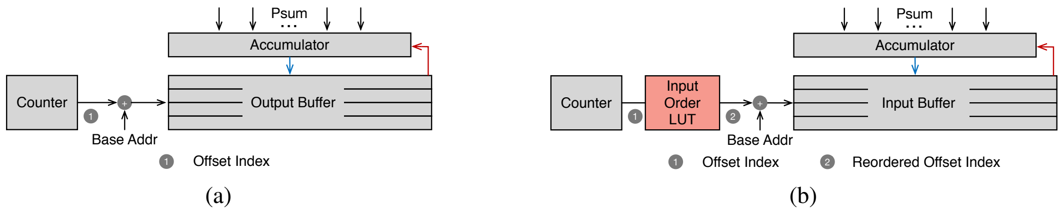

Figure 3: LUT addition to the accumulation buffer to guarantee the correct accumulation and reorder output activations.

Table 1: Training-aware hamming distance optimization for MobileNetV2 on Cifar10 and Cifar100.

DATASET	λ	TOP-1 ACC	TOP-5 ACC	AVERAGE HD REDUCTION	AVERAGE ENERGY REDUCTION
CIFAR10	0.0	94.38	99.82	1.0×	1.0×
	1×10^{-4}	94.22	99.00	7.55×	6.63×
CIFAR100	0.0	78.21	94.53	1.0×	1.0×
	1×10^{-5}	77.98	94.20	1.24×	1.26×
	7×10^{-5}	77.62	94.26	2.00×	1.86×

However, there are two main problems with $\mathcal{L}_{HD}(W)$: 1) $\mathcal{L}_{HD}$ is not differentiable and 2) the input channel clusters and output channel orders are needed to compute $\mathcal{L}_{HD}$ and get updated in the training process. We leverage the straight-through estimator to approximate the gradients of $\mathcal{L}_{HD}$. Meanwhile, after each epoch of training, we leverage the cluster-then-reorder algorithm to cluster the input channels and reorder the output channels.

4 Hardware Support

The direct reordering algorithm only switches the weights along the output channel channels, and hence, change the sequence of the output channel generation. A post-training processing of the model to re-arrange the channels is sufficient with no hardware support needed.

The cluster-then-reorder algorithm clusters the input channels into groups and reorders the output channels for each group separately, which has two implications: 1) the sequences of the output channel generation are different for different clusters; and 2) the sequence of output channel generation may be different than the required sequence of input channel streaming in the next layer.

We add an output address lookup table (LUT) to the accumulation buffer to support the cluster-then-reorder algorithm. As shown in Figure 3, the address LUT translates the index of the counter in the accumulator to the actual address for accumulation. The LUT can help guarantee the correct accumulation of partial sums across different input channel clusters and reorder the output activations for the next layer. If the output buffer depth to be D, the LUT needs to have at least D entries and each entry needs to have $\log_2 D$ bits. For a reasonable output buffer depth, e.g., 1024, the LUT SRAM size is less than 2 KB, which is very small and thus has negligible energy and area overhead.

5 Experimental Results

Experimental Setup We select the 1-by-1 convolution layers in MobileNetV2 (Sandler et al., 2018) and the 3-by-3 convolution layers in ResNet26 (He et al., 2016) trained on the Cifar10 and Cifar100 dataset for the evaluation. To evaluate the datapath energy, we use post-layout simulation. We designed an input-stationary systolic array with 8 rows and 8 columns. Each compute unit supports 8-bit activations and 4-bit weights. The leakage energy is ignored in the evaluation.

Post-Training Hamming Distance Optimization We first compare the effectiveness of the post-training hamming distance optimization algorithms, i.e., the direct reorder and cluster-then-reorder algorithms. As shown in Figure 4, the direct reordering algorithm achieves 1.2 × hamming distance

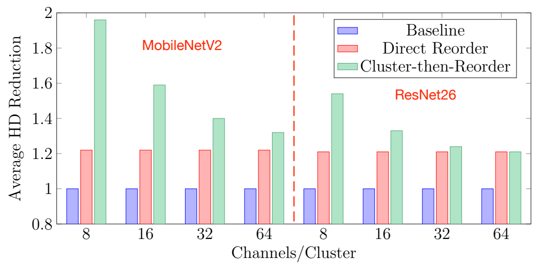

Figure 4: Comparison of the post-training optimization techniques for HD reduction on MobileNetV2 (left) and ResNet26 (right).

	AVERAGE HD REDUCTION	AVERAGE ENERGY REDUCTION
BASELINE	1.0×	1.0×
$\lambda = 0$, CTR	1.96×	1.84×
$\lambda = 7 \times 10^{-5}$	2.00×	1.86×
$\lambda = 7 \times 10^{-5}$, CTR	3.79×	2.85×

Figure 5: Average hamming distance and energy reduction of the combined methods (CTR is short for the cluster-then-reorder).

reduction for MobileNetV2 and ResNet26. The cluster-and-reorder algorithm reduces the hamming distance by 1.96× and 1.54× for MobileNetV2 and ResNet26, respectively, when the number of input channels per cluster is 8, which translate to 1.62× and 1.49× datapath energy reduction.

Training-Aware Hamming Distance Optimization We select MobileNetV2 and train the network on Cifar10 and Cifar100 datasets. We constrain the accuracy degradation within 1%. As shown in Table 1, on Cifar10 dataset, the average hamming distance can be reduced by 7.55×, which leads to 6.63× reduction of the average datapath energy across layers. On Cifar100 dataset, the average hamming distance reduction and the average energy reduction are 2.00× and 1.86×, respectively.

Combined Hamming Distance Optimization As shown in Table 5, by combining the proposed optimization techniques, for MobileNetV2 trained on Cifar100, the average hamming distance can be reduced by 3.79× and the average datapath energy can be reduced by 2.85×.

6 Conclusion

Datapath energy of a neural network accelerator is heavily dependent on the hamming distance of streaming the input operands. The paper proposes hamming-distance-aware training and post-training algorithms to reduce the hamming distance and the datapath energy. Evaluation with MobileNetV2 and ResNet neural networks shows that our proposed methods achieve 2.85× datapath energy reduction on average and up to 8.51× datapath energy reduction for certain layers.

References

Andri, Renzo, Cavigelli, Lukas, Rossi, Davide, and Benini, Luca. Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. *ISVLSI*, pp. 236–241, 2016.

Chen, Yu-Hsin, Krishna, Tushar, Emer, Joel S, and Sze, Vivienne. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *JSSC*, 52(1):127–138, 2016.

Gao, Mingyu, Pu, Jing, Yang, Xuan, Horowitz, Mark, and Kozyrakis, Christos. Tetris: Scalable and efficient neural network acceleration with 3d memory. In *ASPLOS*, 2017.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Jouppi, Norman P., Young, Cliff, Patil, Nishant, Patterson, David, and et al. In-datacenter performance analysis of a tensor processing unit. In *ISCA*, 2017.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *nature*, 521(7553):436, 2015.

Park, Eunhyeok, Kim, Dongyoung, and Yoo, Sungjoo. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *ISCA*, 2018.

Sandler, Mark, Howard, Andrew, Zhu, Menglong, Zhmoginov, Andrey, and Chen, Liang-Chieh. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pp. 4510–4520, 2018.

Sze, V., Chen, Y., Yang, T., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2017.

Wu, Carole-Jean, Brooks, David, Chen, Kevin, Chen, Douglas, Choudhury, Sy, Dukhan, Marat, Hazelwood, Kim, Isaac, Eldad, Jia, Yangqing, Jia, Bill, et al. Machine learning at facebook: Understanding inference at the edge. In *HPCA*, pp. 331–344. IEEE, 2019.