A 1.40mm^2 141mW 898GOPS Sparse Neuromorphic Processor in 40nm CMOS

Phil Knag, Chester Liu, Zhengya Zhang
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor

Abstract

Sparsity is a brain-inspired property that enables a significant reduction in workload and power dissipation of deep learning. This work presents a 1.40mm^2 40nm CMOS sparse neuromorphic processor that implements a two-layer convolutional restricted Boltzmann machine (CRBM) for inference and a support vector machine (SVM) classifier. The processor incorporates sparse convolvers to realize sparsity-proportional workload reduction. The architecture is parallelized along a non-sparse dimension to minimize stalling. At 0.9V and 240MHz, the processor achieves an effective 898.2GOPS performance, dissipating 140.9mW. Using sparsity, we reduce the workload, datapath power consumption and area by 3.4×, 3.3× and 1.74×, respectively. The design uses latch-based memory to reduce area and dynamic clock gating to save power.

Introduction

Deep learning is a powerful technique for big data analytics. Popular deep learning algorithms, e.g., convolutional deep belief network [1], rely on filtering an input using multiple layers of specialized kernels for detection and classification. These powerful algorithms demand intense computation for practical applications, as the input is often high in dimensionality, and the number of kernels and the kernel size need to be sufficiently large. Moreover, as we increase the depth of the network, the computational intensity and memory size grow much further. Prior work has demonstrated chip-, package-, board-level integration, and custom ASICs that accelerate deep learning up to 1.93TOPS/W [2], [3]. In this work, we present a sparse neuromorphic processor to exploit sparsity that is inherent in biologically inspired deep learning algorithms to enable the next order of magnitude improvement in performance and efficiency.

A Sparse Neuromorphic Processor

Sparsity is a key advantage of our approach, not only because of the physiological evidence that the brain uses a sparse representation to encode sensory inputs, but also due to its significant reduction of computational complexity. Enforcing sparsity has been shown to learn better features for classification [1], [4]. To introduce sparsity, a term of neuron activity is added to the cost function, and learning is done via sparsity regularization.

We design a sparse neuromorphic processor for object recognition based on convolutional deep belief network trained with sparsity regularization [1]. The processor adopts a representative 3-layer configuration (Fig. 1) that supports common image and video processing tasks. Layer 1 and 2 (L1 and L2) are CRBM layers that perform feature extraction. The outputs of L1 and L2 are max-pooled to create H1 and H2 respectively. Layer 3 (L3) is a SVM that performs classification using the sum of each channel from H2. The majority of the workload, i.e., 76M OP per 100×100 input patch (an OP is defined as an 8b multiply or a 16b add), is done in L2, followed by 18M OP in L1. Trained with sparsity regularization, the outputs of L1 and L2, i.e., H1 and H2, become sparse (Fig. 1). To achieve a high classification accuracy, the sparsity targets of H1 and H2 are tuned to ≥87.5%, i.e., ≥87.5% of H1 and H2 are zeros, enabling power reduction.

Sparse Convolver

The effective use of sparsity to reduce workload is a challenge for parallel architectures, as random sparse inputs result in redundant operations and memory contention. A conventional k×k patch convolver (Fig. 2(a)) that computes the convolution of a k×k kernel with an input works well for dense inputs. However, if the input is sparse, the patch convolver is inefficient, as multiplying by zero is wasteful. We design a sparse convolver (Fig. 2(b)) that utilizes a priority encoder to scan a stream of inputs at a time and forwards only the non-zero entries to a line convolver, skipping redundant operations. A k-pixel line convolver achieves k-way parallelism using k multipliers, and a (2k–1)-entry register bank with a k:(2k–1) selector for data alignment. The line convolver computes the convolution of an input pixel with a row in the kernel (Fig. 2(c)). The row address of the input pixel determines which row of the kernel to use, and the column address of the input pixel sets the selector for data alignment. Thanks to the predictable data dependency between consecutive line convolutions, the memory contention between consecutive convolutions is resolved by a simple selector. The selector only needs to access a small 1-D line of temporary outputs, as opposed to a large 2-D block. The line convolver outputs a line that is written to a contiguous memory region, which is not accessed again for the current kernel operation to save memory accesses.

With a target sparsity no less than 87.5%, an 8-pixel sparse convolver matches the throughput of an 8×8 patch convolver, but its power and area are 3.3× and 1.74× lower than the patch convolver. Since the input to L1 cannot be assumed to be sparse, L1 is implemented using patch convolvers. The more computationally intensive L2 uses sparse convolvers to save significant power and area.

Parallel Architecture Optimized for Sparse Inputs

A parallel architecture consisting of multiple convolvers could incur inefficiencies in processing a sparse input, as the irregularity of sparse entries causes variable completion times and stalling. A common way to allocate multiple convolvers is along one primary dimension (Fig. 3): pixel dimension (P-parallel), channel dimension (C-parallel), or kernel dimension (K-parallel). If an input is sparse in a dimension (often the P and C dimension), allocating convolvers along the sparse dimension results in uneven workloads among the convolvers. Instead, we choose to parallelize along the non-sparse K dimension. However, aggressively pursuing parallelism along one dimension leads to uneven memory bandwidths (Fig. 3), e.g., K-parallel requires a low input bandwidth, a high kernel and output bandwidth, whereas P-parallel has some opposite characteristics.

Guided by the insights, we design a 3× P-parallel and 2× K-parallel architecture for L1 using 6 8×8 patch convolvers (Fig. 4) to balance the input and weight bandwidths. Since the input to L2 is sparse, we design a 16× K-parallel and 2× P-parallel architecture for L2 using 32 8-pixel sparse convolvers (Fig. 4) to minimize stalling. Note that the 2× P-parallel is used to halve the weight bandwidth at the cost of 11% stalling. The latency of L1 and L2 processing are balanced for interleaving. The fluctuations in sparsity are smoothed out by H1 and H2 buffering. At 160MHz, the architecture meets the 30fps 1920×1080 HD video data rate.

Test Chip Design and Measurement

A 1.40mm^2 sparse neuromorphic processor (Fig. 5) is implemented in 40nm CMOS. The chip uses 40Kb registers to store weights and 12Kb registers to queue L2 outputs. As the weight and L2 output storage are not updated in a pipelined fashion, we replace the registers by latches that are 25% more compact than registers after routing. Since the weight storage, and H1 and H2 interface buffers are infrequently updated, dynamic clock gating is applied to turn off the clock input to save the power by 47%.

With a 0.9V supply and 240MHz frequency, the processor's measured throughput at room temperature is 96.4M pixel/s, consuming 140.9mW (Fig. 6). The processor takes advantage of sparsity to achieve an effective performance of 898.2GOPS. Tested with the Caltech 101 dataset [5] to identify faces (with 50% faces, 50% non-faces, 434 training images and 434 testing images), the processor achieves an 89% classification accuracy. The chip demonstrates a competitive power efficiency of 6.37TOPS/W and area efficiency of 641.6GOPS/mm^2, which are 3.3× and 15.6× higher than the state-of-the-art non-sparse deep learning processor [2] (Table I). With a scaled supply voltage of 0.65V, the efficiency improves to 10.98TOPS/W. The techniques demonstrated in this work are applicable to a class of sparse processing problems.

Acknowledgements
This work was supported in part by Intel, DARPA and SONIC.

References

[1] H. Lee, *et al.*, *ACM Commun.*, Oct. 2011.
[2] S. Park, *et al.*, *ISSCC*, 2015.
[3] J. Lu, *et al.*, *ISSCC*, 2014.
[4] J. K. Kim, *et al.*, *VLSI Symp.*, 2015.
[5] L. Fei-Fei, *et al.*, *IEEE CVPR*, 2004.

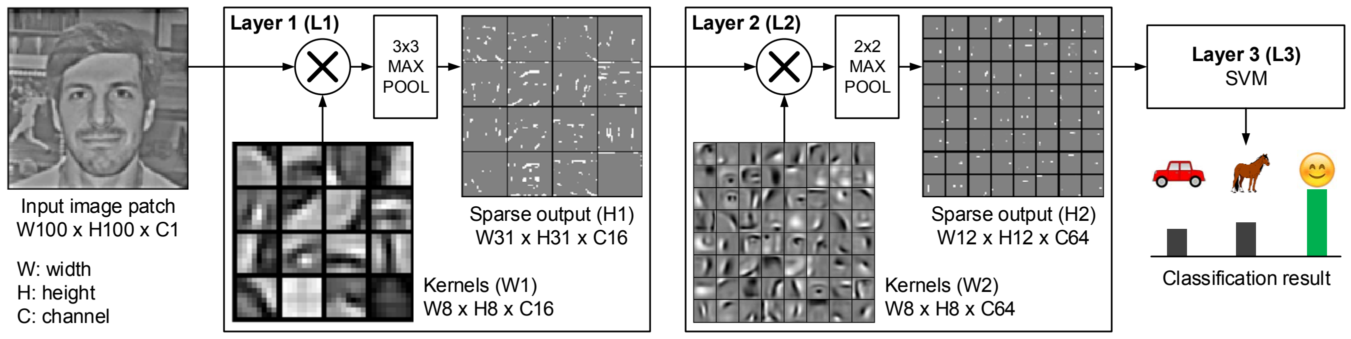


Fig. 1. Convolutional deep belief network composed of two layers of convolutional restricted Boltzmann machine (CRBM) layers and a support vector machine (SVM).

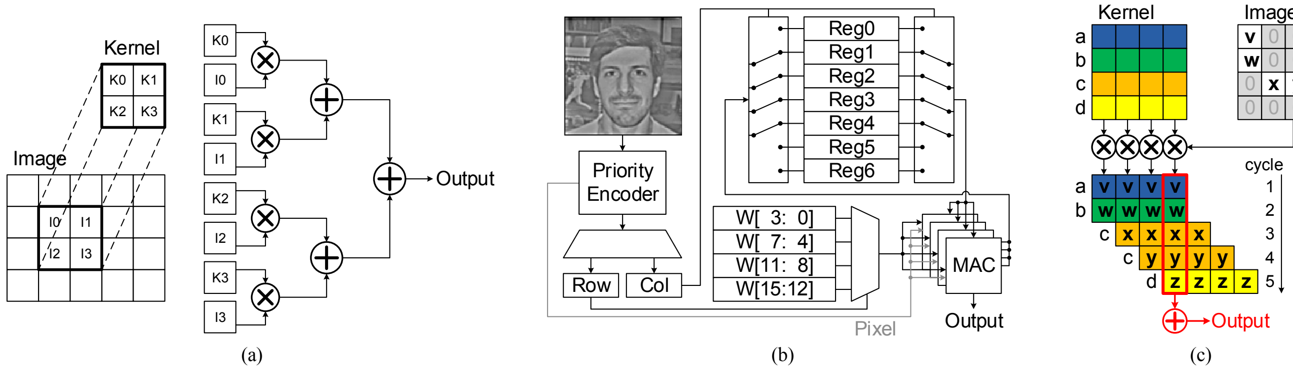
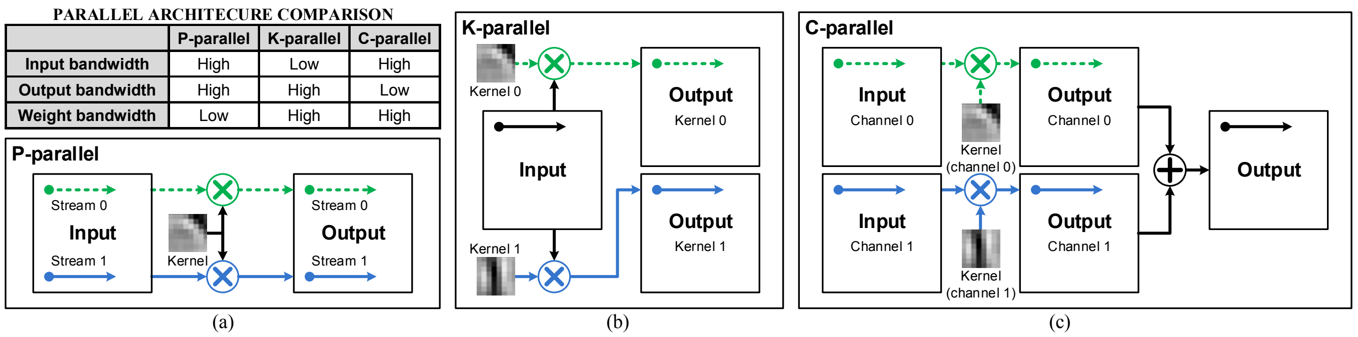


Fig. 2. (a) 2x2 patch convolver, (b) 4-pixel sparse convolver consisting of a priority encoder and a line convolver, (c) illustration of line convolver operation.



PARALLEL ARCHITECURE COMPARISON

	P-parallel	K-parallel	C-parallel
Input bandwidth	High	Low	High
Output bandwidth	High	High	Low
Weight bandwidth	Low	High	High

Fig. 3. Comparison of three types of parallel architectures: (a) pixel-parallel (P-parallel), (b) kernel-parallel (K-parallel), and (c) channel-parallel (C-parallel).

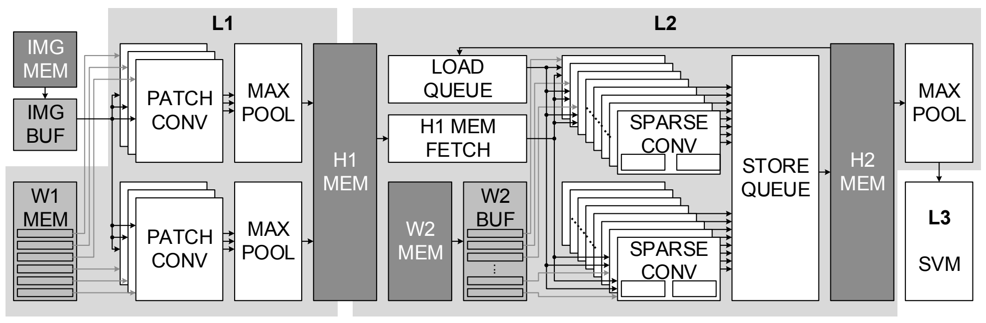


Fig. 4. Sparse deep learning processor architecture consisting of two layers of CRBM and an SVM. Layer 1 uses a 3× P-parallel and 2× K-parallel architecture, and layer 2 uses a 16× K-parallel and 2× P-parallel architecture.

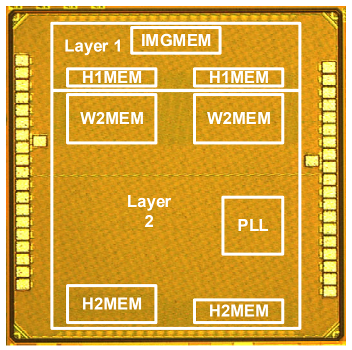


Fig. 5. Chip microphotograph.

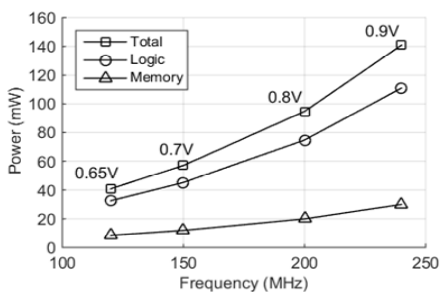


Fig. 6. Measured power and frequency of the test chip at room temperature.

TABLE I: COMPARISON WITH PRIOR WORKS

Reference	This Work		ISSCC'15 Park [2]	ISSCC'14 Lu [3]
Application	Object recognition		Big data analysis	Pattern recognition
Function	Deep neural network		Deep neural Network	Unsupervised online clustering
Technology	40nm		65nm	0.13um
Area	1.4mm^2		10.0mm^2	0.36mm^2
Voltage	0.9V	0.65V	1.2V	3V
Frequency	240MHz	120MHz	200MHz	8.3kHz
Performance	898.2GOPS	449.1GOPS	411.3GOPS	0.012GOPS
Power	140.9mW	40.9mW	213.1mW	11.4uW
Power Efficiency	6.37TOPS/W	10.98TOPS/W	1.93TOPS/W	1.04TOPS/W
Area Efficiency	641.6GOPS/mm^2	320.8GOPS/mm^2	41.13GOPS/mm^2	0.03GOPS/mm^2

* Performance, power efficiency, and area efficiency of this work are based on effective number of operations