ETH zürich

# Diannao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning

Presenter : Jiawei Liao

# What is Diannao

- **Diannao as a word** – Electric Brain (from Mandarin)
- **Diannao** – Accelerator for Neural Networks
- **Features:** High throughput ; Energy efficiency; Small area

# Sarcasm from TPU

## In-Datacenter Performance Analysis of a Tensor Processing Unit™

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon

*Google, Inc., Mountain View, CA USA*

Email: {jouppi, cliffy, nishantpatil, davidpatterson} @google.com

*To appear at the 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, June 26, 2017.*

[Che16a]. All use 16-bit integer operations and all designs dove down to layout, but no chips were fabricated. The original

No tape-out for Diannao family !!?

# The successor of Diannao in Huawei SoC

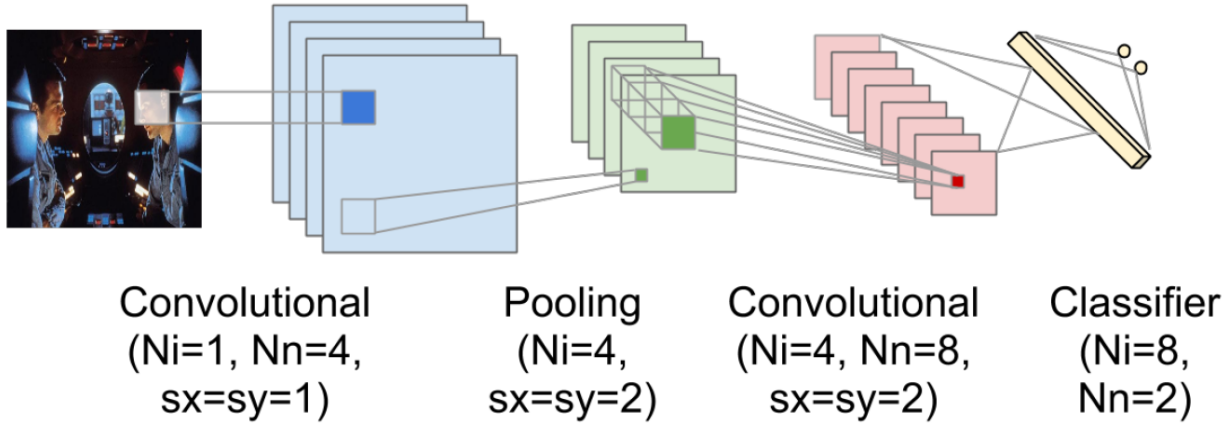

# What algorithm to accelerate

- **Two Requirements:**
  - **Efficiency**

    Throughput, energy, area
  - **Flexibility**

    Can be applied to more scenarios

- **CNN & DNN**



# What are the challenges

- **Trend towards large neural networks.**
  - **VGG-16**: over 100 million parameters.
  - also example from 1 billion to 10 billion parameters.

- **Challenges:**
  - **Scalability:**

    How to do the computation for large NN?
  - **Memory:**

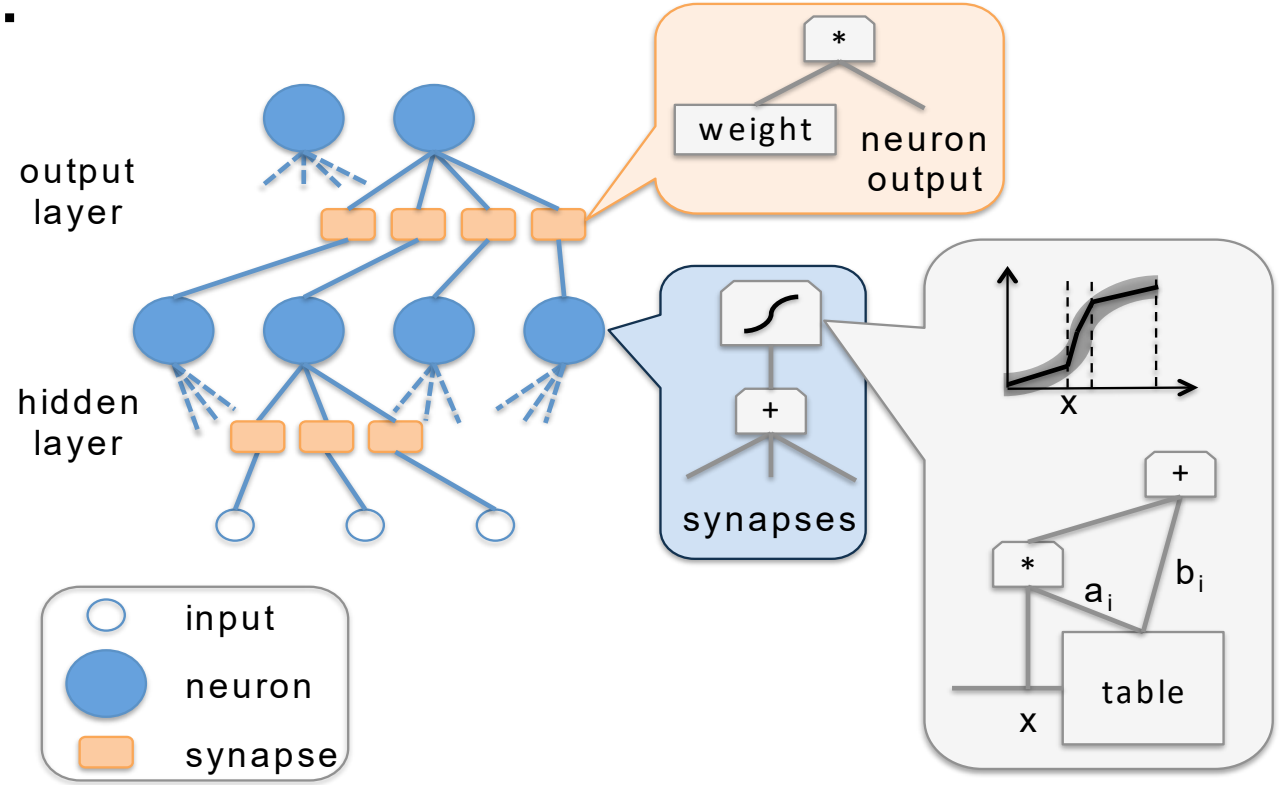
    How to reduce memory access?

# Why do we say scalability is a problem?

**One traditional accelerator for NN:**
All the neurons and synapses (parameters) laid out in hardware.

**Neuron**
– logic circuits
**Synapses(parameters)**
– latches or RAMs

**Memory**
Only store input and output of the whole NN.

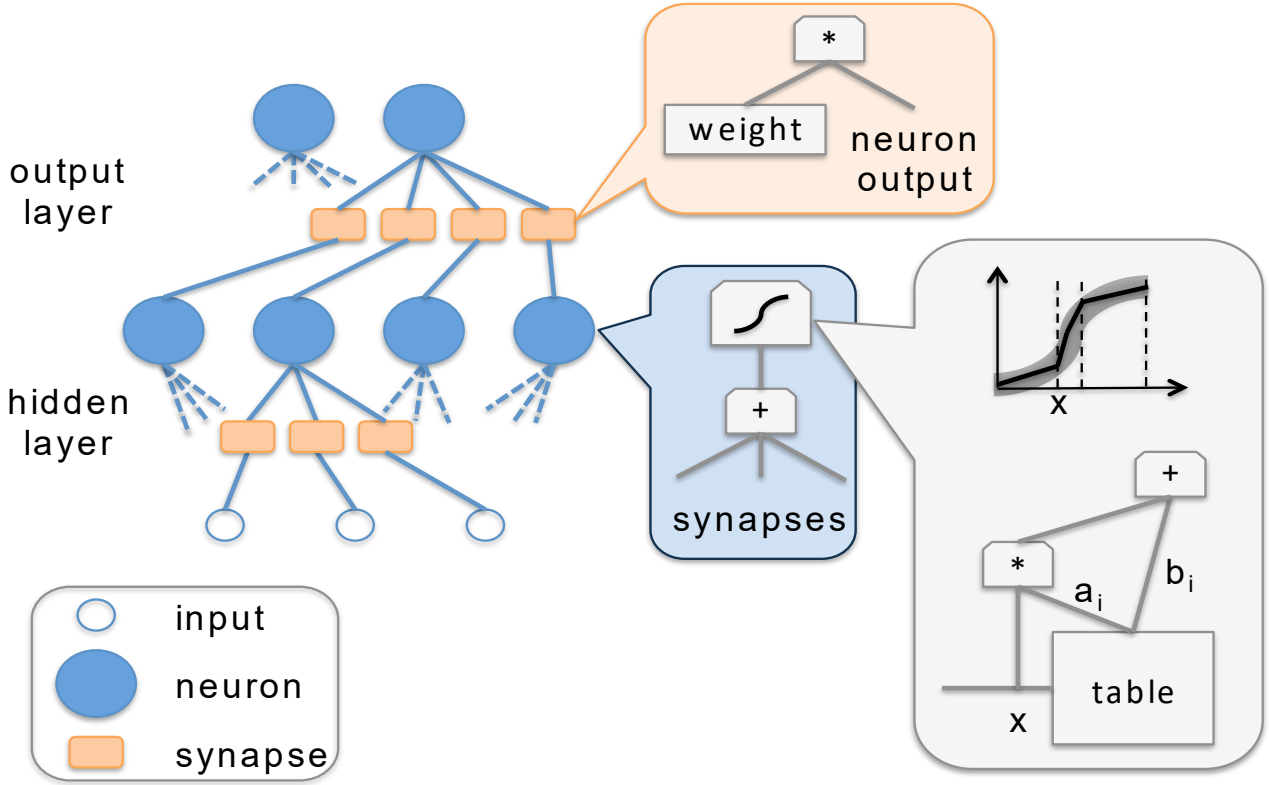


# Advantages of traditional design

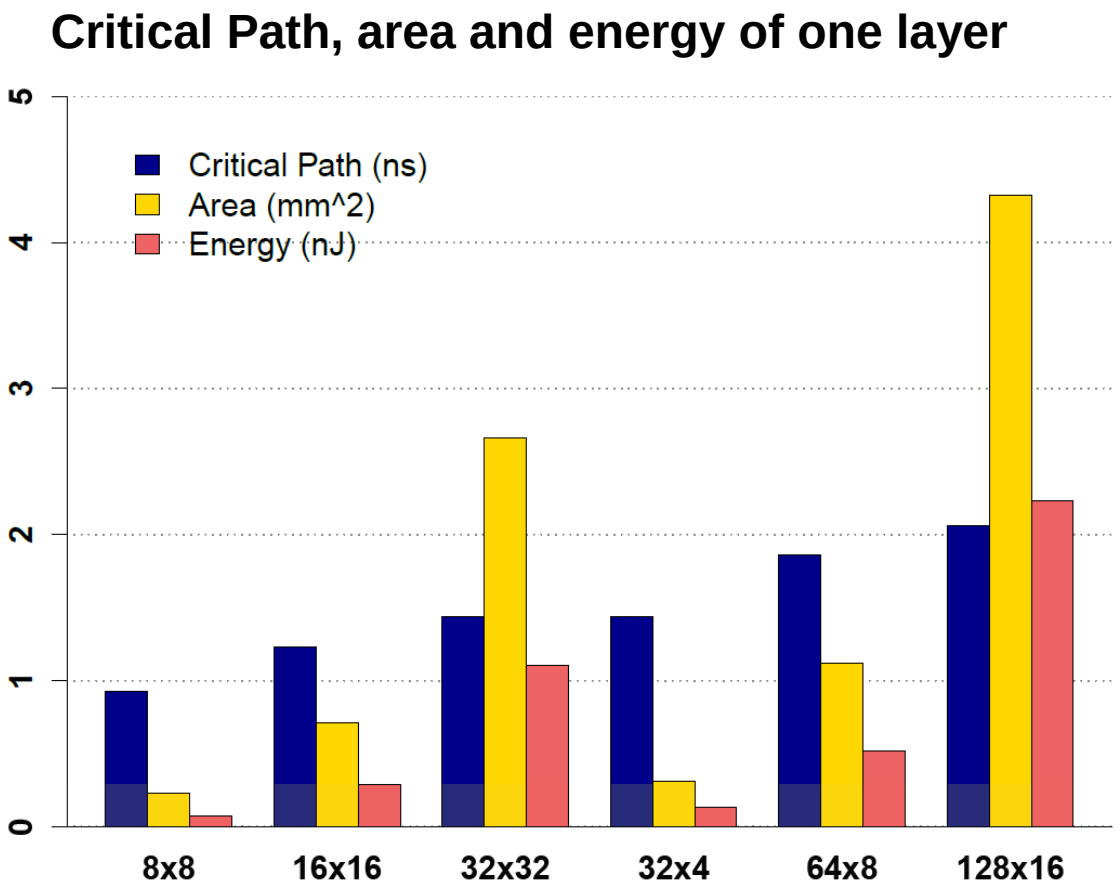
**A high-throughput and energy efficient design:**

**1. Less memory access**
Intermediate values don't need to be stored in the memory.

**2. Faster computation**
Neurons fully mapped to hardware



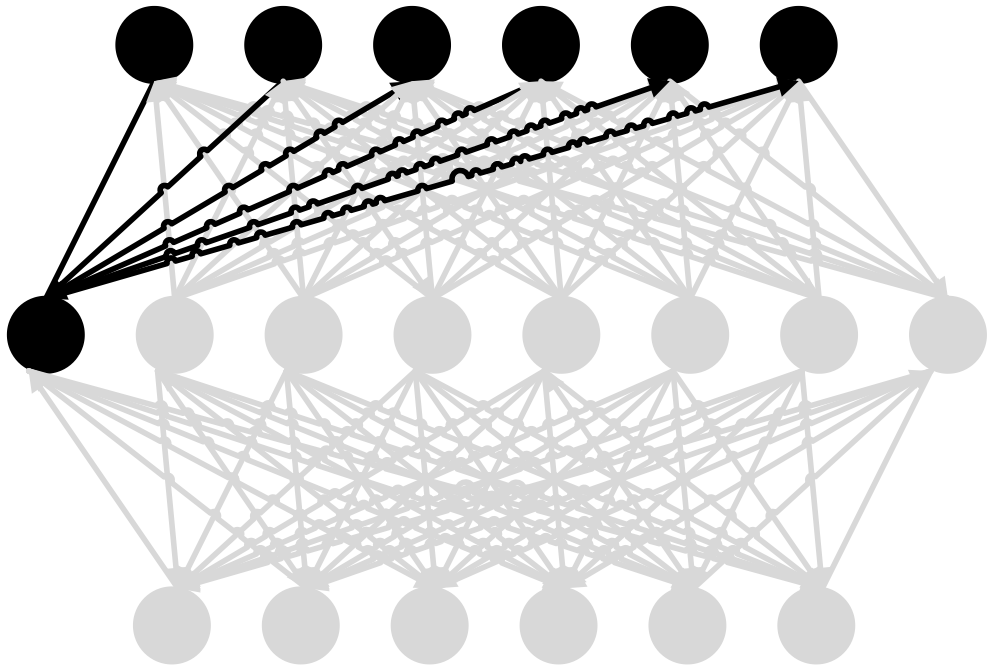
# Scalability of full-hardware implementation




With thousands of neurons, area can be **hundreds or thousands $mm^2$ per layer**

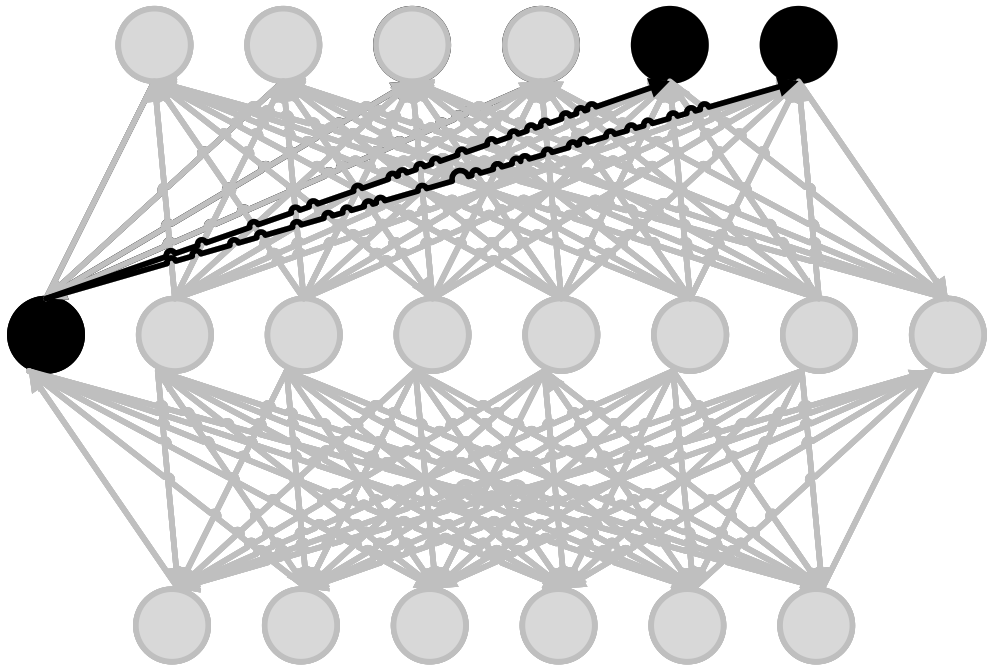
Not realistic for large scale NN

# Reuse hardware within each layer



**In full-hardware implementation:**
All neurons calculated in parallel

# Reuse hardware within each layer



- Share the limited resources
- Step by step execution to calculate partial result

# Decomposition for different layers

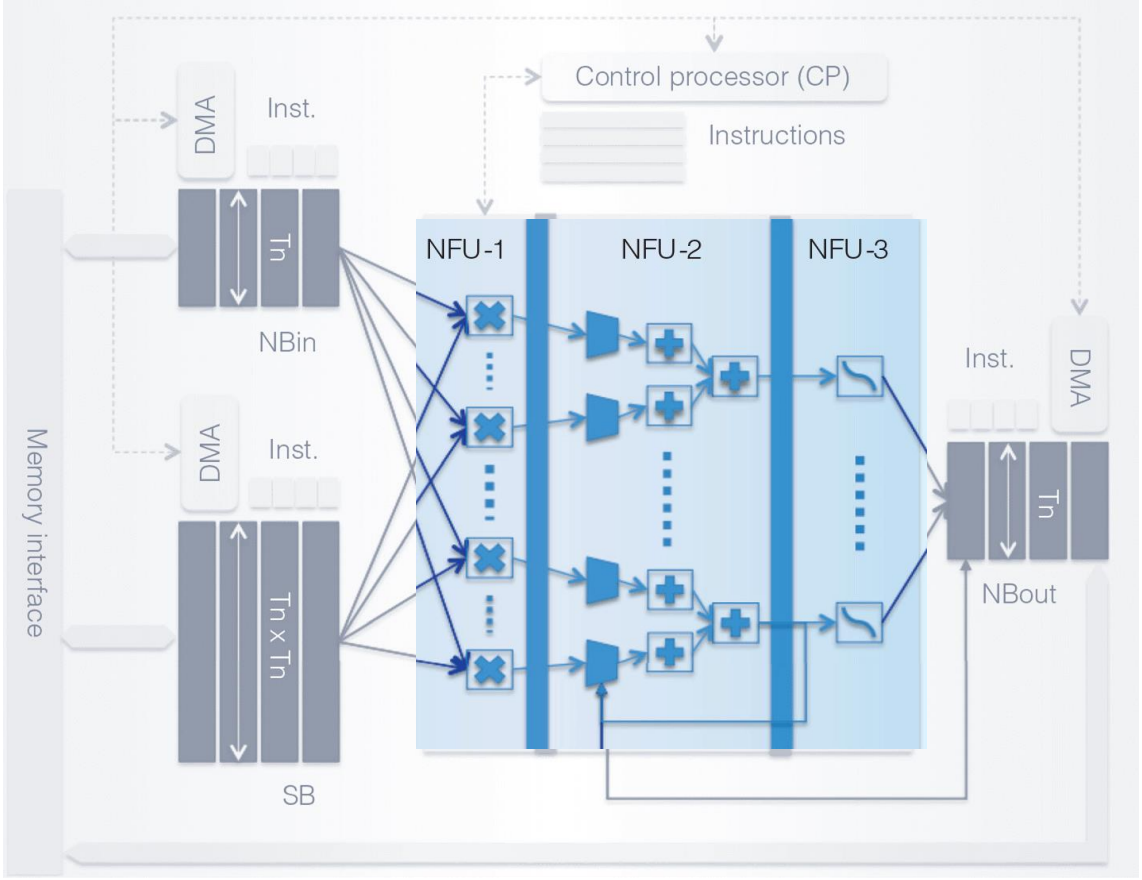
## Three main layers in CNN:

- Convolution
- Pooling
- Fully connected

Executing different layers in the same architecture

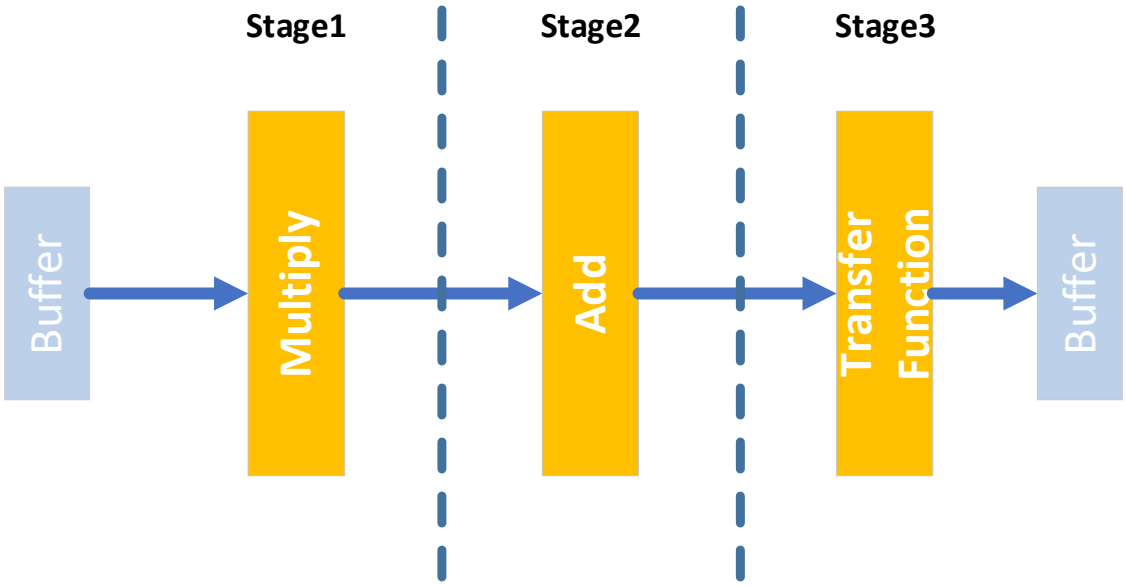
## NFU ( Neural Functional Unit)

- Three-stage pipeline
  - Stage1: Multiplication
  - Stage2: Add, Max operation
  - Stage3: Transfer function ( sigmoid or other non-liner function)

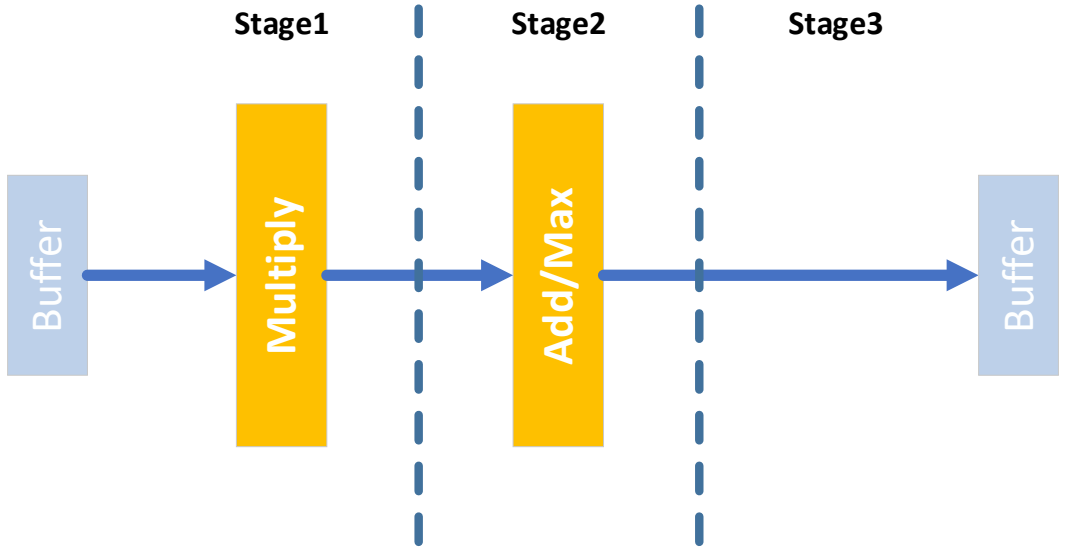


# Decomposition for different layer

## Fully connected / Convolution




## Polling Layer

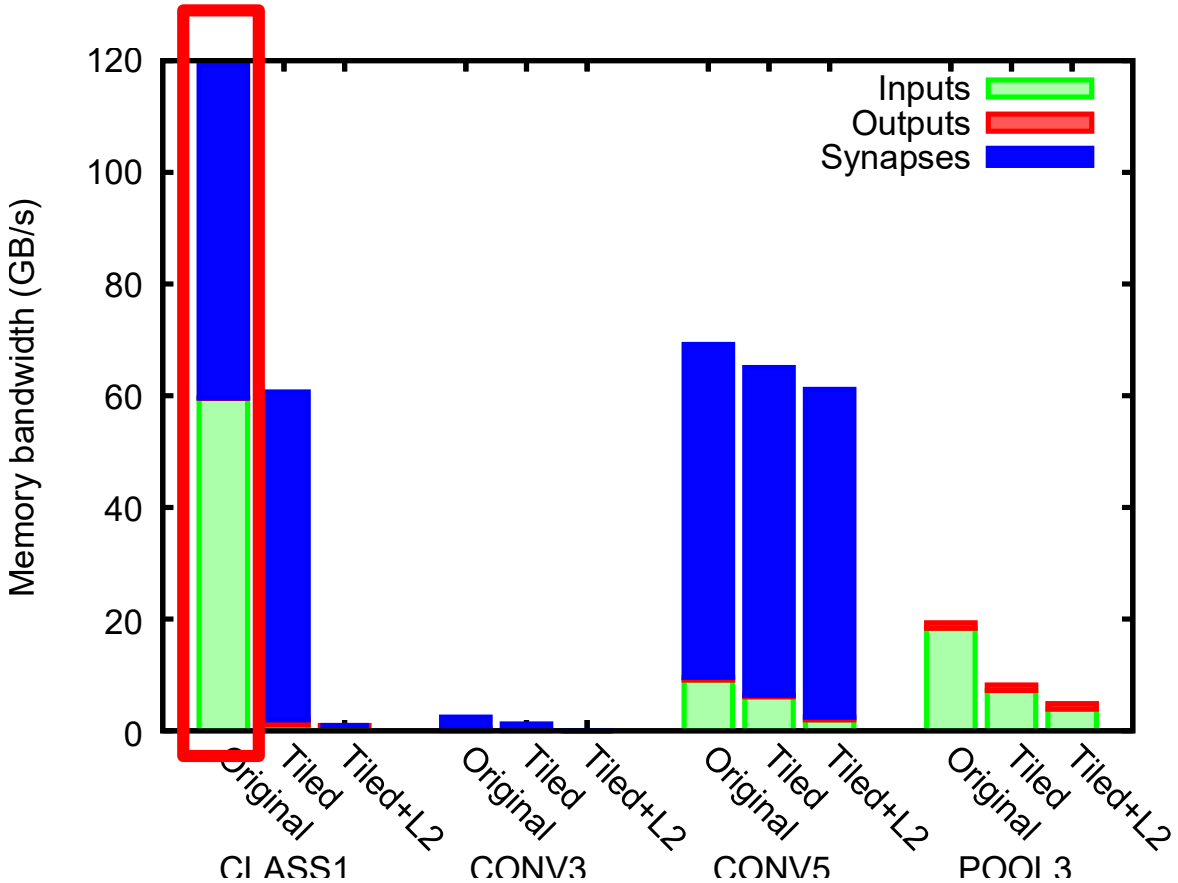


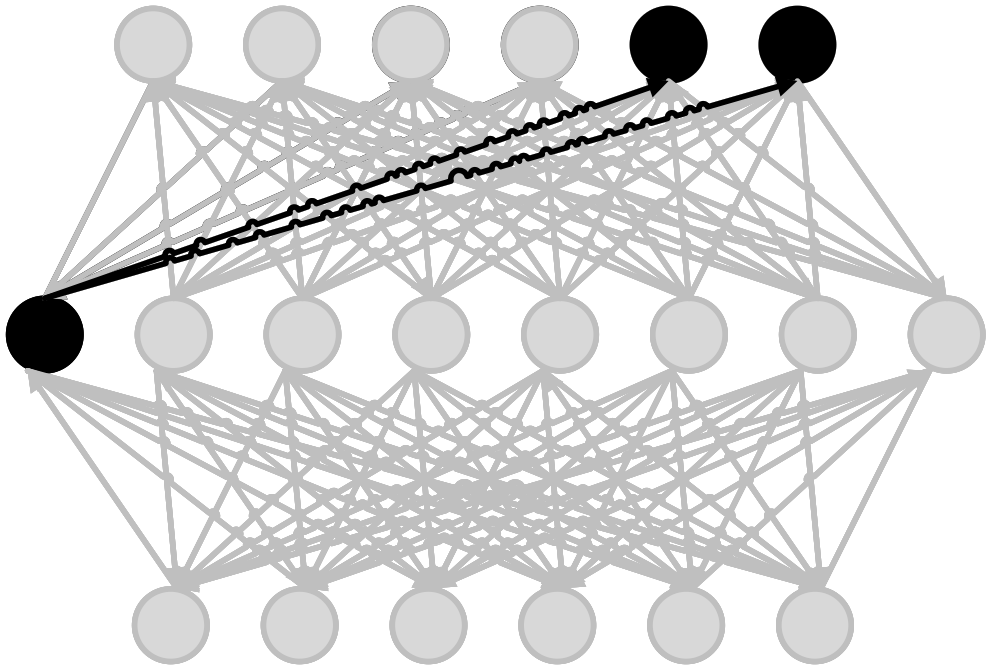
# High memory bandwidth requirement for NN

- For their NFU, processing 16 input neurons and 16 output neurons per cycle at 0.98 GHz. The peak bandwidth is 467.30 GB/s.

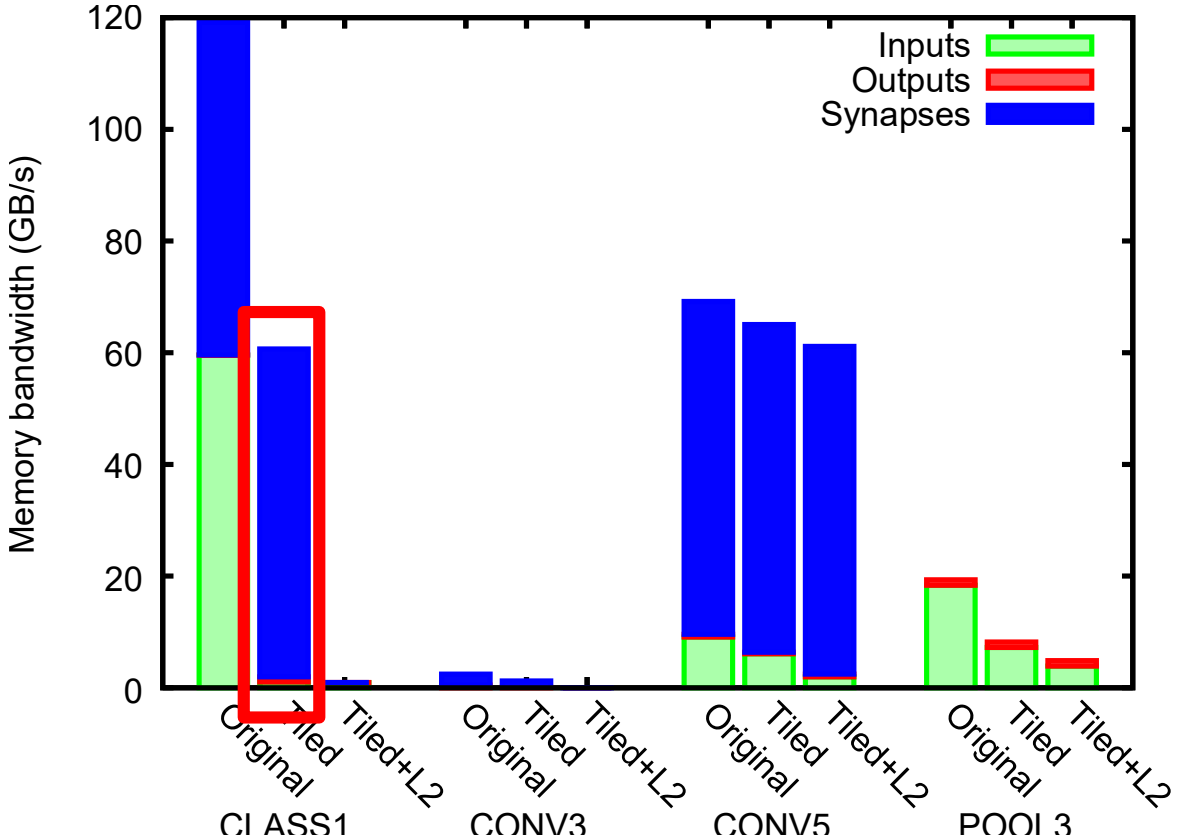
- To understand the source of the high memory traffic of NN and to conduct optimization, they analyzed the memory bandwidth on a simulator for processor-based implementation.

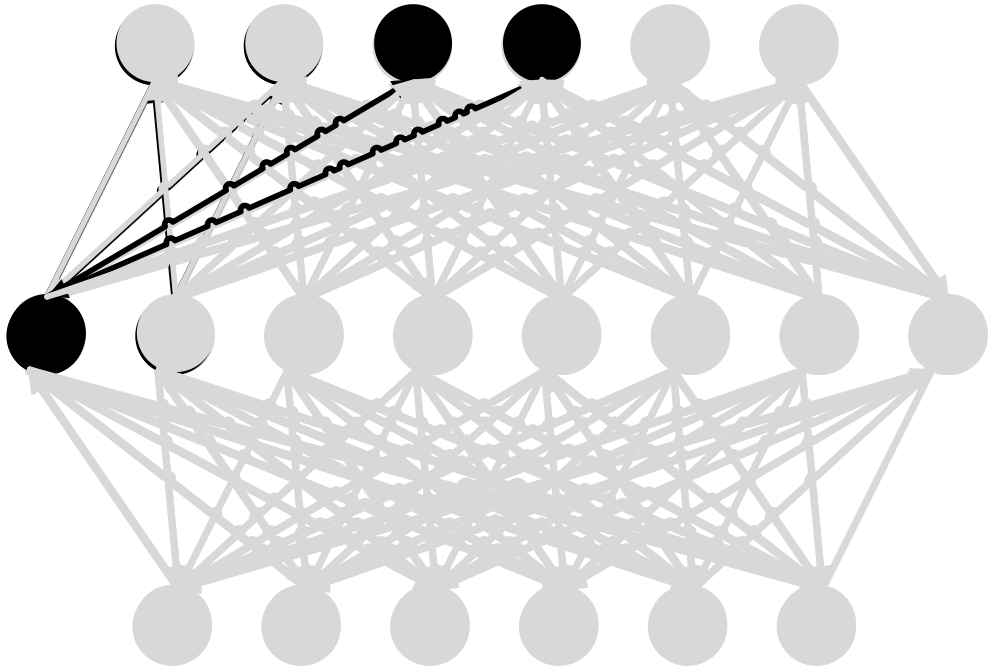
# Memory traffic analyzation result



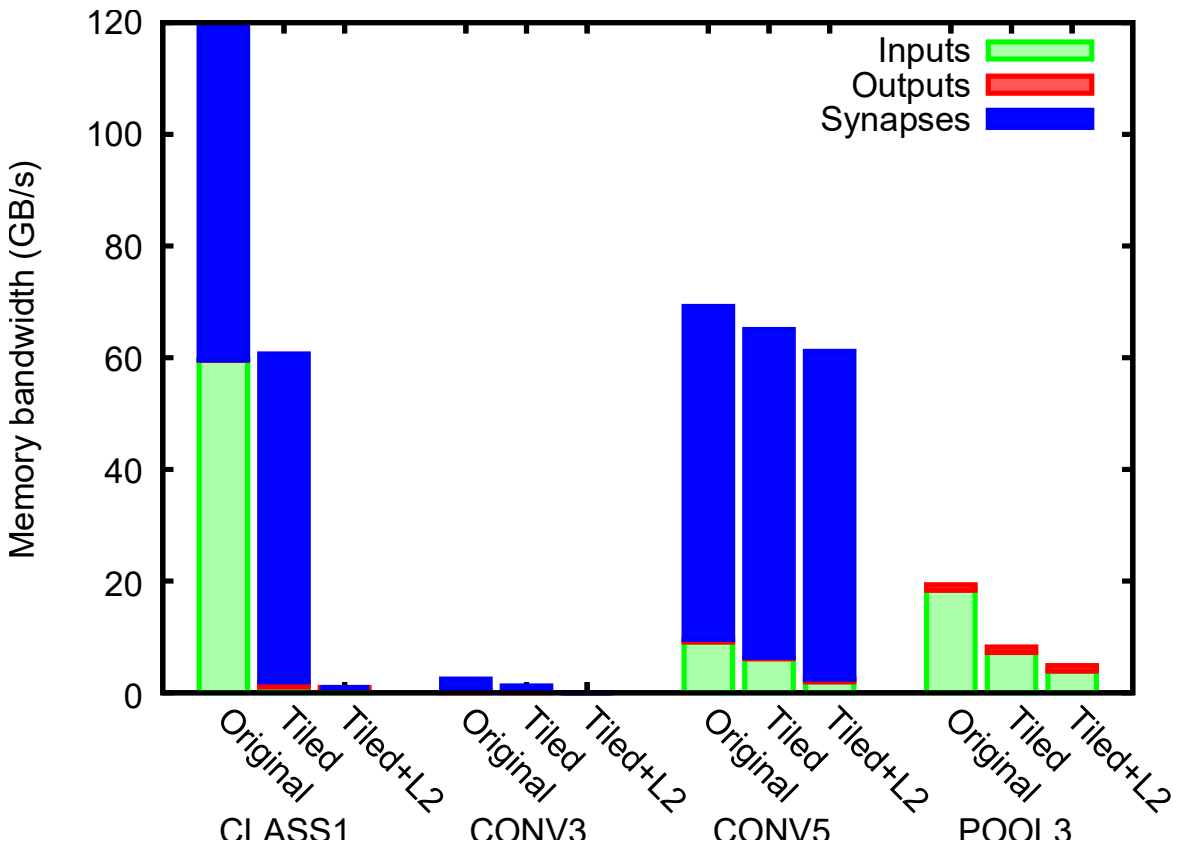



# Memory traffic optimization — Tiling



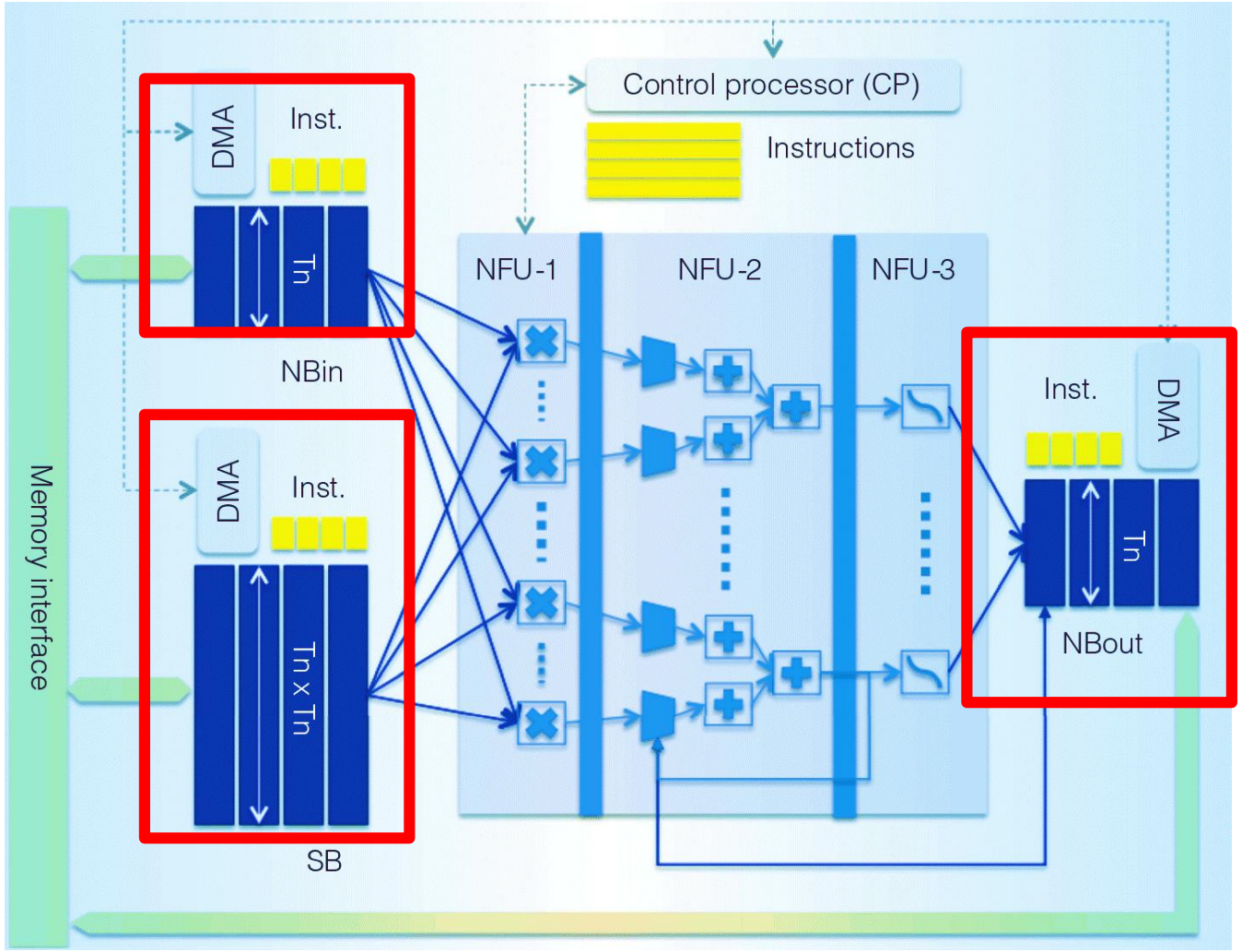
# Memory traffic – Difference



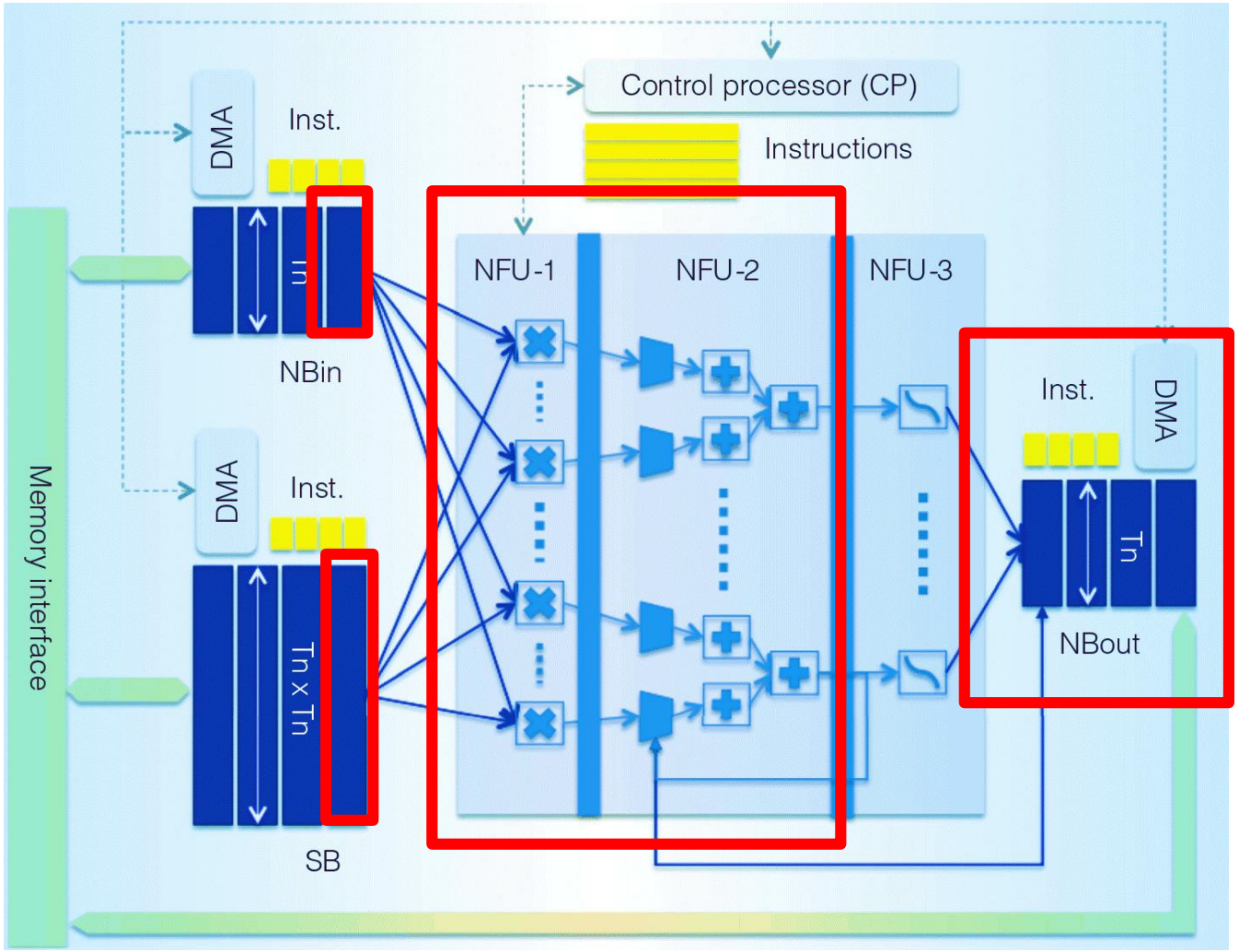

- **CONV3 – convolution layer using shared kernel**
  - Shift window is inherently reusing data
- **CONV5 – convolution layer using private kernel**
  - Parameters have to be loaded for every new input
- **POOL3 – pooling layer**
  - No parameters need to be loaded

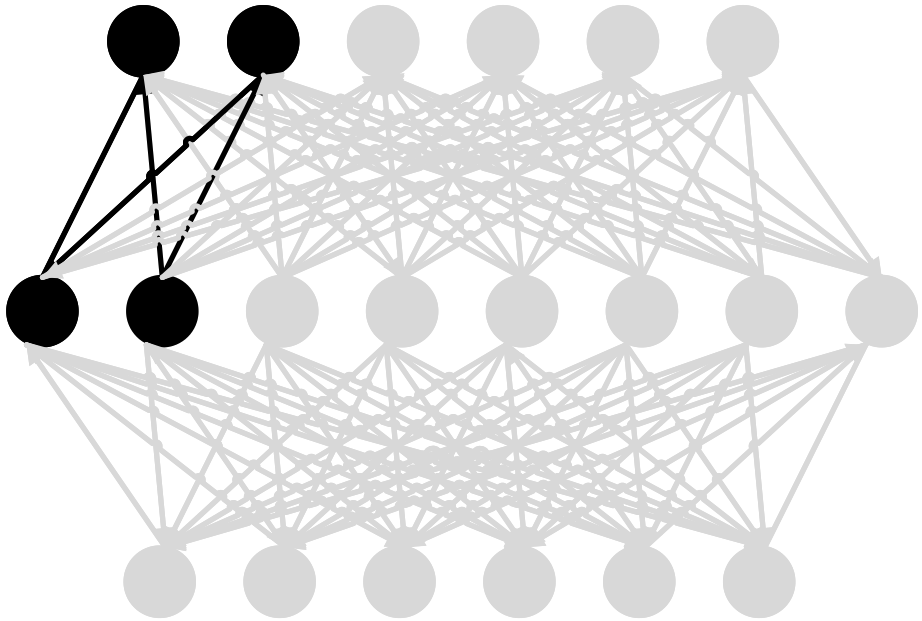
# Mapping optimization to hardware

- **NBin: Input data**
- **SB: Parameters**
- **NBout: output data**

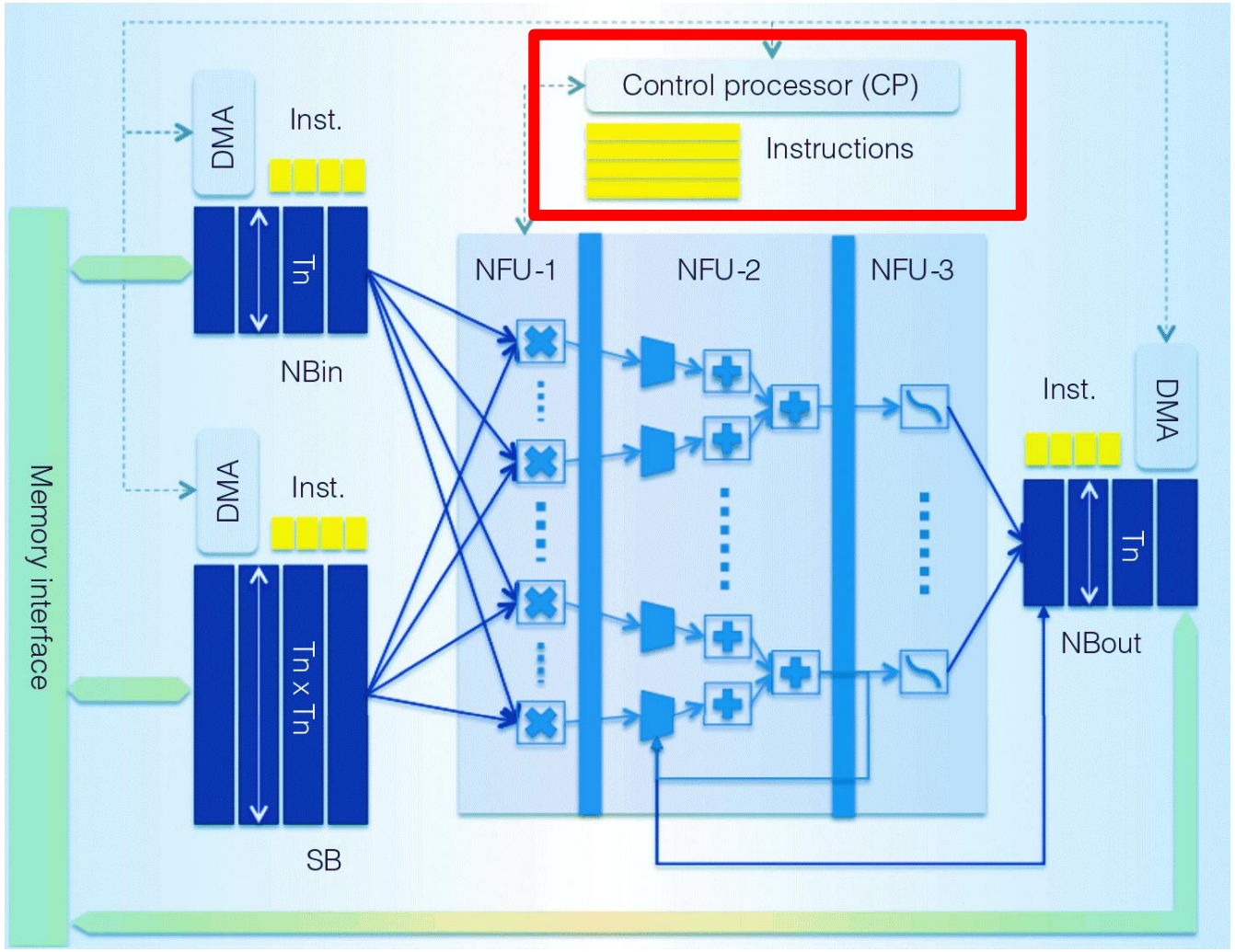


# Mapping optimization to hardware

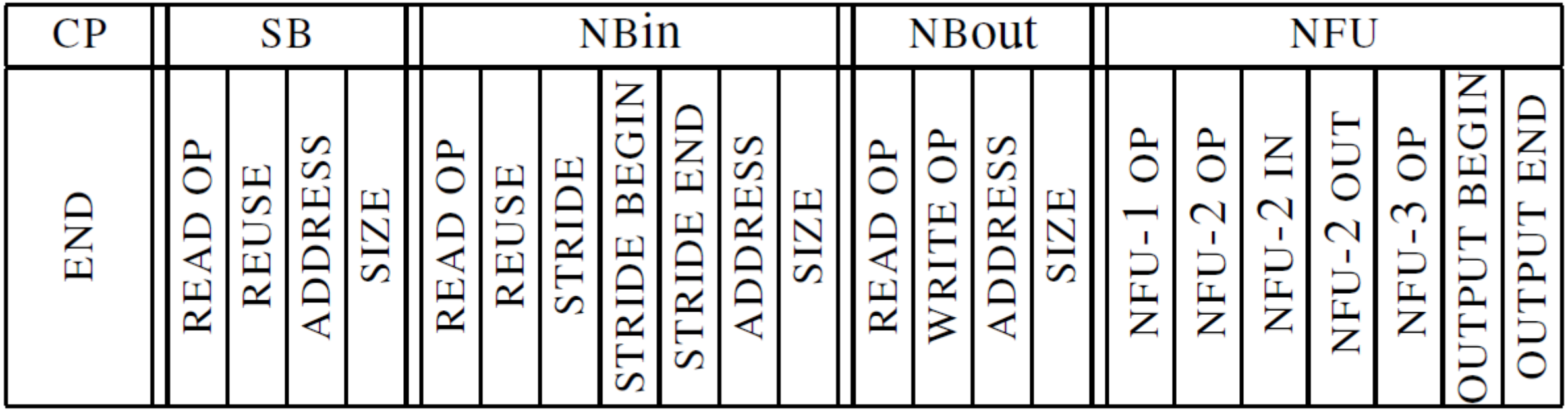


# How to solve control problem?



# Control instruction format



# Experiment

- **Base line**

SIMD:GEM5(simulator) + McPAT ( integrated power, area, and timing modeling frame work for multicore architecture.)

- **Accelerator**

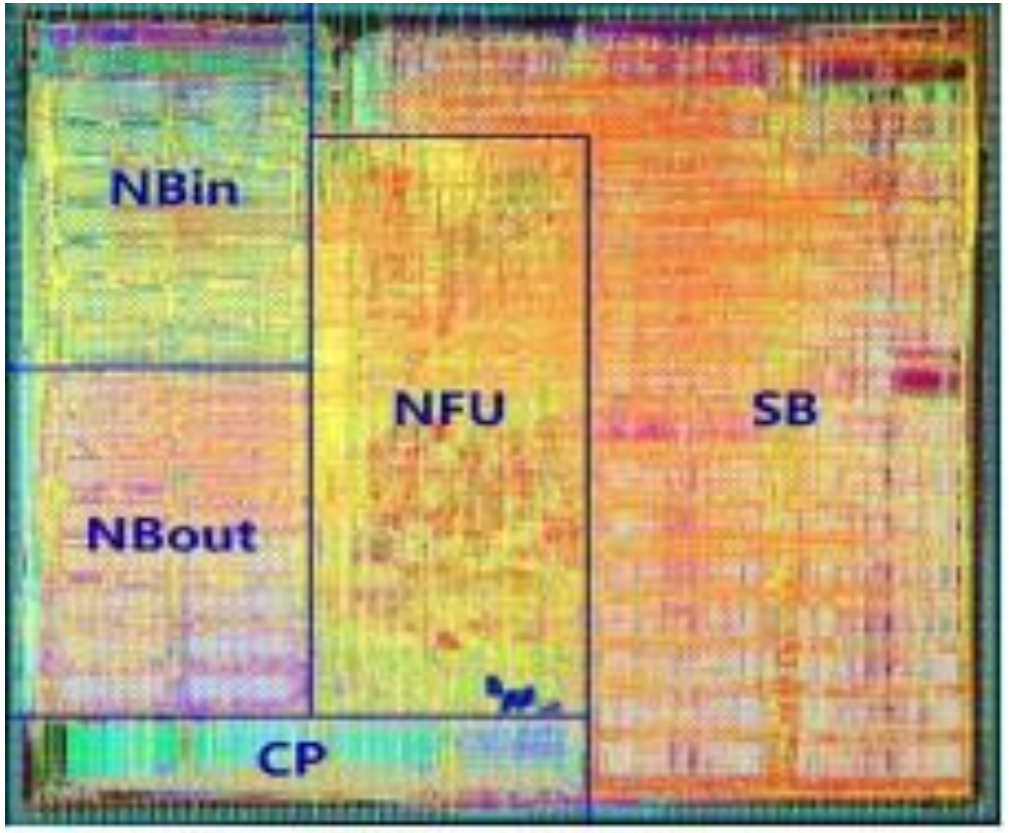
a custom cycle-accurate, bit-accurate C++ simulator

# Bench Mark

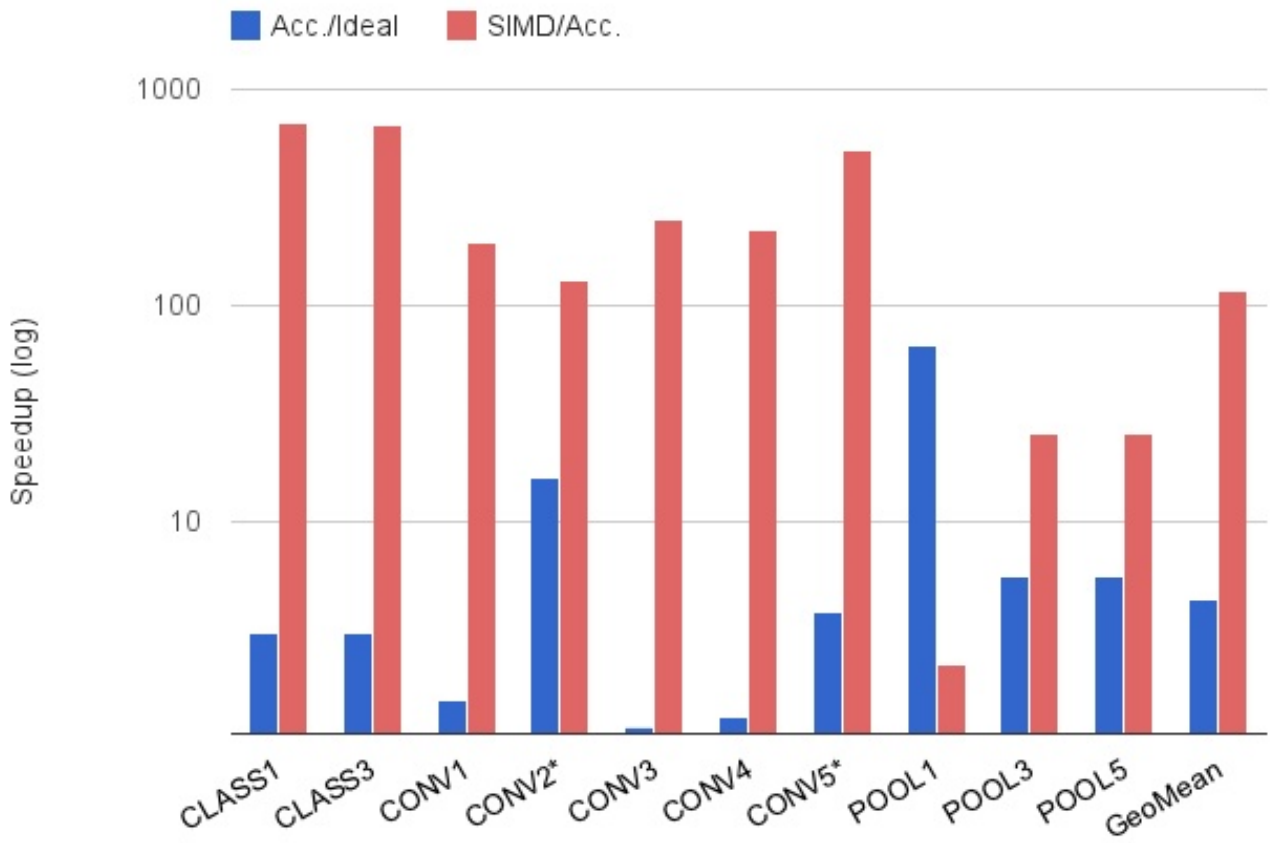
| Layer | $N_x$ | $N_y$ | $K_x$ | $K_y$ | $N_i$ | $N_o$ | Description |
|---|---|---|---|---|---|---|---|
| CONV1 | 500 | 375 | 9 | 9 | 32 | 48 | Street scene parsing (CNN) [13], (e.g., identifying "building", "vehicle", etc) |
| POOL1 | 492 | 367 | 2 | 2 | 12 | - | |
| CLASS1 | - | - | - | - | 960 | 20 | |
| CONV2* | 200 | 200 | 18 | 18 | 8 | 8 | Detection of faces in YouTube videos (DNN) [26], largest NN to date (Google) |
| CONV3 | 32 | 32 | 4 | 4 | 108 | 200 | Traffic sign identification for car navigation (CNN) [36] |
| POOL3 | 32 | 32 | 4 | 4 | 100 | - | |
| CLASS3 | - | - | - | - | 200 | 100 | |
| CONV4 | 32 | 32 | 7 | 7 | 16 | 512 | Google Street View house numbers (CNN) [35] |
| CONV5* | 256 | 256 | 11 | 11 | 256 | 384 | Multi-Object recognition in natural images (DNN) [16], winner 2012 ImageNet competition |
| POOL5 | 256 | 256 | 2 | 2 | 256 | - | |

# Result: Layout after P&R

| Components | Area(in %) | Power(in %) |
|---|---|---|
| RAM | 56 | 60 |
| NFU | 28 | 27 |



# Result: Speedup

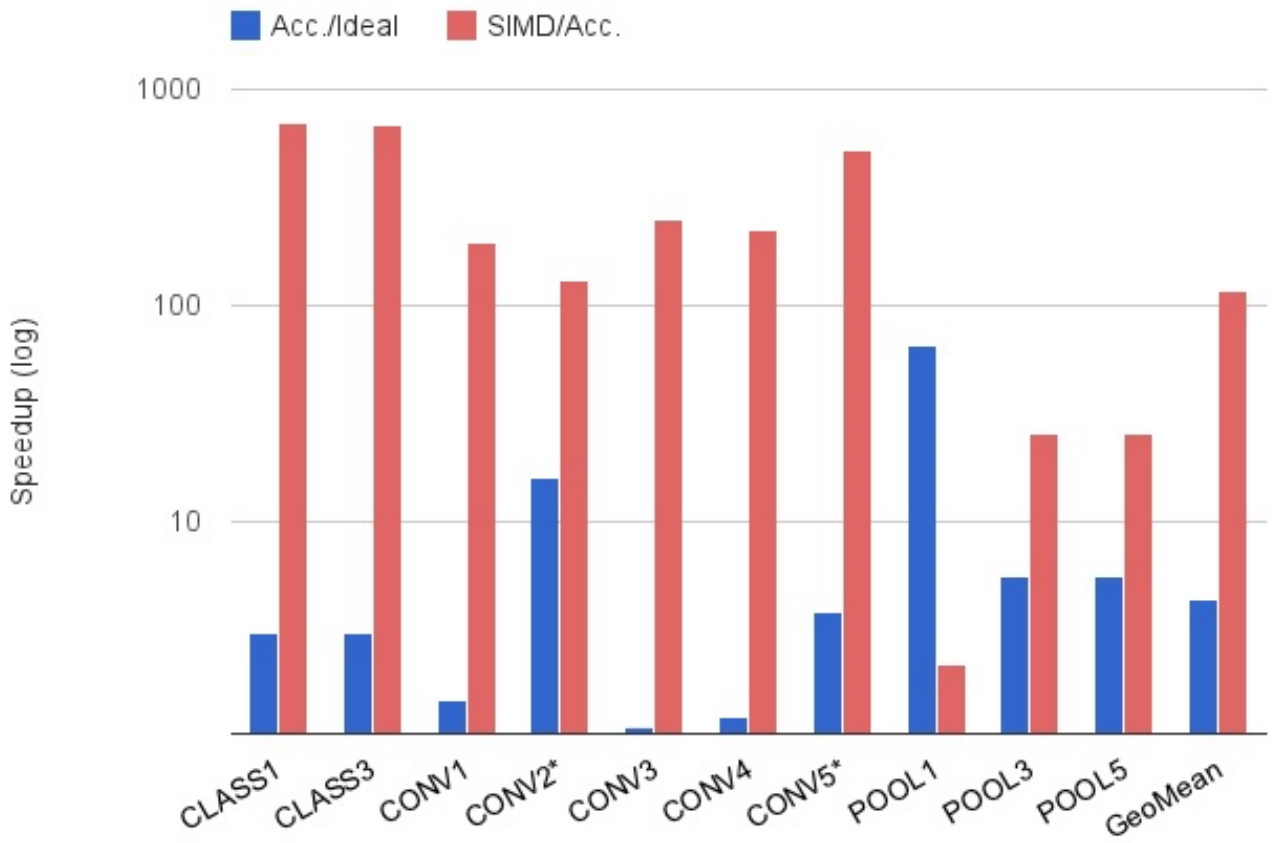



## Comparison:

- Diannao over SIMD
- Ideal model over Diannao

**Ideal model:** no off-chip fetching.

# Analysis of the speedup



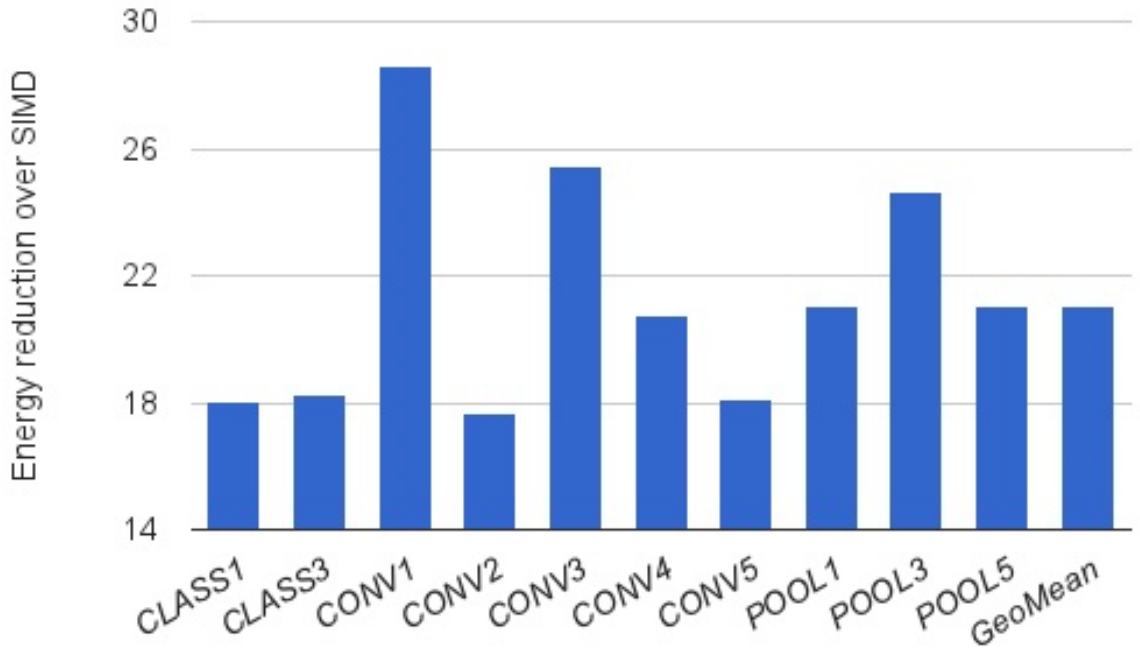

**Diannao outperforms SIMD**

- More operators
- Preload the data
- Simpler control logic

**Diannao far from ideal**

- Memory access

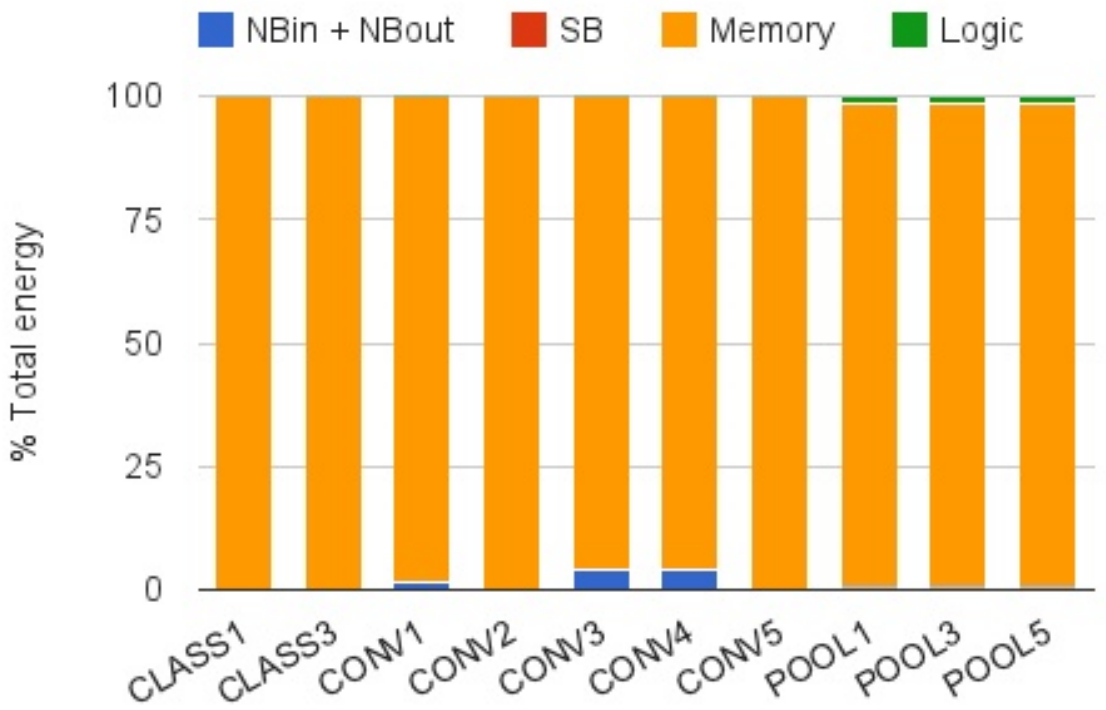
# Result: Energy




The Diannao outperforms the SIMD model 18~28X

However, other similar structure can achieve an energy ratio of 500

# Result: Energy Decomposition



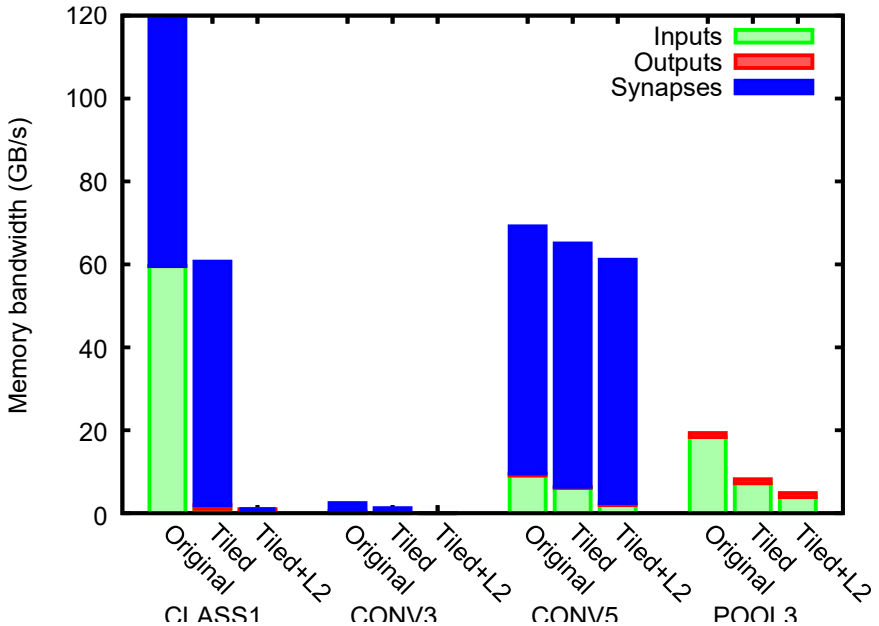

Most of energy consumed by memory access

Reduce the memory access energy is the future work

# Conclusions

- **Major idea:**
  - Reuse hardware to save area
  - Reuse data to save memory traffic
- **Limitations of results:**
  - no tape out for this design. All comparisons are based on simulation results
  - No comparison to GPU(they can’t beat GPU!), or other state-of-the-art accelerator
- **Limitations of design:**
  - Memory traffic still too large.
  - Not flexible enough

# Follow-up work —saving memory access energy cost in Dadiannao



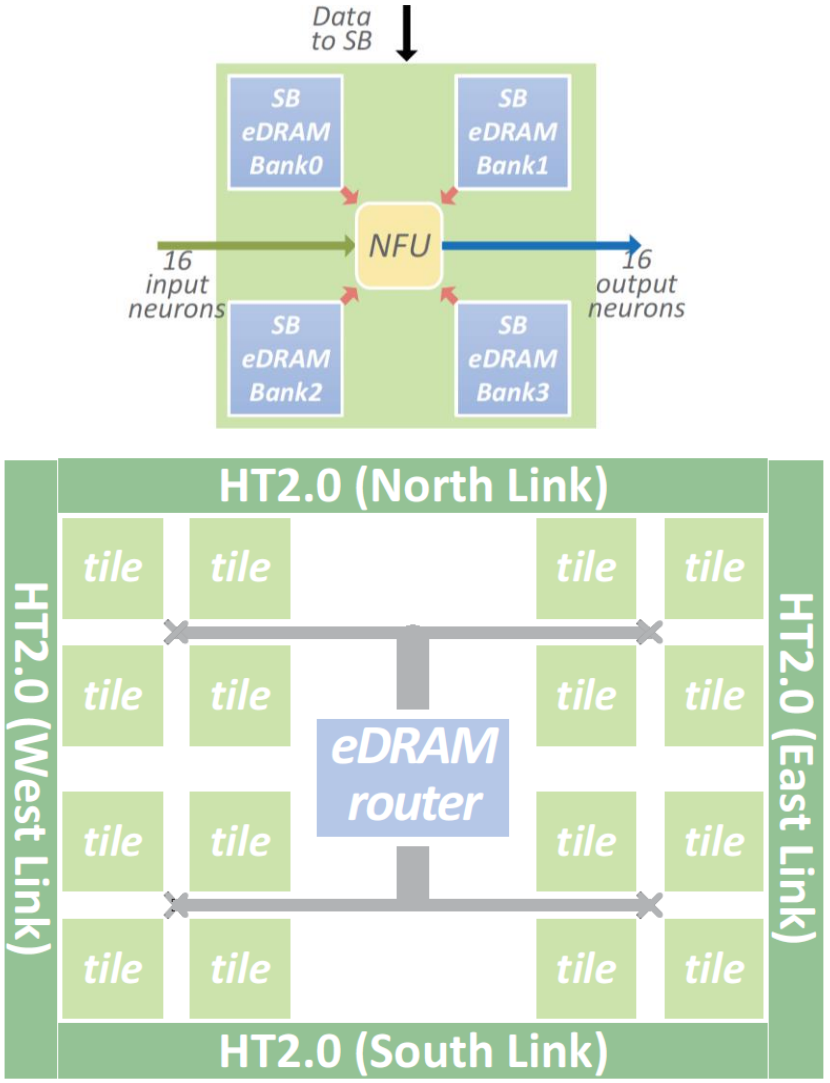

**Parameter fetching accounts for most of memory access for Fully-connected layer and convolution layers.**

**Solution: Store parameters locally.**

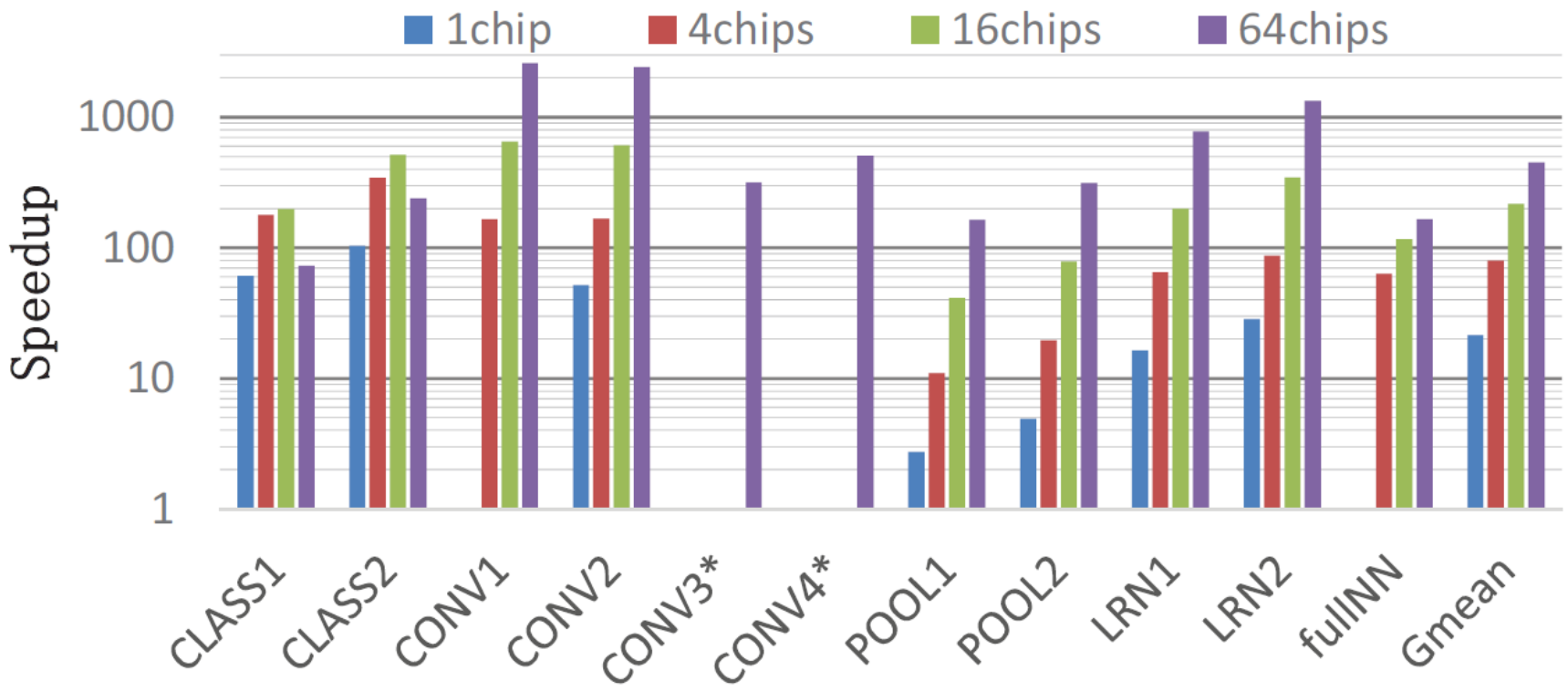
# Saving memory access energy cost in Dadiannao

## Design principles:

- Store parameters near NFU in eDRAM (denser than SRAM)
- Break down local storage into tiles to enable high internal bandwidth.
- Supporting multi-chip system for even larger NN (up to tens of GB corresponds to billions of parameters)



# Speedup of Dadiannao over GPU



# Continuation of the Story

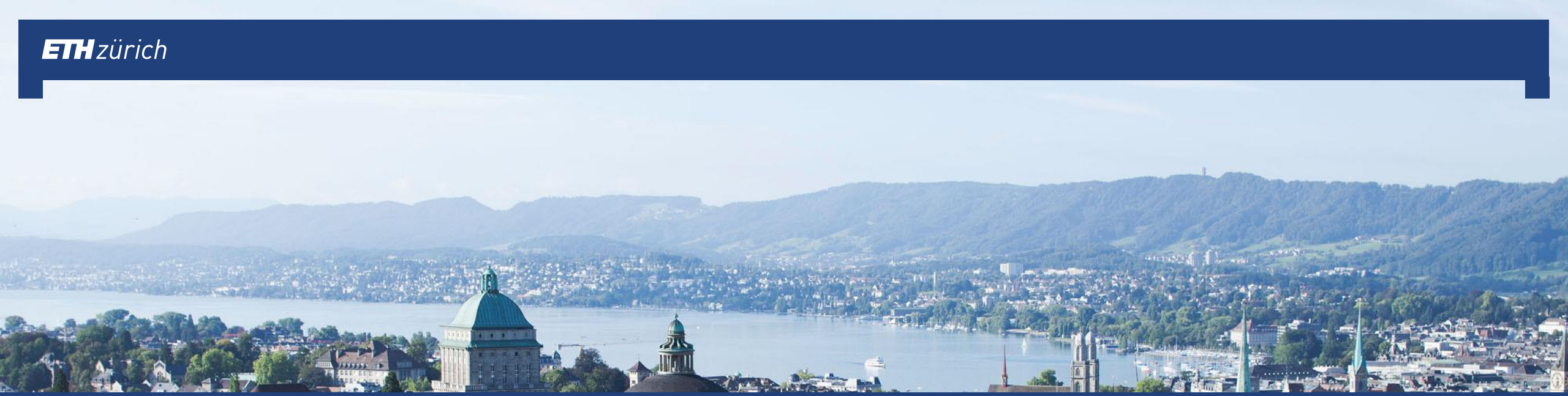
- DaDiannao (Big Diannao)
- ShiDiannao (Vision Diannao)
- PuDiannao (General Diannao)
- Diannaoyu (Diannao Language)
- Cambricon ( ISA ; more like general purpose processor)

# Comments on the paper itself

- **Cons:**
  - Annotations not explained immediately
  - Using pseudocode but not easy to read
  - In the result section, the comparisons are not convincing

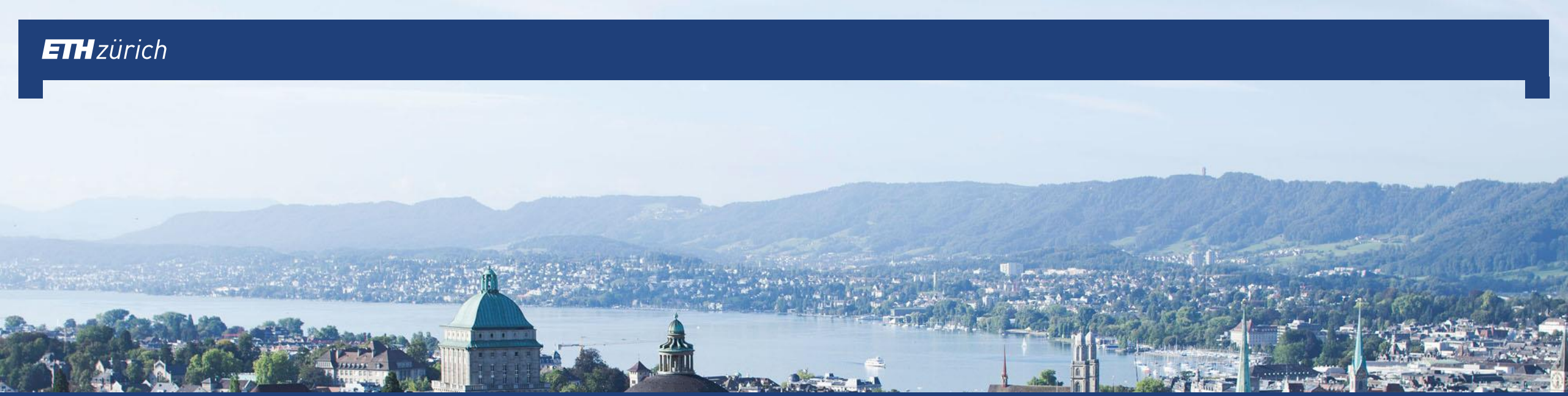
- **Pros:**
  - Relatively comprehensive consideration when solving problem
  - Relatively comprehensive analysis of their results
  - The starter of a direction

ETH zürich



# Thanks for your attention

ETH zürich



# Back up slides

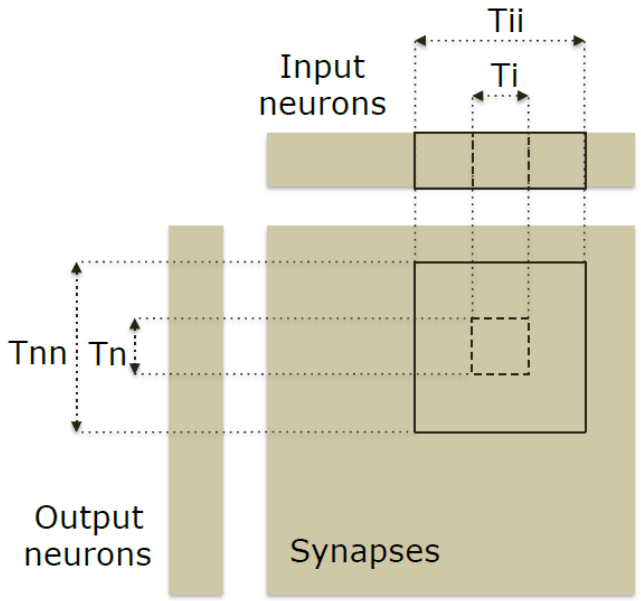



**Figure 2.** *Classifier layer tiling.*

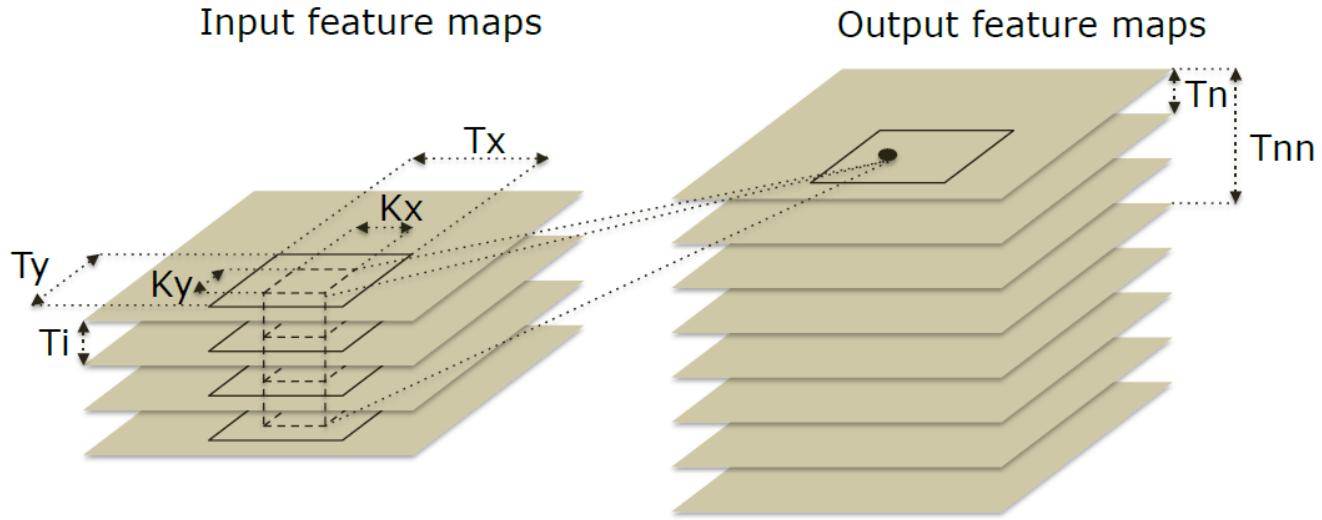



**Figure 3.** *Convolutional layer tiling.*

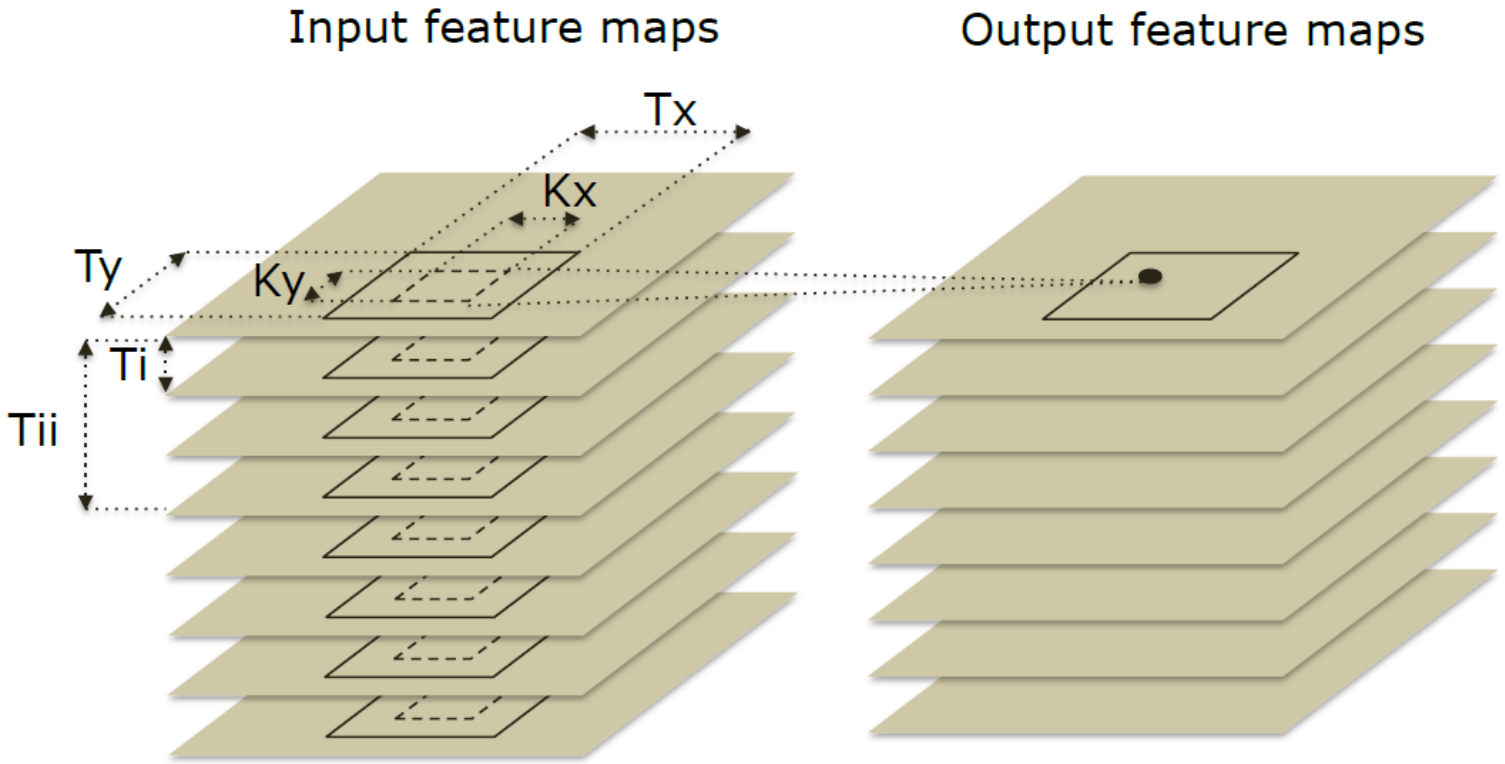



**Figure 4.** *Pooling layer tiling.*

| CP | SB | | | | NBin | | | | | | NBout | | | | NFU | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOP | LOAD | 0 | 0 | 32768 | LOAD | 1 | 0 | 0 | 0 | 4194304 | 2048 | NOP | WRITE | 0 | 0 | MULT | ADD | RESET | NBOUT | SIGMOID | 1 | 0 |
| NOP | LOAD | 0 | 32768 | 32768 | READ | 1 | 0 | 0 | 0 | 0 | 0 | NOP | WRITE | 0 | 0 | MULT | ADD | RESET | NBOUT | SIGMOID | 0 | 0 |
| ..... | | | | | | | | | | | | | | | | | | | | | | |
| NOP | LOAD | 0 | 7864320 | 32768 | LOAD | 1 | 0 | 0 | 0 | 4225024 | 2048 | READ | STORE | 8388608 | 512 | MULT | ADD | NBOUT | NFU3 | SIGMOID | 1 | 0 |
| ..... | | | | | | | | | | | | | | | | | | | | | | |

**Table 4.** *Subset of classifier/perceptron code ($N_i = 8192$ $N_o = 256$, $T_n = 16$, 64-entry buffers).*