SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks

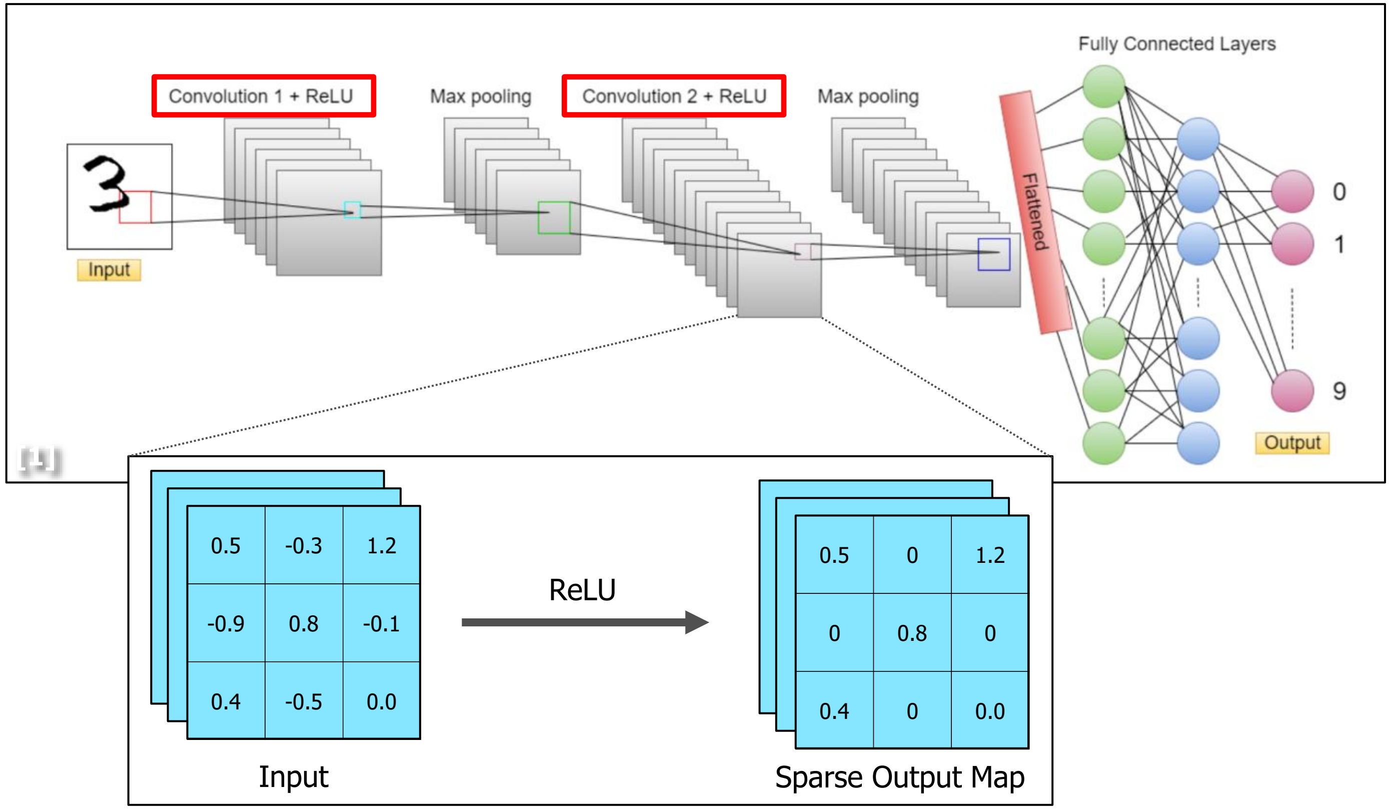
Original Paper by: Angshuman Parashar et al. (NVIDIA, ISCA ‘17)

Presented by Jongyun Hur

Introduction

Introduction

Exploiting Sparsity in CNN Inference

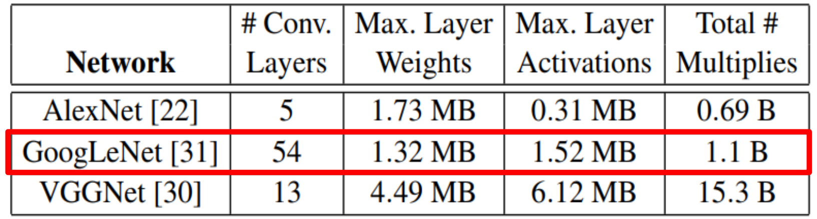


Component	Sparsity	Key Cause
Weights	**20% ~ 80%**	Pruning
Activations	**50% ~ 70%**	ReLU

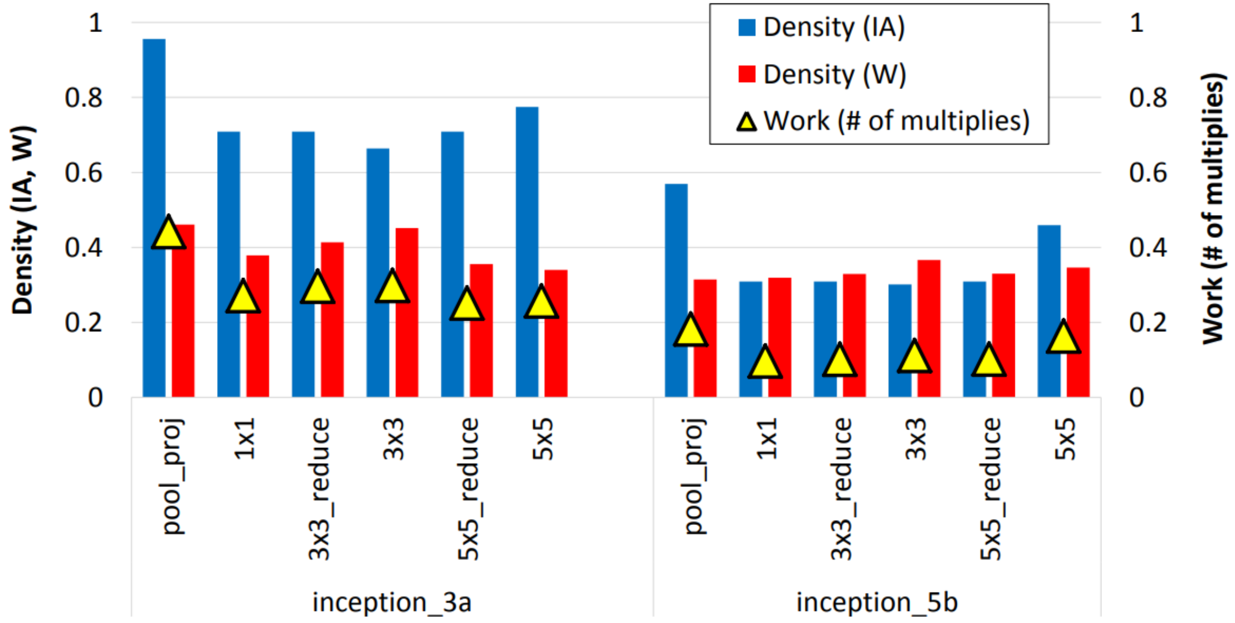
[1] Al Bataineh et al., "Automated CNN Architectural Design: A Simple and Efficient Methodology for Computer Vision Tasks," Mathematics (2023)

Introduction

Workload characterization: Why GoogLeNet?



Network	# Conv. Layers	Max. Layer Weights	Max. Layer Activations	Total # Multiplies
AlexNet [22]	5	1.73 MB	0.31 MB	0.69 B
GoogLeNet [31]	54	1.32 MB	1.52 MB	1.1 B
VGGNet [30]	13	4.49 MB	6.12 MB	15.3 B

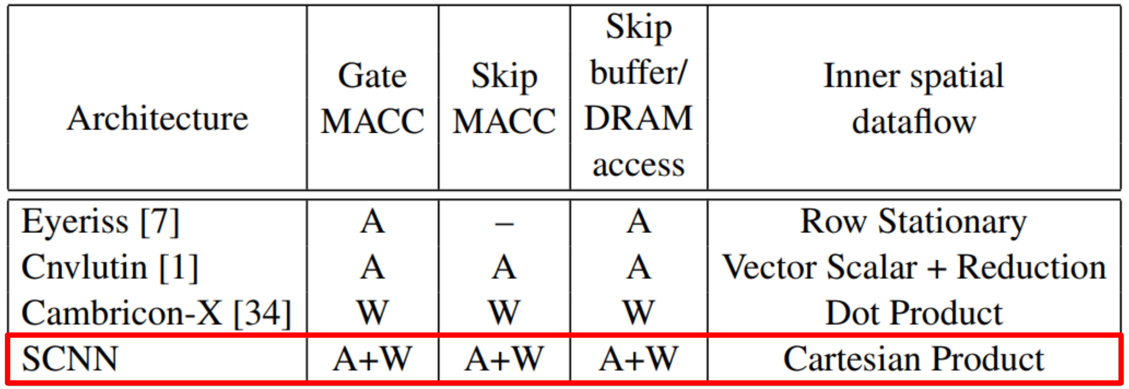


(b) GoogLeNet

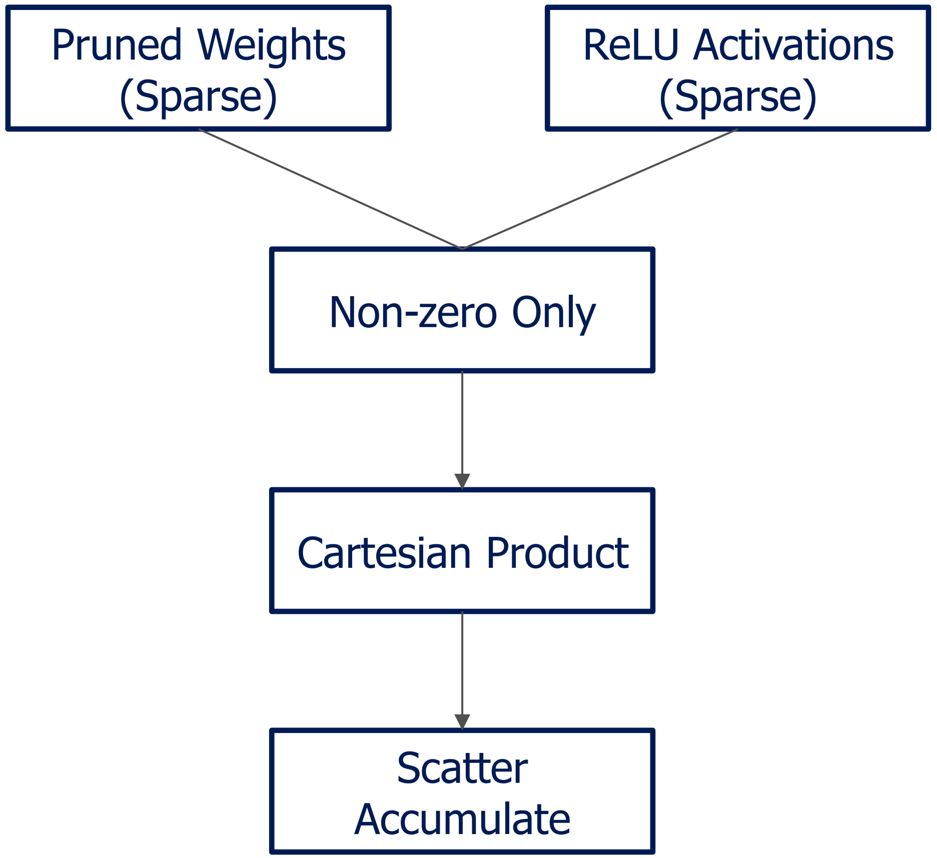
Introduction

Proposed solution: Dual sparsity & Cartesian dataflow

Table 2: Qualitative comparison of sparse CNN accelerators.



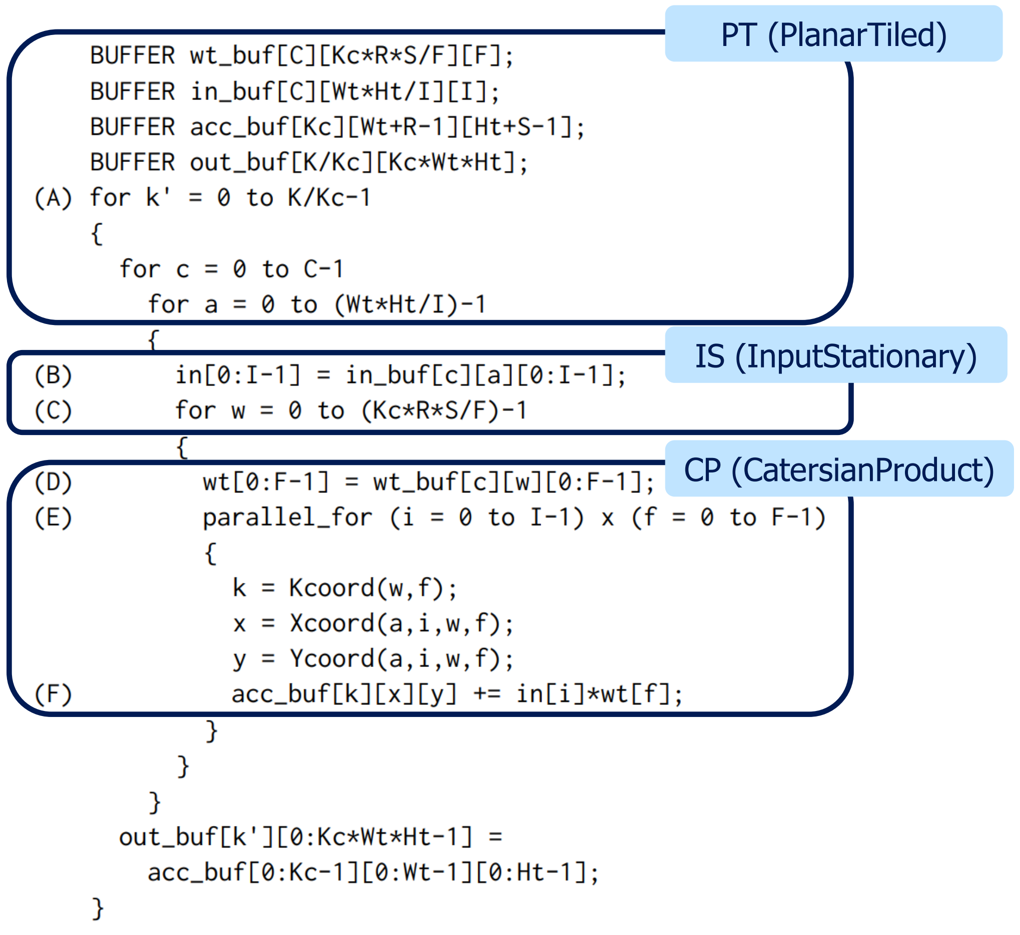
Architecture	Gate MACC	Skip MACC	Skip buffer/ DRAM access	Inner spatial dataflow
Eyeriss [7]	A	–	A	Row Stationary
Cnvlutin [1]	A	A	A	Vector Scalar + Reduction
Cambricon-X [34]	W	W	W	Dot Product
SCNN	A+W	A+W	A+W	Cartesian Product



Dataflow

Dataflow

PT-IS-CP-dense dataflow



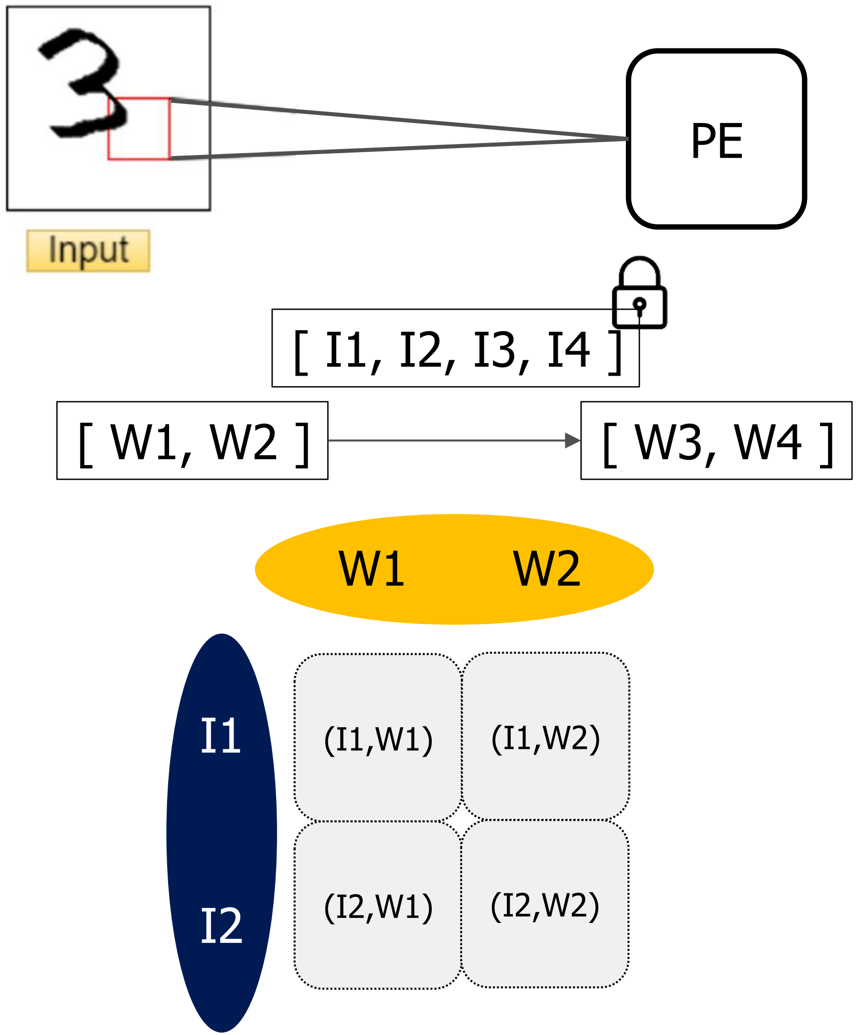
PT (PlanarTiled)

IS (InputStationary)

CP (CatersianProduct)

```
    BUFFER wt_buf[C][Kc*R*S/F][F];
    BUFFER in_buf[C][Wt*Ht/I][I];
    BUFFER acc_buf[Kc][Wt+R-1][Ht+S-1];
    BUFFER out_buf[K/Kc][Kc*Wt*Ht];
(A) for k' = 0 to K/Kc-1
    {
     for c = 0 to C-1
      for a = 0 to (Wt*Ht/I)-1
      {
(B)    in[0:I-1] = in_buf[c][a][0:I-1];
(C)    for w = 0 to (Kc*R*S/F)-1
       {
(D)     wt[0:F-1] = wt_buf[c][w][0:F-1];
(E)     parallel_for (i = 0 to I-1) x (f = 0 to F-1)
        {
         k = Kcoord(w,f);
         x = Xcoord(a,i,w,f);
         y = Ycoord(a,i,w,f);
(F)      acc_buf[k][x][y] += in[i]*wt[f];
        }
       }
      }
     out_buf[k'][0:Kc*Wt*Ht-1] =
      acc_buf[0:Kc-1][0:Wt-1][0:Ht-1];
    }
```

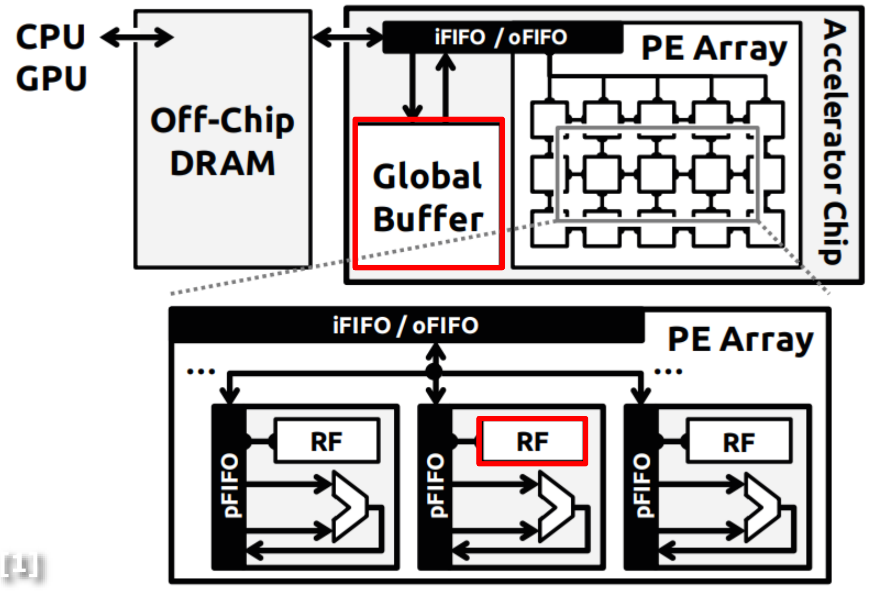
Figure 4: PT-IS-CP-dense dataflow, single-PE loop nest.

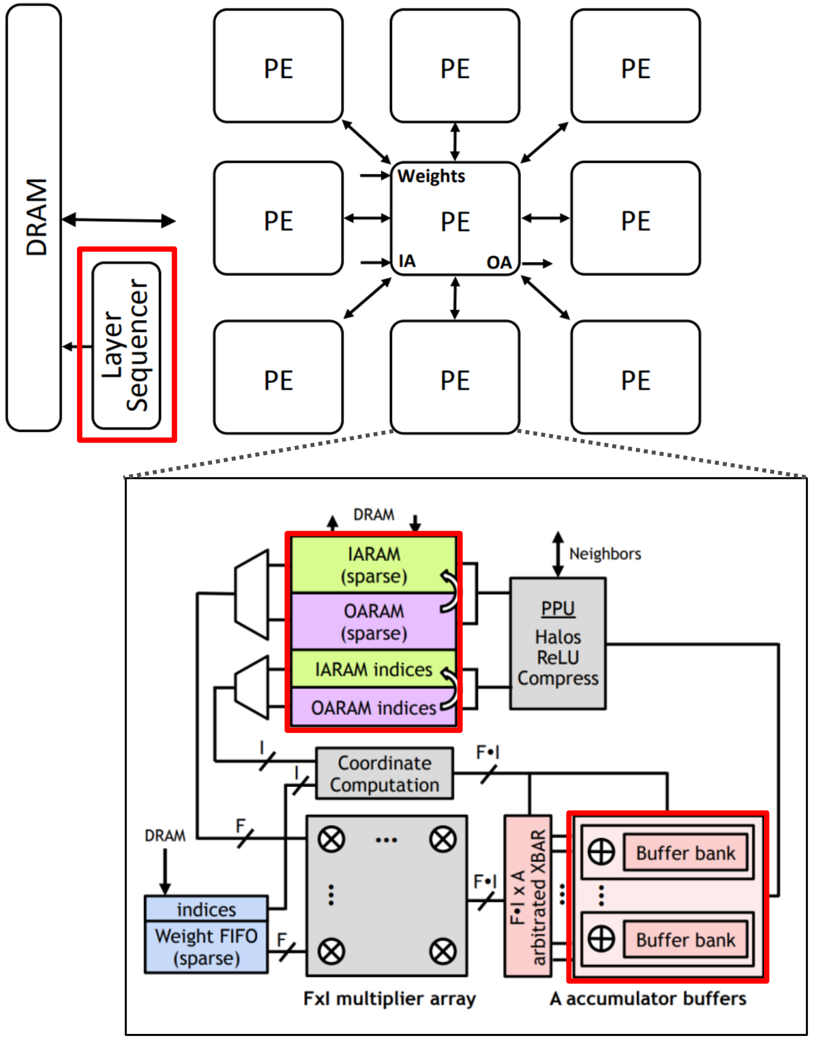


Architecture

Architecture

Standard CNN vs. Sparse CNN

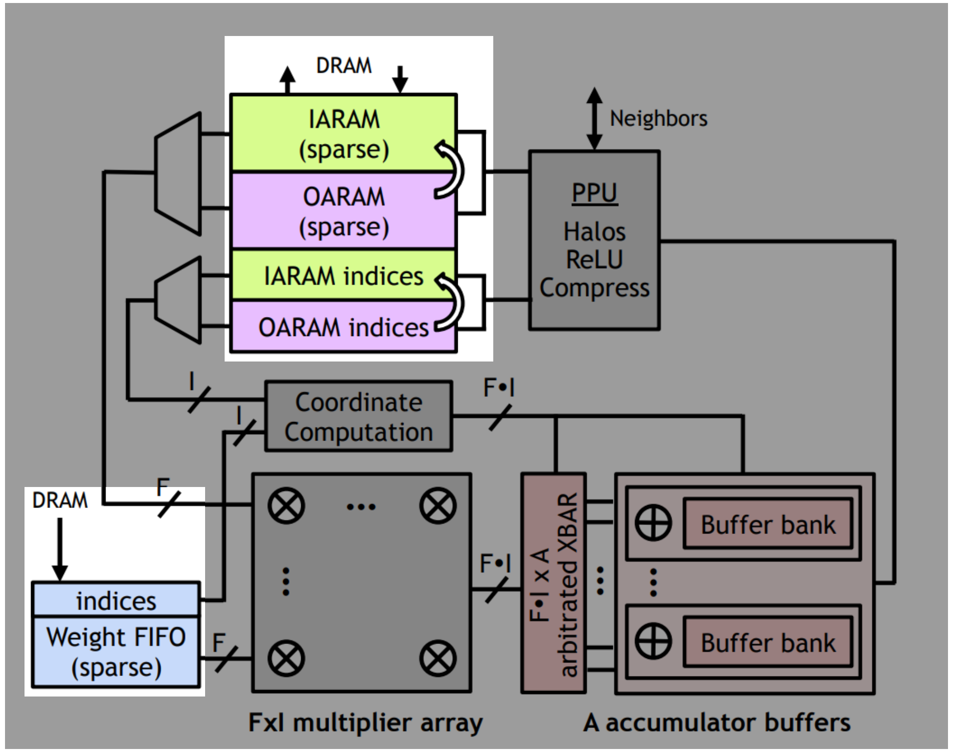




[1] Chen et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for CNNs," ISCA 2016

Architecture

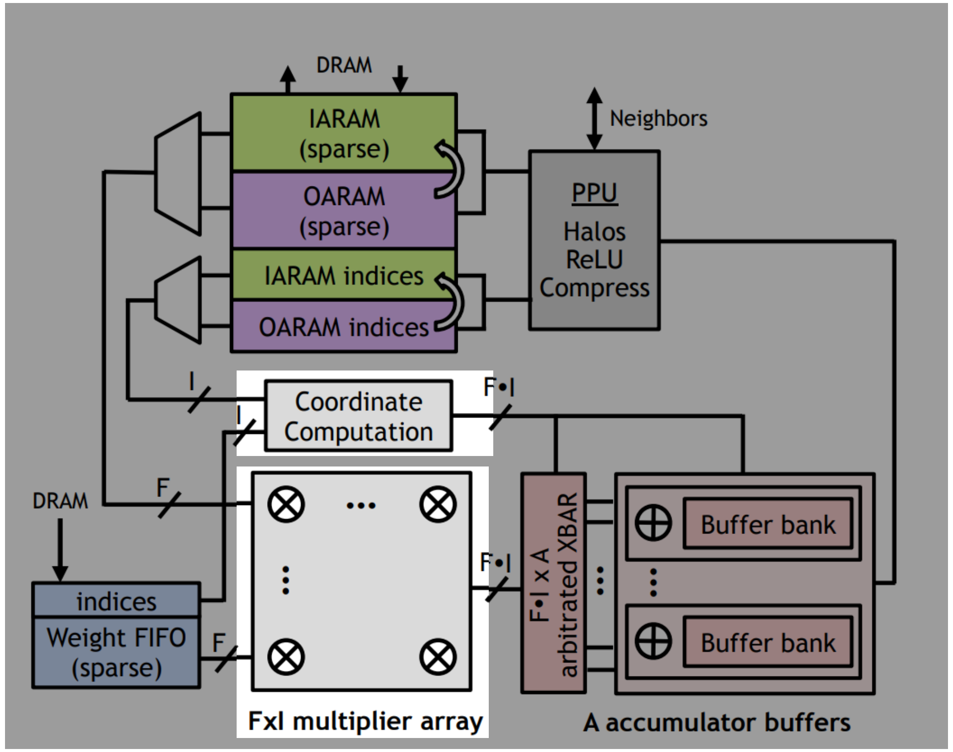
PE Microarchitecture I: Compressed storage units



Component	Input Data	Key Action
IARAM & OARAM	Compressed Activations	**Swaps roles** *(Keeps data on-chip)*
Weight FIFO	Compressed Weights *(Broadcasted from DRAM)*	**Streams non-zero vectors**

Architecture

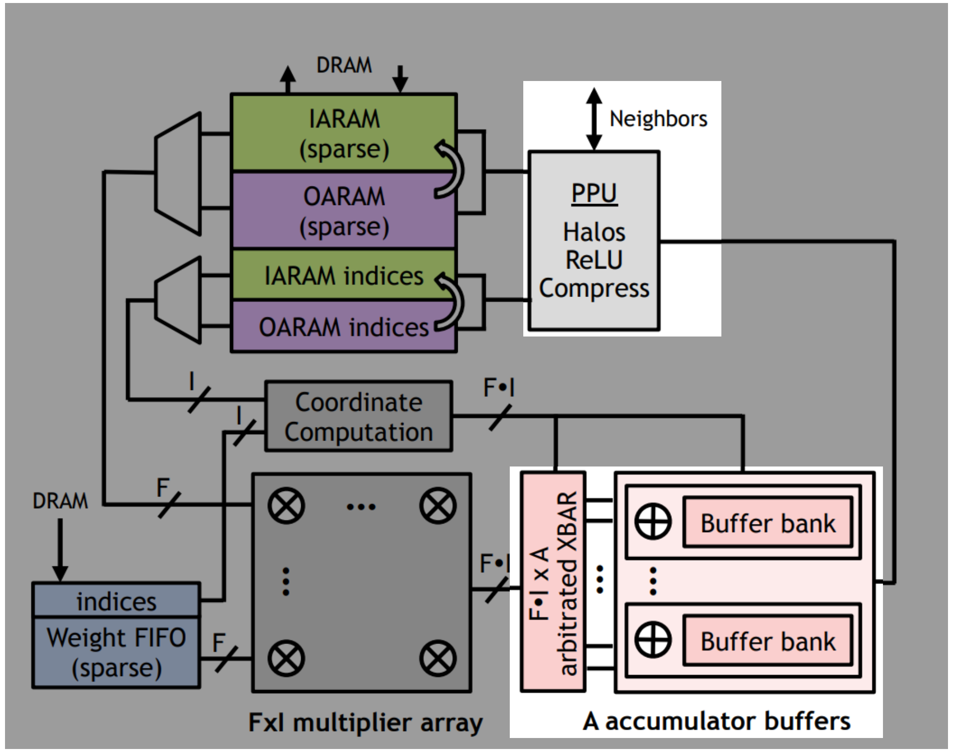
PE Microarchitecture II: The Cartesian compute core



Component	Input Data	Key Action
Coordinate Computation (Address Calculator)	Compressed Indices	**Calculates** (k, x, y)
Multiplier Array (Compute Unit)	Non-zero Values	**Cartesian Product**

Architecture

PE Microarchitecture III: Scatter & Accumulation



Component	Input Data	Key Action
PPU *(Post-Processing Unit)*	Accumulated Results *(Dense format)*	**ReLU & Re-compression**
Scatter Crossbar	Partial Sums	**Routes to correct bank** *(Prevents conflicts)*
Accumulator Banks	Routed Partial Sums *(Directed data)*	**Dense Accumulation** *(Handles random writes)*

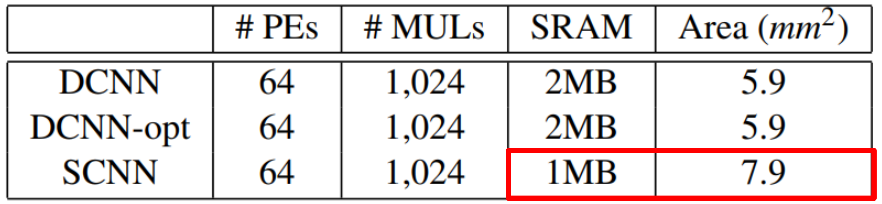
Evaluation

Evaluation

Accelerator specifications: CNN, DCNN, DCNN-opt, and SCNN

Category	Sparsity	Mechanism	Dataflow
CNN	A or W (only)	Gate MACC or Skip MACC	Row Stationary, etc.
DCNN	None	Standard MAC	Dot Product
DCNN-opt	A + W (Energy Only)	Zero-Gating	Dot Product
SCNN	A + W (Both)	Zero-Skipping	Cartesian Product

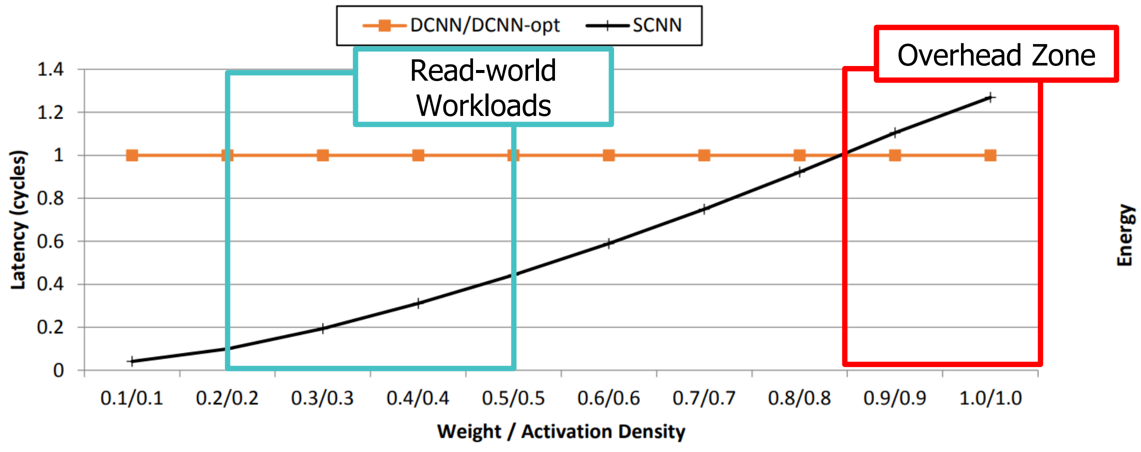
Table 5: CNN accelerator configurations.



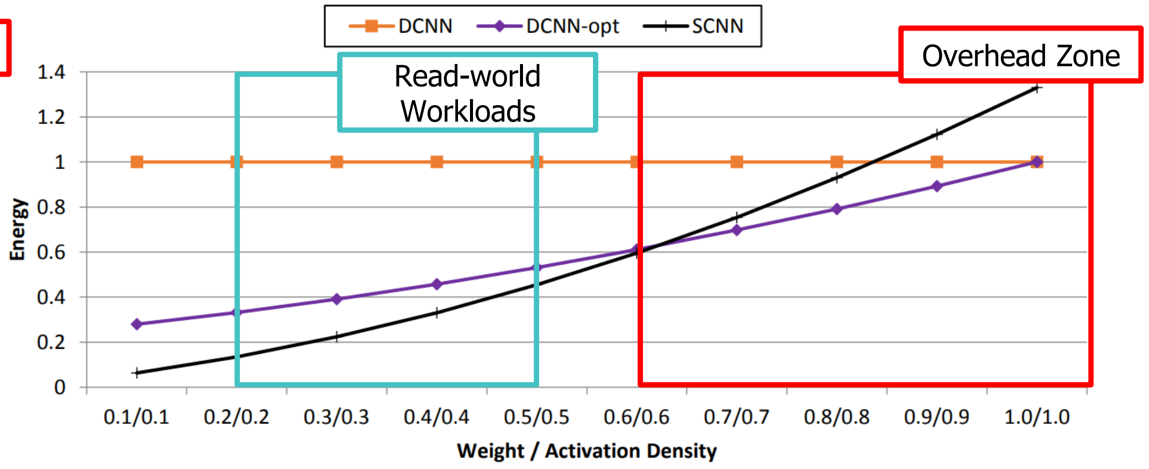
	# PEs	# MULs	SRAM	Area (mm^2)
DCNN	64	1,024	2MB	5.9
DCNN-opt	64	1,024	2MB	5.9
SCNN	64	1,024	1MB	7.9

Evaluation

GoogLeNet performance and energy versus density



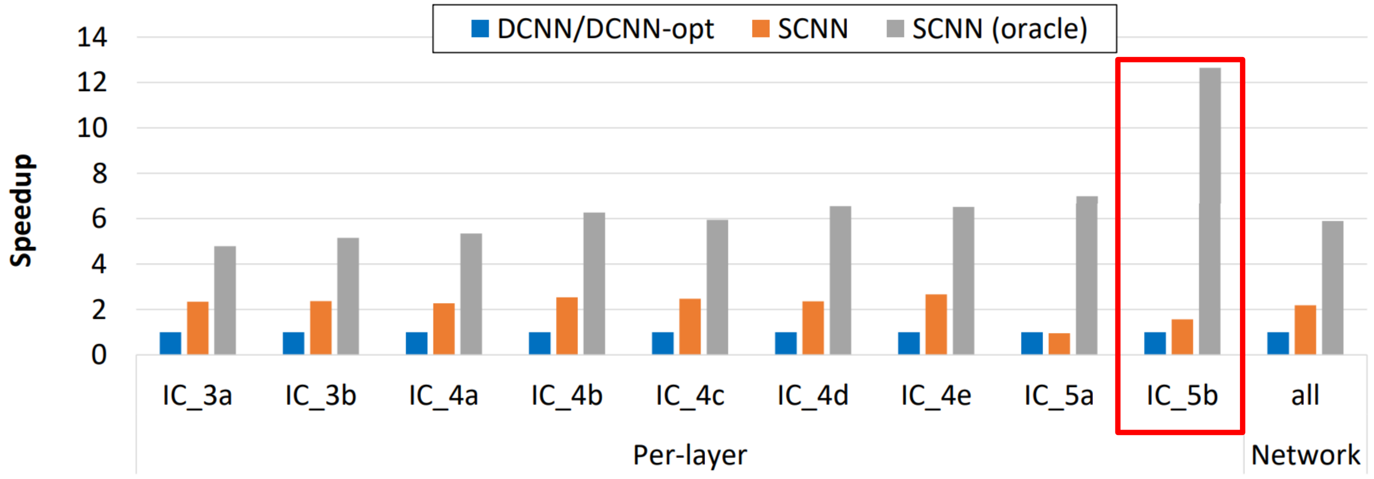
(a) Performance



(b) Energy

Evaluation

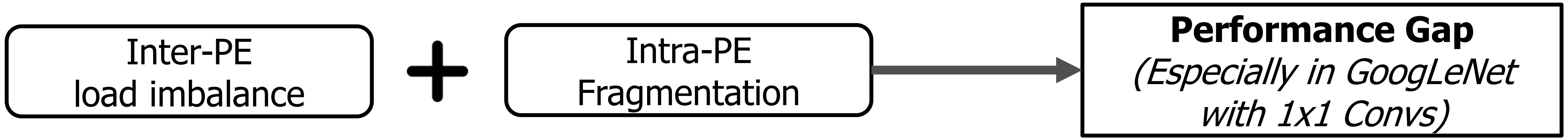
GoogLeNet performance comparison



(b) GoogLeNet

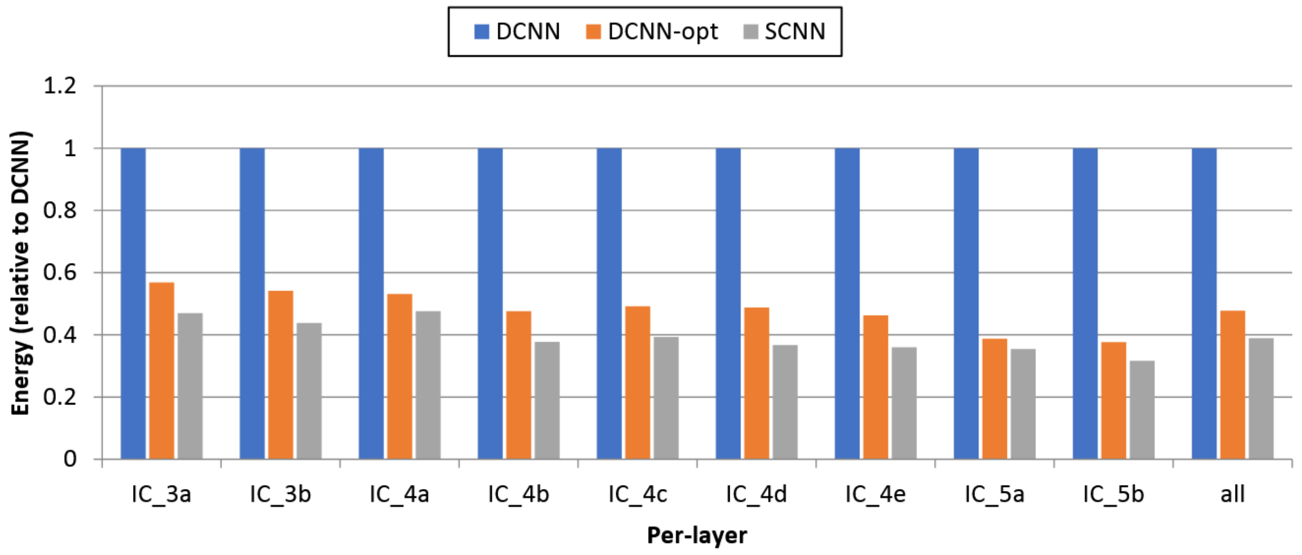
Network	Speedup (vs. DCNN)
AlexNet	2.37
GoogLeNet	2.19
VGGNet	3.52
Average	2.7

"The Performance Gap: Why does SCNN fall short of the Oracle?"



Evaluation

GoogLeNet energy-efficiency comparison

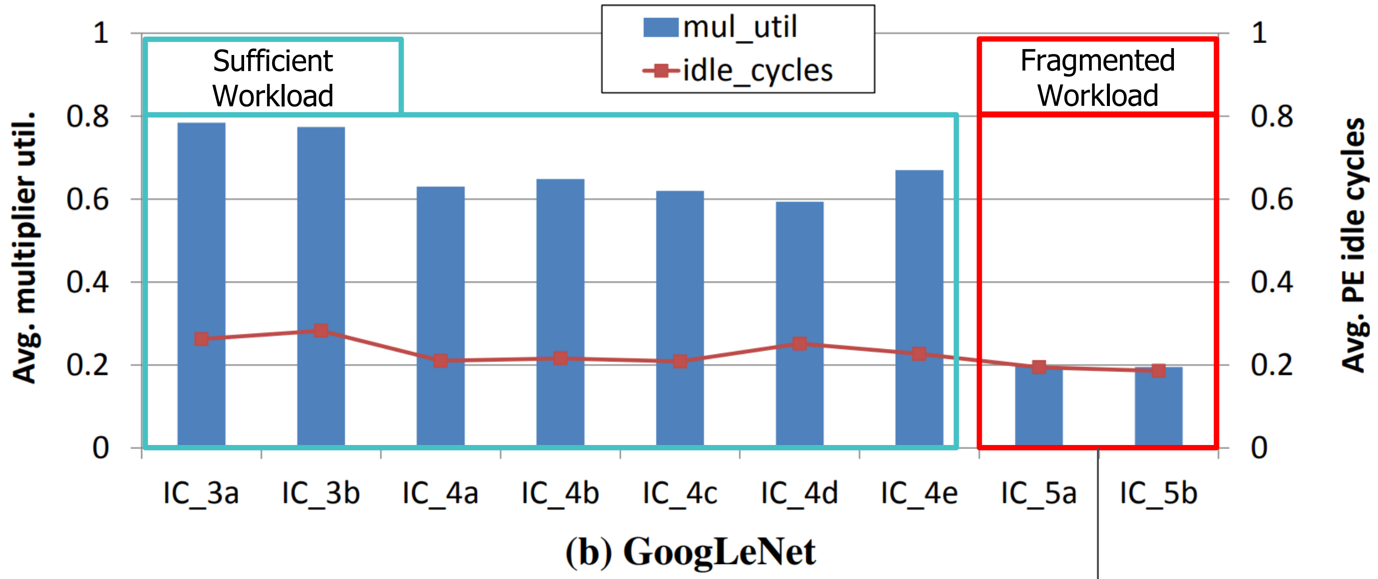


(b) GoogLeNet

Architecture	**Energy Eff (vs. DCNN)**
DCNN-opt	2.0x
SCNN	2.3x

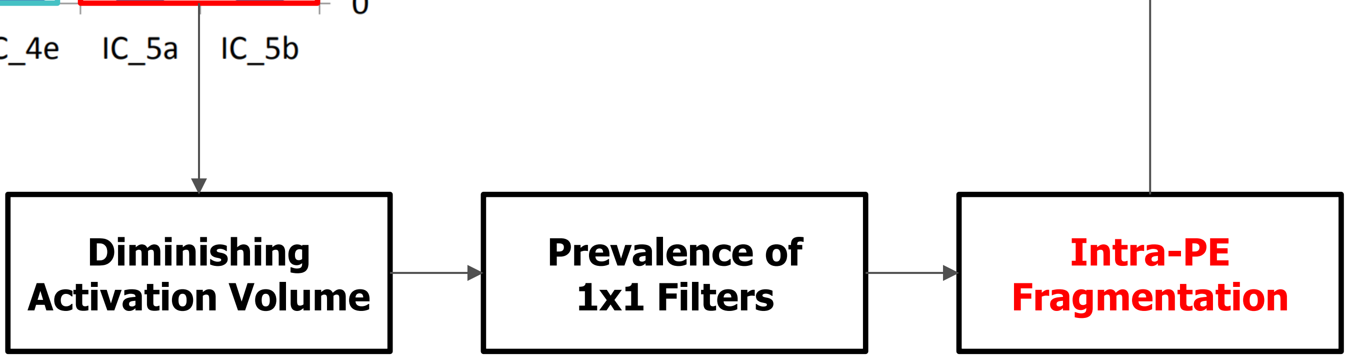
Evaluation

The Bottleneck: Multiplier underutilization and Load imbalance



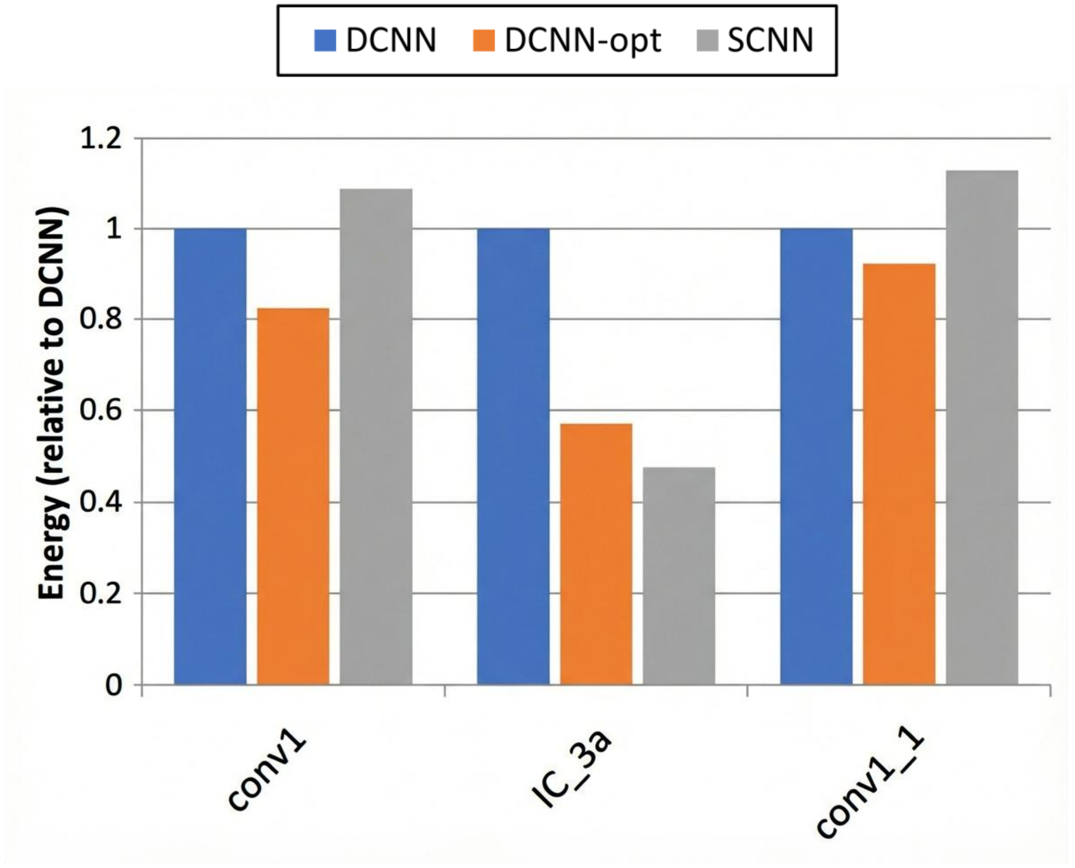
(b) GoogLeNet

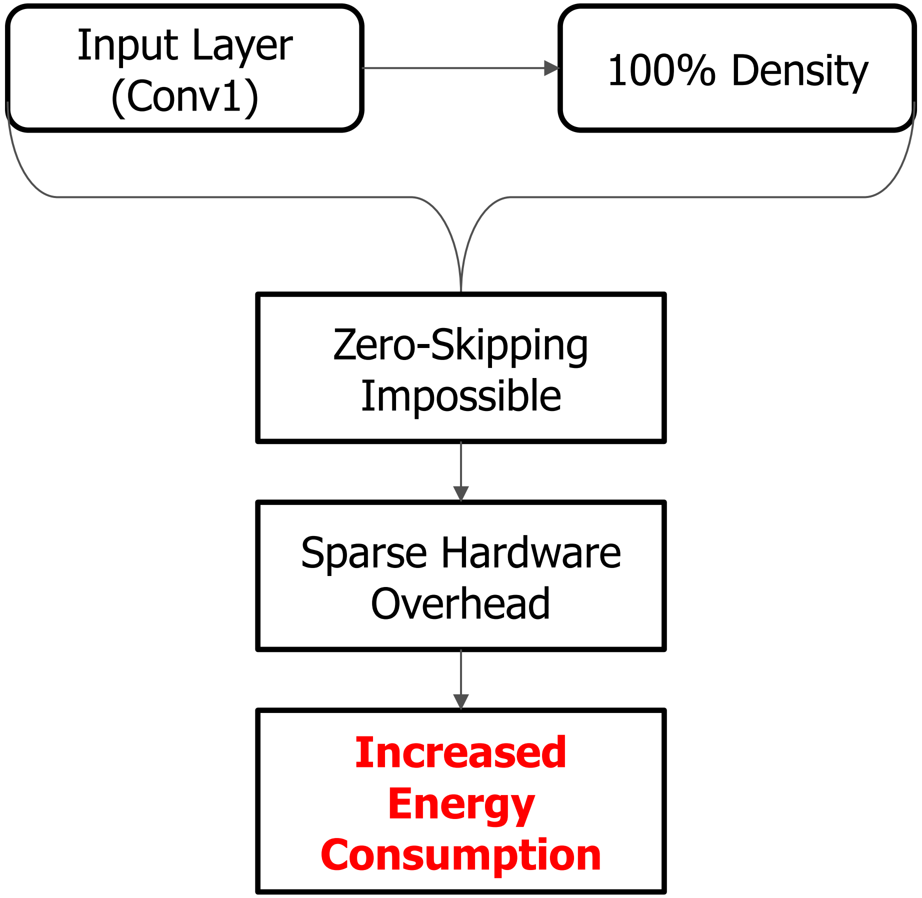
Network	Target Layer	Mul_util
GoogLeNet	Network Avg	59%
	IC_5a, IC_5b	< 20%



Evaluation

Why SCNN struggles with non-sparse input layers





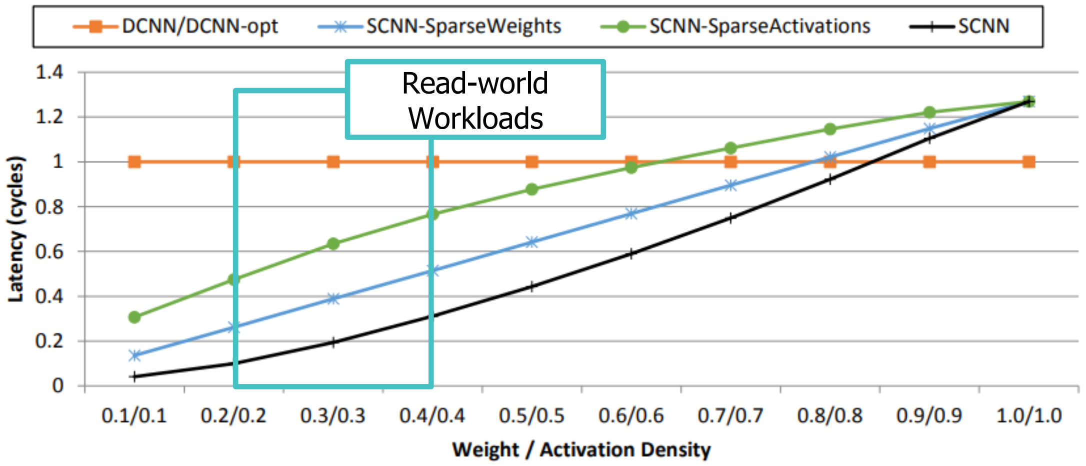
Evaluation

Comprehensive comparison: Full support for dual-way sparsity

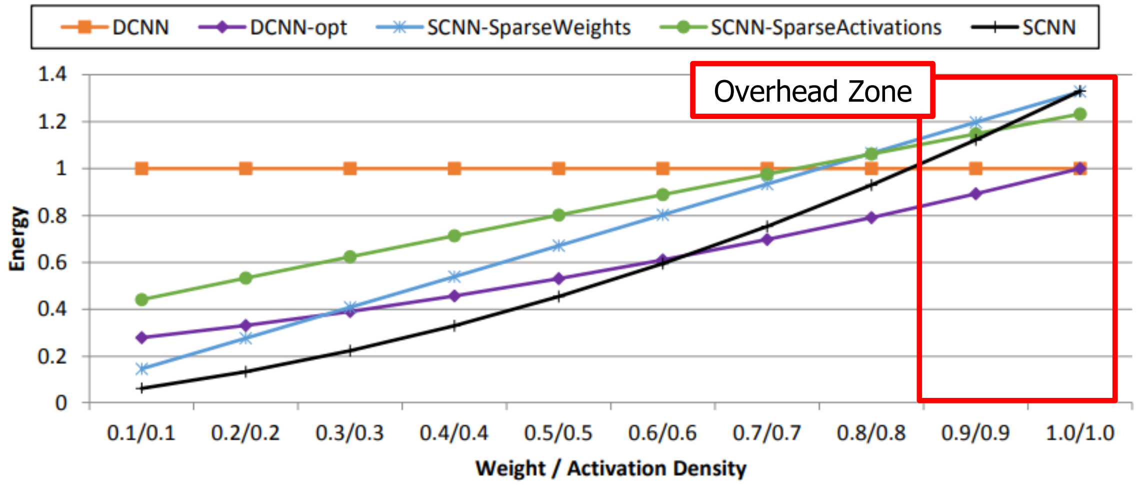
Category	Architecture	Sparsity Target	Processing Method	Memory Access	Impact	Dataflow
Baseline	**DCNN**	**None**	Standard MAC	Full Access	Baseline (1.0x)	Dot Product
	DCNN-opt	**A + W**	**Gating** (Blocks zeros)	Compressed DRAM	**Energy Saving Only** (No Speedup)	Dot Product
Comparison (Partial)	**SCNN-SparseA** (like Cnvlutin)	**A** (Only)	**Skipping**	**Skip** (A only)	**Speedup + Energy ↑** (Dependent on A)	Cartesian Product
	SCNN-SparseW (like Cambricon-X)	**W** (Only)	**Skipping**	**Skip** (W only)	**Speedup + Energy ↑** (Dependent on W)	Cartesian Product
Proposed	**SCNN**	**A + W (Both)**	**Skipping**	**Skip** (Both)	**Maximized Speedup & Efficiency** (Both Optimized)	**Cartesian Product**

Evaluation

High Efficiency in Low-Density vs. Marginal Overhead in High-Density



(a) Performance



(b) Energy

Conclusion

Conclusion

Comparative analysis of SCNN accelerators

Core Innovations

Both-way Skipping
Efficiently skips zeros in inputs & weights.

PT-IS-CP-sparse Dataflow

Cartesian Product
New dataflow for sparse matrix multiplication.

Key Results

2.7x Speedup
(vs. DCNN)

2.3x Energy Efficient

Challenges

High Density Overhead

Intra-PE Fragmentation

Inter-PE Load Imbalance