Deep CNNs and Energy-efficient Hardware Accelerators

Gopinath V. Mahale

7 February, 2023

Learning

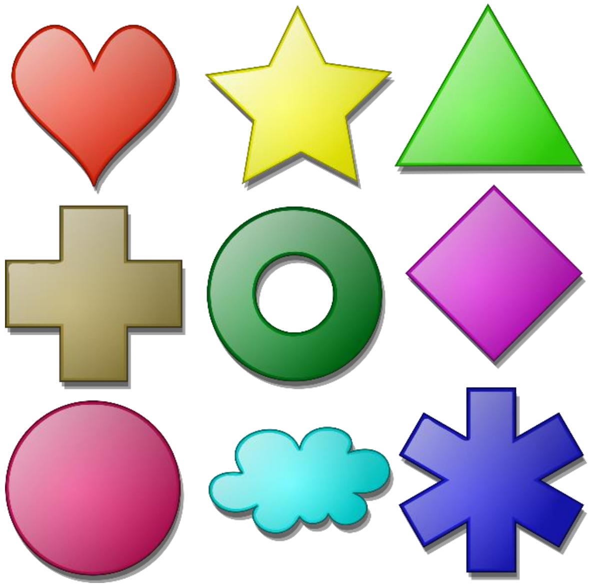
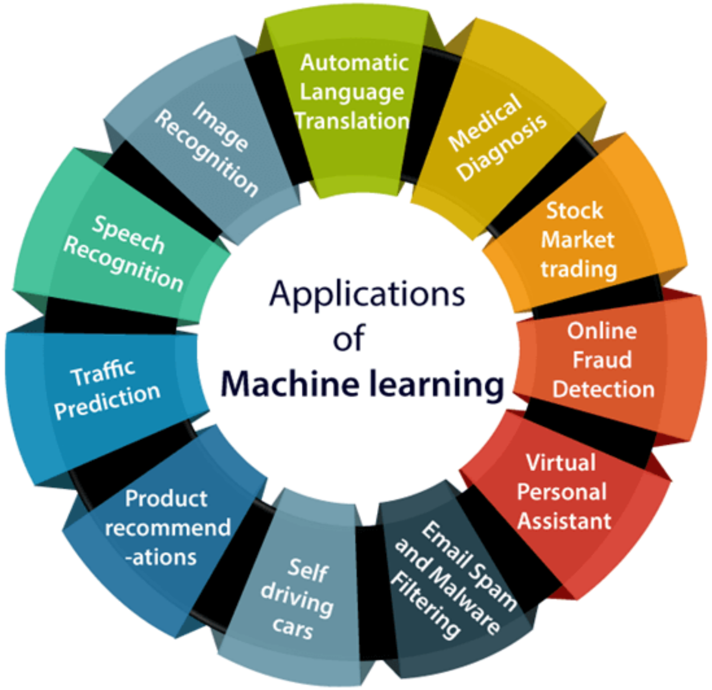


Image source: https://www.clipartmax.com/

Artificial intelligence (AI)

"Simulation of human intelligence processes by machines"



Global AI market size in Billion USD

Machine learning → Intelligent systems

Image source: https://www.javatpoint.com/applications-of-machine-learning

Agenda

- A brief introduction to Neural Networks and deep learning
- Popular networks in the field of object recognition
- Objectives in CNN hardware acceleration, well-known accelerators
- WinDConv, IKW, …
- Further topics of interest

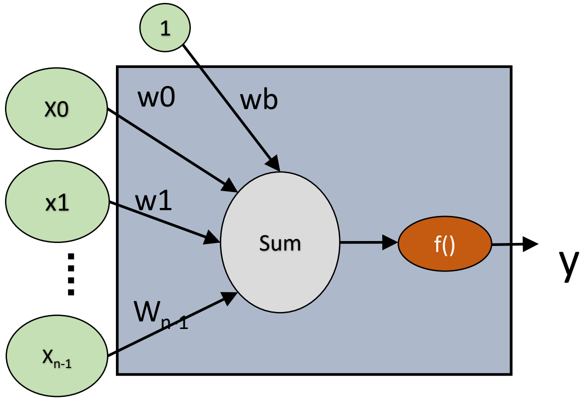
Artificial neural networks

Neurons -> Building blocks of NNs

Neuron = weighted sum + activation function

$$y = f\left(\sum_{i=0}^{n-1} (xi * wi) + wb \right)$$

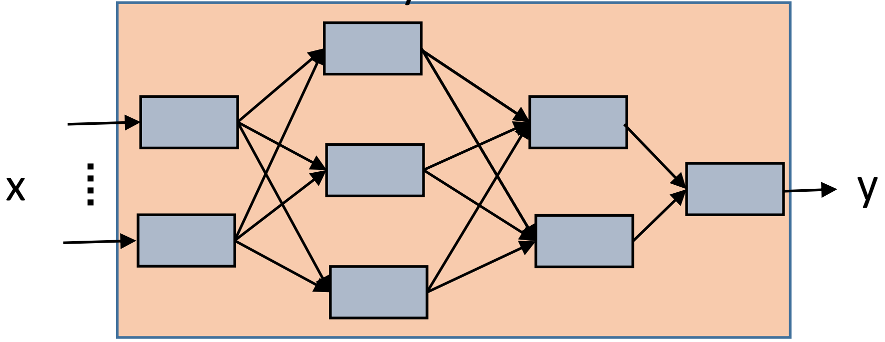
Activation functions: sigmoid, tanh, ReLU, etc.



Two phases:

1. Training
2. Inference

Multi-layer network



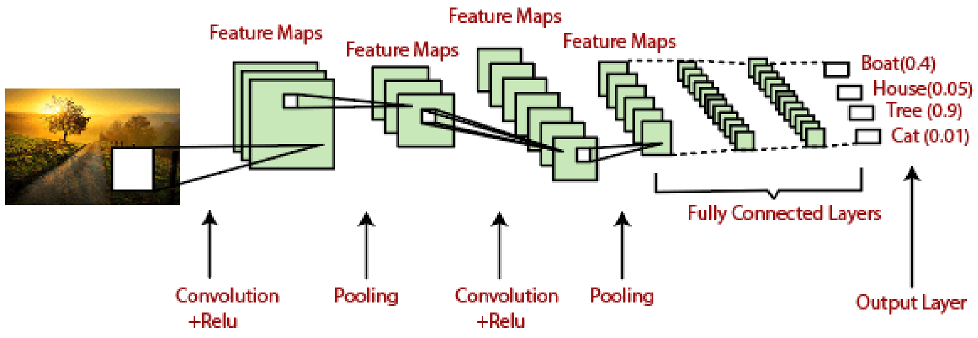
How to train?

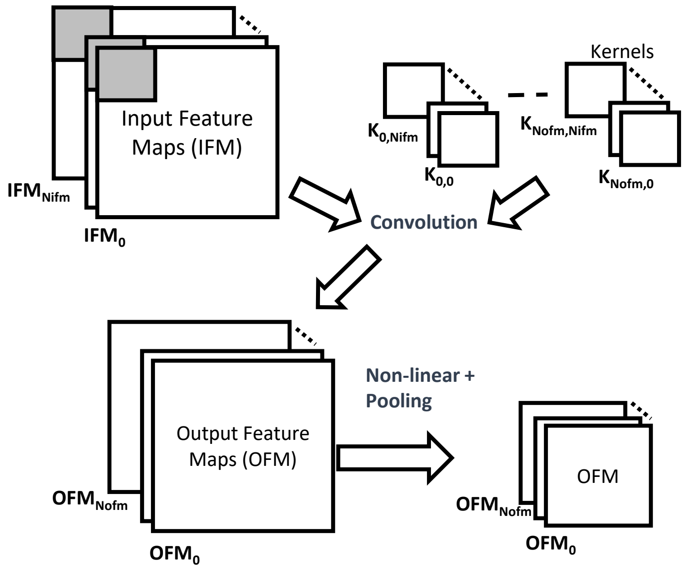
→ Gradient descent

For multi-layer networks → Error Back-propagation

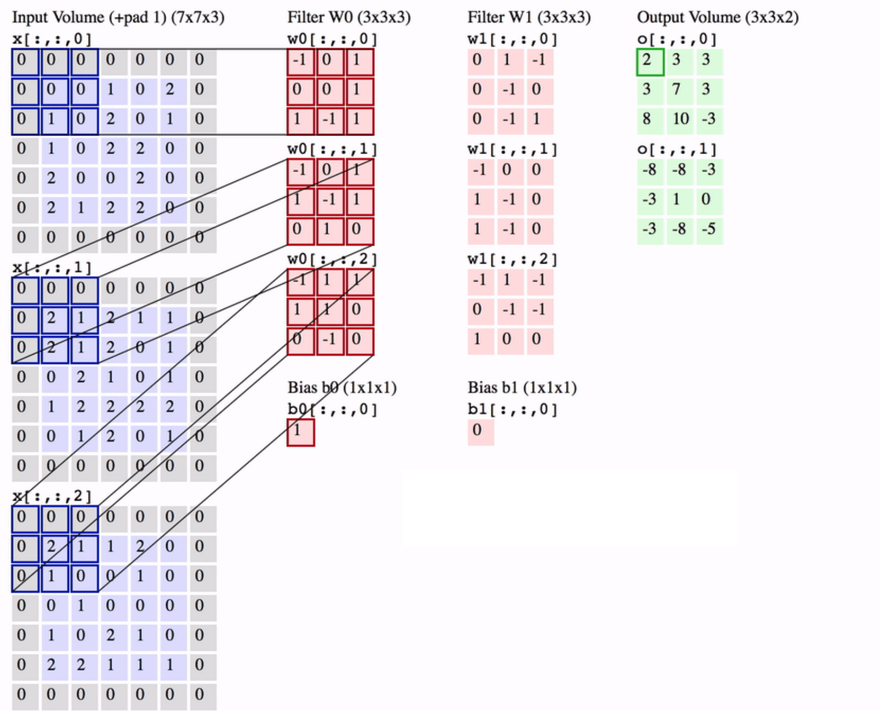
NNs → Universal approximation

Convolutional Neural Network (CNN)

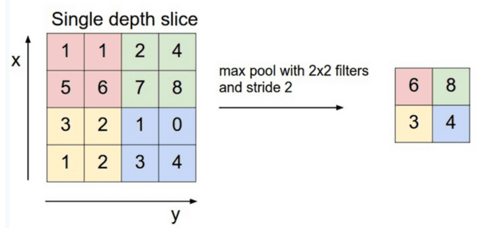




3-D Convolution



Pooling

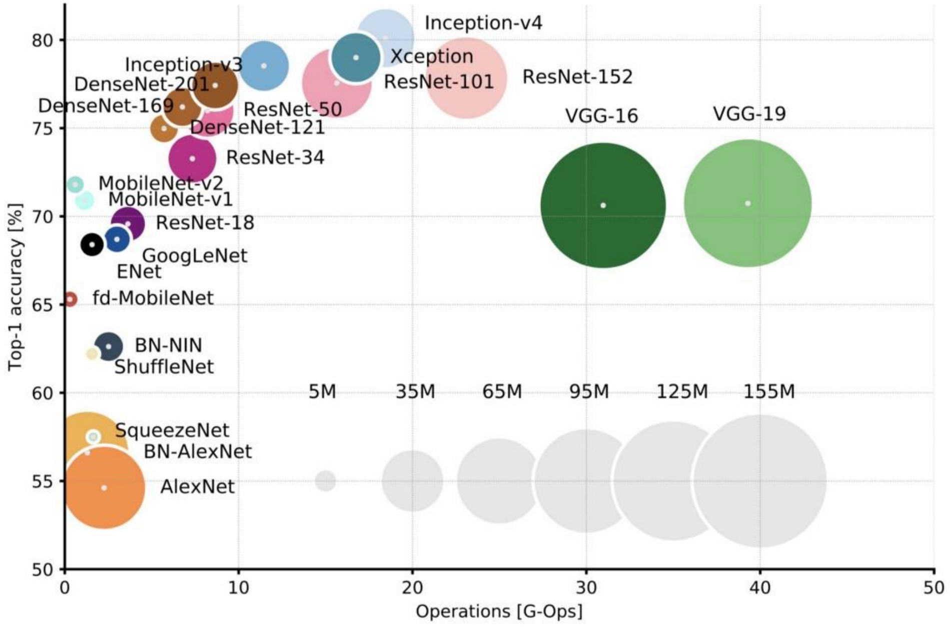


Convolution is the most compute intensive operation of CNN

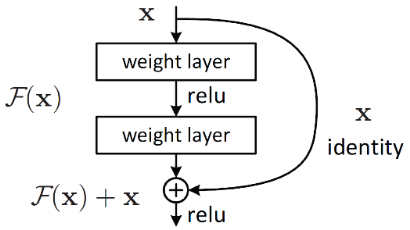
Imager source: https://mukulrathi.com/demystifying-deep-learning/convolutional-neural-network-from-scratch/

Deep CNNs

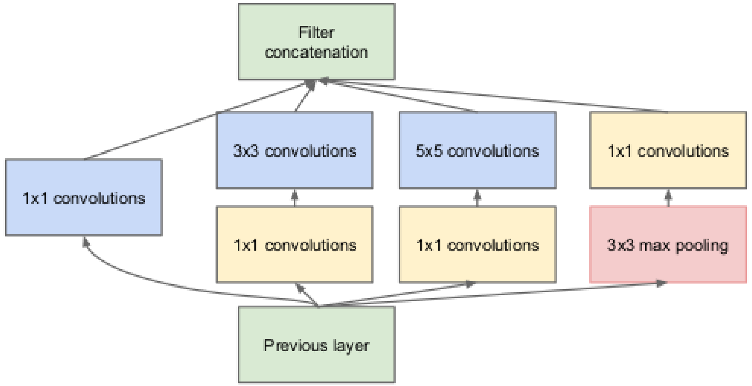
- Lenet : For digit recognition (MNIST) -> 2conv+ 2FC
- Alexnet: recognition on Imagenet dataset -> 5 Conv + 2 FC
- VGGNet: 13conv+ 3FC
- ResNet: residual layers, up to 150 layers
- Inception V1, V2, V3,V4: “Inception layer”
- Mobilenet: “Depth-wise separable convolutions”
- Learned models: NASNet, EfficientNet



Residual layer



Inception layer



Depth-wise (2-D) convolution layer

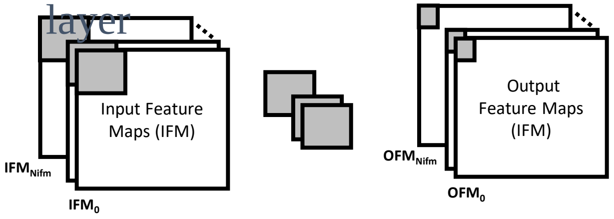


Image source: https://culurciello.medium.com/analysis-of-deep-neural-networks-dcf398e71aae

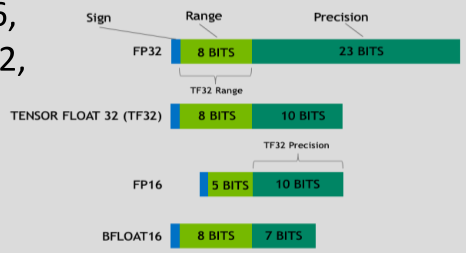
CNN accelerators: Objectives and design parameters

Target networks:

- 3-D convolution
- depth-wise
- 3-D and depth-wise convolution
- group-wise convolution
- Fully connected layer
- etc..

Precision:

- Int8, Int16, FP16, FP32, FP64
- Training?
- Mixed-precision



Energy/power

- Low Energy/power budget for embedded devices
- Different data traversals
- Power efficiency = f(comp_throughput, power)
- Energy efficiency = f(performance, energy)

Performance:

- Response time: Real-time?
- Batching : Better performance, Increased latency
- alternate methods?

Data sparsity:

- Number of zeros in data
- Introduce zeros by kernel-pruning
- Data compression
- Zero-skipping for performance
- Kernel or IFM zero-skipping?
- Power savings

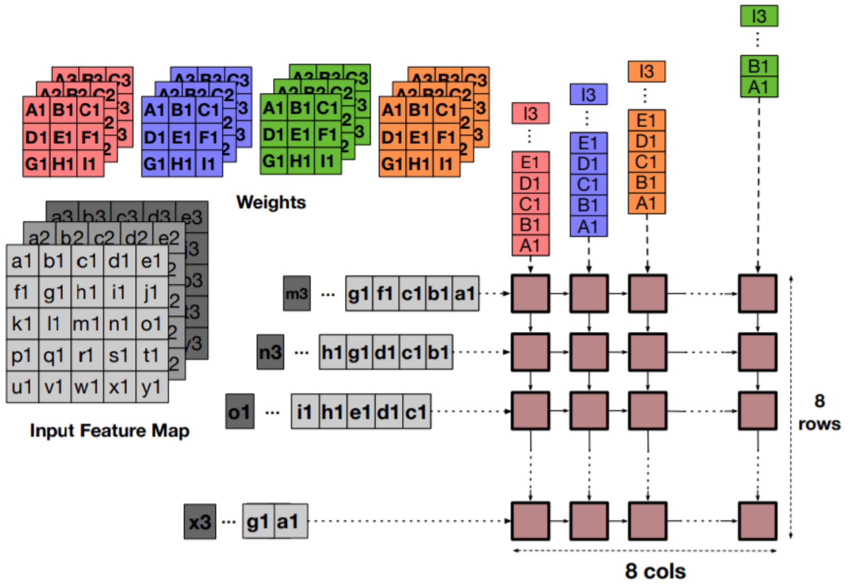
Area:

- Related to cost of the solution

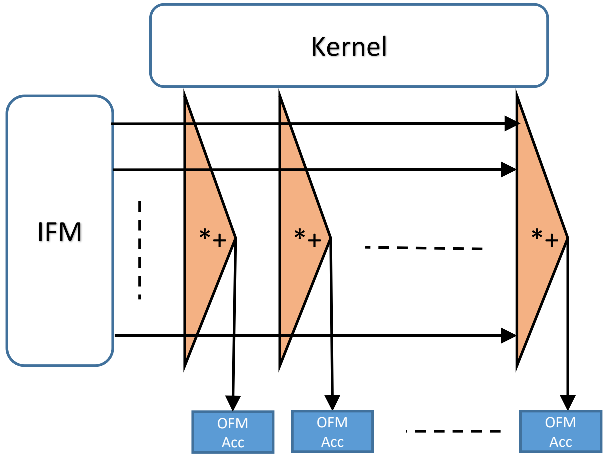
CNN accelerators

Accelerator	Features
Eyeriss	Data traversal : Row stationary
NVDLA	Open-source DL accelerator from NVIDIA, z-first data layout
Cnvlutin	Implemented IFM zero-skipping
Cnvlutin2, Zena	IFM, kernel zero skipping
Bit-Tactical	look-ahead, look-aside zero-skipping
Google TPU	Systolic array accelerator
Samsung NPU	For mobile devices

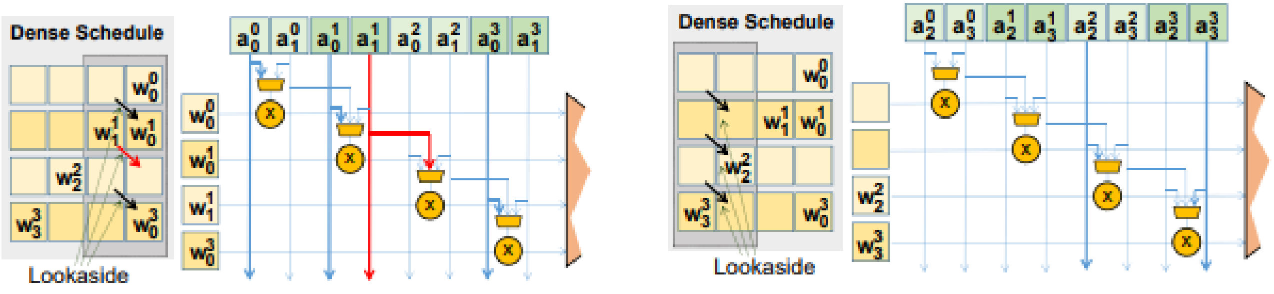
Systolic array based:



Adder tree based:



Look-aside zero-skipping:



(a) Cycle 0 (b) Cycle 1

CNN hardware acceleration on edge-devices

- **Motivation** :
  - Time-critical applications
  - unavailability of network
  - data privacy
- **Challenges :** Power constrained devices
- Vision based inference: Medical applications, Autonomous driving, etc.
- Support for on-device processing:

  \- A baseline z-first storage CNN accelerator architecture

Performance enhancement : *An alternative method of Convolution*

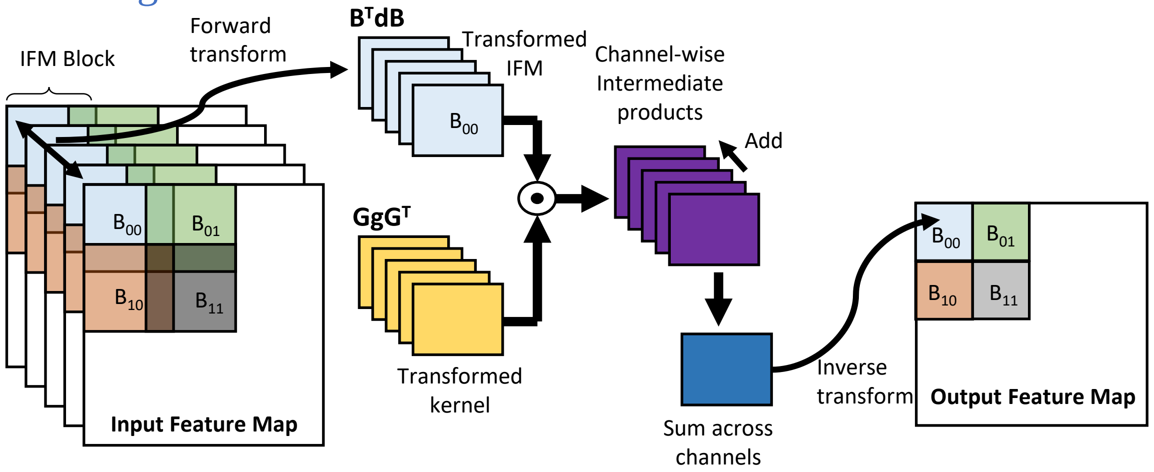
-> Better Energy Efficiency?

Power optimization : *Need: End-to-end Power optimization*

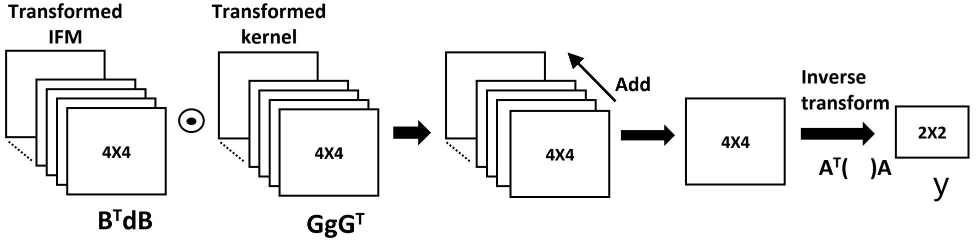
- memory hierarchy
- data traversal
- compute power

Winograd Convolution (WgConv)

(a) Generic WgConv



(b) 3×3 WgConv (for 2X2 OFM)



$$y = A^T[(GgG^T) \odot (B^TdB)]A$$

A^T =

1	1	1	0
0	1	-1	-1

B^T =

1	0	-1	0
0	1	1	0
0	-1	1	0
0	1	0	-1

G =

1	0	0
½	½	½
½	-½	½
0	0	1

Motivation to support 3×3 WgConv:

- Reduction in multiplications by 2.25X for 3×3 layers
- Majority of 3×3 convolutions
- Simple transform matrices consisting of addition and subtractions for 3×3 WgConv (for 2x2 OFM)

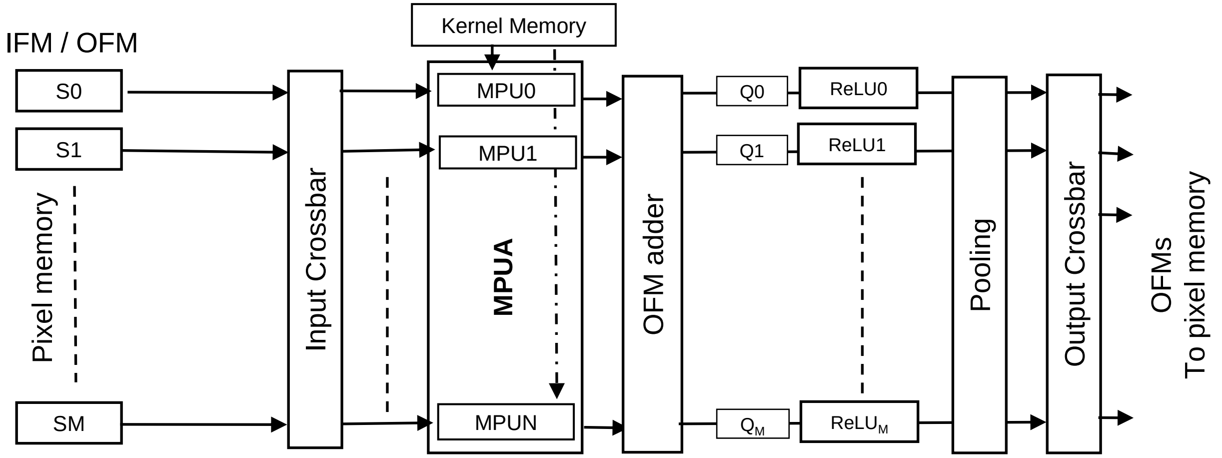
Challenges:

- Problem of sparsity -> WgConv_m
- Problem of common traversal -> Fused datapath

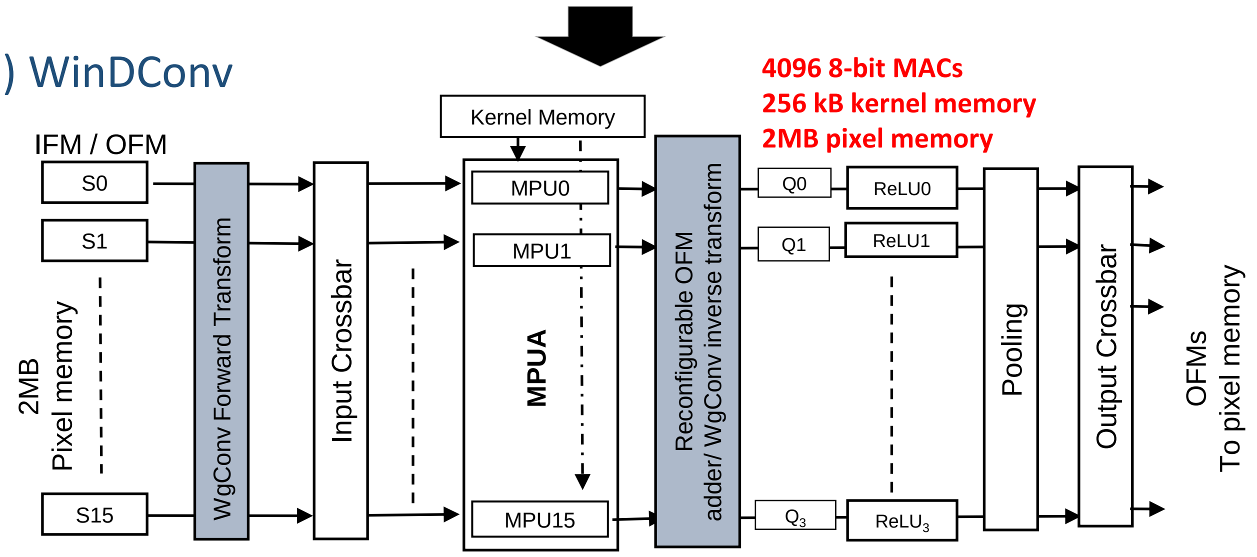
3X3 WgConv has simpler transforms, and provides performance improvement of 2.25X

A fused datapath to support Winograd Convolution

(a) Baseline architecture



(b) WinDConv

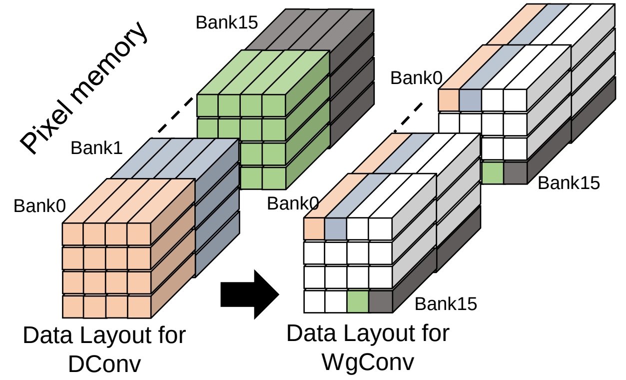


4096 8-bit MACs
256 kB kernel memory
2MB pixel memory

Salient features:

- Support for DConv and 3X3, 3X1 and 1X3 WgConv
- Complete utilization of compute units in both DConv and WgConv through efficient data traversals
- Specialized memory and data layouts
- Small increase in power and area
  => improved performance
  => improved energy efficiency
- 4096 8-bit MACs, 256kB kernel memory, 2MB pixel memory, 128 bit memory word

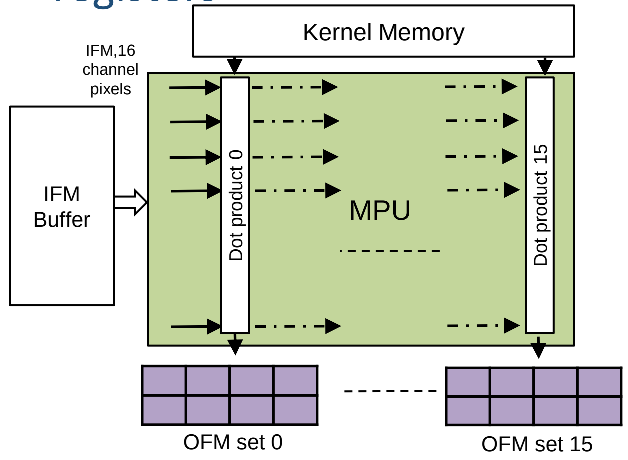
(c) Data layout of IFM



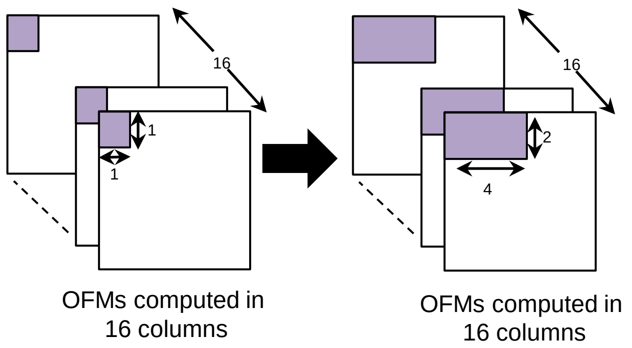
Power-efficient hybrid traversal apparatus and method for convolutional neural network accelerator architecture
GV Mahale, PP Udupa, KK Chandrasekharan, SH Lee - US Patent App. 17/033,132, 2021

The Hybrid traversal (WinDConv_h)

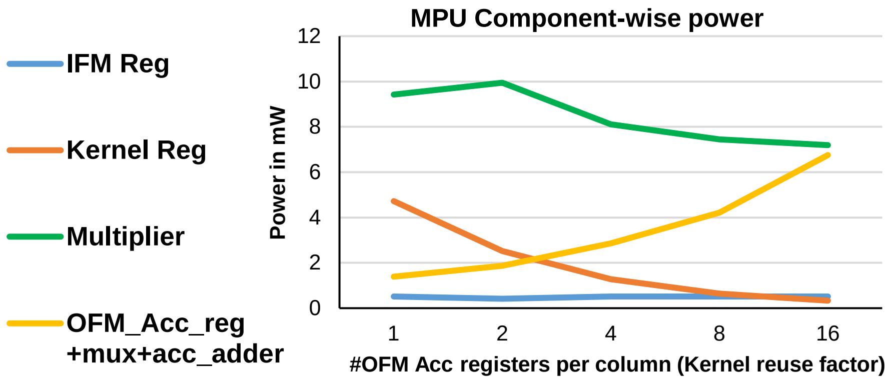
(a) MPU with additional accumulator registers

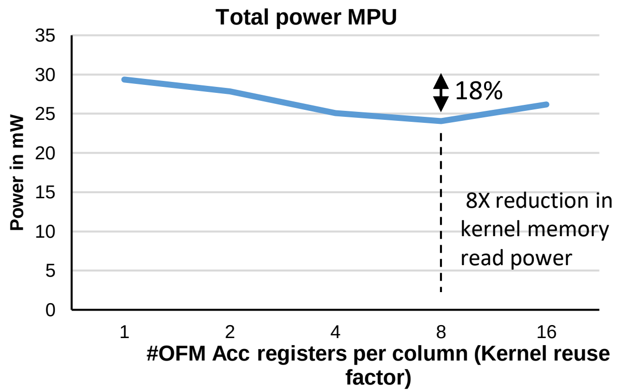


(b) OFM pixels computed under each MPU



(c) Reduction in MPU power by kernel reuse for DConv





Salient features:

- Reduced kernel memory access power, kernel register power and multiplier power
- Hybrid traversal for both DConv and WgConv
- Hybrid traversal for variants of Dconv: dilated, depth-wise, strided and deconvolution
- Hybrid traversal shows **2.8X** and **2.1X** power reduction in DConv and WgConv modes respectively

Z-first reference neural processing unit for mapping winograd convolution and a method thereof
GV Mahale, PP Udupa, Kiran KC, SH Lee - US Patent App. 17/239,892, 2021

The hybrid traversal

```
// OUTPUT STATIONARY
FOR (K=0; K<NOFM; K++)               //NUMBER OF OFMS
  FOR (C=0; C<N_CH_IFM/16; C++)      //NUMBER OF MICROBATCHES
    FOR (I=0; I<NROW_OFM; I++)       //OFM HEIGHT
      FOR (J=0; J<NCOL_OFM; J++)     //OFM WIDTH
        FOR (K1=0; K1<3; K1++)       //KERNEL HEIGHT
          FOR (K2=0; K2<3; K2++)     //KERNEL WIDTH
            FOR (C1=0; C1<16; C1++)//MICROBATCH
              OFM(K,I,J) = OFM(K,I,J)
              + IFM(C,I+K1,J+K2)*KER(K,C,K1,K2)
```

```
// PROPOSED HYBRID TRAVERSAL DConv
FOR (K=0; K<NOFM; K++)                    //NUMBER OF OFMS
  FOR (C=0; C< N CH IFM/16; C++)          //NUMBER OF MICROBATCHES
    FOR (I=0; I<NROW_OFM; I=I+4)          //OFM HEIGHT
      FOR (J=0; J< NCOL OFM; J=J+2)       //OFM WIDTH
        FOR (K1=0; K1<3; K1++)            //KERNEL HEIGHT
          FOR (K2=0; K2<3; K2++)          //KERNEL WIDTH
            FOR (T1=0; T1<4;T1++)         //PARTIAL OFM ROWS
              FOR (T2=0; T2<2; T2++)      //PARTIAL OFM COLUMNS
                FOR (C1=0; C1< 16; C1++)//MICROBATCH
                  OFM(K,I+T1,J+T2) = OFM(K,I+T1,J+T2)
                  + IFM(C,I+T1+K1,J+T2+K2)*KER(K,C,K1,K2)
```

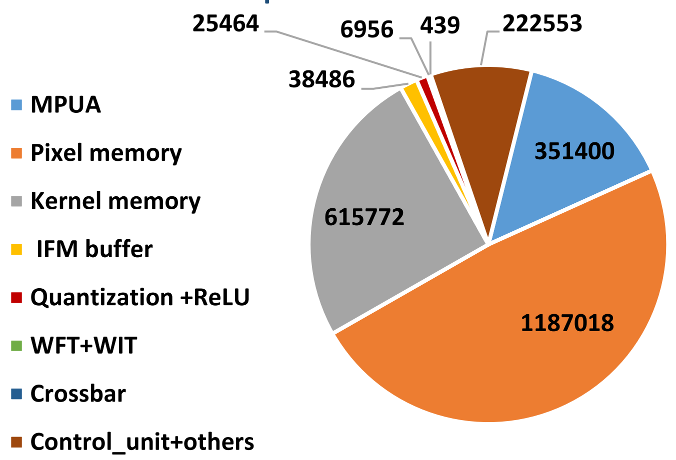
```
// WEIGHT STATIONARY
FOR (K=0; K<NOFM; K++)                //NUMBER OF OFMS
  FOR (C=0; C<N CH IFM/16; C++)       //NUMBER OF MICROBATCHES
    FOR (K1=0; K1<3; K1++)            //KERNEL HEIGHT
      FOR (K2=0; K2<3; K2++)          //KERNEL WIDTH
        FOR (I=0; I<NROW_OFM; I++)    //OFM HEIGHT
          FOR (J=0; J<NCOL_OFM; J++)//OFM WIDTH
            FOR (C1=0; C1<16; C1++) //MICROBATCH
              OFM(K,I,J) = OFM(K,I,J)
              + IFM(C,I+K1,J+K2)*KER(K,C,K1,K2)
```

```
// PROPOSED HYBRID TRAVERSAL WgConv
FOR (K=0; K<NOFM; K=K+2)                   //NUMBER OF OFMS
  FOR (C=0; C<N_CH_IFM/16; C++)            //NUMBER OF MICROBATCHES
    FOR (I=0; I<NROW_OFM; I=I+4)           //OFM HEIGHT
      FOR (J=0; J<NCOL_OFM; J=J+2)         //OFM WIDTH
        FOR (K1=0; K1<4; K1++)           //TRANSFORMED KERNEL HEIGHT
          FOR (K2=0; K2<4; K2++)         //TRANSFORMED KERNEL WIDTH
            FOR (S=0; S<2; S++)            //TWO KERNEL REGISTERS
              FOR (T1=0; T1<2;T1++)        //PARTIAL OFM ROWS
                FOR (T2=0; T2<2; T2++)     //PARTIAL OFM COLUMNS
                  FOR (C1=0; C1<16; C1++)//MICROBATCH
                    OFM(K+S,I+T1,J+T2) = OFM(K+S,I+T1,J+T2)
                    + IFM(C,I+T1+K1,J+T2+K2)*KER(K+S,C,K1,K2)
```

For DConv => Hybrid of weight and output stationary
For WgConv => Hybrid of input, weight and output stationary

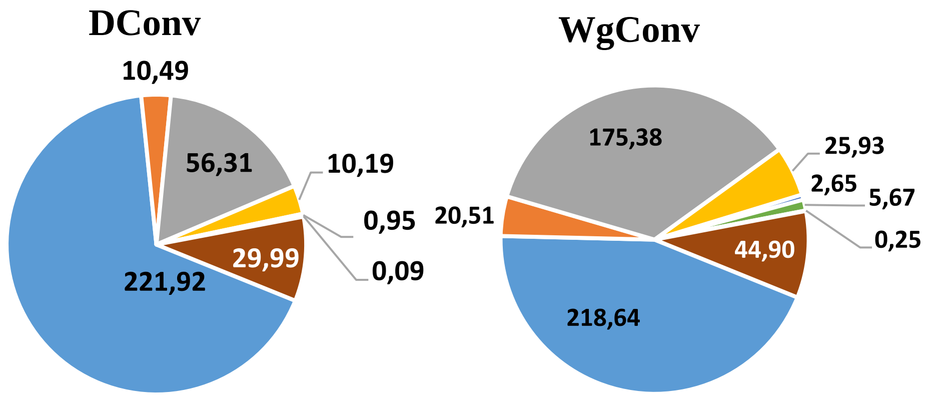
Synthesis results

(a) Area break-up of individual modules in Area in um2



(Samsung 10nm, 800MHz)

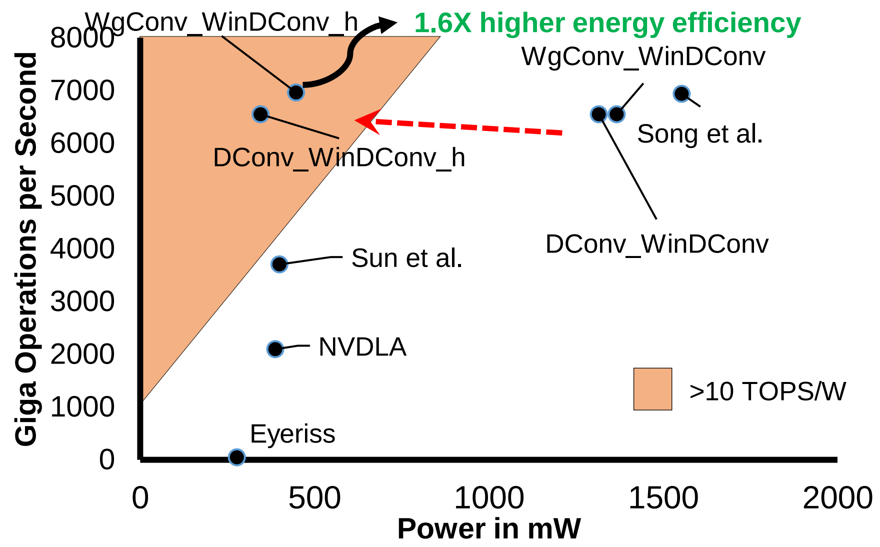
(b) Power break-up of individual modules in mW (run for Inception-v3)



(c) Power and power efficiency for practical CNNs

CNN	Power DConv (mW)	Power $WgConv_m$ 3X3 layers (mW)	Power DConv +$WgConv_m$ (mW)	Averag e TOPS/ W	Energy/OFM improvement by $WgConv_m$ (3X3 layers)
VGG-16	358.44	497.51	497.51	12.4	1.6X
ResNet-101	374.01	483.94	483.94	12.35	1.7X
Inception-V3	333.74	455.46	363.45	13.9	1.9X
Inception-V4	337.35	452.25	361.1	14.4	1.9X

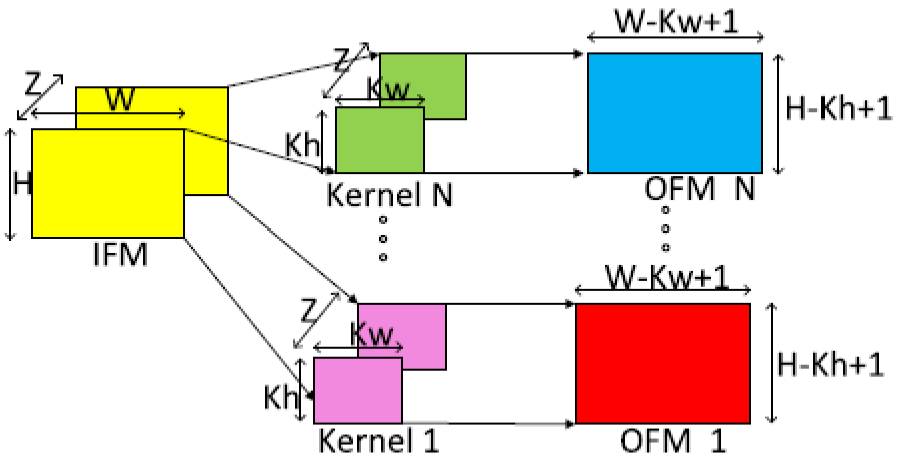
(d) Power-performance plot

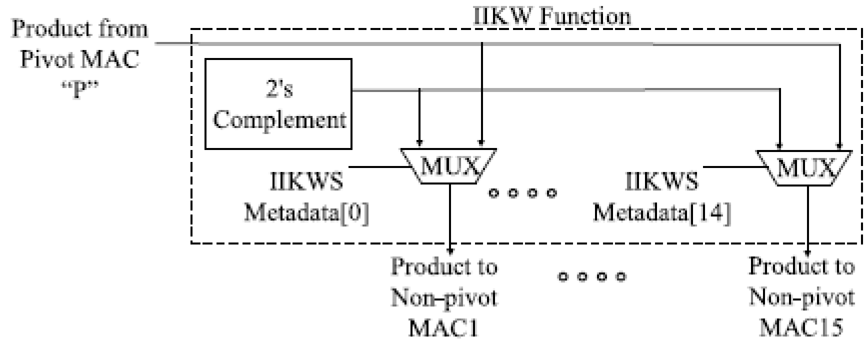


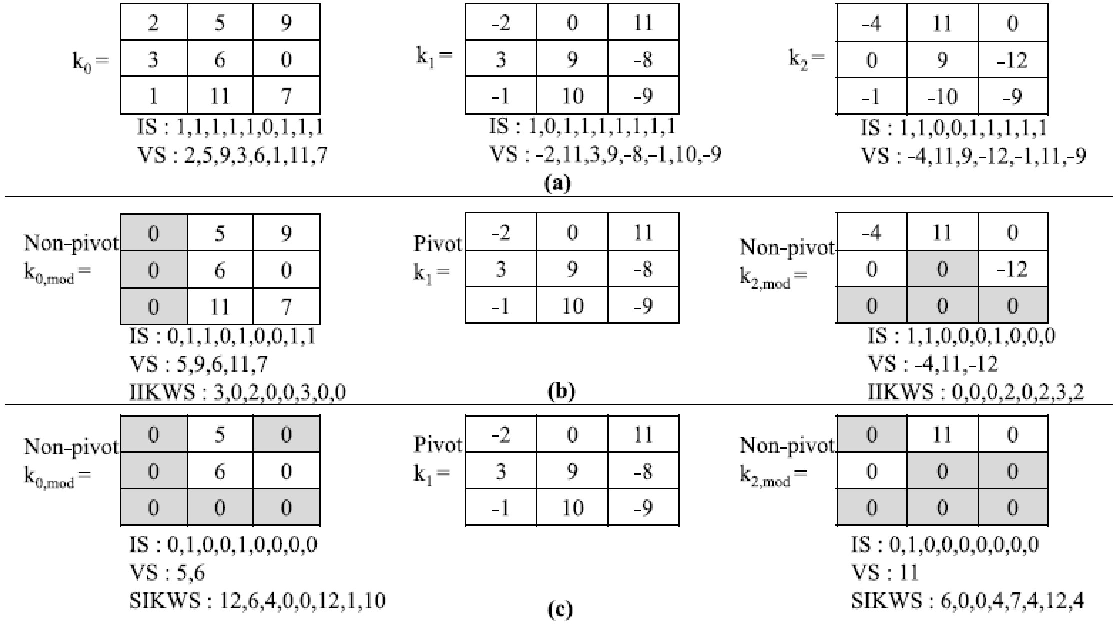
~3% increase in area from additional accelerators.
over **2X gain** in power efficiency and over **3X gain** in energy efficiency

G. Mahale, P. Udupa, K. K. Chandrasekharan and S. Lee, "WinDConv: A Fused Datapath CNN Accelerator for Power-Efficient Edge Devices," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 4278-4289, Nov. 2020

Inter-kernel weights







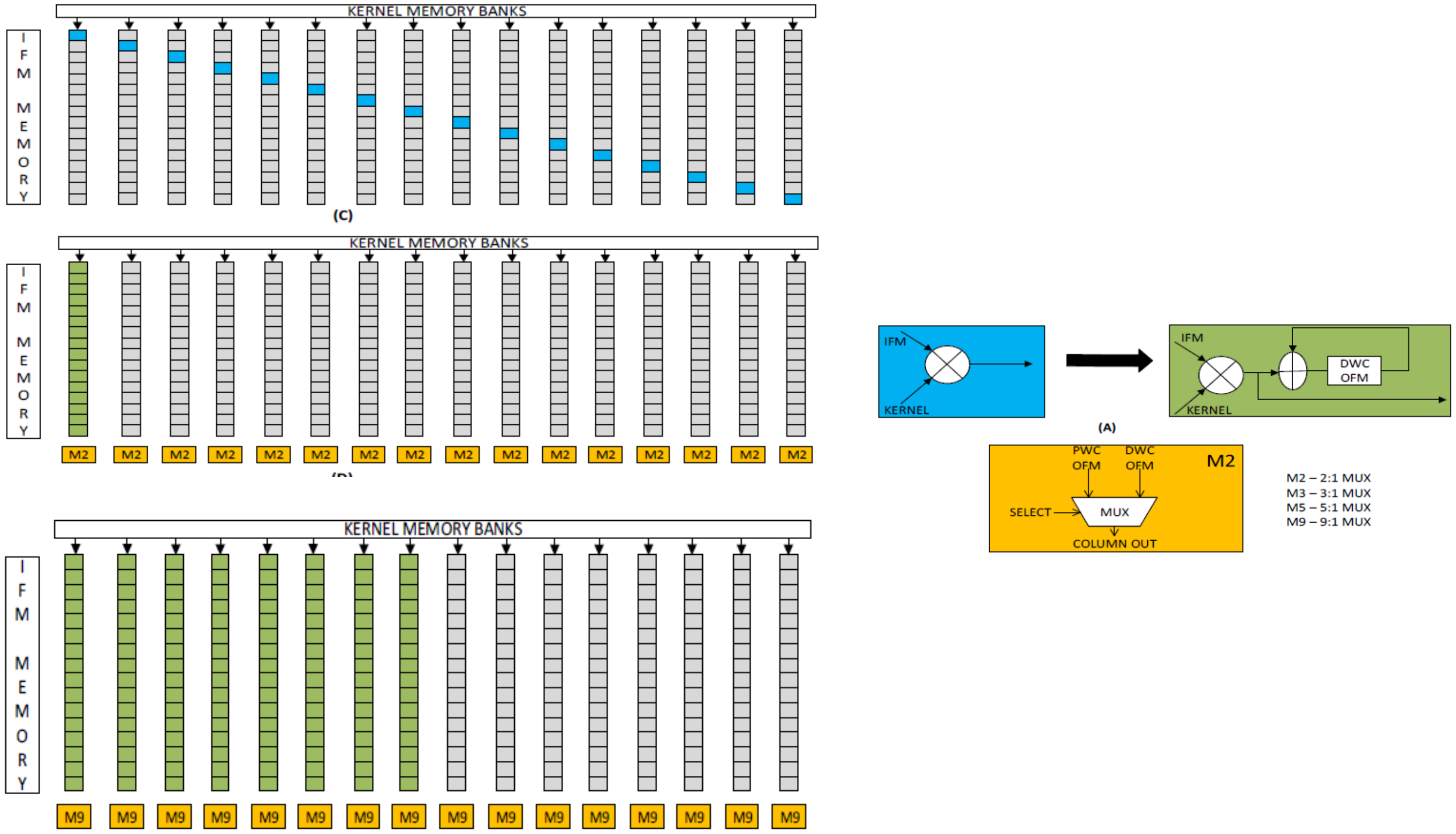
Kernel weights having the same co-ordinate and same channel number across two kernels in a CNN layer multiply with same IFM pixel

- About 30% sparsity enhancement in Practical CNNs
- Low precision -> higher sparsity
- 15% to 30% reduction in power
- Can be an alternative to kernel pruning process

P. Udupa, G. Mahale, K. K. Chandrasekharan and S. Lee, "IKW: Inter-Kernel Weights for Power Efficient Edge Computing," in IEEE Access, vol. 8, pp. 90450-90464, 2020

Support for Depth-wise convolution (DWC) layers and pooling

53.5% of total cycles for all DWC layers which have only 8% of total computations



Kernel Size	Kernel Stride	Throughput	Power (in mW)	Throughput per mW	Power Efficiency Improvement
DWC mapped on the baseline architecture					
3 × 3	1, 2	1.78 (16/9)	5.23	0.34	-
5 × 5	1, 2	0.64 (16/25)	5.23	0.12	-
DWC mapped on the proposed 8-MCA architecture					
3 × 3	1	7.11 (64/9)	14.23	0.49	1.46×
3 × 3	2	3.55 (32/9)	6.04	0.58	1.73×
5 × 5	1	2.56 (64/25)	13.28	0.19	1.57×
5 × 5	2	2.56 (64/25)	14.24	0.179	1.47×
Pooling mapped on the baseline architecture					
3 × 3	1, 2	1.78 (16/9)	4.1	0.433	-
2 × 2	2	4 (16/4)	4.1	0.974	-
Pooling mapped on the proposed 8-MCA architecture					
3 × 3	1	7.11 (64/9)	8.5	0.84	1.92×
3 × 3	2	3.55 (32/9)	3.62	0.98	2.26×
2 × 2	2	4 (16/4)	2.16	1.85	1.89×

# of Augmented Columns	MAC Tile Area (um^2)	Increase
6	36091.54	8.80%
8	37064.4	11.73%
10	38037.26	14.66 %

P. Udupa, G. Mahale, K. K. Chandrasekharan and S. Lee, "Accelerating Depthwise Convolution and Pooling Operations on z-First Storage CNN Architectures," 2020 IEEE International Symposium on Circuits and Systems (ISCAS)

Topics of interest

- Support for different convolution types
- Winograd convolution on other platforms
- Z-first data layout for practical CNNs
- Near-memory computing
- Sequential networks:

    LSTMs, Transformers for Images, NLP, speech recognition etc.

Summary

- AI is ubiquitous in our everyday life
- Neural networks are universal approximations
- Workload of recent CNNs is significantly high, requiring energy-efficient acceleration
- Mapping computations from different layer types on a common accelerator has been a challenge
- Accelerator needs mixed-precision acceleration
- Introduction of networks like Transformers have opened scope for new lines of research

Thank you!

gopinath.mahale@bsc.es