清華大學
Tsinghua University

Going Deeper with Embedded FPGA Platform for Convolutional Neural Network

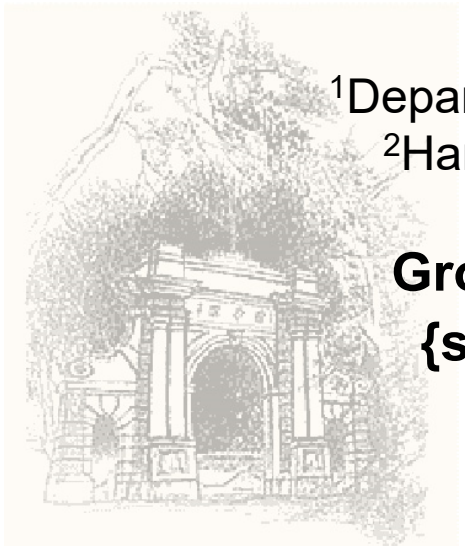
Jiantao Qiu[1], Jie Wang[1], Song Yao[1], Kaiyuan Guo[1], Boxun Li[1], Erjin Zhou[1], Jincheng Yu[1], Tianqi Tang[1], Ningyi Xu[2], Sen Song[3], Yu Wang[1], Huazhong Yang[1]

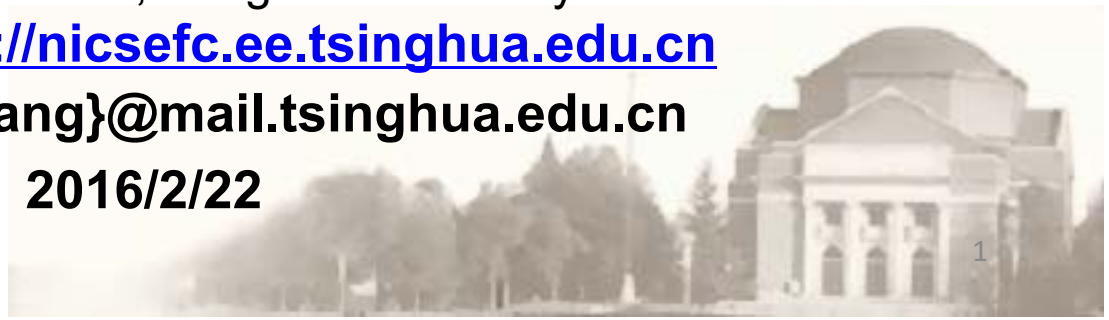
[1]Departmentt of Electronic Engineering, Tsinghua University
[2]Hardware Computing Group, Microsoft Research Asia
[3]School of Medicine, Tsinghua University

Group URL: http://nicsefc.ee.tsinghua.edu.cn

{songyao, yu-wang}@mail.tsinghua.edu.cn

2016/2/22



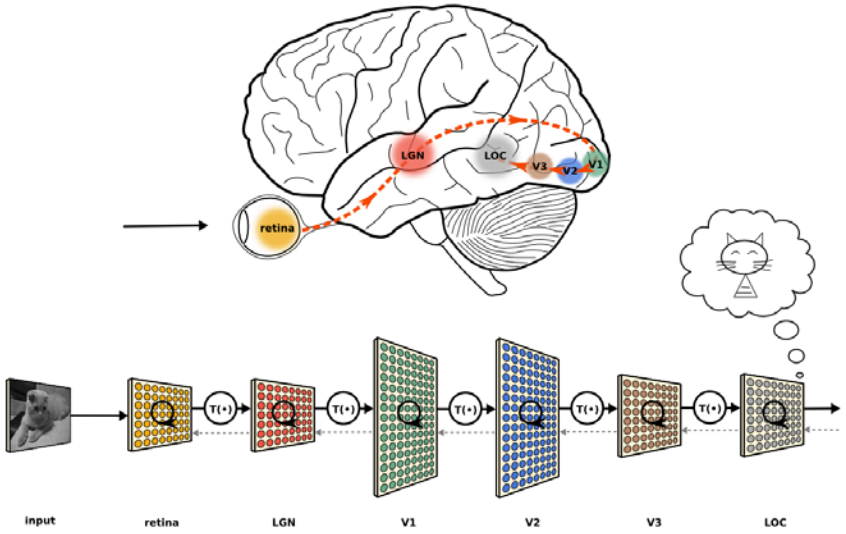


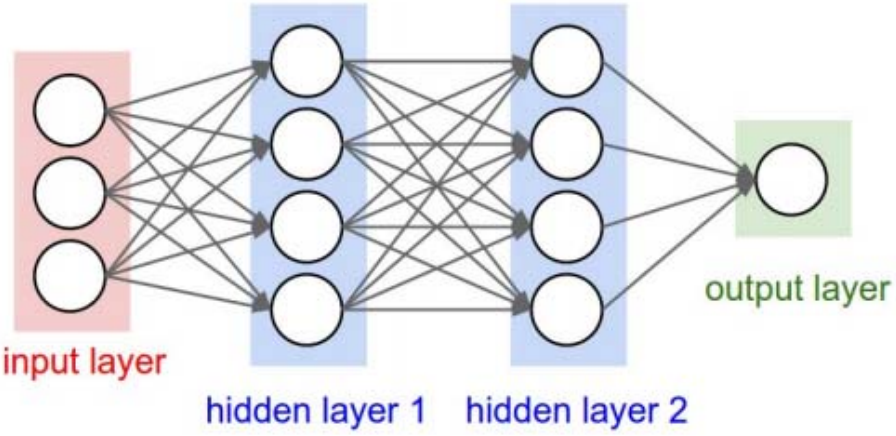
Contents

- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

Deep Learning

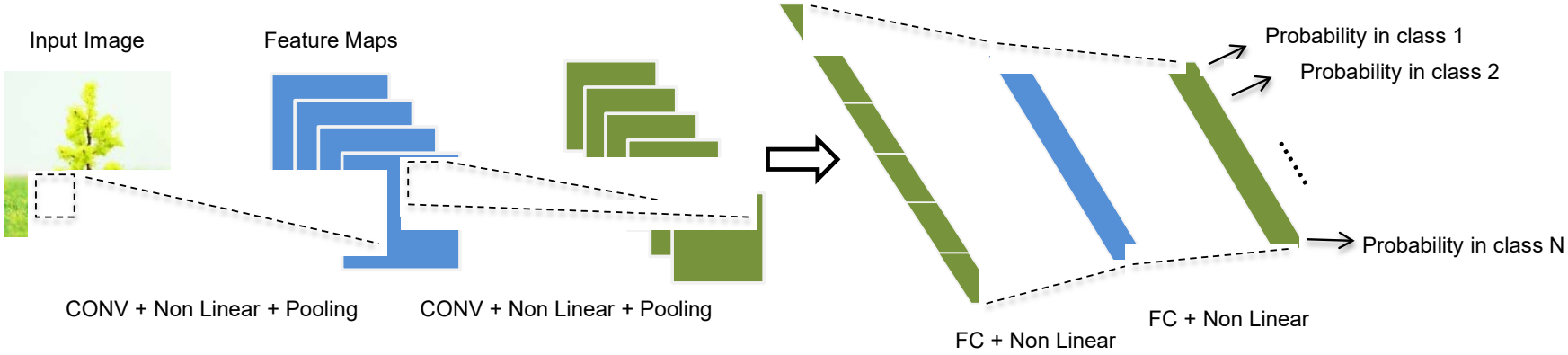
- Deep Learning: The new tide in artificial intelligence
  - Inspired by neuroscience
  - A collection of simple trainable mathematical units, which collaborate to compute a complicated function.
  - Deep Neural Network (DNN)/Recurrent Neural Network (RNN)/Long-Short Term Memory (LSTM)/Convolutional Neural Network (CNN)





Convolutional Neural Network (CNN)

- CNN: State-of-the-art in visual recognition applications



	Year	Team	Top-5 Accuracy
Non CNN	2010	NEC	71.8%
	2011	XRCE	74.2%
CNN	2012	SuperVision	84.7%
	2013	Clarifai	88.3%
	2014	GoogLeNet	93.3%
	2015	MSRA	96.4%

Top-5 accuracy of image classification in Image-Net Large-Scale Visual Recognition Challenge (ILSVRC)

Contents

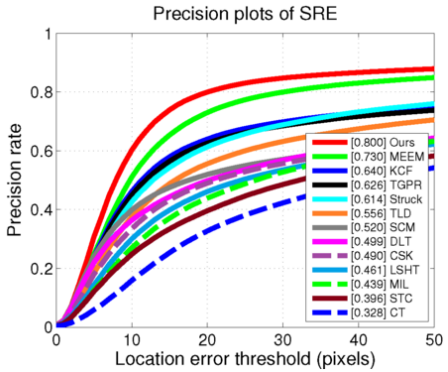
- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

CNN: Mainstream in Computer Vision

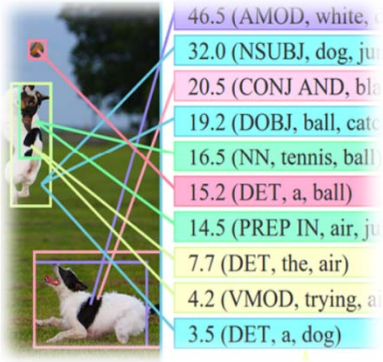
- CNN: State-of-the-art in visual recognition applications

Pedestrian Detection [NUS2015]

Vehicle and Lane Detection [Stanford2015]



Tracking [UIUC2015]



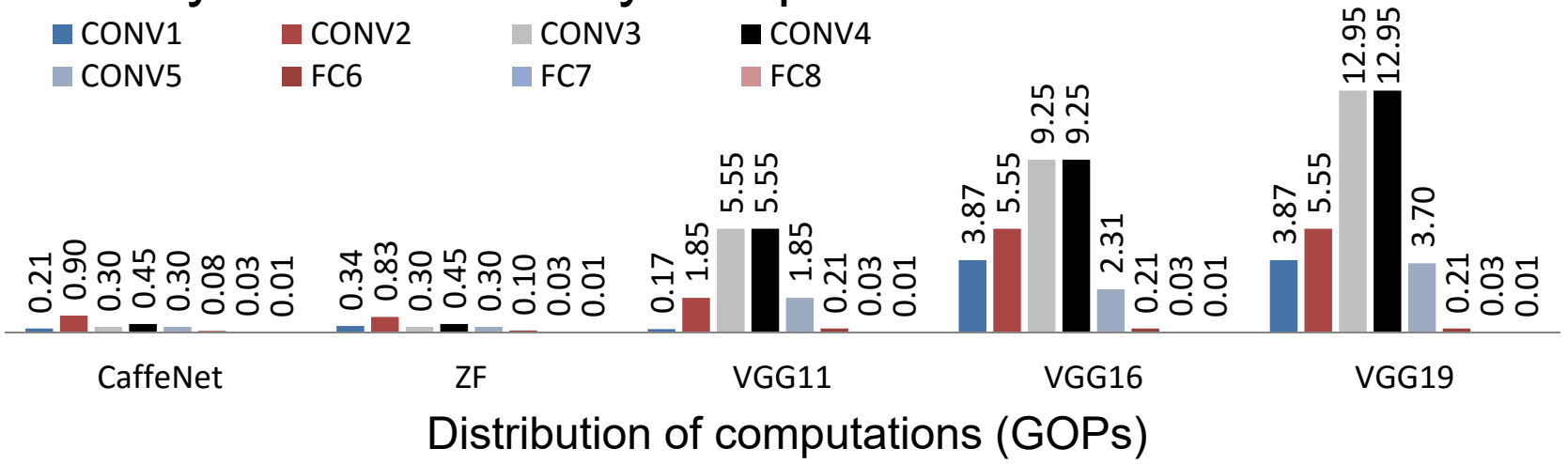
Object detection

Posture estimation

Face recognition

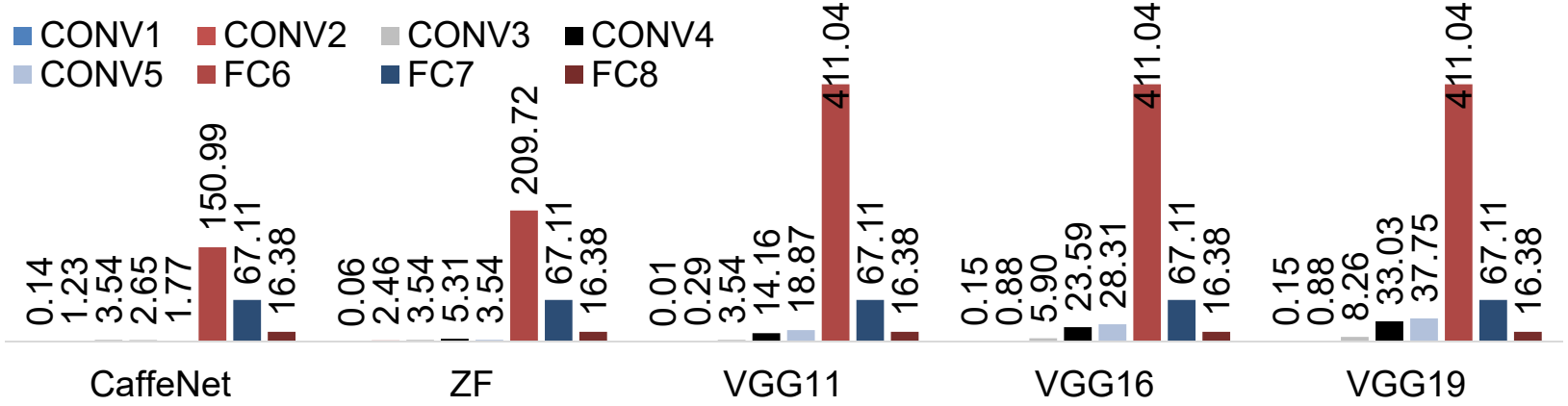
CNN: High Complexity

- Conv Layers: bounded by computations



Distribution of computations (GOPs)

- FC Layers: bounded by memory access



Distribution of storage demands (GOPs)

Motivation

Why customized hardware?

- High complexity versus Limited energy
- CPU and GPU are not efficient enough

How to accelerate CNN with FPGA?

- High Complexity under Limited Resource
  - CNN Model Compression
  - Highly efficient computing units
  - Using convolver for FC layers
- High Complexity under Limited Bandwidth
  - CNN model compression
  - Shorter representations
  - Reducing memory access

CNN acceleration is more than hardware
Complete compilation tool is expected

Contents

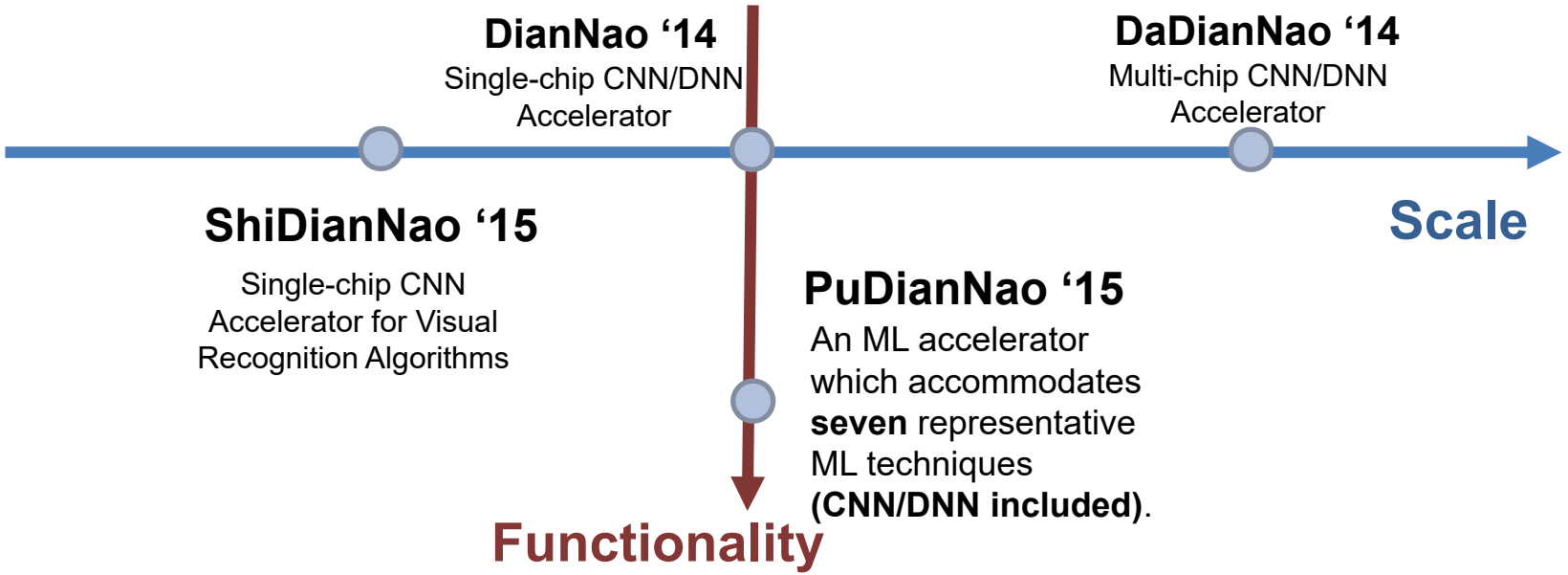
- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

Related Work

- **(MSR)** K. Ovtcharov et al. Accelerating deep convolutional neural networks using specialized hardware. Microsoft Research Whitepaper, 2, 2015
- **(MSR Slides)** K. Ovtcharov et al. Toward Accelerating Deep Learning at Scalue Using Specialized Hardware in the Datacennter, Hotchips 2015.
- **(Baidu Slides)**
- **(NYU)** C. Farabet et al. An FPGA-based Stream Processor for Embedded Real-Time Vision with Convolutional Networks, ECVW 2009.
- **(NYU)** C. Farabet et al. CNP: An FPGA-based Processor for Convolutional Networks, FPL 2009.
- **(NEC)** M. Sankaradas et al., A massively parallel coprocessor for convolutional neural networks. ASAP 2009.
- **(NEC)** S. Cadambi et al., A programmable parallel accelerator for learning and classification. PACT 2010.
- **(NEC)** S. Chakradhar et al., A dynamically configurable coprocessor for convolutional neural networks. In ACM SIGARCH Computer Architecture News 2010.
- **(NYU/Yale)** Farabet et al. Large-Scale FPGA-based Convolutional Networks, 2011.
- **(NYU/Yale)** Farabet et al. NeuFlow: A Runtime Reconfigurable Dataflow Processor for Vision, ECVW 2011.
- **(NYU/Yale)** Farabet et al. Hardware Accelerated Convolutional Neural Network for Synthetic Vision Systems.
- **(Purdue/NYU)** P. Pham, NeuFlow: Dataflow Vision Processing System-on-a-chip, MWSCAS 2012.
- **(Eindhoven University of Technology)** M. Peemen et al. Memory-centric accelerator design for convolutional neural networks. ICCD 2013.
- **(Purdue)** V. Gokhale et al. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks, CVPRW 2014.
- **(CAS)** T. Chen et al. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine learning, ASPLOS 2014
- **(CAS)** Y. Chen et al. DaDianNao: A Machine-Learning Supercomputer, MICRO 2014
- **(CAS)** Y. Chen et al. PuDianNao: A Machine Learning Accelerator, ASPLOS 2015
- **(CAS)** Z. Du et al. Shidiannao: Shifting vision processing closer to the sensor, ISCA 2015
- **(PKU)** C. Zhang et al., Optimizing fpga-based accelerator design for deep convo-lutional neural networks. FPGA 2015.
- **(MIT)** Y. Chen et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. ISSCC 2016.
- **(KAIST)** J. Sim et al. A 1.42tops/w deep convolutional neural network recognition processor for intelligent ioe systems. ISSCC 2016.
- **(Stanford)** S. Han et al. EIE: Efficient Inference Engine on Compressed Deep Neural Network, arxiv

Related Work

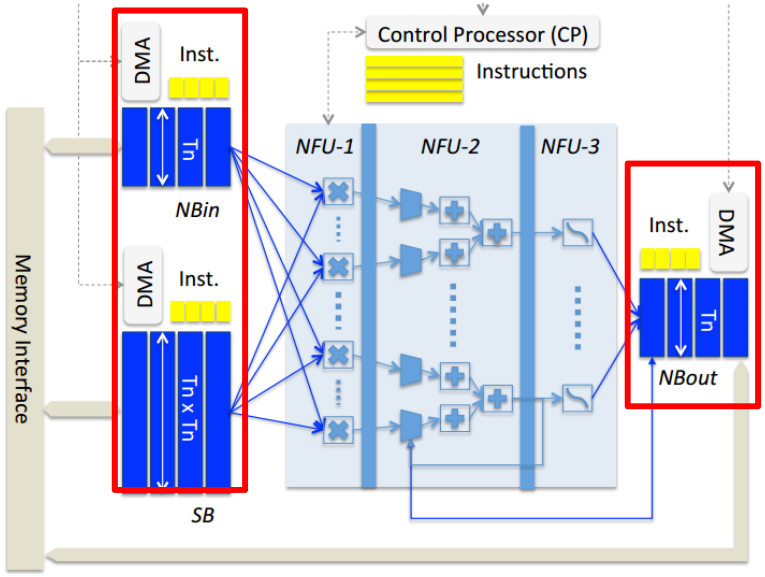
- Memory System Optimization
  - DianNao Series

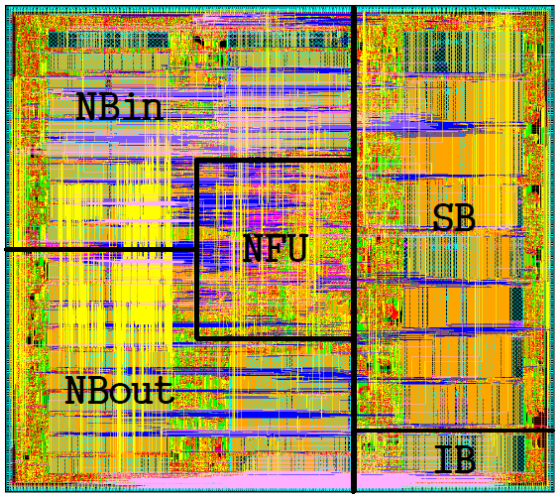


*1 Diannao: A small-footprint high-throughput accelerator for ubiquitous machine learning, Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. ASPLOS ’14
*2 DaDianNao: A Machine-Learning Supercomputer, Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, O. Temam, MICRO ’14
*3 PuDianNao: A Machine Learning Accelerator, Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, Jia Wang, L. Li, T. Chen, Z. Xu, N. Sun, O. Temam, ASPLOS ‘15
*4 Shidiannao: Shifting vision processing closer to the sensor, Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, ISCA ’15

Related Work

- Memory System Optimization
  - DianNao Series





How to solve the memory problem?

Strategy 1: Tiling and Data Reuse

Cut down memory traffic

Strategy 2: Storage Buffer

Dedicated buffer for data reuse

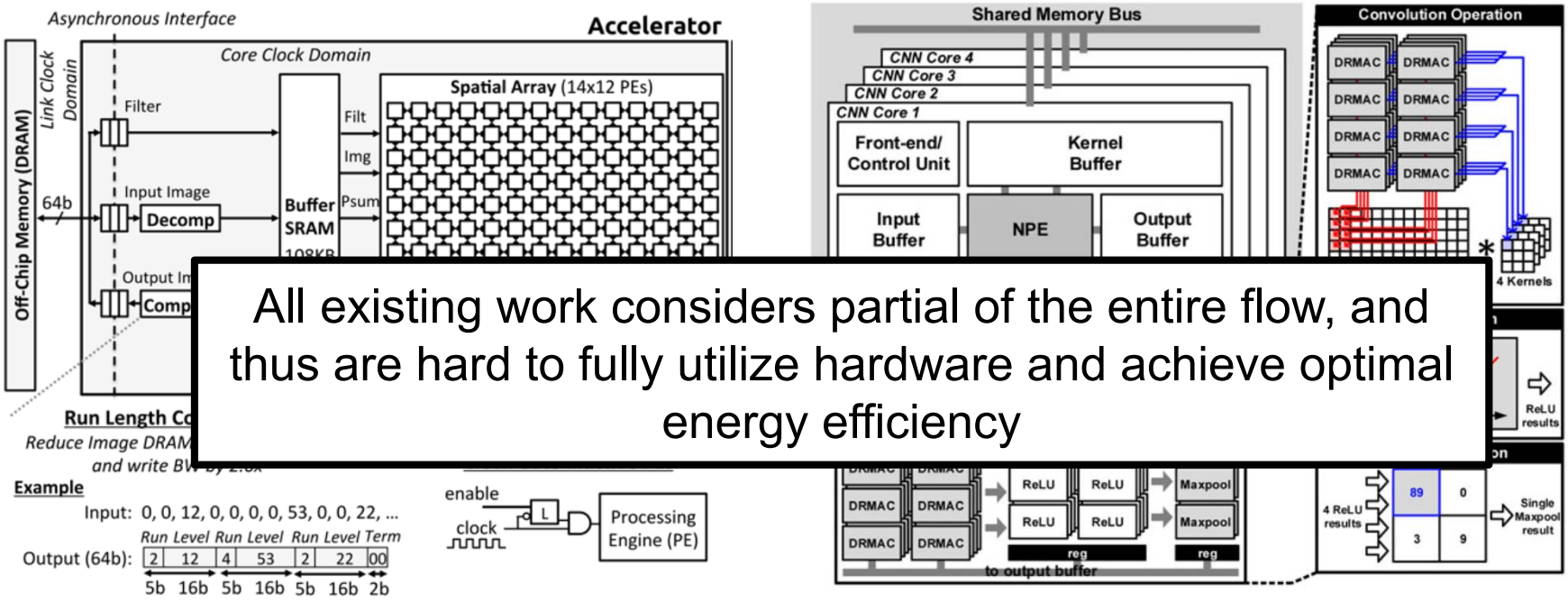
Strategy 3: On-Chip Memory

Using on-chip memory to store all parameters

Problem: Using on-chip memory to store parameters in each layer of the CNN model, hard to be used for state-of-the-art large CNN models

Related Work

- Computing Engine Optimization



All existing work considers partial of the entire flow, and thus are hard to fully utilize hardware and achieve optimal energy efficiency

Small PE: Eyeriss [MIT ISSCC2016]

Complex PE: [KAIST ISSCC2016]

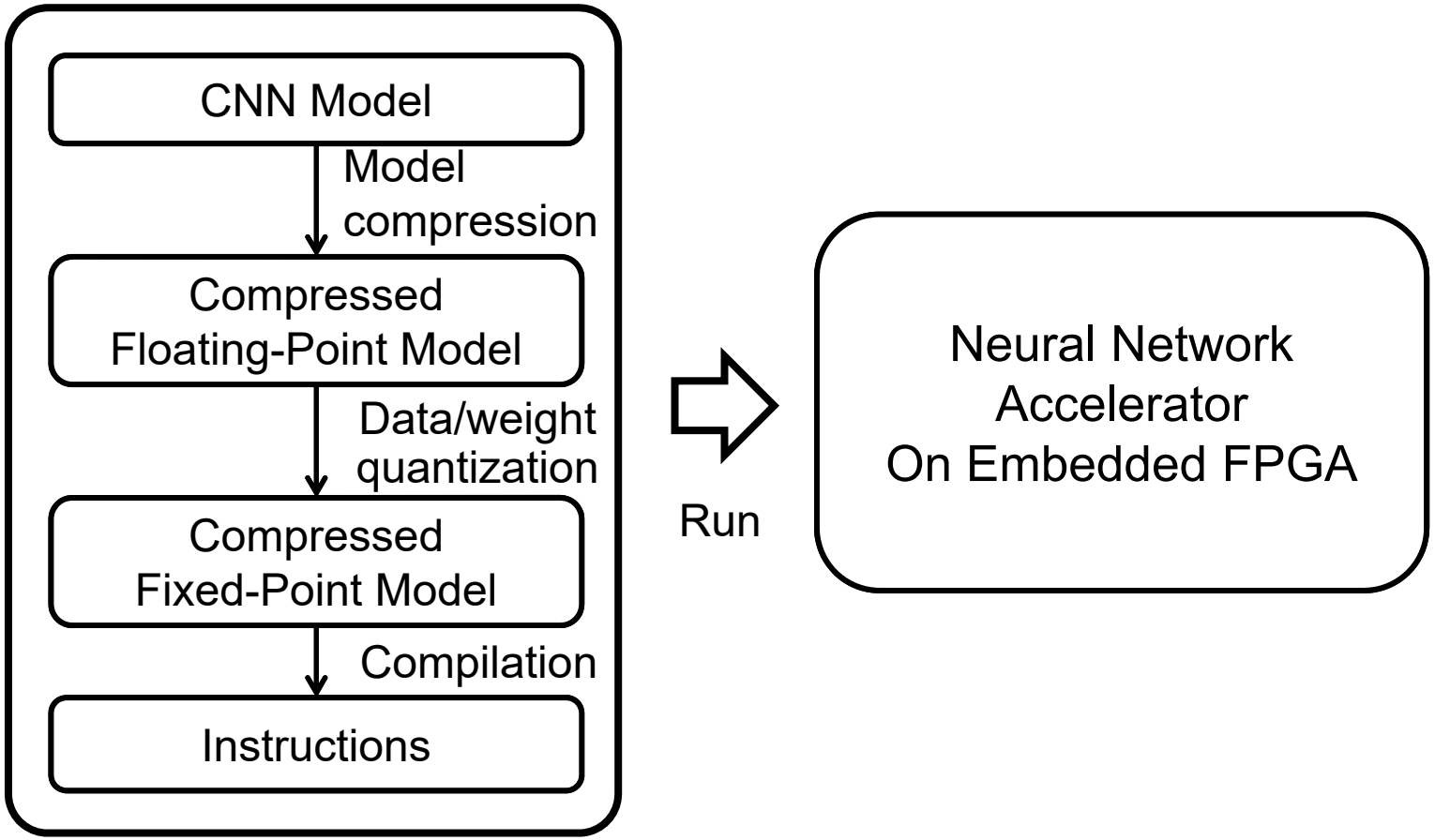
- [MIT ISSCC2016] Y. Chen et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. ISSCC 2016.
- [KAIST ISSCC2016] J. Sim et al. A 1.42tops/w deep convolutional neural network recognition processor for intelligent ioe systems. ISSCC 2016.

Contents

- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

Overall Flow

- Overall Flow of Angel-Eye



"Goal: accelerate fast algorithms."

Model Compression

- Model Compression
  - Reducing complexity while maintaining comparable accuracy

Original Shallow Thin Decomposition Pruning

- Singular Value Decomposition (SVD)
  - No demand for specific computation unit
  - Moderate compression
  - Computation model

$$f^{out} = W f^{in} + b \Rightarrow W \approx U_d S_d V_d \quad U_d \in \mathbb{R}^{n_{out} \times d},\ V_d \in \mathbb{R}^{d \times n_{in}},\ S_d \in \mathbb{R}^{d \times d}$$

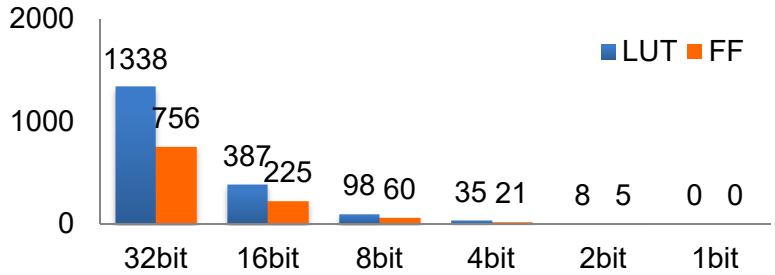
  - Storage complexity

$$O(n_{in} n_{out}) \Rightarrow O(d n_{in} + d n_{out}) \quad d \ll n_{in}, n_{out}$$

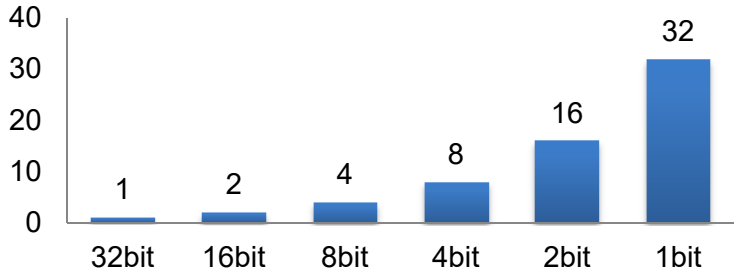
Network	FC6	# of total weights	# of operations	Top-5 accuracy
VGG-16	25088×4096	138.36M	30.94G	88.00%
VGG-16-SVD	25088×500 + 500×4096	50.18M	30.76G	87.96%

Data Quantization

- Data Quantization
  - Uses shorter fixed-point numbers

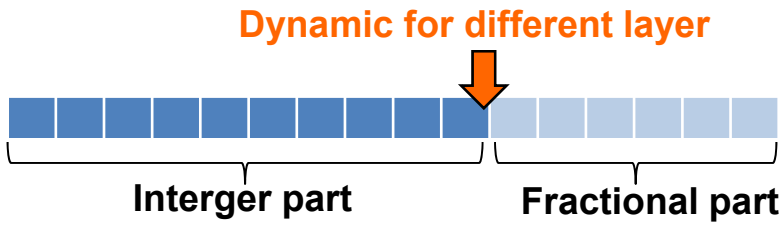


Resources needed by a multiplier

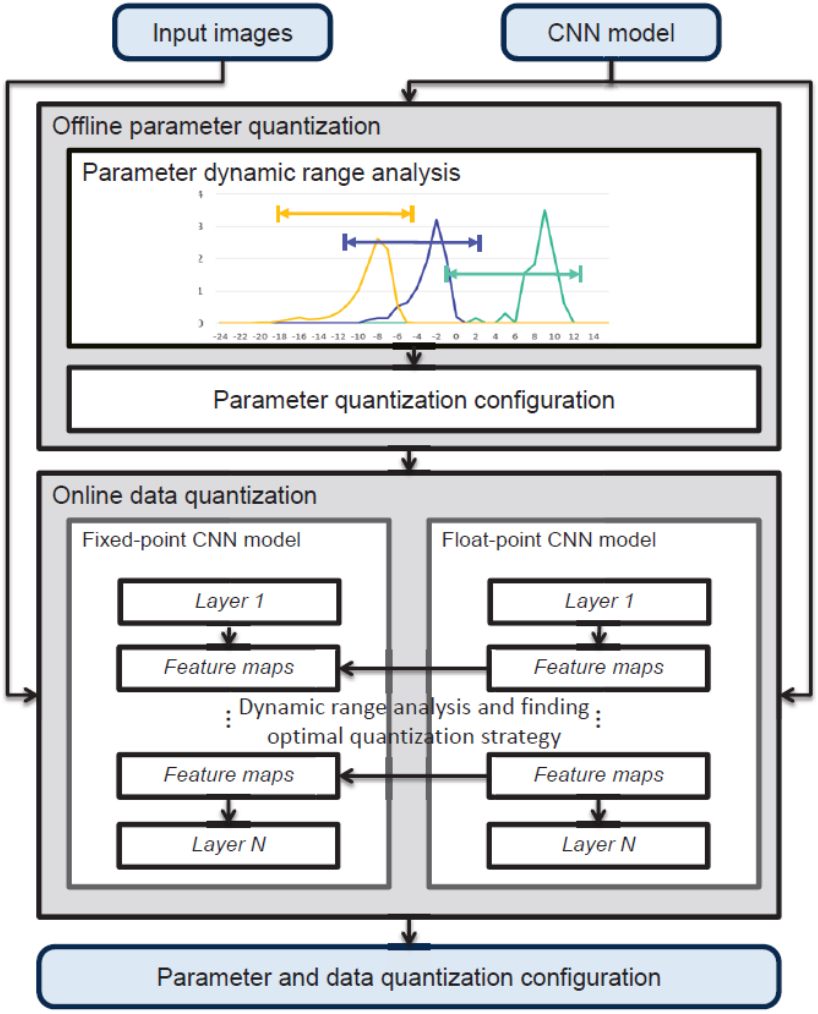


Bandwidth utilization

- Dynamic-Precision Data Quantization



- Proposed Flow



Data Quantization

- Dynamic-Precision Data Quantization Results (Simulation results)

Network	VGG16						
Data Bits	Single-float	16	16	8	8	8	8
Weight Bits	Single-float	16	8	8	8	8	8 or 4
Data Precision	N/A	2^{-2}	2^{-2}	Impossible	$2^{-5}/2^{-1}$	Dynamic	Dynamic
Weight Precision	N/A	2^{-15}	2^{-7}	Impossible	2^{-7}	Dynamic	Dynamic
Top-1 Accuracy	68.1%	68.0%	53.0%	Impossible	28.2%	66.6%	67.0%
Top-5 Accuracy	88.0%	87.9%	76.6%	Impossible	49.7%	87.4%	87.6%

Network	CaffeNet			VGG16-SVD		
Data Bits	Single-float	16	8	Single-float	16	8
Weight Bits	Single-float	16	8	Single-float	16	8 or 4
Data Precision	N/A	Dynamic	Dynamic	N/A	Dynamic	Dynamic
Weight Precision	N/A	Dynamic	Dynamic	N/A	Dynamic	Dynamic
Top-1 Accuracy	53.9%	53.9%	53.0%	68.0%	64.6%	64.1%
Top-5 Accuracy	77.7%	77.1%	76.6%	88.0%	86.7%	86.3%

Instruction Set

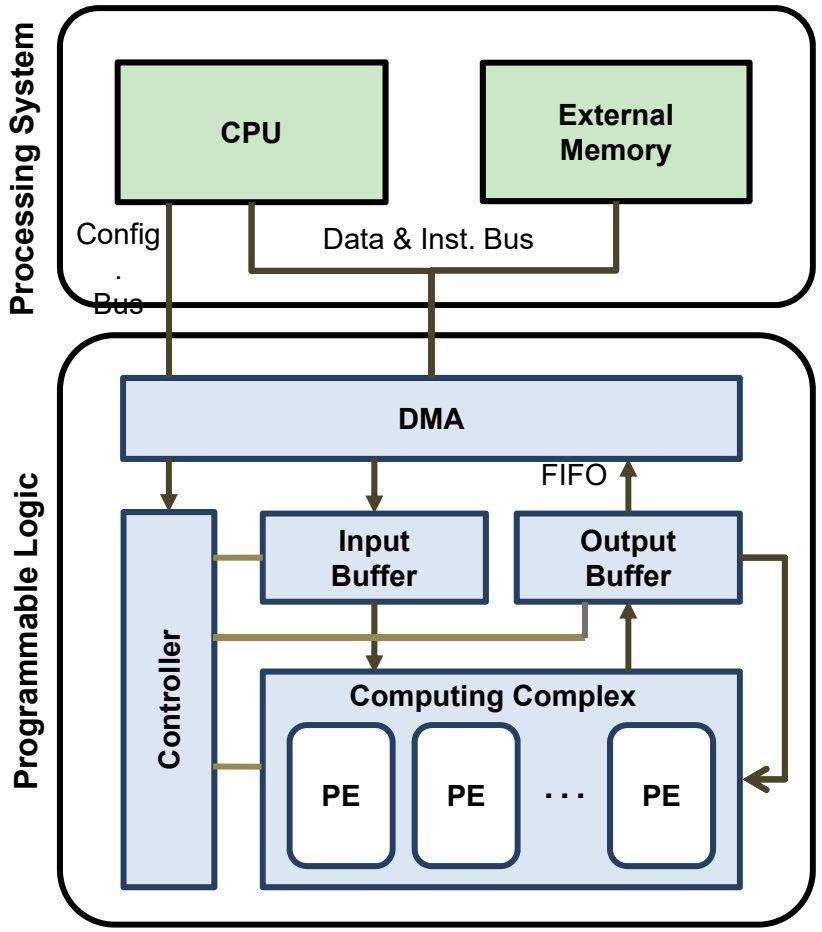
- Coarse-grained Instructions

Index	Pool Bypass	NL Bypass	Zero Switch	Result Shift	Bias Shift	Write En	PE En	Phase Type	Pic Num	Tile Size	Layer Type
1	X	X	X	X	X	No	2	First	2	Tr	Conv
2	Yes	Yes	Bias	X	BS	No	2	Calculate	2	Tr	Conv
3	No	No	Zero	X	X	PE	2	Calculate	2	Tr	Conv
4	X	X	X	RS	X	DDR	2	Last	2	Tr	Conv

- **Hardware handles fine-grained operations**
- Inst 1: commands Input Buffer to load all the needed data
- Inst 2: starts calculating the four tiled blocks in the output layer
- Inst 3: Write En is set as “PE” to command Output Buffer send the intermediate results back to the Pes
- Inst 4: Write EN is set as “DDR” to command the Output Buffer write results back to the external memory (last layer)

Architecture and Implementation Details

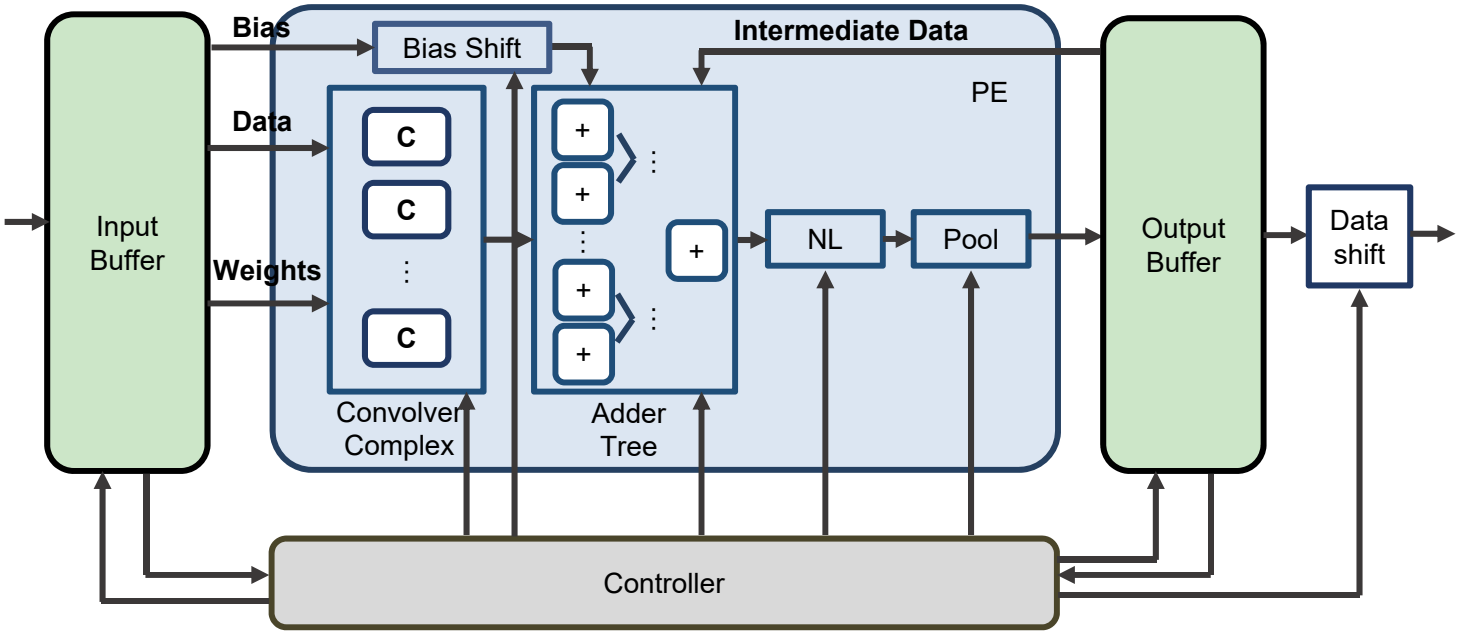
- Overall Architecture



- **Processing System**
  - Flexibility
  - CPU + DDR
  - Scheduling operations
  - Prepare data and instructions
  - Realize Softmax function
- **Programmable Logic**
  - Hardware acceleration
  - Computing Complex + On-chip Buffers + Controller + DMA
  - Few Complex PEs
- **Achieve three-level parallelism**
  - Inter-output: multiple PEs
  - Intra-output
  - Operator-level
- **16-bit dynamic-precision data quantization**

Architecture and Implementation Details

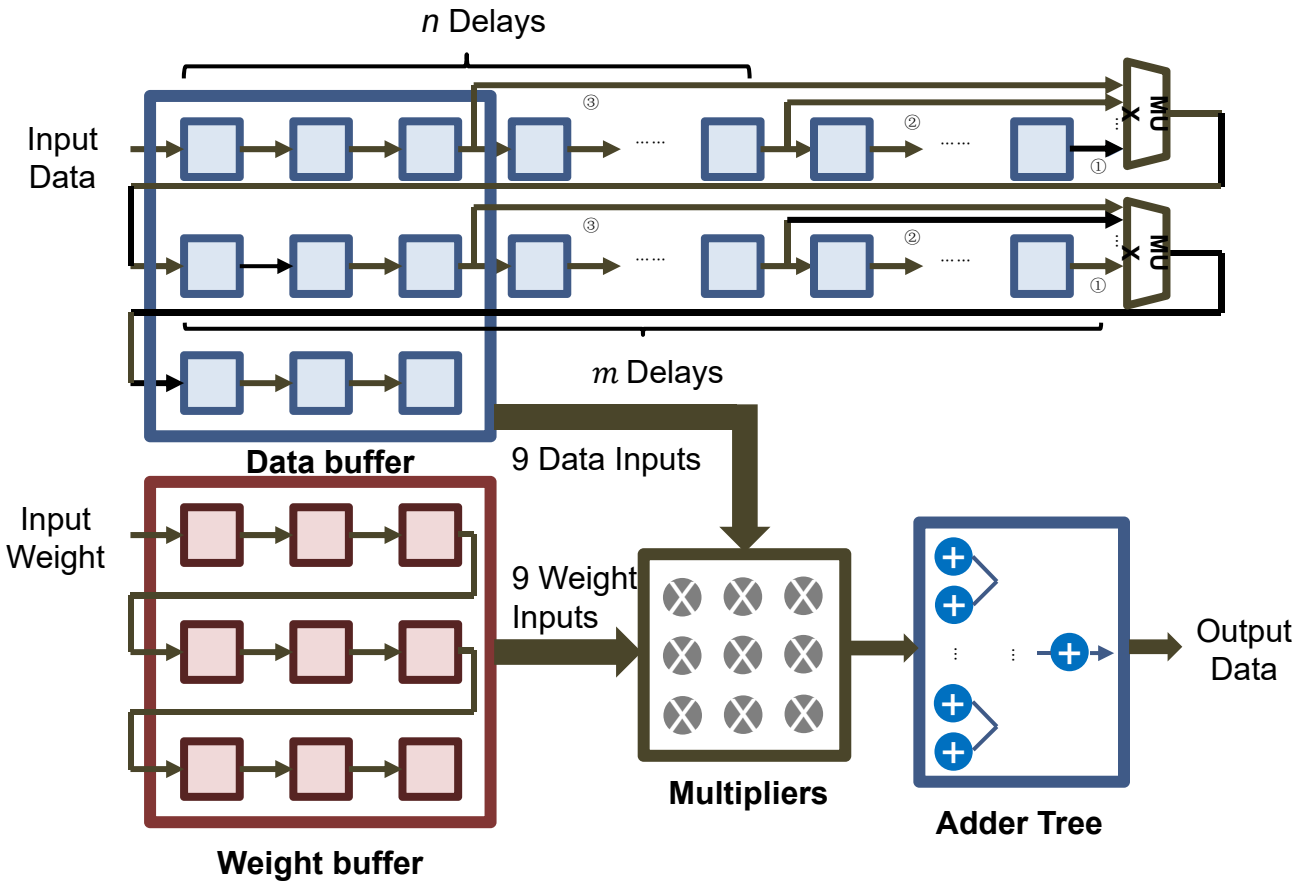
- Processing Engine Architecture



- Achieve intra-output parallelism by placing multiple Convolvers
- Convolver: optimized for 3x3 convolution operation
- Adder Tree: sum up results of one convolution operation
- NL: supports non-linear function (ReLU)
- Pool: supports max-pooling
- Bias Shift & Data Shift: support dynamic-precision fixed-point numbers

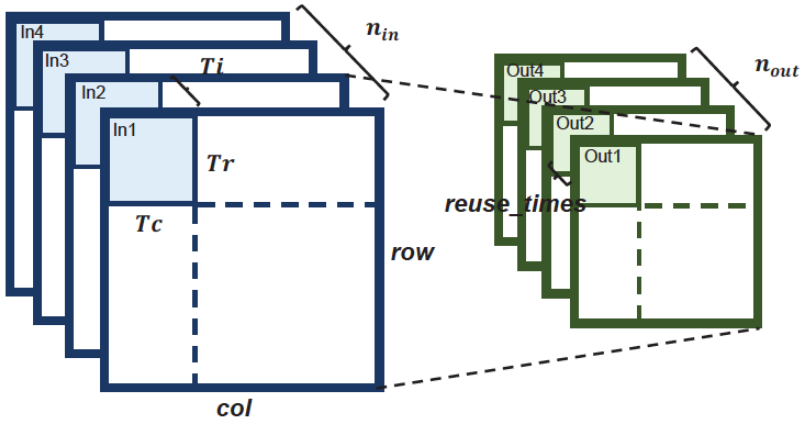
Architecture and Implementation Details

- Line-buffer design
  - Optimized for 3x3 Convolver
  - Supports operator-level parallelism



Architecture and Implementation Details

- Tiling and Data Reuse Strategy



- Using Convolver for FC layers

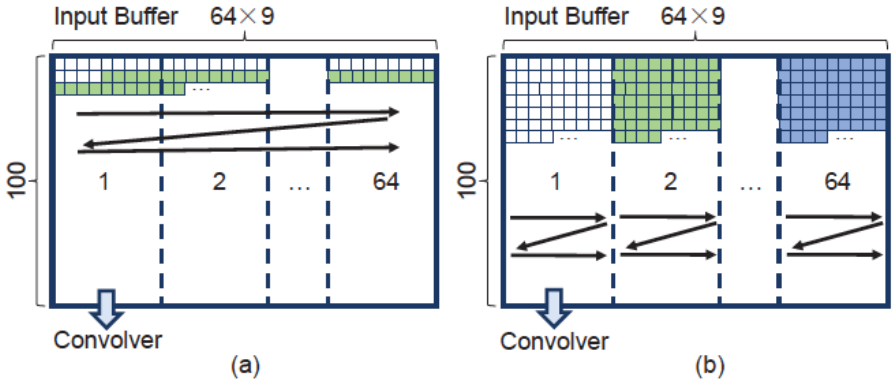


Figure 9: Data arrangement in external memory: (a) Linear arrangement; (b) DMA-oriented arrangement.

- FC layers are bandwidth-bounded
- Convolvers are enough to compute FC layers
- Save resource to accelerate Conv layers

Performance Comparison

- Performance and Energy Efficiency Comparison

	Chakaradhar 2010	Gokhale 2014	Zhang 2015	Ours
Platform	Virtex 5 SX240t	Zynq XC7Z045	Virtex7 VX485t	Zynq XC7Z045
Clock (MHz)	120	150	100	150
Bandwidth (GB/s)	-	4.2	12.8	4.2
Quantization	48-bit fixed	16-bit fixed	32-bit float	16-bit fixed
Problem Complexity (GOP)	0.52	0.552	1.33	30.76
Performance(GOP/s)	16	23.18	61.62	136.97 (Overall) 187.89 (Conv)
Power (W)	14	8	18.61	9.63
Power Efficiency (GOP/J)	1.14	2.90	3.31	14.22 (Overall) 19.50 (Conv)

Angel-Eye V2

- Similar overall architecture
- Fully parameterized design
  - Supports different data quantization settings
  - Supports different PE and Convolver number
- Supports different Conv kernel size
- User-friendly compiler
- Fine-grained instructions
  - Increase the flexibility of compiler
  - More optimization in compiler back-end

Platform	Performance	Power	Price
Angel-Eye V2 on Xilinx 7020	30GOP/S	~2W	~40 dollar
Nvidia Tegra K1	60-90GOP/S	~10-15W	199 dollar

Angel-Eye V2: Face Det + Alig

- Overall Flow

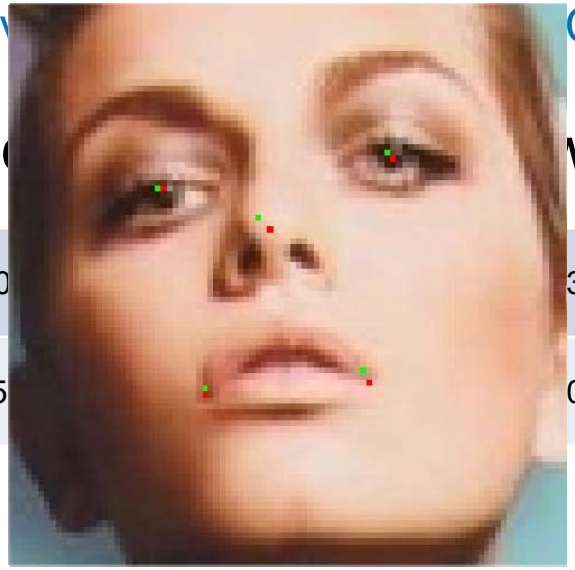
RGB image

Haar-like feature on ARM

9-layer CNN in FPGA

16Conv3x3 -> 16Conv3x3 -> 32Conv3x3 -> 32Conv3x3 -> 48Conv3x3 -> 64Conv3x3 -> 64Conv Conv3x3 -> FC10

- 8-bit dynamic-pre vithout fine-tuning



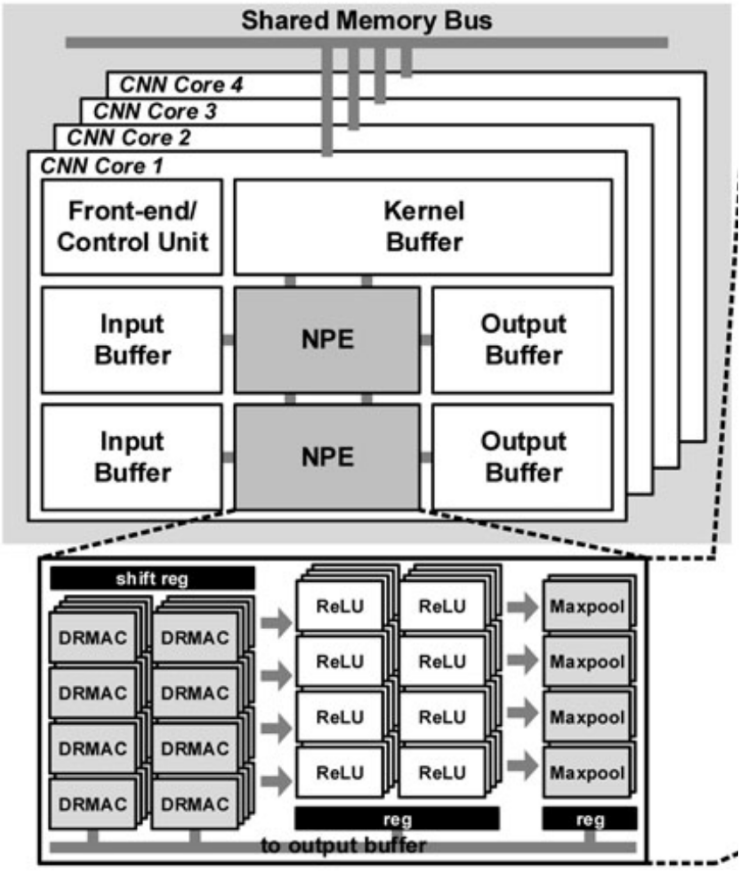
Fixed-point Network	27.3794	32.0	39	33.6290	67.4658	62.2705	
Original Network	26.0281	31.5	01	33.8175	66.2999	61.1216	

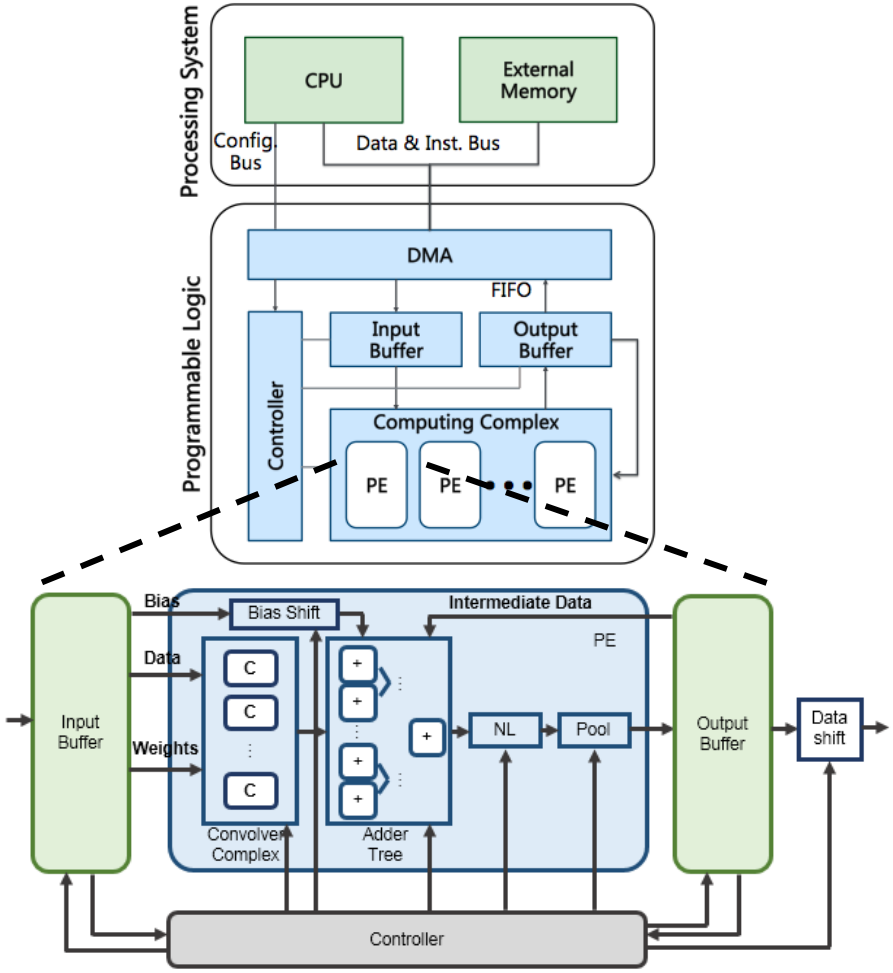
Contents

- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

Open question: Computation Granularity

- Computer Engine Architecture Comparison
  - [KAIST ISSCC2016] and ours

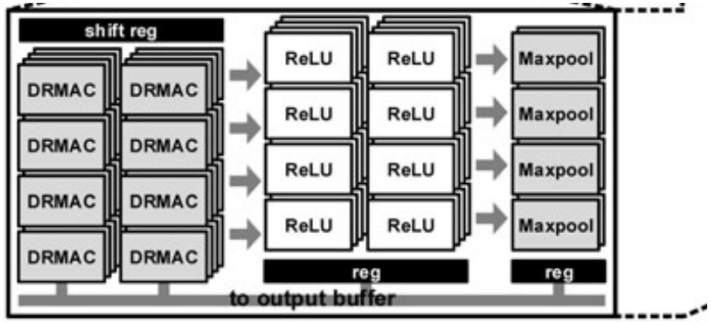




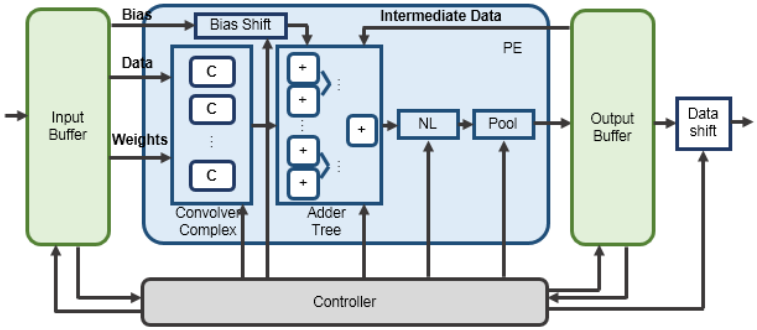
Few complex compute elements

Open question: Computation Granularity

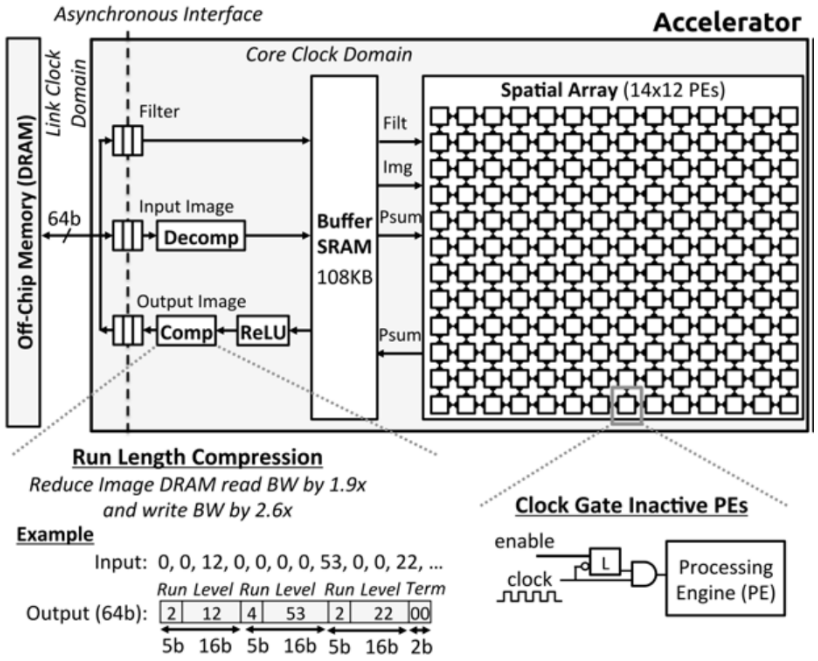
- Computer Engine Architecture Comparison



[KAIST ISSCC2016]



[Ours]



[MIT ISSCC2016]

Few complex PEs VS. Many simple PEs

Guess: Neural networks are highly predictable and serial. Few complex PEs can better utilize these characters.

Conclusion

- Deep Learning: Mainstream in AI
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity

Thanks!
Q&A