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

Hardware for Machine Learning

Lecture 13: Sparsity

Sophia Shao

OSKI: Optimized Sparse Kernel Interface

The Optimized Sparse Kernel Interface (OSKI) Library is a collection of low-level C primitives that provide automatically tuned computational kernels on sparse matrices, for use in solver libraries and applications. OSKI has a BLAS-style interface, providing basic kernels like sparse matrix-vector multiply and sparse triangular solve, among others.

Oski: "Go Bears!"

Rich Vuduc

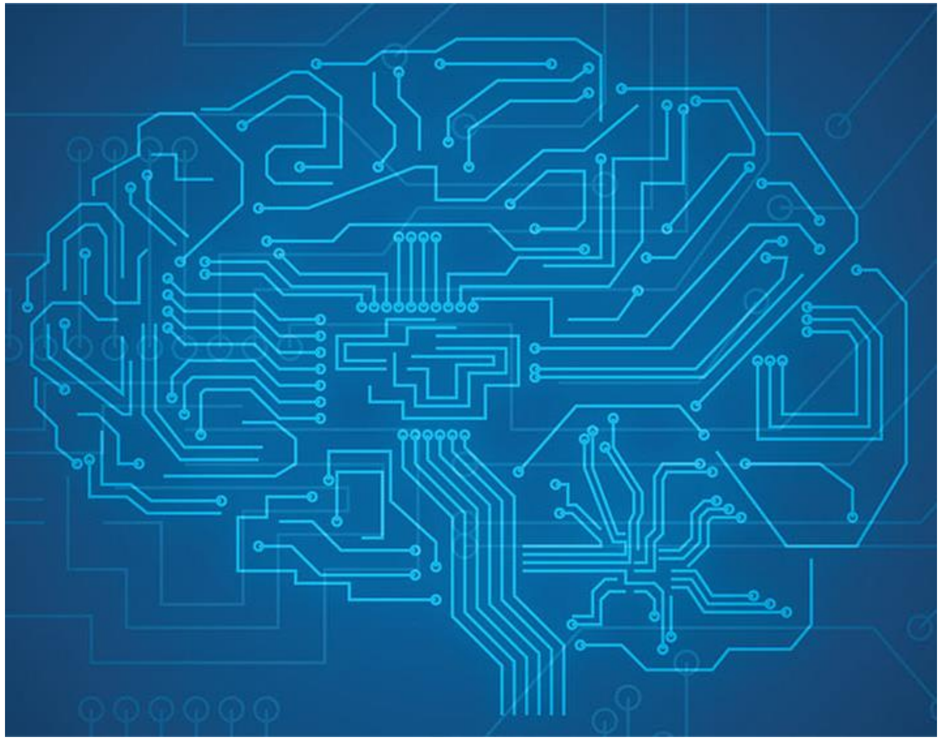
James Demmel

Kathy Yelick

http://bebop.cs.berkeley.edu/oski/

Review

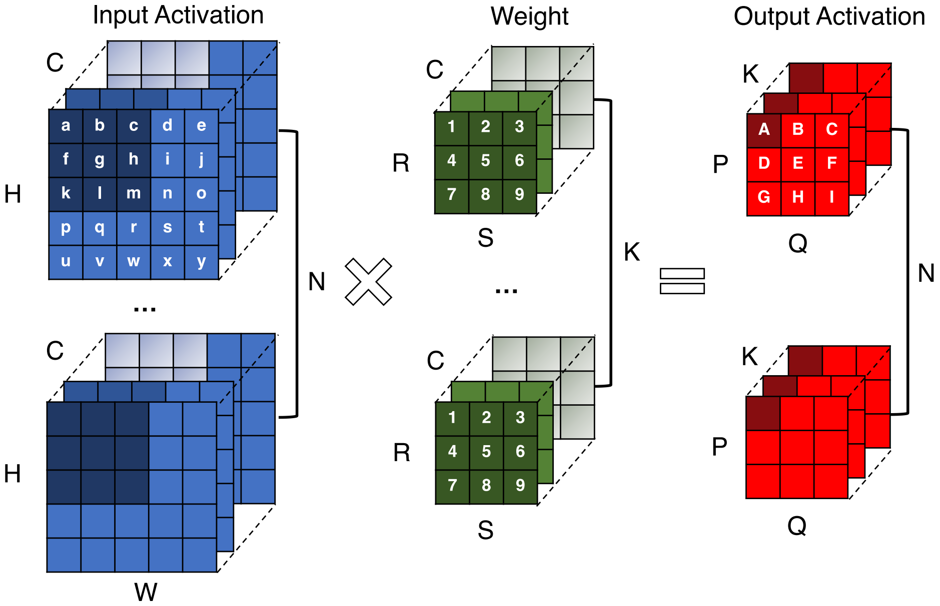
- Core computation in DNN
- Execution order of the core computation
- Hardware realization of the core computation
- Mapping DNNs to hardware
- Last lecture: data transfer mechanisms across storage hierarchy
  - Guiding principles
  - Taxonomy:
    - Implicit vs Explicit
    - Coupled vs Decoupled
  - Case studies:
    - Cache, DAE, GPU shared memory, DMA



Sparsity

- Motivation
  - Source of Sparsity
- Sparsity in Storage:
  - Compression Formats
- Sparsity in Compute:
  - Single Operand
  - Two Operands
- Case study

Convolution Loop Nest



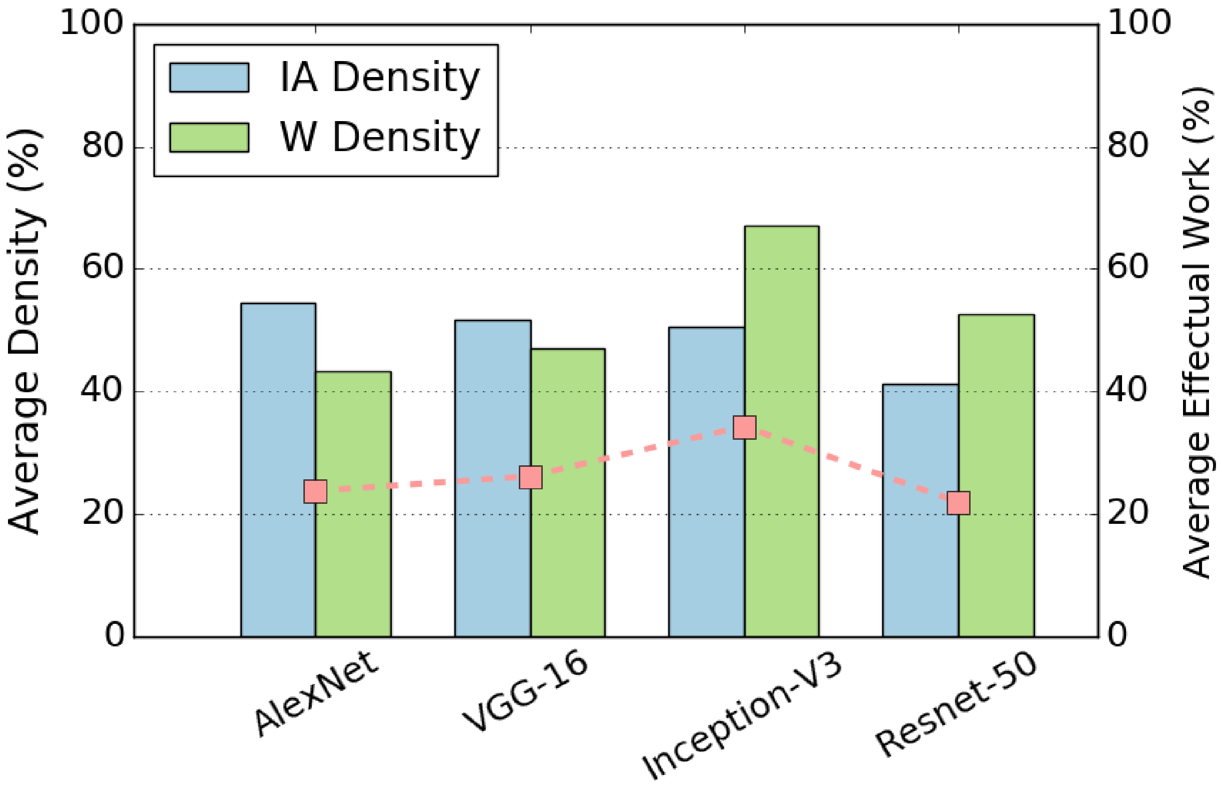

```
for (n=0; n<N; n++) {
  for (k=0; k<K; k++) {
    for (p=0; p<P; p++) {
      for (q=0; q<Q; q++) {
        OA[n][k][p][q]= 0;
        for (r=0; r<R; r++) {
          for (s=0; s<S; s++) {
            for (c=0; c<C; c++) {
              h = p * stride - pad + r;
              w = q * stride - pad + s;
              OA[n][k][p][q] +=
                         IA[n][c][h][w]
                         * W[k][c][r][s];
            }
          }
        }
        OA[n][k][p][q]= Activation(OA[n][k][p][q]);
      }
    }
  }
}
```

$\left.\right\}$ for each output activation (the four outer loops over n, k, p, q)

$\left.\right\}$ convolution window (the loops over r, s, c)

Sparsity in DNNs

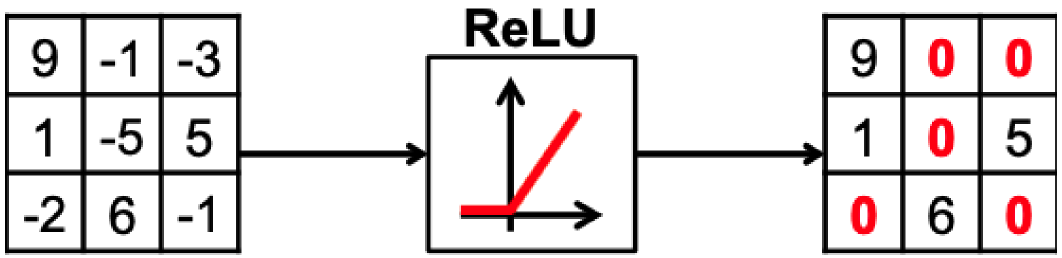
- Zero exists in both input activations and weights.
- Effectual work: both operands are non-zero.



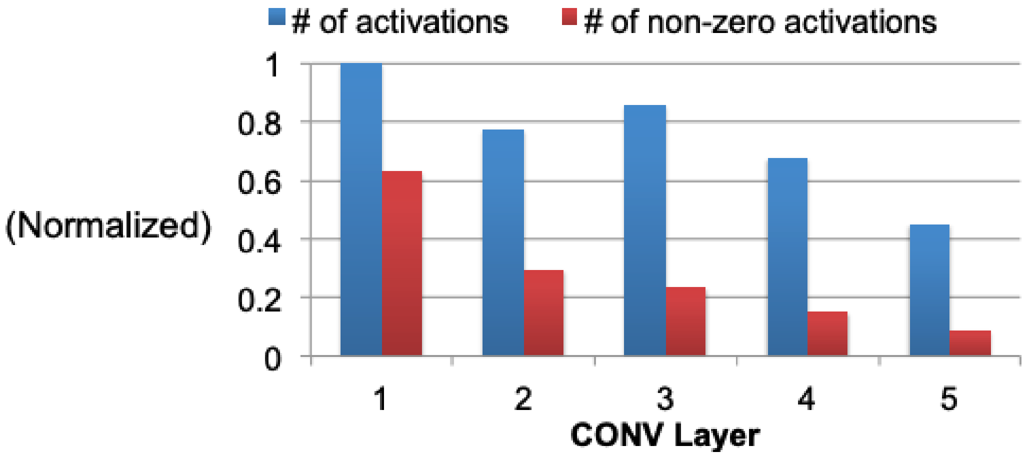
SNAP, VLSI'2019

Source of Sparsity: Activation

- ReLU operator.



(a) ReLU non-linearity



(b) Distribution of activation after ReLU of AlexNet

Eyeriss tutorial

Source of Sparsity: Activation

- ReLU operator.

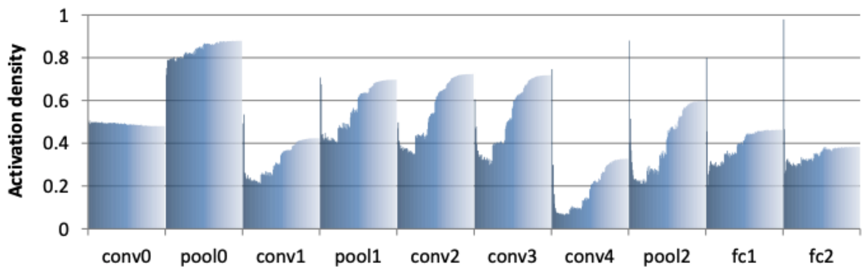
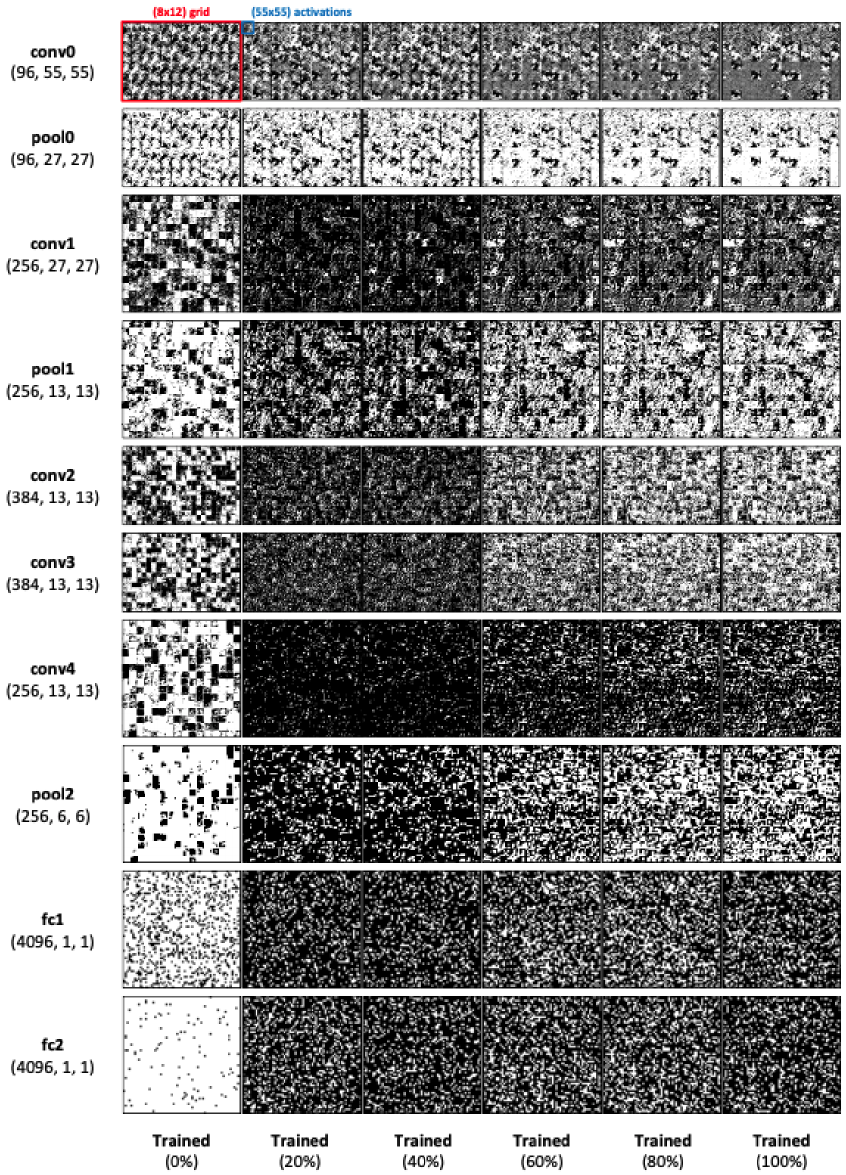


Fig. 4: Average activation density of each layer in AlexNet over time during training (going from dark to light blue colored bars, per layer). Activation density is sampled at every 2K iterations of training and a total of 226K iterations were spent to reach the fully trained model (53.1%/75.1% top-1/top-5 accuracy).

Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks, HPCA'2018

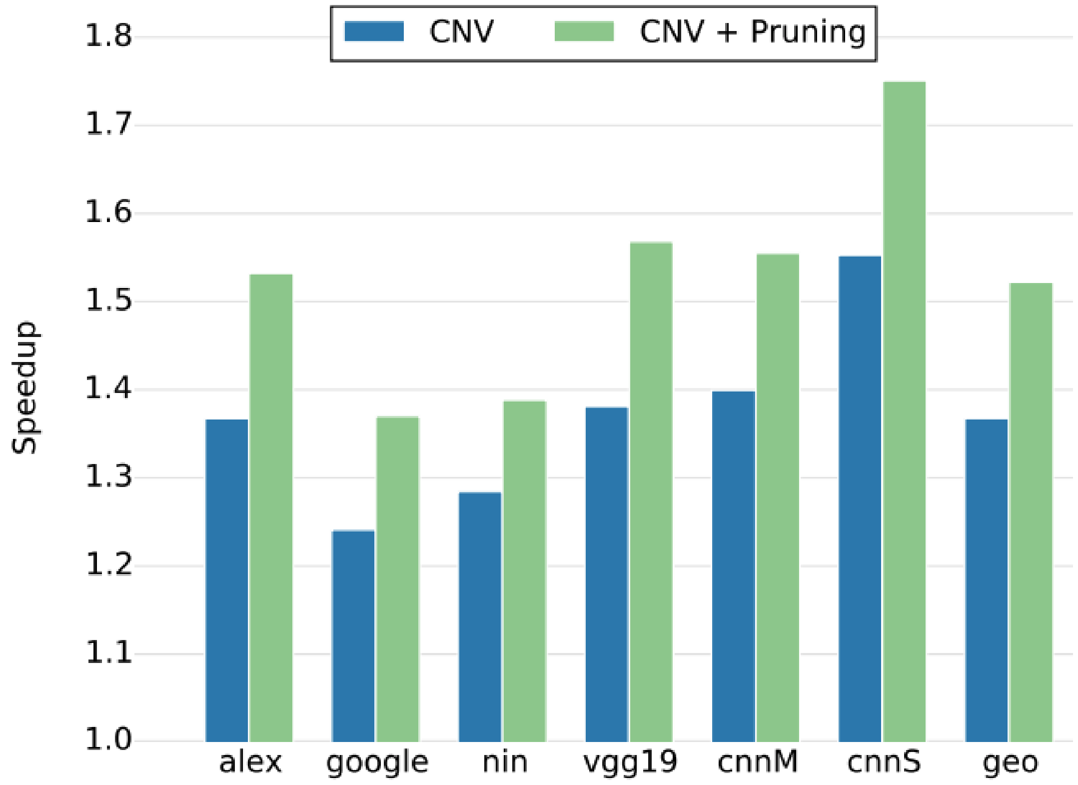


Source of Sparsity: Activation

- Pruning: if val. < threshold, val = 0;

Network	Thresholds per layer	Speedup
alex	8,4,8,16,8	1.53
nin	4,8,16,16,16,16,32,32,16,8,16,4	1.39
google	4,4,8,16,4,4,4,4,2,2,2	1.37
cnnM	8,2,4,4,2	1.56
cnnS	4,4,8,4,4	1.75
vgg19	8,4,16,64,64,64,64,128,256, 256,256,128,64,32,16,16	1.57

TABLE II: Lossless Ineffectual Neuron Thresholds



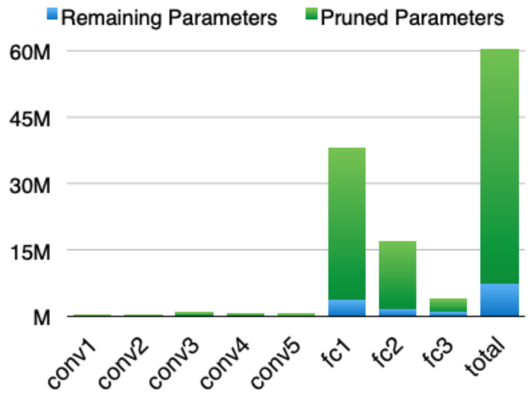
Cnvlutin, ISCA'2016

Source of Sparsity: Weights

- Pruning: if val. < threshold, val = 0;
- Regularization (Weight decay):
  - Expressing preferences for smaller weights
    - $CF = MSE_{train} + \lambda \sum_i w_i^2 \ (L2)$
    - $CF = MSE_{train} + \lambda \sum_i |w_i| \ (L1)$

Table 4: For AlexNet, pruning reduces the number of weights by $9\times$ and computation by $3\times$.

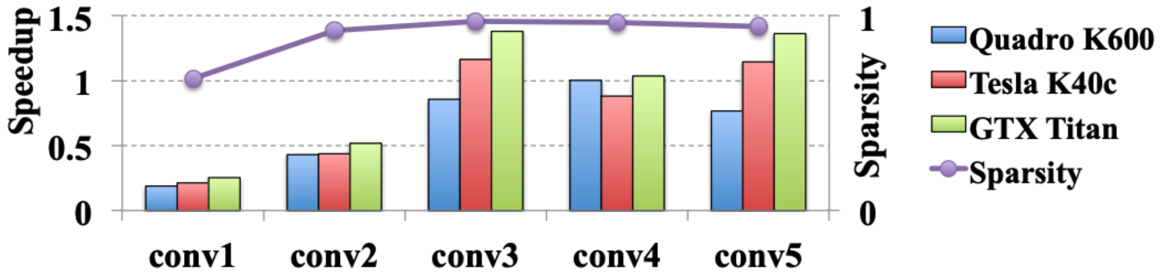
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**



Learning both Weights and Connections for Efficient Neural Networks, NIPS'2015

Source of Sparsity: Unstructured vs Structured

- Unstructured sparsity does not directly translate to speedup.



- Structured sparsity.

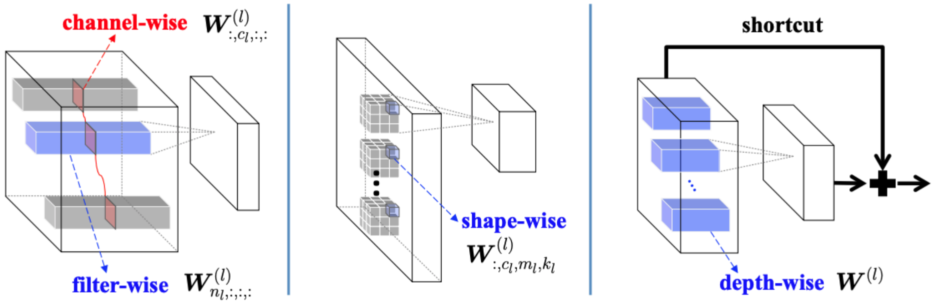
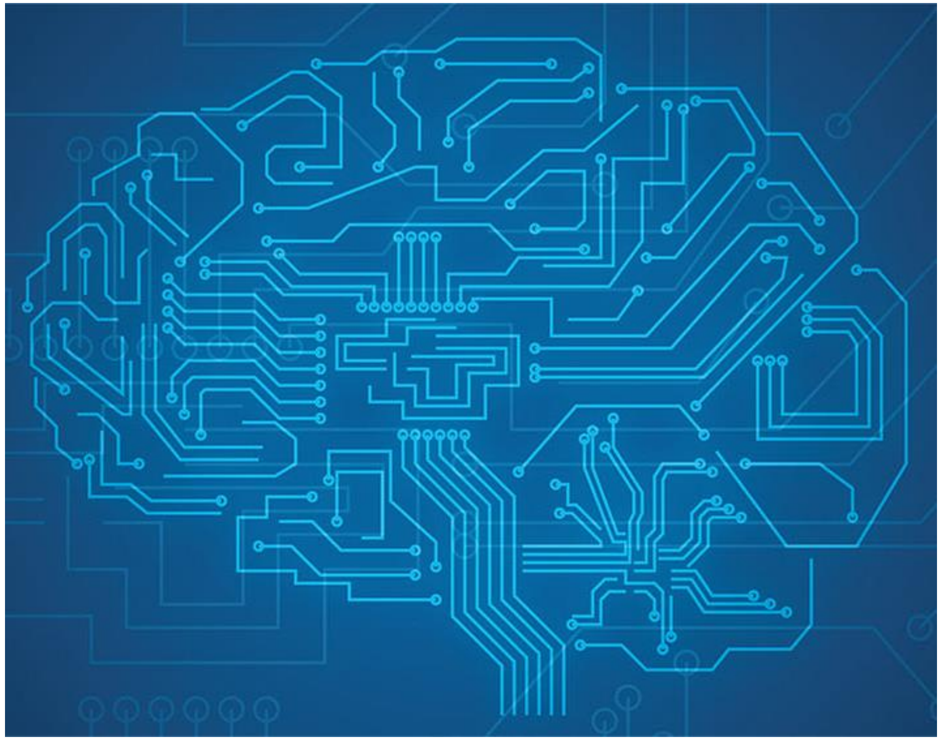


Figure 2: The proposed *Structured Sparsity Learning* (SSL) for DNNs. The weights in filters are split into multiple groups. Through group Lasso regularization, a more compact DNN is obtained by removing some groups. The figure illustrates the filter-wise, channel-wise, shape-wise, and depth-wise structured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



Sparsity

- Motivation
  - Source of Sparsity
- Sparsity in Storage:
  - Compression Formats
- Sparsity in Compute:
  - Single Operand
  - Two Operands
- Case study

Leveraging Sparsity in Storage

- Goal: reduce memory usage and/or bandwidth
- Commonly-used compression formats:
  - Bitmask (e.g, NVDLA)
  - Run-length encoding (e.g., Eyeriss)
  - Compressed-sparse row (CSR) and compressed sparse column (CSC)
  - Tensor compression (Taco)

Bitmask compression

- Using 1-bit/element to indicate whether the value is zero or not.
- Example: NVDLA
- Pros: simple and regular
- Cons: fixed overhead independent of density, e.g., 1/8 overhead for 8-bit values

Original Format

Column

Row				
	A	0	B	C
	0	0	0	0
	E	0	0	F

Compressed Format

Bitmask	1	0	1	1	0	0	0	0	1	0	0	1
Value	A	B	C	E	F							

Run-Length Encoding

- Runs of data (sequences where the same data value, e.g., 0, occurs in consecutive elements) are stored as a single value and count.
- Example: Eyeriss
- Pros: compression rate linear with nnz (number of non-zero) elements.
- Cons: overhead: ~nnz * (run-bitwidth)

Original Format

Column

	A	0	B	C
Row	0	0	0	0
	E	0	0	F

Compressed Format

	Run		Run		Run		…			
Value	0	A	1	B	0	C	4	E	2	F
		Level		Level		Level		…		

Compressed-Sparse Row (CSR)

- Represents a matrix with 3 one-dimension arrays
  - Bound of each row (size: num of rows)
  - Column index for each nnz element within the row (size: nnz)
  - Non-zero values (size: nnz)
- Pros: allow fast row access

Original Format

Column

Row				
	A	0	B	C
	0	0	0	0
	E	0	0	F

Compressed Format

Bounds (row)	0	3	3	5	
Index (col)	0	2	3	0	3
Value	A	B	C	E	F

Compressed-Sparse Column (CSC)

- Represents a matrix with 3 one-dimension arrays
  - Bound of each column (size: num of columns)
  - Row index for each nnz element within the column (size: nnz)
  - Non-zero values (size: nnz)
- Pros: allow fast column access

Original Format

Column

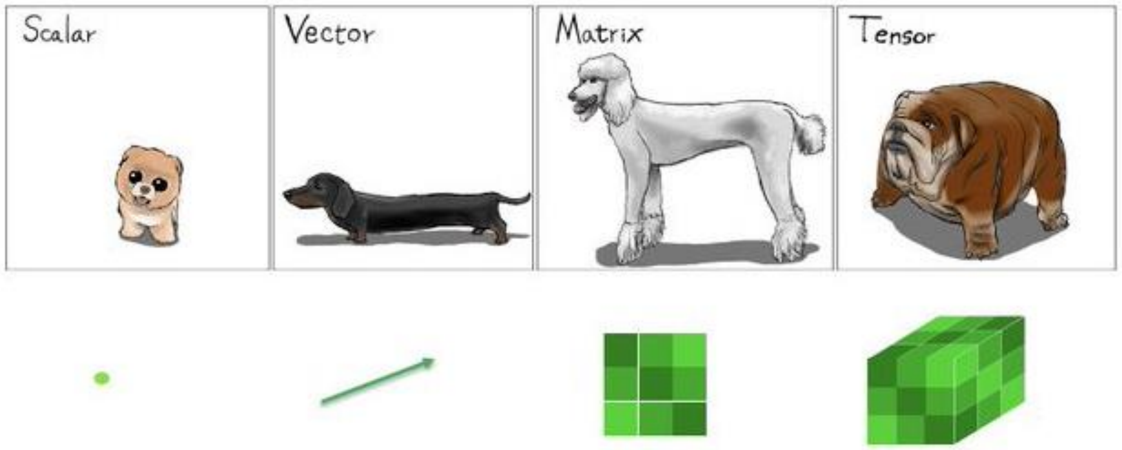
Row				
	A	**0**	**B**	**C**
	0	**0**	**0**	**0**
	E	**0**	**0**	**F**

Compressed Format

Bounds (col)	**0**	**2**	**2**	**3**	**5**
Index (row)	**0**	**2**	**0**	**0**	**2**
Value	**A**	**E**	**B**	**C**	**F**

The Taco Notation

- Tensors: multi-dimensional arrays.
- Tensor storage format:
  - Separately designate each dimension as dense or sparse
  - Specify the order in which dimensions are stored
  - E.g. CSR is a row-dense, column-sparse format.



Anima Anandkumar, ScaledML'2018

Column

Row	A	0	B	C
	0	0	0	0
	E	0	0	F

Bounds (row)	0	3	3	5	
Index (col)	0	2	3	0	3
Value	A	B	C	E	F

The Tensor Algebra Compiler, OOPSLA'2017

The Taco Notation

- Tensors: multi-dimensional arrays.
- Tensor storage format:
  - What if nnz < n?
  - Row-sparse, column-sparse

Original

Column

Row				
	A	0	B	C
	0	0	0	0
	E	0	0	F

CSR

Bounds (row)	0	3	3	5	
Index (col)	0	2	3	0	3
Value	A	B	C	E	F

Sparse-Sparse
Doubly compressed sparse row (DCSR)

Bounds (-)	0	2			
Index (row)	0	2			
Bounds (row)	0	3	5		
Index (col)	0	2	3	0	3
Value	A	B	C	E	F

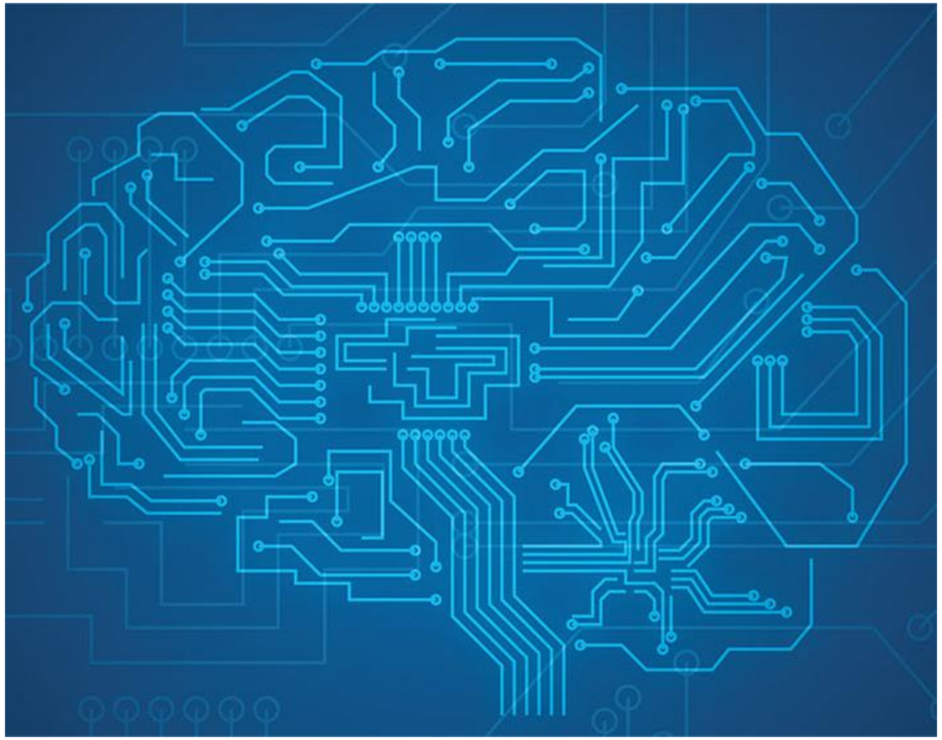
The Tensor Algebra Compiler, OOPSLA'2017
On the Representation and Multiplication of Hypersparse Matrices, IPDPS'2008

Administrivia

- Guest Lecture next Monday:
  - Bita Darvish Rouhani, Microsoft
- Use late days to finish Lab 3 if you haven't done already!
- Project starts this week.
  - Proposal (1-2 page) due 3/19.

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)



Sparsity

- Motivation
  - Source of Sparsity
- Sparsity in Storage:
  - Compression Formats
- Sparsity in Compute:
  - Single Operand
  - Two Operands
- Case study

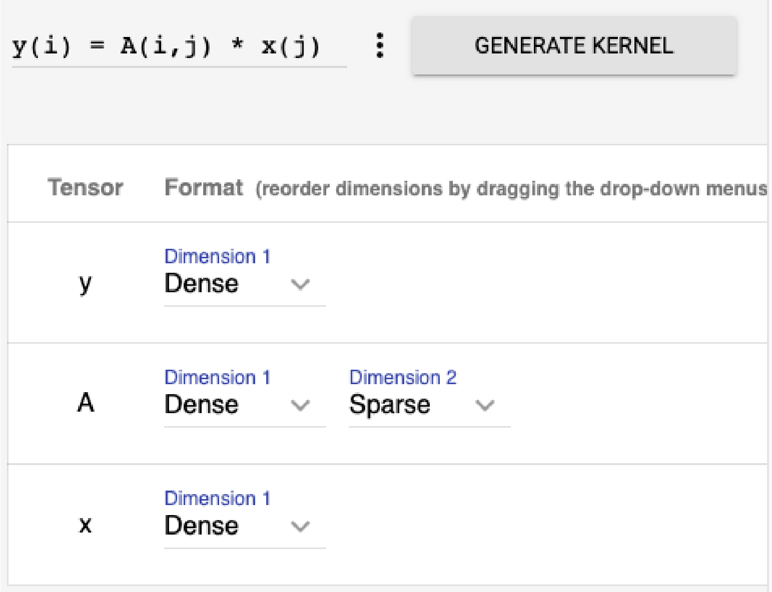
DNN Compute on Sparse Data

- Skip ineffectual computation when either W or IA is zero.
- Approaches:

	Single-Operand Sparse	Two-Operand Sparse
Align W & IA	Indirection (e.g., Cnvlutin, …)	Intersection (e.g., SNAP, …)
Align OA	Arbitration (e.g., SCNN)	

Align Non-zero W & IA w/ 1-Operand Sparse

- **Indirection** for one-operand sparse case:
  - Use the index of non-zero elements of the sparse tensor to index the dense tensor.




```
#pragma omp parallel for schedule(runtime)
for (int32_t i = 0; i < A1_dimension; i++) {
  for (int32_t pA2 = A2_pos[i]; pA2 < A2_pos[(i + 1)]; pA2++) {
    int32_t j = A2_crd[pA2];
    y_vals[i] = y_vals[i] + A_vals[pA2] * x_vals[j];
  }
}
```

http://tensor-compiler.org/codegen.html

Align Non-zero W & IA w/ 1-Operand Sparse

- **Indirection** for one-operand sparse case:
  - Use the index of non-zero elements of the sparse tensor to index the dense tensor.

`y(i, j) = A(i,k) * x(k, j)`

GENERATE KERNEL

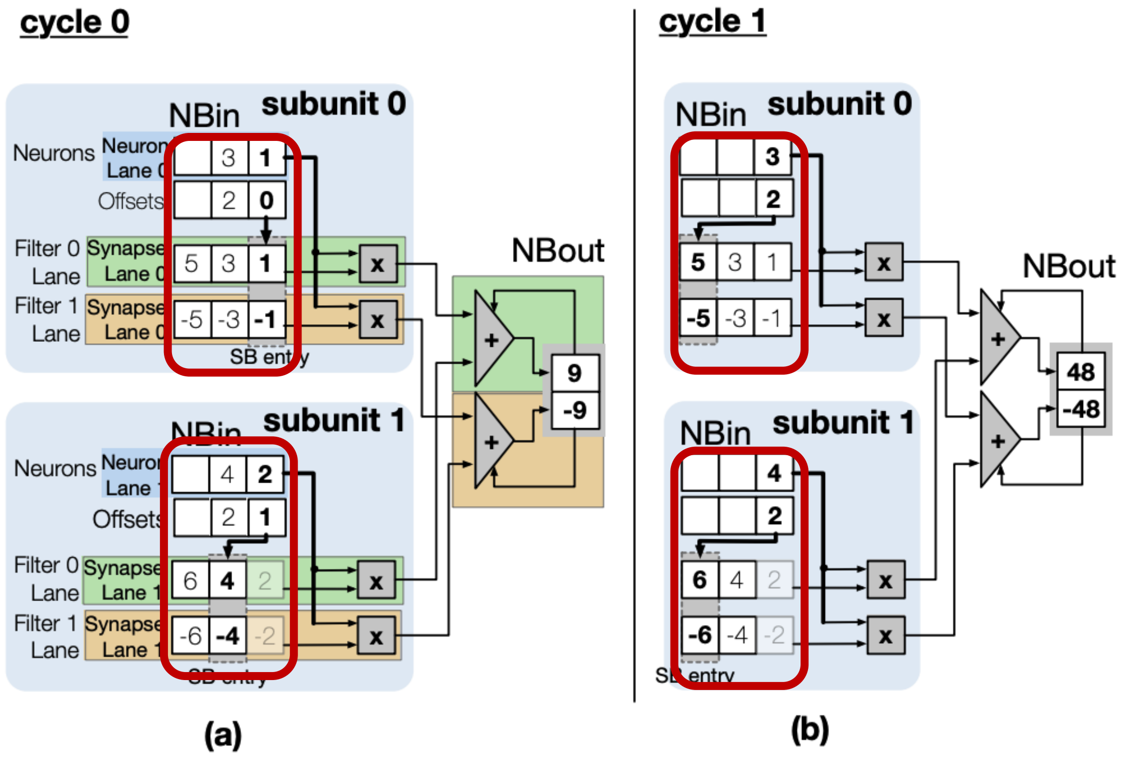
Tensor	Format (reorder dimensions by dragging the drop-down menus)	
y	Dimension 1: Dense	Dimension 2: Dense
A	Dimension 1: Dense	Dimension 2: Sparse
x	Dimension 1: Dense	Dimension 2: Dense

```
#pragma omp parallel for schedule(runtime)
for (int32_t i = 0; i < A1_dimension; i++) {
  for (int32_t pA2 = A2_pos[i]; pA2 < A2_pos[(i + 1)]; pA2++) {
    int32_t k = A2_crd[pA2];
    for (int32_t j = 0; j < x2_dimension; j++) {
      int32_t py2 = i * y2_dimension + j;
      int32_t px2 = k * x2_dimension + j;
      y_vals[py2] = y_vals[py2] + A_vals[pA2] * x_vals[px2];
    }
  }
}
```

http://tensor-compiler.org/codegen.html

Align Non-zero W & IA w/ 1-Operand Sparse

- **Indirection** for one-operand sparse case:
  - Use the index of non-zero elements of the sparse tensor to index the dense tensor.



Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing, ISCA'2016

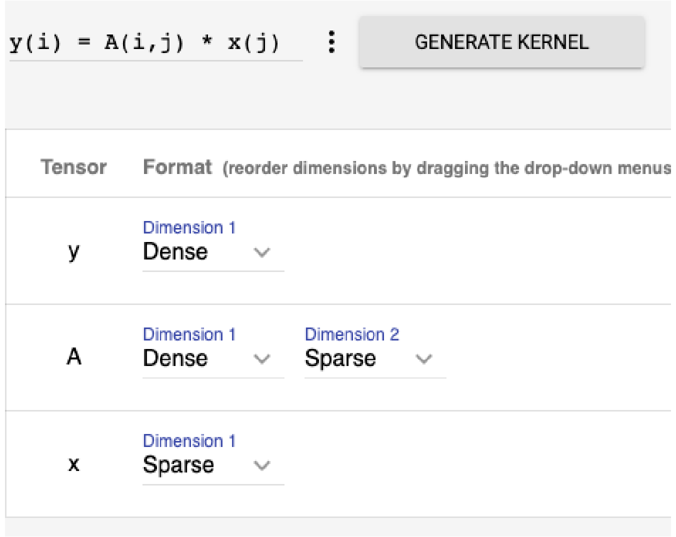
DNN Compute on Sparse Data

- Skip ineffectual computation when either W or IA is zero.
- Approaches:

	Single-Operand Sparse	Two-Operand Sparse
Align W & IA	Indirection (e.g., Cnvlutin, …)	Intersection (e.g., SNAP, …)
Align OA	Arbitration (e.g., SCNN)	

Align Non-zero W & IA w/ 2-Operand Sparse

- **Intersection** for two-operand sparse case:
  - Find the non-zero elements of both W and IA



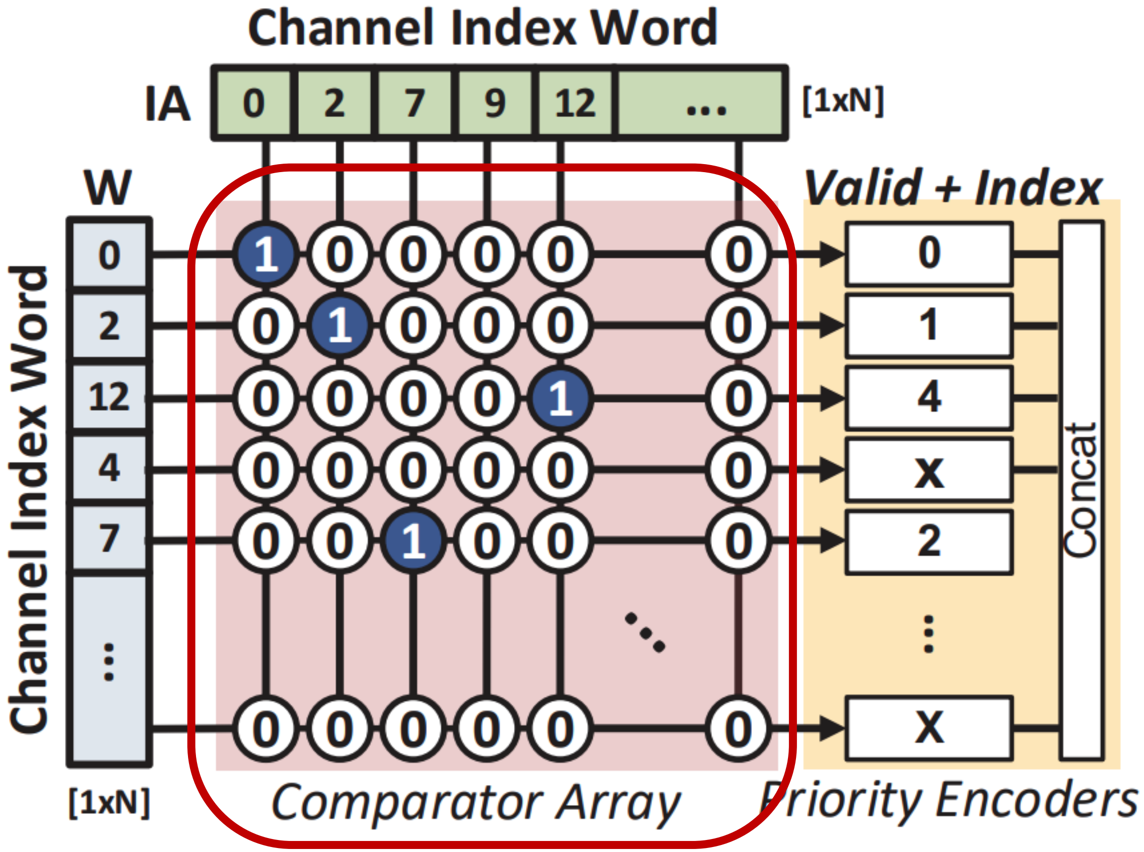

```
#pragma omp parallel for schedule(runtime)
for (int32_t i = 0; i < A1_dimension; i++) {
  int32_t pA2 = A2_pos[i];
  int32_t pA2_end = A2_pos[(i + 1)];
  int32_t px1 = x1_pos[0];
  int32_t px1_end = x1_pos[1];

  while (pA2 < pA2_end && px1 < px1_end) {
    int32_t jA = A2_crd[pA2];
    int32_t jx = x1_crd[px1];
    int32_t j = TACO_MIN(jA,jx);
    if (jA == j && jx == j) {
      y_vals[i] = y_vals[i] + A_vals[pA2] * x_vals[px1];
    }
    pA2 += (int32_t)(jA == j);
    px1 += (int32_t)(jx == j);
  }
}
```

http://tensor-compiler.org/codegen.html

Align Non-zero W & IA w/ 2-Operand Sparse

- **Intersection** for two-operand sparse case:
  - Find the non-zero elements of both W and IA



SNAP, VLSI'2019

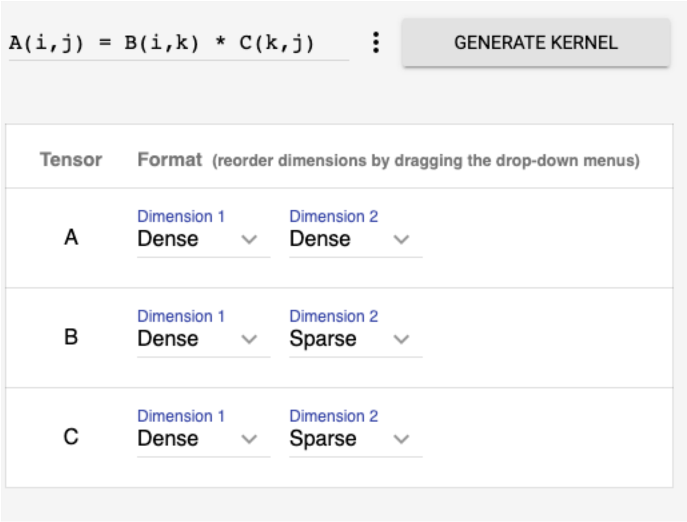
DNN Compute on Sparse Data

- Skip ineffectual computation when either W or IA is zero.
- Approaches:

	Single-Operand Sparse	Two-Operand Sparse
Align W & IA	Indirection (e.g., Cnvlutin, …)	Intersection (e.g., SNAP, …)
Align OA	Arbitration (e.g., SCNN)	

Arbitration with Cartesian Product

- All non-zero activations must (at some point in time) be multiplied by all non-zero weights (holds true for convolution stride of 1).
  - The lack of alignment in IA & W -> a large number of scattered partial sums
    - Need arbitration logic to send to corresponding output SRAM.




```
#pragma omp parallel for schedule(runtime)
for (int32_t i = 0; i < B1_dimension; i++) {
  for (int32_t pB2 = B2_pos[i]; pB2 < B2_pos[(i + 1)]; pB2++) {
    int32_t k = B2_crd[pB2];
    for (int32_t pC2 = C2_pos[k]; pC2 < C2_pos[(k + 1)]; pC2++) {
      int32_t j = C2_crd[pC2];
      int32_t pA2 = i * A2_dimension + j;
      A_vals[pA2] = A_vals[pA2] + B_vals[pB2] * C_vals[pC2];
    }
  }
}
```

http://tensor-compiler.org/codegen.html

Arbitration with Cartesian Product

- All non-zero activations must (at some point in time) be multiplied by all non-zero weights (holds true for convolution stride of 1).
  - The lack of alignment in IA & W -> a large number of scattered partial sums
    - Need arbitration logic to send to corresponding output SRAM.

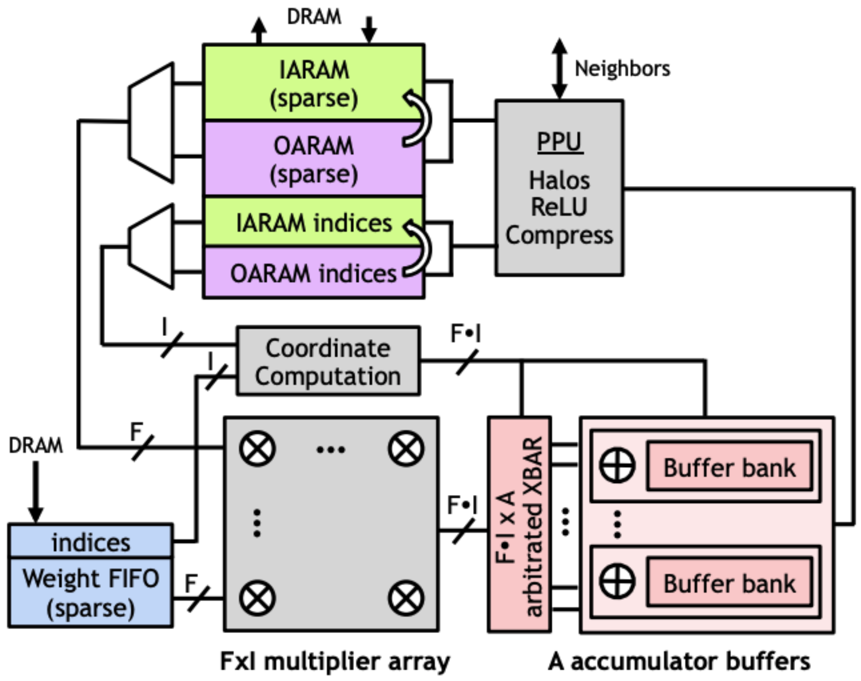


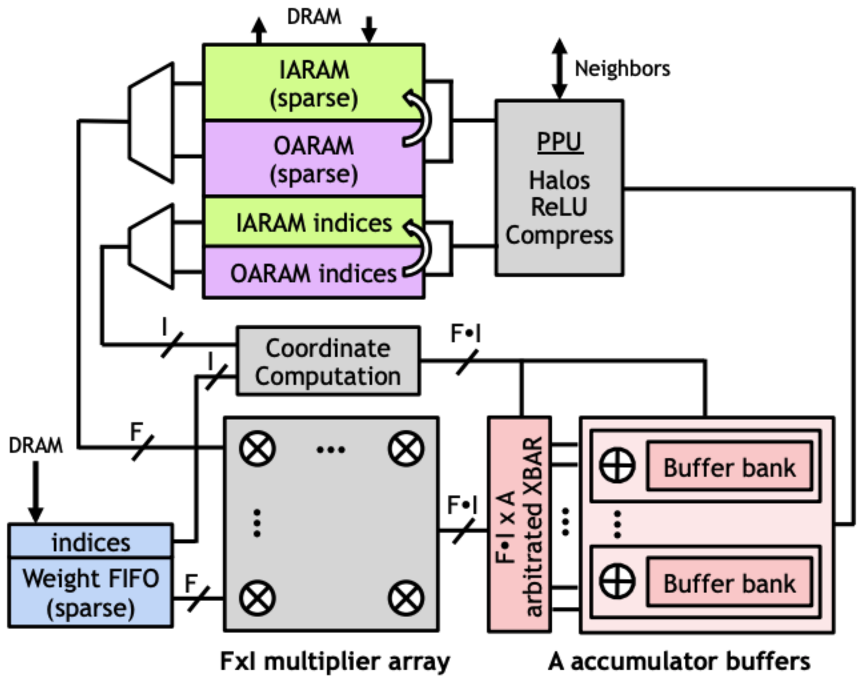
Table 4: SCNN PE area breakdown.

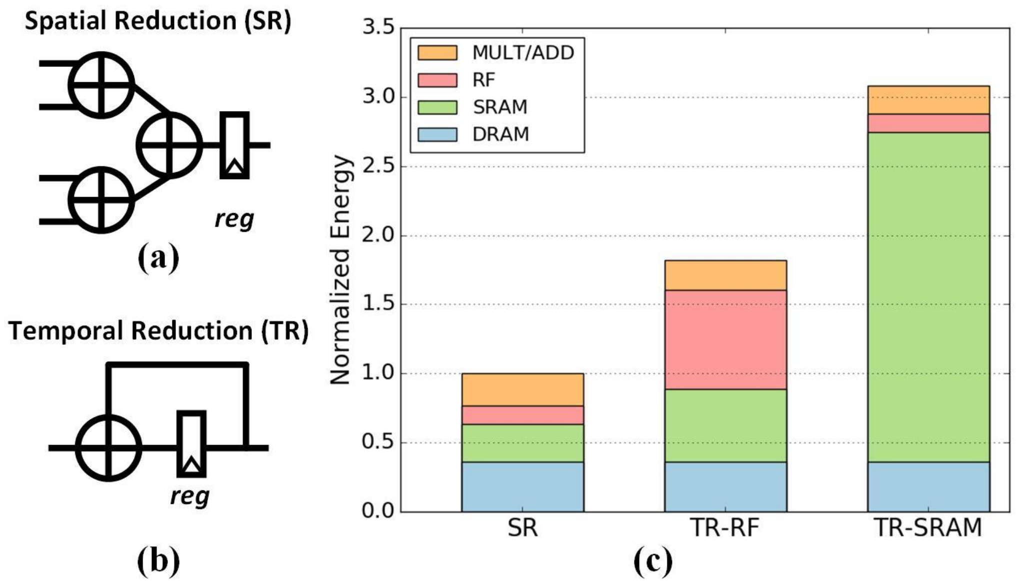
PE Component	Size	Area (mm^2)
IARAM + OARAM	20 KB	0.031
Weight FIFO	0.5 KB	0.004
Multiplier array	16 ALUs	0.008
Scatter network	16×32 crossbar	0.026
Accumulator buffers	6 KB	0.036
Other	—	0.019
Total	—	0.123
Accelerator total	64 PEs	7.9

SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks, ISCA'2017

Arbitration with Cartesian Product

- All non-zero activations must (at some point in time) be multiplied by all non-zero weights (holds true for convolution stride of 1).
  - The lack of alignment in IA & W -> a large number of scattered partial sums
    - Need arbitration logic to send to corresponding output SRAM.





Stitch-X, SysML'2018

DNN Compute on Sparse Data

- Skip ineffectual computation when either W or IA is zero.
- Approaches:

	Single-Operand Sparse	Two-Operand Sparse
Align W & IA	Indirection (e.g., Cnvlutin, …)	Intersection (e.g., SNAP, …)
Align OA	Arbitration (e.g., SCNN)	

Review

- Last lecture: data transfer mechanisms across storage hierarchy
- This lecture: sparsity in DNNs
  - Source of sparsity
  - Sparsity in storage:
    - Compression formats
  - Sparsity in compute:
    - Indirection and intersection