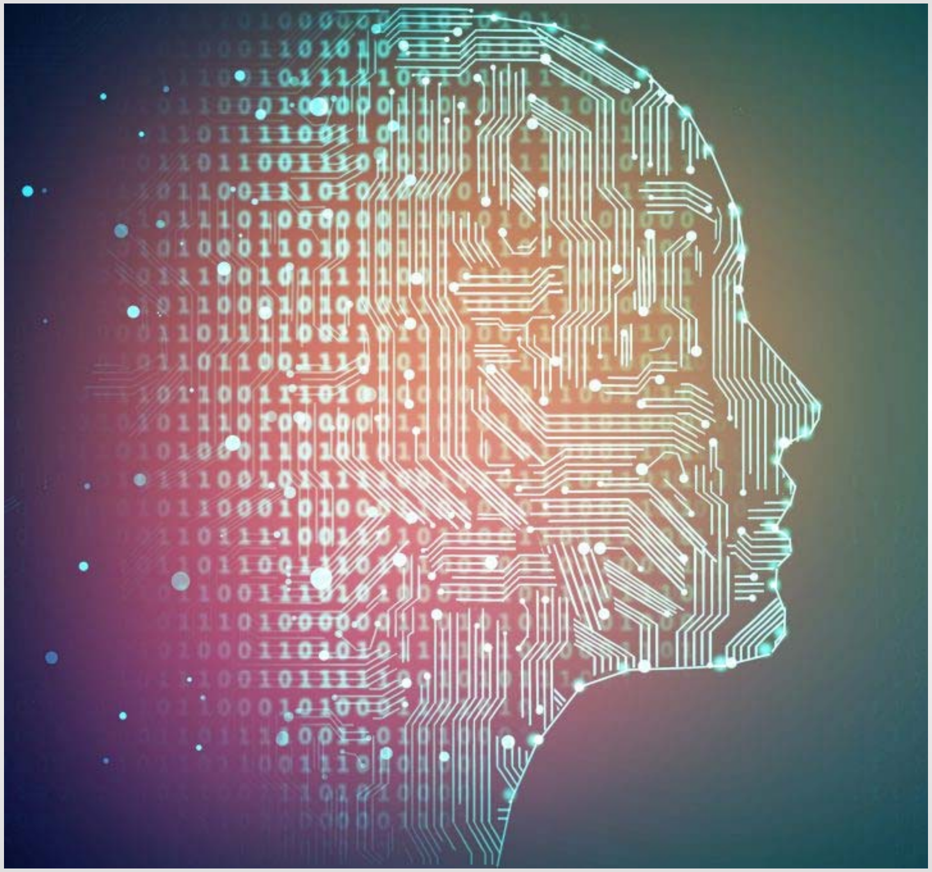


Efficient Machine Learning Architectures

T. N. Vijaykumar

EMC^2 Workshop, HPCA, Feb 17 2019

Machine Learning is Exploding

- Breakthrough progress in ML
  - Neural networks more accurate than human experts
  - E.g., Google AlphaGo, IBM Debater
- Great commercial interest
  - Self-driving cars, personal assistants, drug design, investment, medical diagnosis, smart home, OMG!
- Barely scratched the surface
  - Breakthroughs in ML for faster deeper learning

New “application pull”

Technology Landscape

- Moore's Law is slowing down
  - Dennard's scaling stopped
  - Thermal limit in handheld devices
- New exciting directions opening
  - 2.5-D/3-D stacking, processing-in-memory
- Efficiencies through domain-specific designs

New "technology push"

Architecture's Role

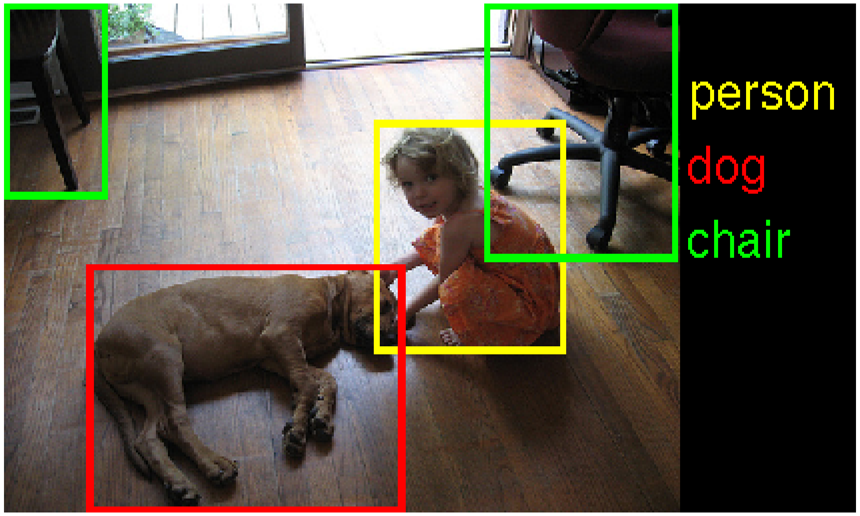
- Match up technology and ML workloads
  - Exploit ML workload characteristics
- Better performance, energy, programmability, reliability, …
  - At same ML model accuracy

- Improved accuracy or new models
  - OR accuracy-time/energy trade-off
  - Architects alone can't evaluate
  - Requires ML input

Let loose our innovation!

Theme

- Efficiency through
  - fine-grain
  - regularity
  - parallelism
  - reuse



[ImageNet]

Outline

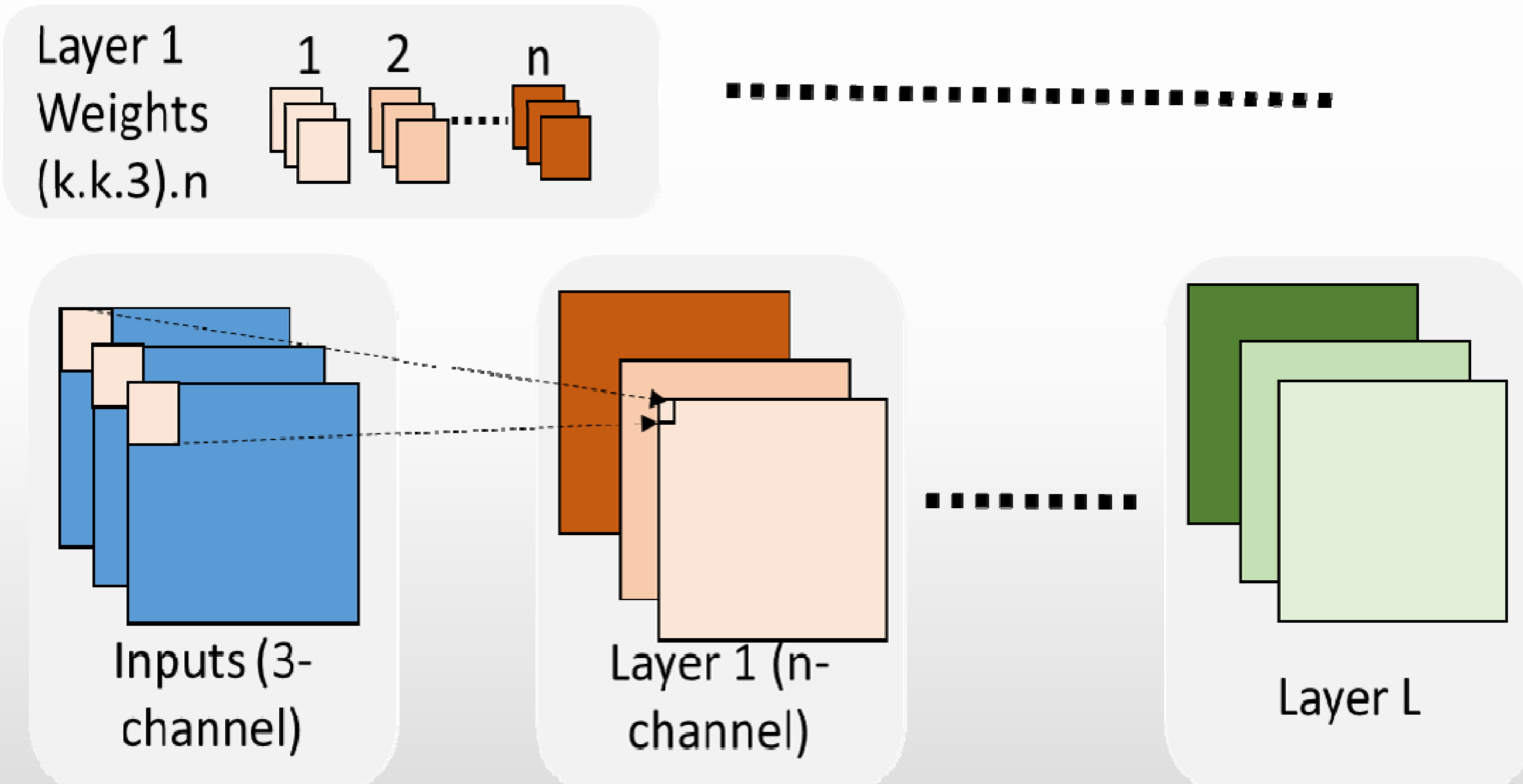
- Introduction
- Workload characteristics
- ML architectures
- Looking forward
- Conclusion

ML Workload Characteristics (1/4)

- Convolutional Neural Networks (CNNs), Long Short-Term Memories (LSTMs), Multi-level Perceptrons (MLPs), Recurrent Neural Networks (RNNs), Reinforcement Learning (RL)
  - Support Vector Machines (SVM), Regression
- Models extract features by applying filters to input
  - Filters trained
- Mostly, matrix-matrix or -vector multiplication
  - Training or inference
- And some “local” operations (e.g., ReLU, pooling)

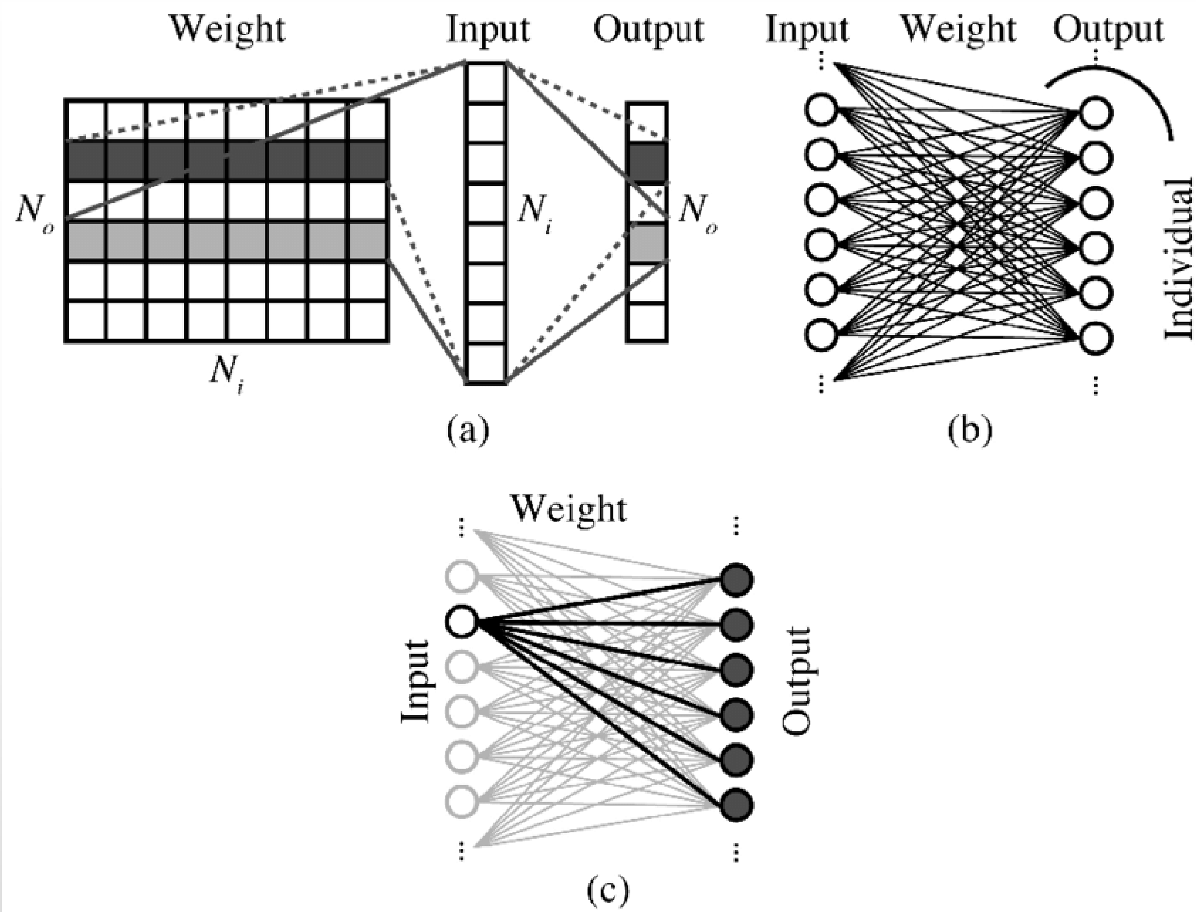
Fine-grained, regular

An Example CNN Inference



A Fully Connected Layer

[Ando'17]



Fine-grained, regular compute and memory access

ML Workload Characteristics (1/4)

Each layer of network:

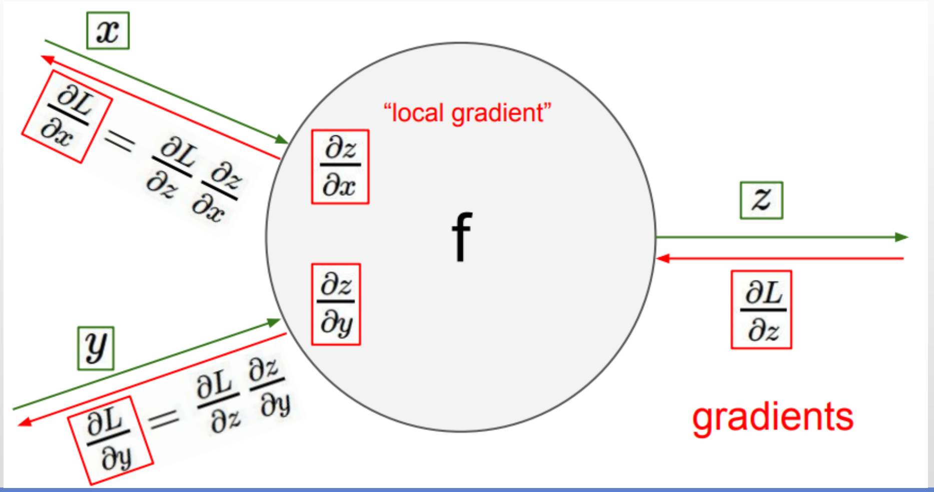
- Input x filters (aka weights) = output feature map
- CNNs: matrix x matrix = matrix
- RNNs: vector x matrix = vector

$$(i_1, i_2, i_3, \ldots . I_{n-1}, i_n) \times \begin{Bmatrix} f_{11}, f_{12}, f_{13}, \ldots . f_{1\ k-1}, f_{1\ k} \\ \ldots . \\ f_{n1}, f_{n2}, f_{n3}, \ldots . f_{n\ k-1}, f_{n\ k} \end{Bmatrix}$$

Fine-grained, regular compute and memory

CNN Backpropagation

- Error propagation
  - Convolution with a rotated filter
  - Matrix-matrix multiplication
- Gradient descent
  - Matrix-matrix multiplication and summation
  - [CS231n Stanford]



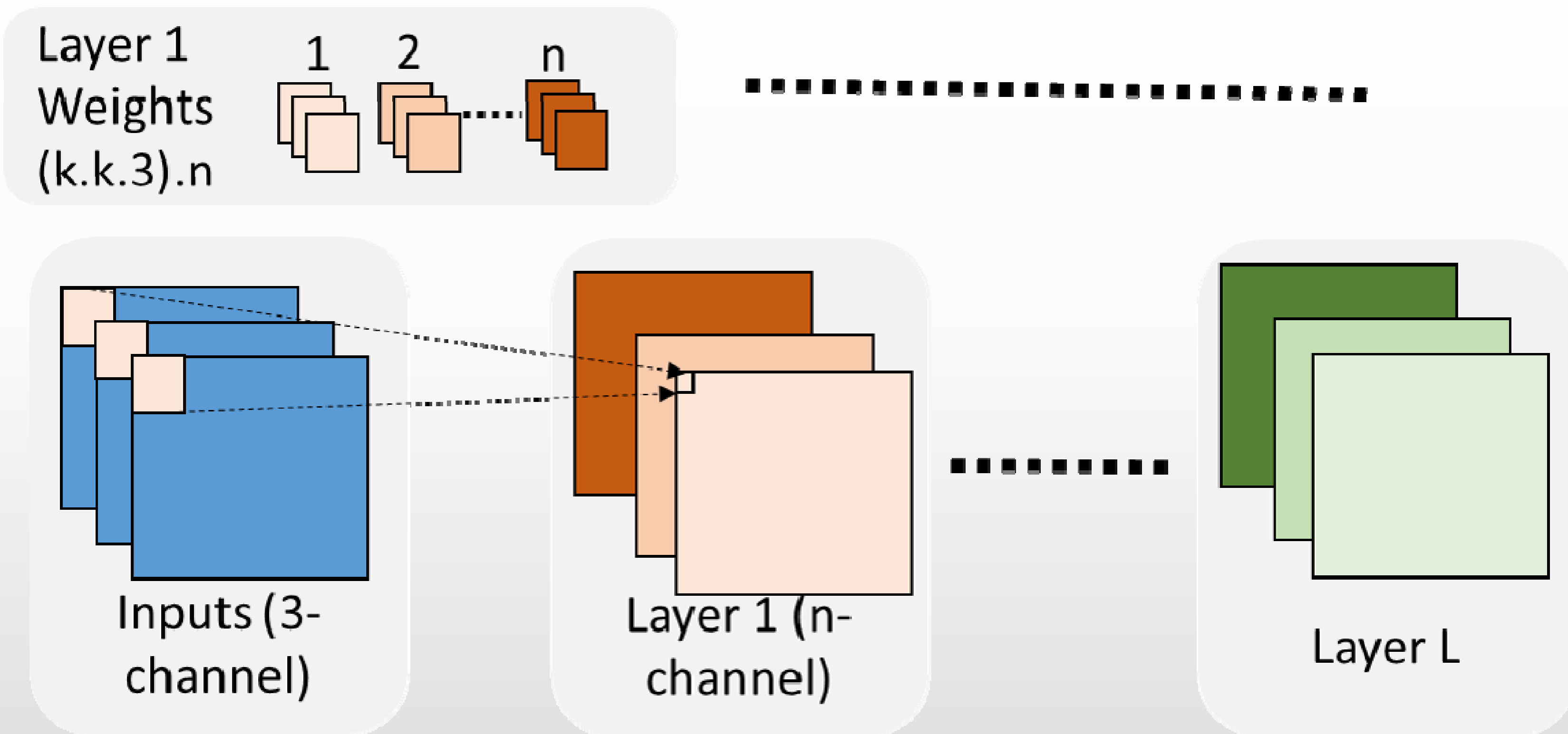
Matrix-matrix multiplications

ML Workload Characteristics (2/4)

- Highly parallel (fine-grain)
- In parallel
  - Apply numerous filters to the input
  - Apply same filter to numerous parts of input
    - CNNs
  - Apply same filters to numerous inputs
    - Batching
- Contrast to Spec or TPC, which pose parallelism-scarcity, ML poses parallelism-abundance

Highly parallel

Parallelism in CNNs



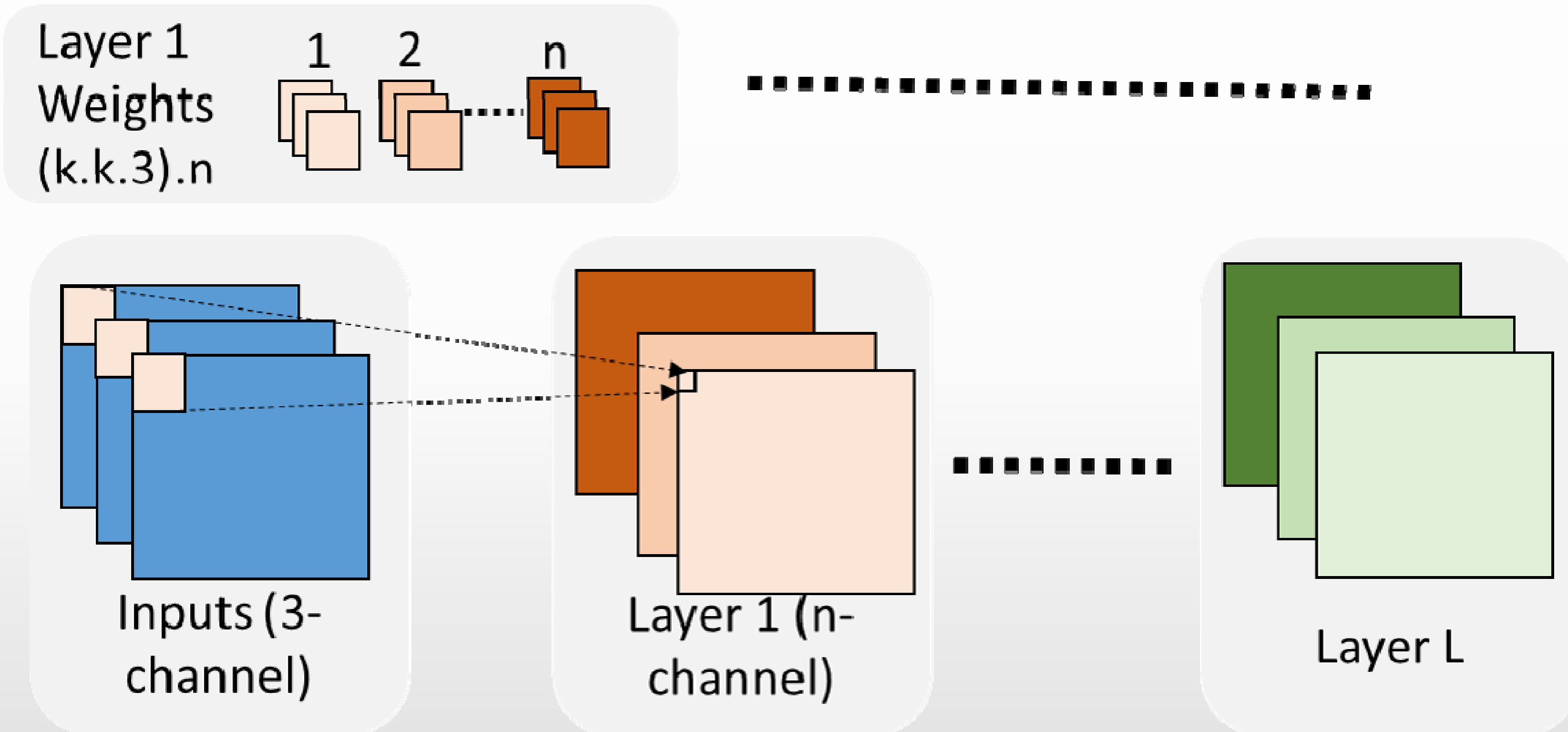
Highly parallel

ML Workload Characteristics (3/4)

- High reuse
- Apply numerous filters to the input
  - Input reuse
- Apply same filter to numerous parts of input
  - Filter reuse (CNNs)
- Apply same filters to numerous inputs
  - Filter reuse (batching)
- One layer's output is next layer's input
  - Output reuse

High reuse

Reuse in CNNs



High reuse

ML Workload Characteristics (4/4)

- Quantized arithmetic
  - Reduces compute and data
- Low-precision arithmetic for inference
  - Simple quantization for fast arithmetic
  - E.g., int8
- Higher-precision for training
  - E.g., 16-bit FP
  - Post-training quantization
  - Quantization itself is fairly involved

Low precision suffices

Spiky Neural Networks

- Based on spikes and not values
  - Closer to biological brains?
- Neuron activated only upon a spike
  - Energy-efficient
- But accuracy lower than CNNs’
  - Training hard: gradient hard to compute for spikes
- Should model be more accurate before hardware is built?

Outline

- Introduction
- Workload characteristics
- ML architectures
- Looking forward
- Conclusion

ML Architectures

A few examples

- DianNao and successors
- GPGPU, Systolic (IBM, TPU, Trillium), Brainwave FPGA, Wavecomputing Dataflow
- Reuse: Eyeriss, Layer Fusion
- Sparse: Convlutin, EIE, Cambricon, SCNN
- Spiky: True North
- Many more .....

ML architectures everywhere!

ML Architecture Features (1/4)

- Matrix multiply via multiply-accumulate (MAC) units
- Fine-grained, regular compute and memory access
  - SIMT in GPGPUs
  - Systolic as in IBM, TPU
  - SIMD in Microsoft Brainwave
- Simple logic, low control/instruction overhead
- Energy- and area-efficient

Exploiting fine-grain regularity

ML Architecture Features (2/4)

Numerous MAC units to match high compute

- CNNs compute-bound → more MACs → higher speed

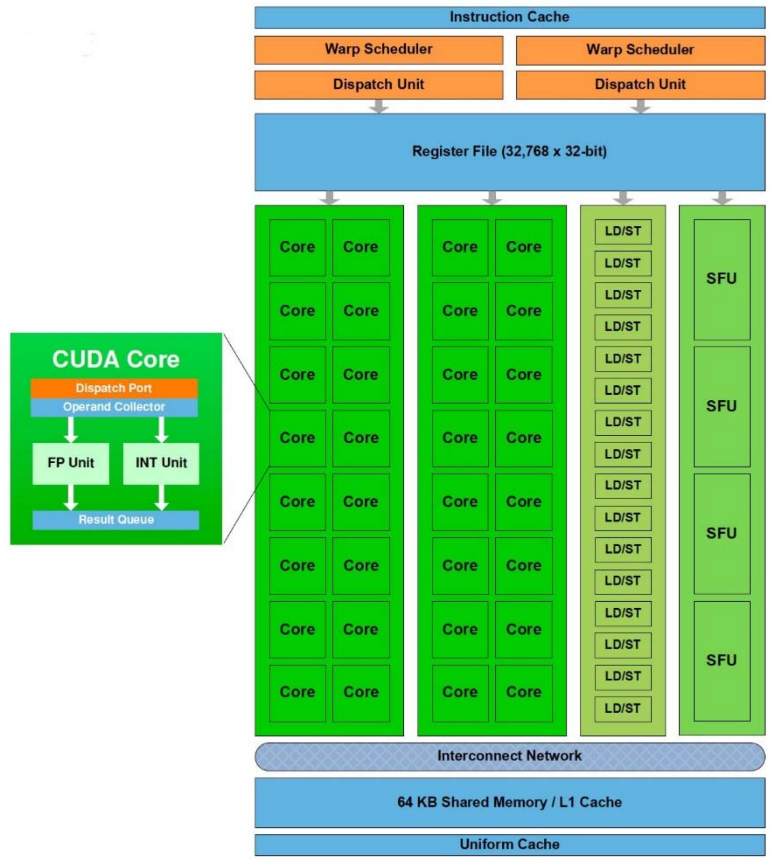
- 128-lane SMs in GPGPUs
- 128x128 = 16K MAC units in a TPU cluster
- 2048-wide SIMD in FPGA

- Challenge is managing immense parallelism

Exploiting fine-grain parallelism

GPGPU

[Nvidia]

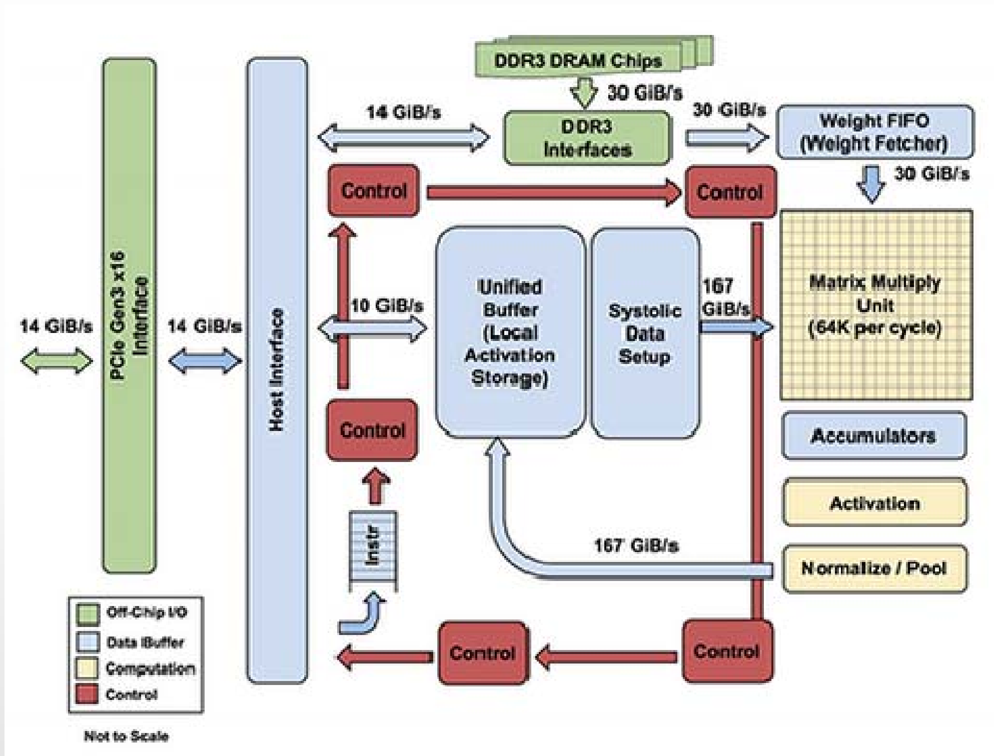


Fermi Streaming Multiprocessor (SM)

Exploiting fine-grain parallelism

TPU

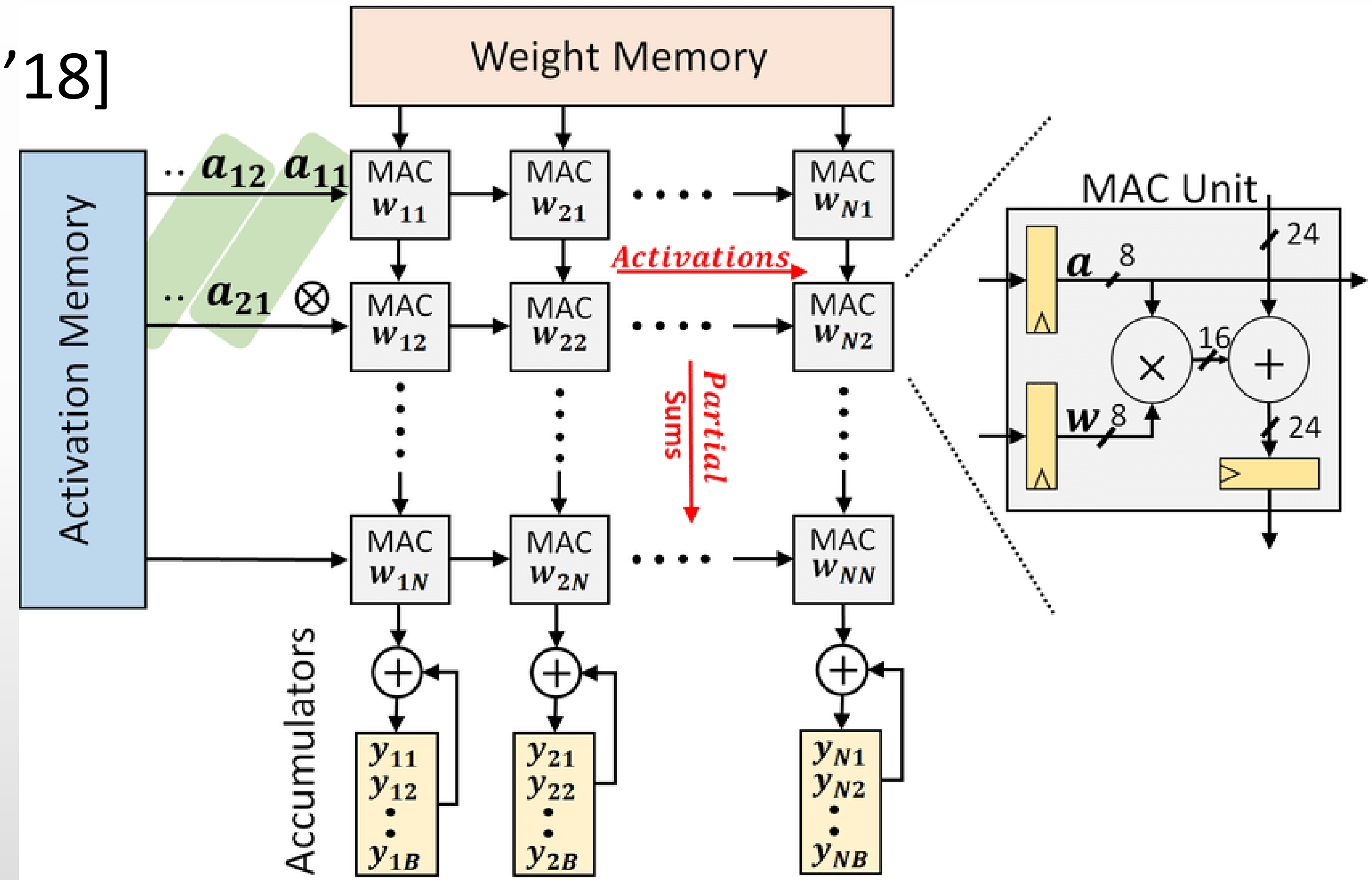
- [Sato'17]



Exploiting fine-grain parallelism

Systolic Matrix Multiplier

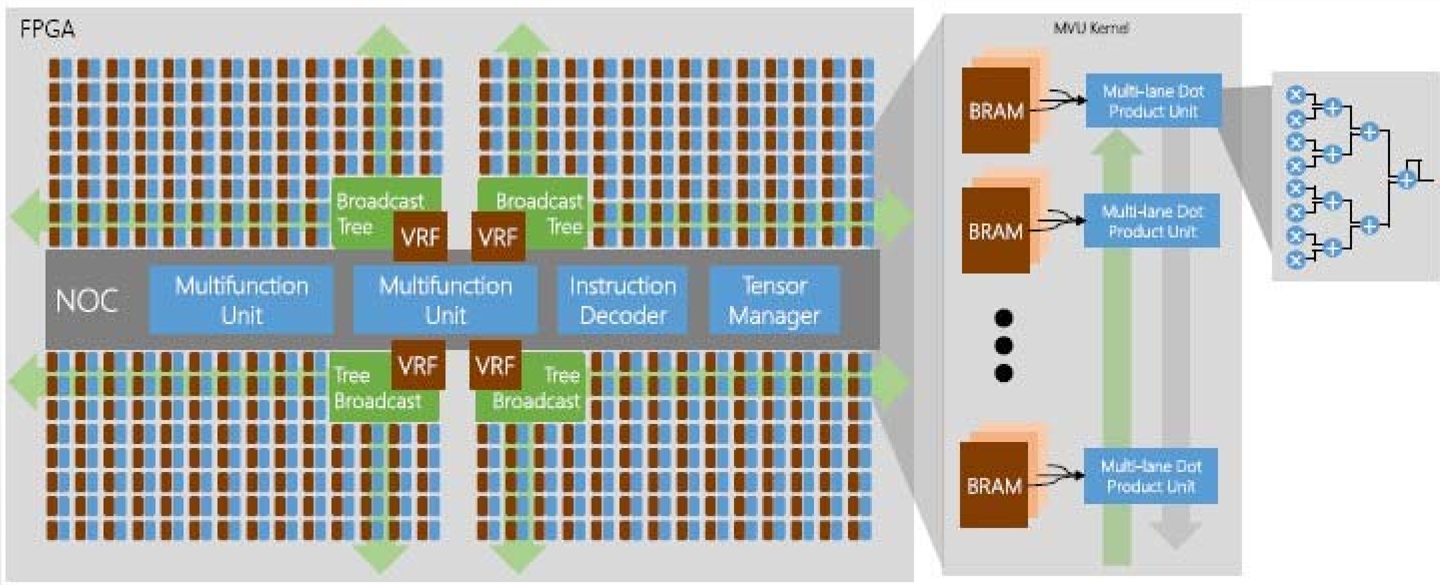
[Zhang'18]



Exploiting fine-grain parallelism

Brainwave FPGA

[BrainWave Hotchips '17]



Exploiting fine-grain parallelism

Exploiting Regular Memory Access

- Regular memory → amenable to simple prefetch
- Prefetch next matrix row under current row computation
    - Small prefetch buffers (eg 8 KB)
- GPGPUs' multithreading unnecessary
    - Originally for unpredictable texture cache misses
    - Huge register files (256 KB per SM = 8 MB per die)
    - Area and energy overheads
    - Yet, fundamentally enabled ML's recent success

Regular memory access → Efficient

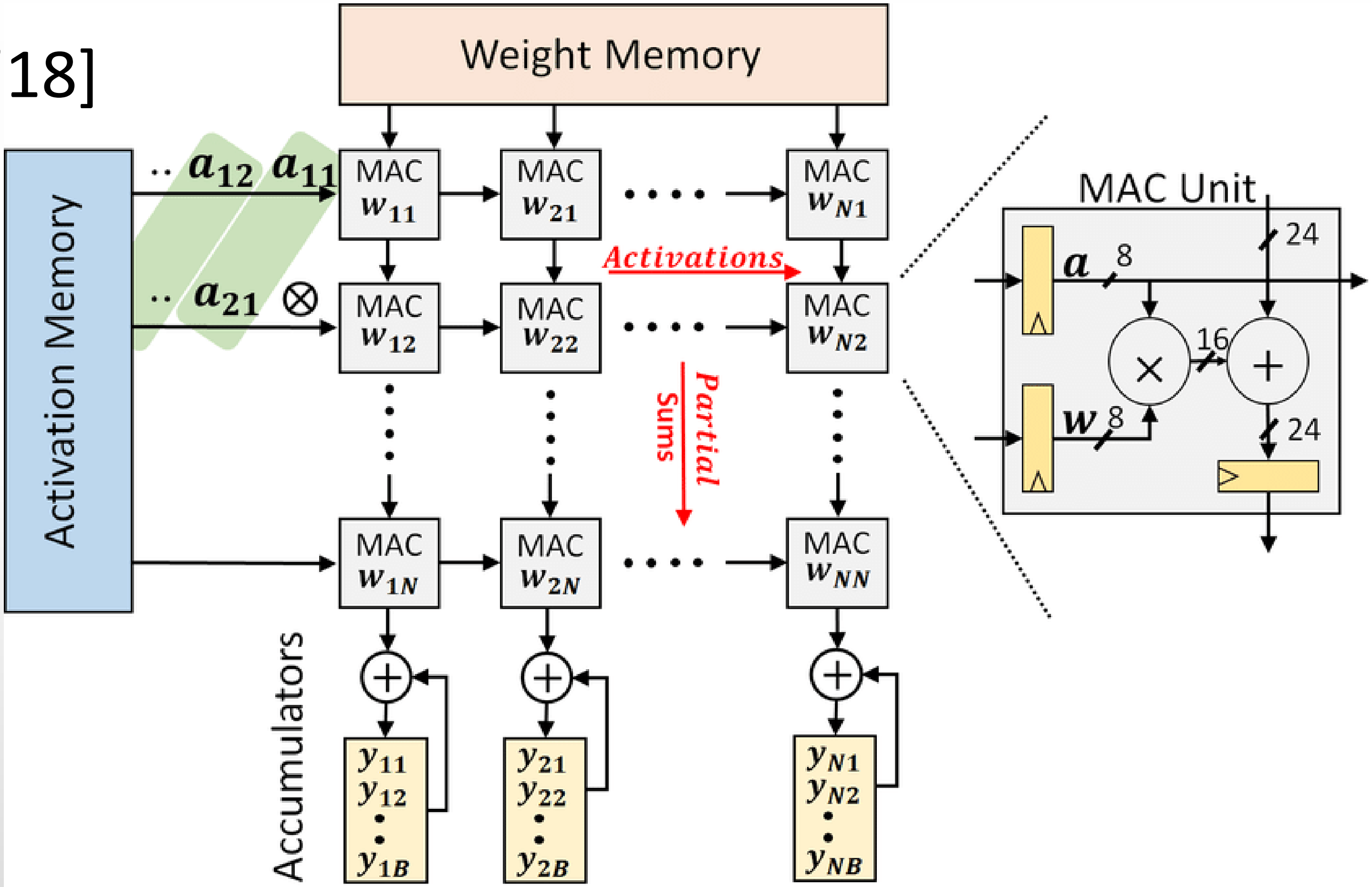
ML Architecture Features (3/4)

- Reuse of filters and inputs
  - Reduce memory bandwidth demand
- Hold filters near MAC units
  - Reuse filters across inputs
- Broadcast input to MAC units
  - Reuse input across filters
- More MAC units → more reuse

Exploiting reuse

Reuse in Systolic Array

[Zhang'18]



Exploiting reuse

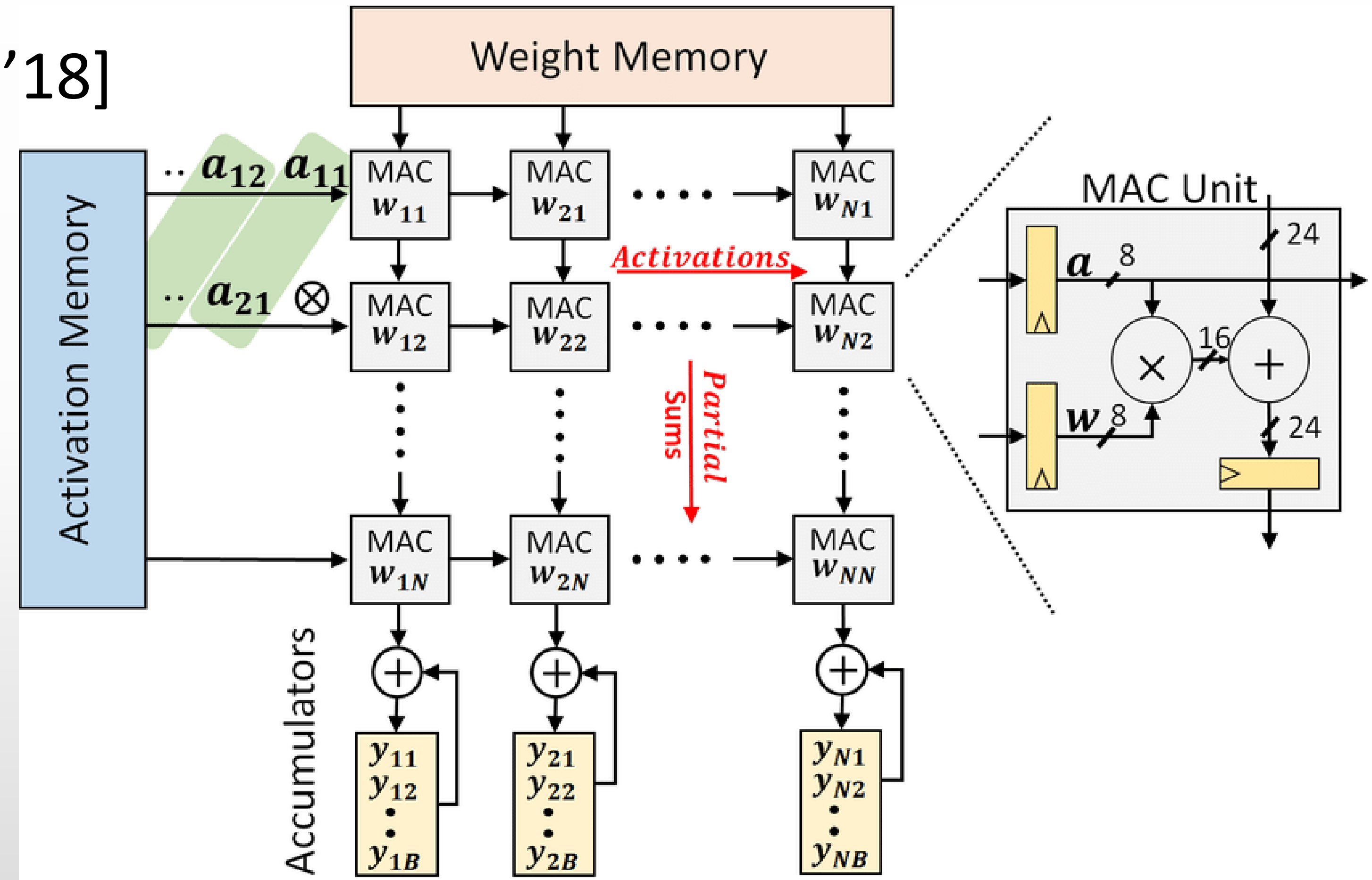
Concurrent Reuse

- Reuse spread over time → buffering
- Here, concurrent reuse → little buffering
- Systolic pipelines input-filter vector-vector multiply
- V-V multiply is a recurrence → no parallelism
  - Converting accumulation into reduction unnecessary given numerous concurrent vector-vector multiplies
- But pipelining V-V multiply reduces buffering/MAC
  - Eg GPGPU holds 128-B filter/MAC
  - Systolic pipelines 128-B filter across 128 MACs (1 B/MAC)

Reduced buffering

Buffering in Systolic Array

[Zhang'18]



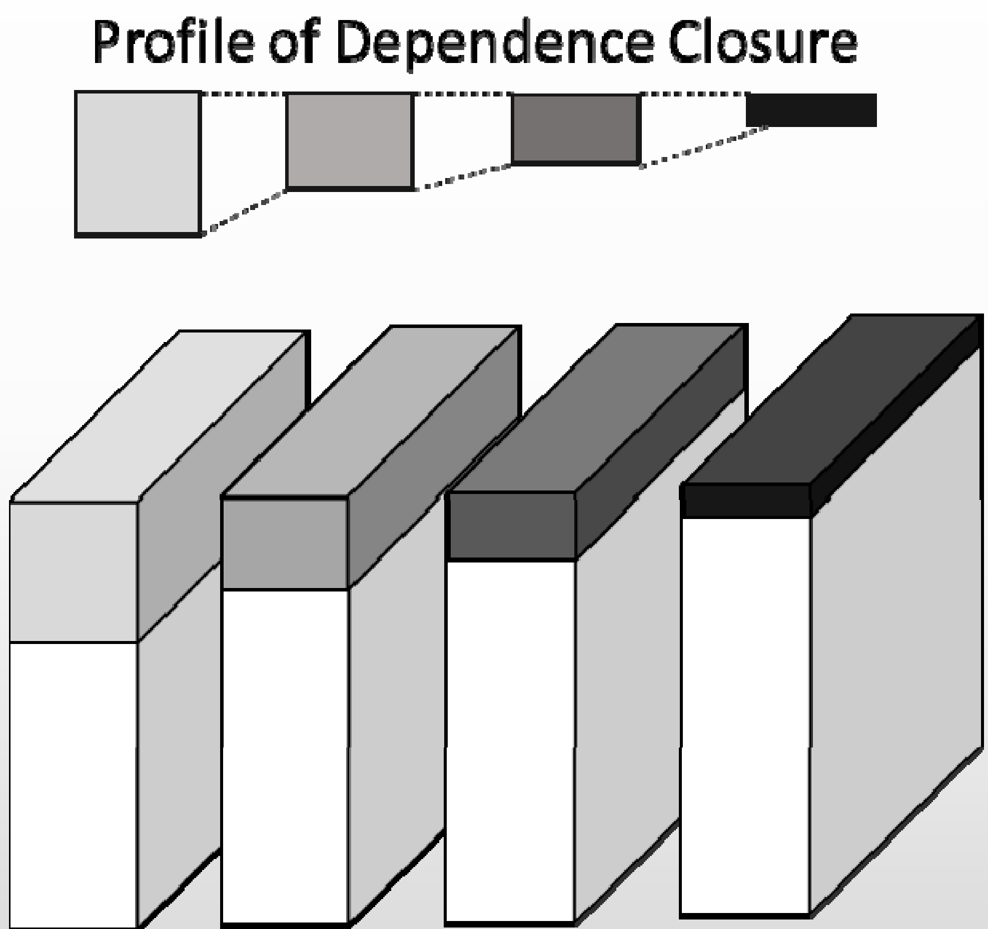
Reduced buffering

Output Reuse

- Reuse of output as next layer input is more involved
  - For convolutional layers
- Current systems write each layer output to memory
  - Unless fits in on-chip cache
- For each output cell, enough to hold dependence parents
  - No need to hold all of previous output
- Going across multiple network layers, enough to hold dependence ancestors [Layer Fusion]

Exploiting output reuse can be efficient

Output Reuse in CNNs



Exploiting output reuse

ML architecture features (4/4)

- Hardware support for int8, 16-bit FP
  - Order of magnitude lower area, energy than 32-bit
  - Simple arithmetic
  - 4-bit, 12-bit?

Low-precision arithmetic

Outline

- Introduction
- Workload characteristics
- ML architectures
- Looking forward
- Conclusion

Looking forward

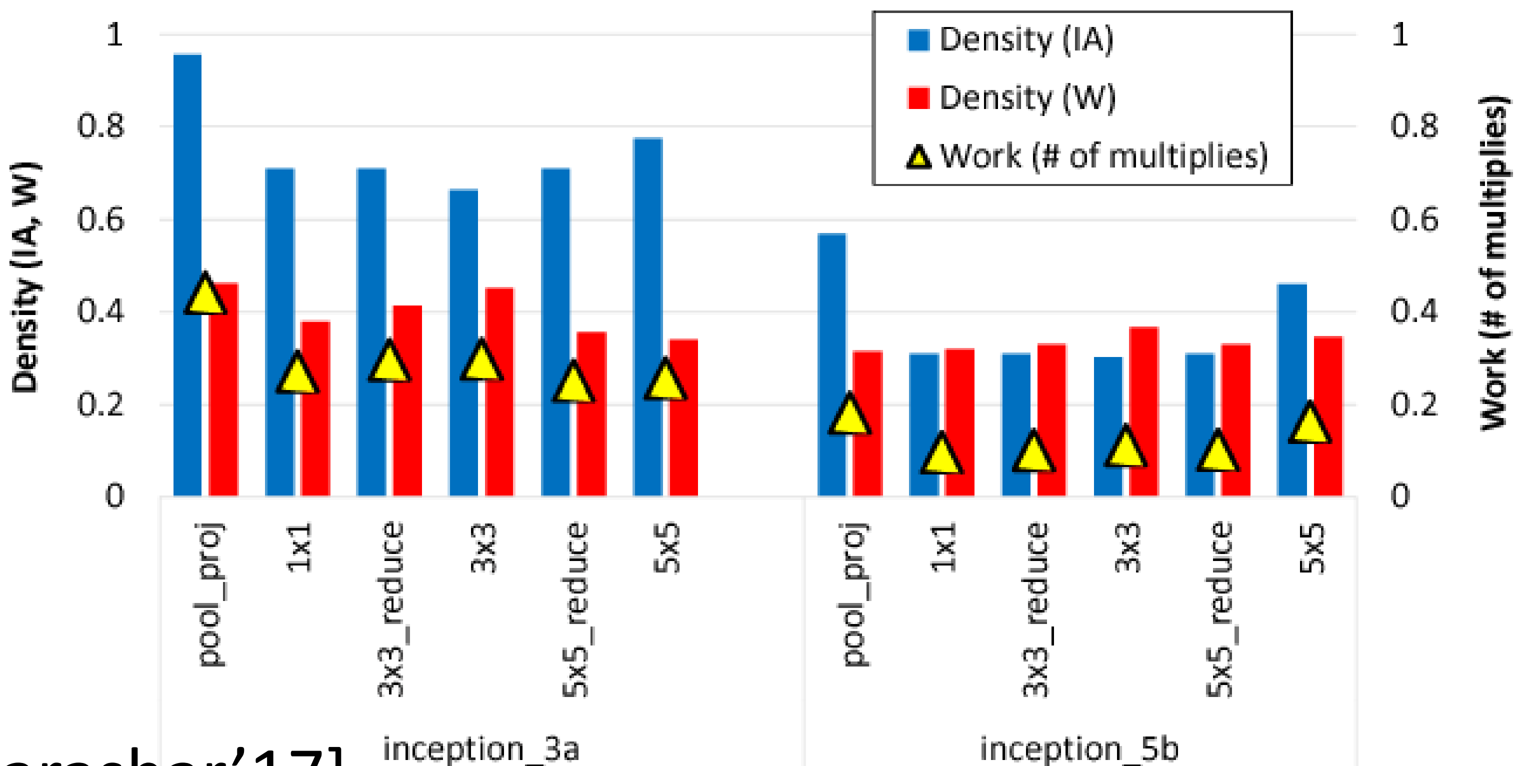
1. ML workload characteristics
    - Sparsity
2. New technologies
    - Processing-in/near-memory
3. Future ML models

ML Models Are Sparse (1/3)

- Many zeros in both filters and feature maps
  - Both convolutional and fully-connected layers
- Naturally sparse [Cnvltin]
- Recent work enhances sparsity through transformations [NIPS ’15, ICLR’16]
  - Pruning by eliminating unimportant connections
  - Maintains accuracy through retraining
  - 25x less compute and 5x less data

Significant sparsity

ML Models Are Sparse



[Parashar'17]

(b) GoogLeNet

Significant sparsity

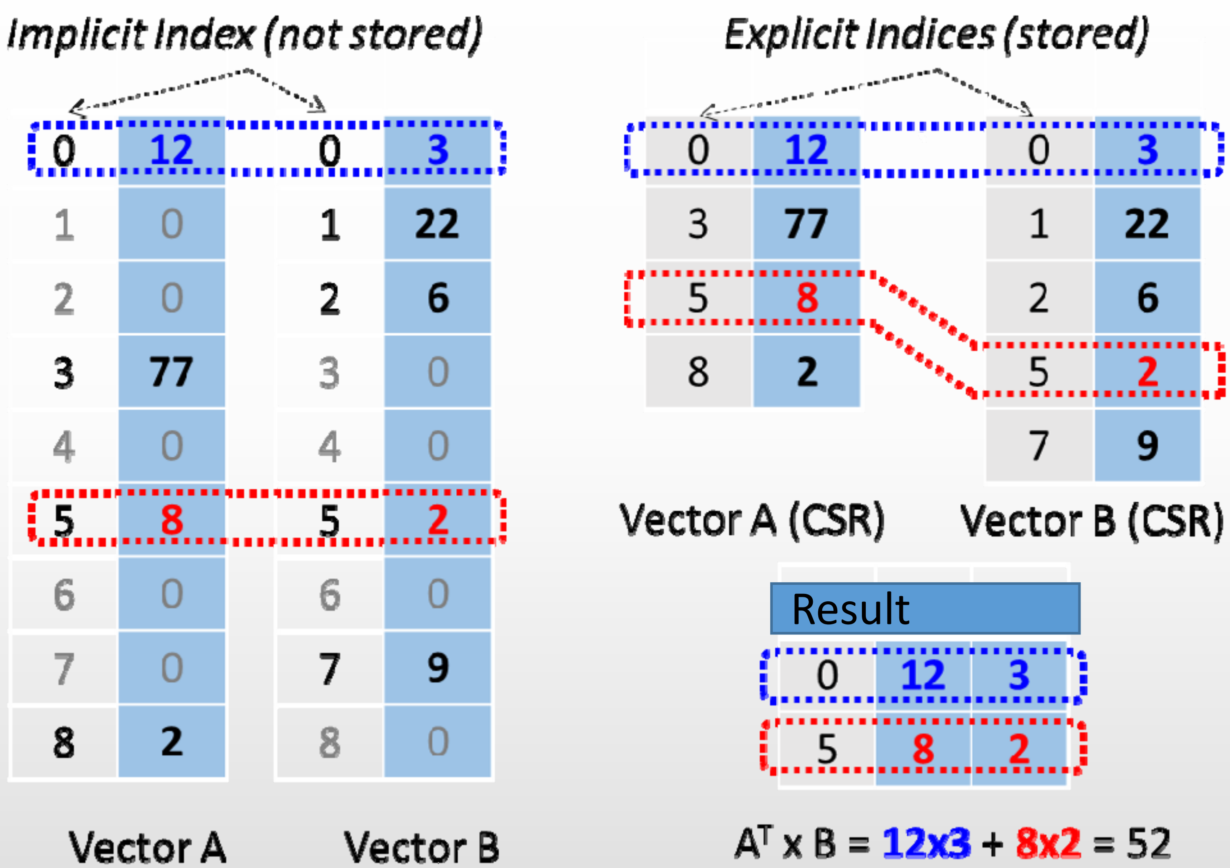
Sparse ML Architectures

- One-sided and two-sided sparsity
  - One-sided: exploit zeros only in filters or feature maps [Cnvlutin, EIE, Cambricon]
  - Two-sided: exploit zeros in both [SCNN]
- Sparsity → irregular computation

  Even for same input, different filters → divergent compute
  - SIMD, vector, SIMT, systolic inefficient
- Memory accesses still regular
  - Non-zero values packed sequentially

Irregular compute

Sparse Matrix Multiply



Sparse ML Architectures

- Pointer/offset representation for non-zeros
  - Compressed Sparse Row in High Performance Computing
- Sparse matrix multiply in hardware
  - Parallelism, reuse remain (modulo sparsity)
  - Compute irregular, memory regular
  - One-sided [Cnvlutin, EIE, Cambricon]
  - Two-sided [SCNN] – unusual dataflow

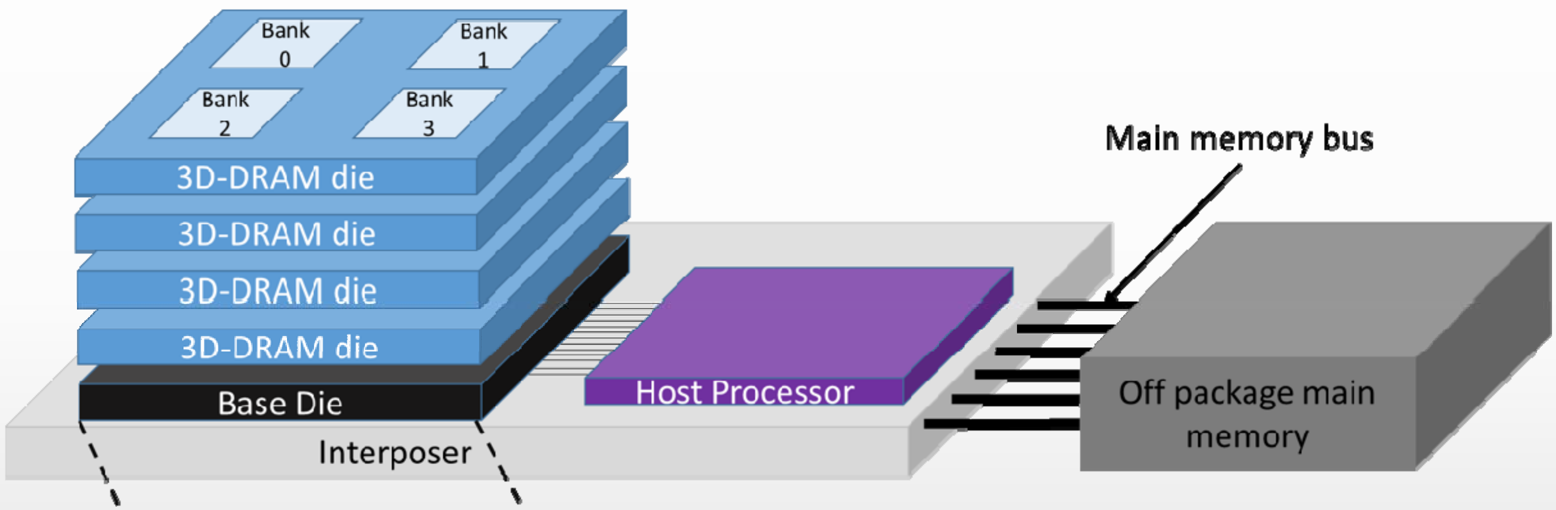
Sparsity key for efficiency

New Technologies (2/3)

- Processing in/near memory
  - DRAM, MRAM, STTRAM, ReRAM, …..
  - Huge memory bandwidth
  - Low energy
- Many ML workloads (eg fully connected layers)
  - Need high memory bandwidth
  - Simple compute
  - Fine-grain parallel
  - Streaming with little reuse
  - Memory-bound

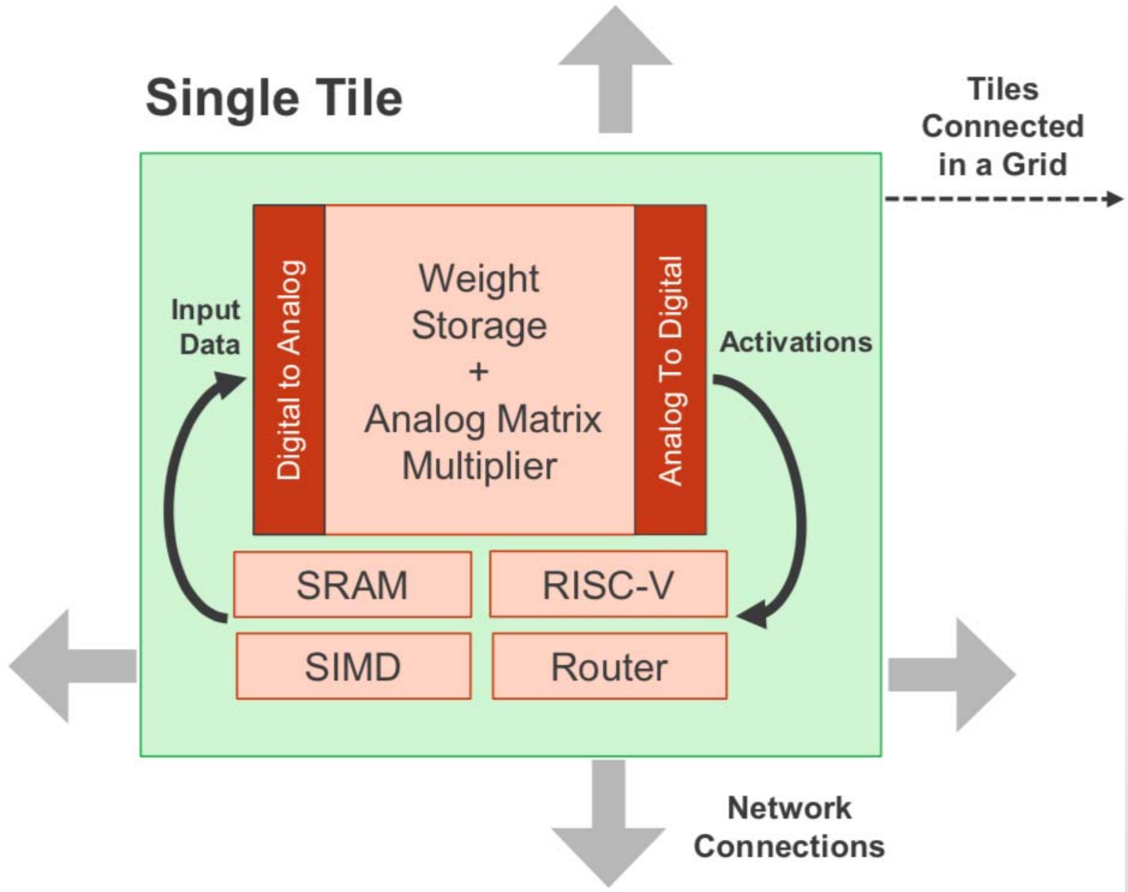
PNM/PIM – ML match made in heaven?

Processing Near Memory



Processing in Memory

- [Mythic Hotchips 30]



Processing in Memory (PIM)

- PIM is not new (70s, 80s, 90s), but three problems
  - CPU-memory process different
    - Die-stacking (PNM) avoids this
    - True PIM → slower logic
  - For 2-input, 1-output operations, compute can be near only one operand
    - Does not work well if > 1 operand large
    - Fundamental
  - Lack of good-fit applications (so far)
- If applications not different, old difficulties will remain

High bandwidth but constraints

PIM implications for ML

- Slow, less compute
  - Process and area constraints
- Limited buffering
  - Area constraints
- Limited connectivity
  - Area/metal layer constraints

- May fit fully-connected ML layers

Future ML Models (3/3)

- ML progressing at breakneck speeds
- Newer, more demanding models
  - Eg Reinforcement learning (RL)
    - Model continually updated and used
  - Computational imaging per-pixel prediction
    - Denoising (Dn) CNNs, Inception Recurrent (IR) CNN
  - Many others
- Multi-modal models
  - Video, speech and language together

Sky is the limit!

A huge thanks to


Mithuna Thottethodi

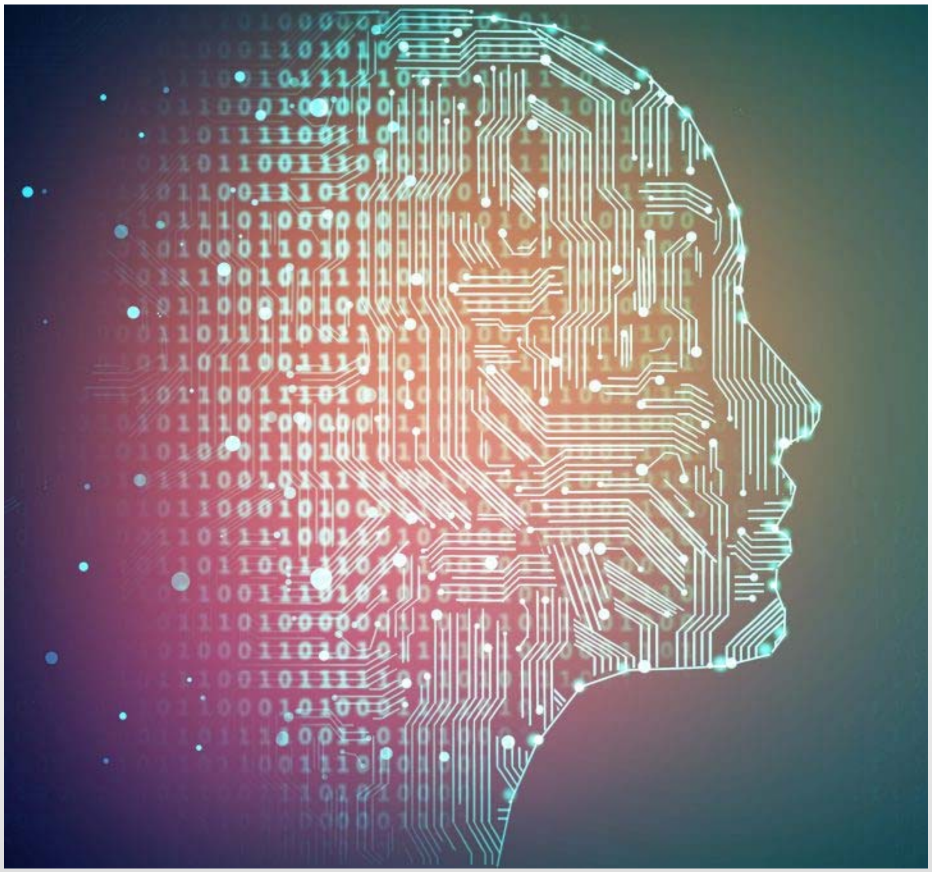

Ashish Gondimalla

- SK Hynix

Conclusion

- Exciting progress in ML
- Huge opportunity for architects
  - SIMD, SIMT, Systolic, Sparse, ….
- Exploit ML workload characteristics
  - Parallelism, regularity, reuse
- New technologies may be a good match for ML
  - Processing in/near memory
- We have barely scratched the surface

We can't get enough of this!



Efficient Machine Learning Architectures

T. N. Vijaykumar

PURDUE UNIVERSITY

EMC^2 Workshop, HPCA, Feb 17 2019