Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks

Chen Zhang, Zhenman Fang, Peipei Zhou $et\ al.$

Presented by Zhuangwei Zhuang

South China University of Technology

October 9, 2016

Content

In the recent years, convolutional neural networks (CNN) is becoming popular for its high accuracy in compute vision task, including face recognition, image and video processing, etc.

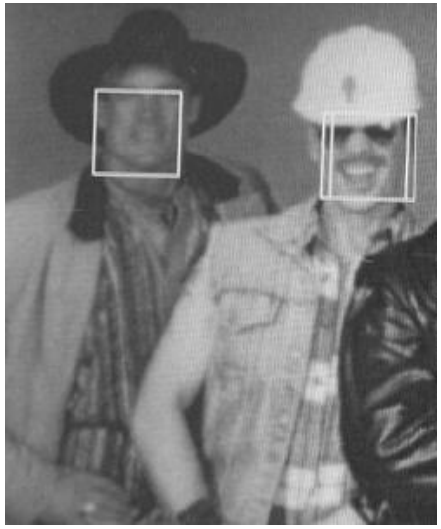


Figure: Face Detection

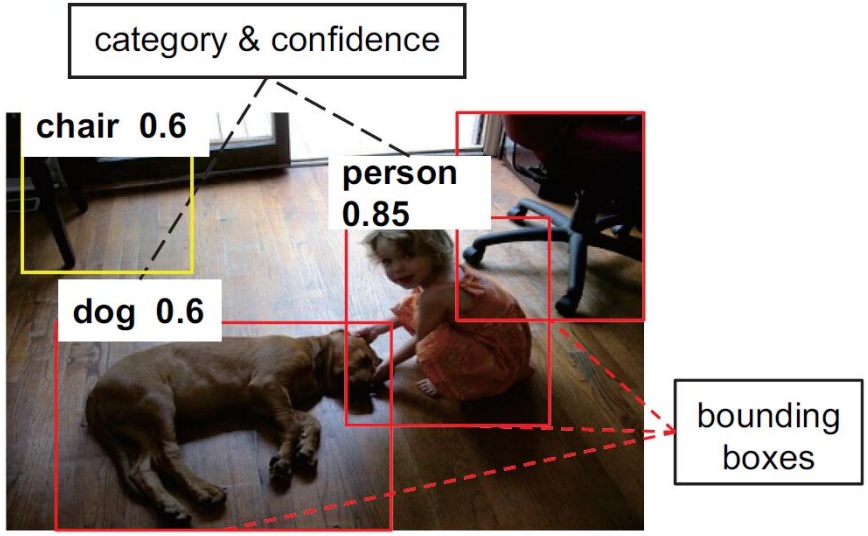


Figure: Classification

Introduction

Convolutional Neural Networks

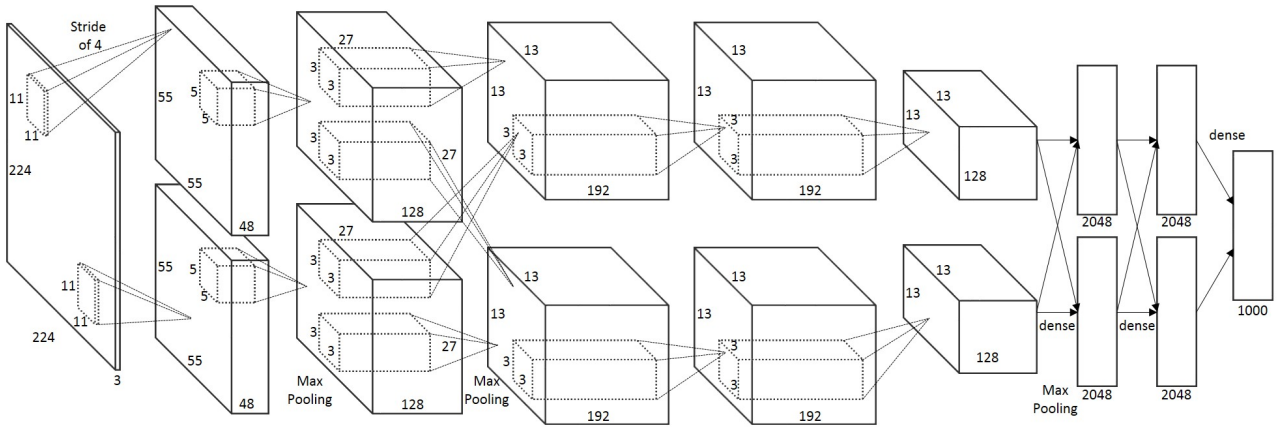


Figure: A real-life CNN model

CNN Models

- VGG16
- AlexNet
- GoogLeNet

Introduction

Convolutional Neural Networks

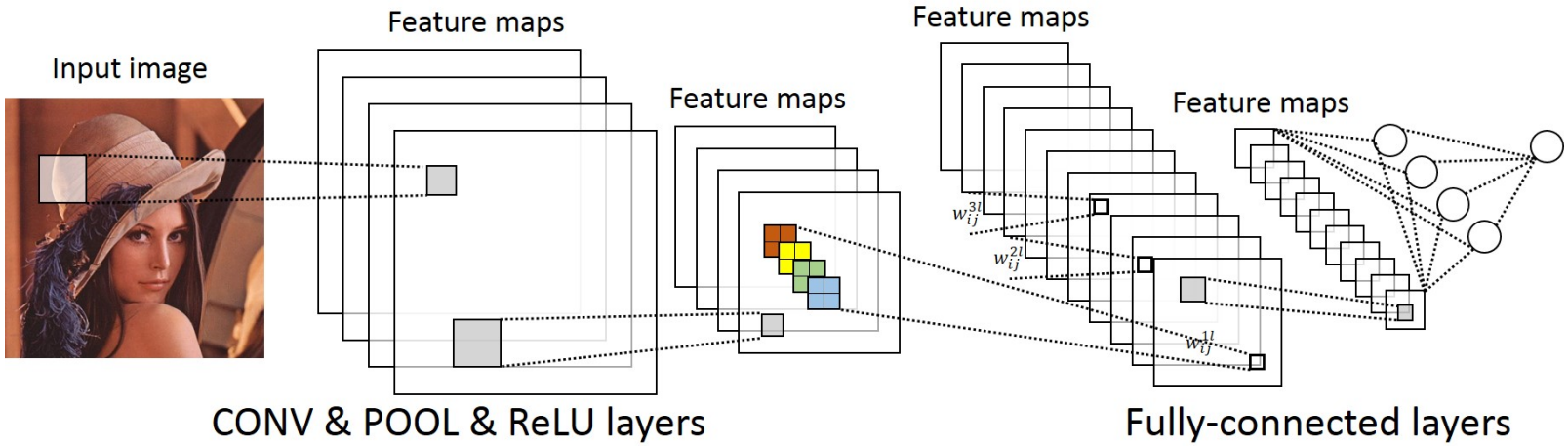


Figure: Inference phase in CNN

Architecture

- Convolutional layers(CONV)
- Pooling layers(POOL)
- Activation layers(ReLU)
- Fully-connected layers(FCN)

Hardware platforms for CNN accelerator: GPU, FPGA, ASIC.

Advantages of FPGA

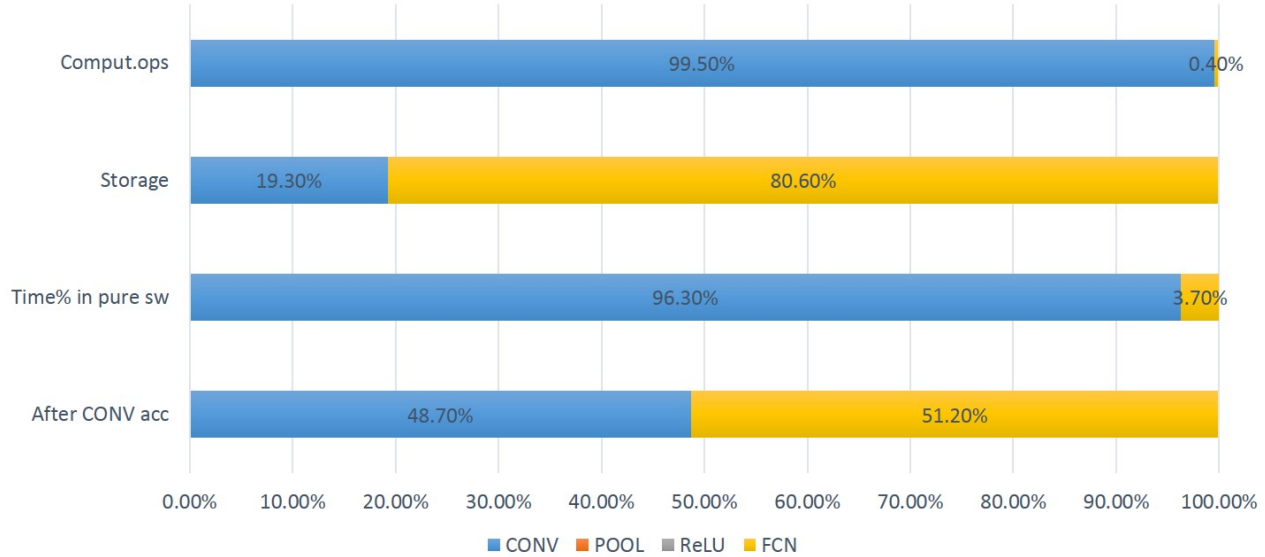
- Low power
- High energy efficiency
- Reprogrammability

Constraints of FPGA

- Limited computation resource
- Limited on-chip memory
- Limited external-memory bandwidth

Motivation

Analysis of Real-Life CNN

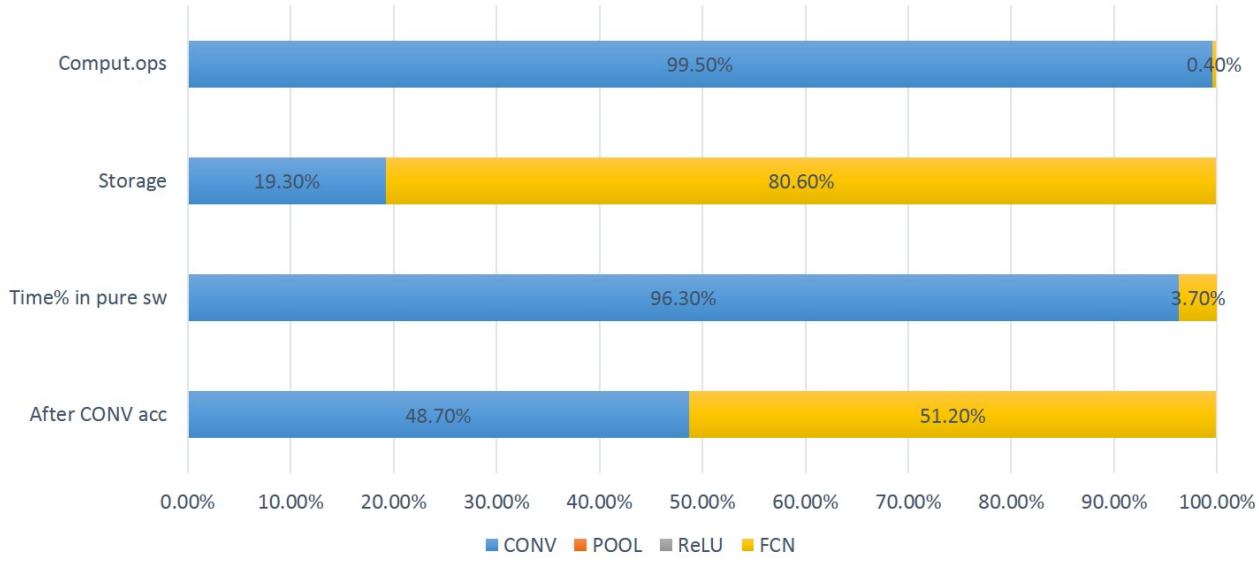


	CONV	POOL	ReLU	FCN
Comput.ops(10^7)	3E3(**99.5%**)	0.6(0%)	1.4(0%)	12.3(0.4%)
Storage(MB)	113(19.3%)	0(0%)	0(0%)	471.6(**80.6%**)
Time% in pure sw	96.3%	0.0%	0.0%	3.7%
After CONV acc	48.7%	0.0%	0.0%	**51.2%**

Table: Analysis of VGG16 model

Motivation

Analysis of Real-Life CNN

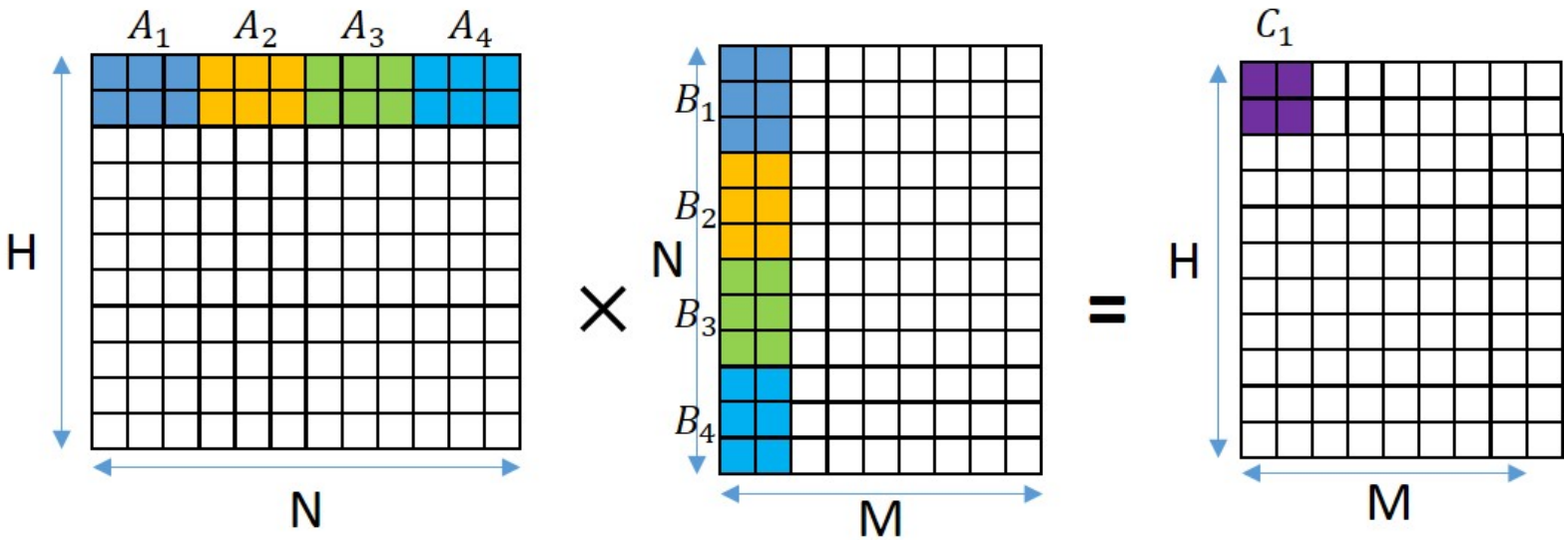


- CONV layers are computation-intensive while FCN layers are memory-intensive
- FCN layers become new bottleneck after CONV layers be accelerated
- However, most prior FPGA acceleration studies on CNN mainly focus on CONV layers in CNN

- What is the right representation for a uniformed acceleration for different layers of CNN?
- How to design and implement efficient and reusable FPGA engine for CNN?

Uniformed CNN Representation

Matrix-Multiplication



$$C_1 = A_1 \times B_1 + A_2 \times B_2 + A_3 \times B_3 + A_4 \times B_4$$

Figure: Matrix-multiplication of FCN

Uniformed CNN Representation

Input-Major Mapping

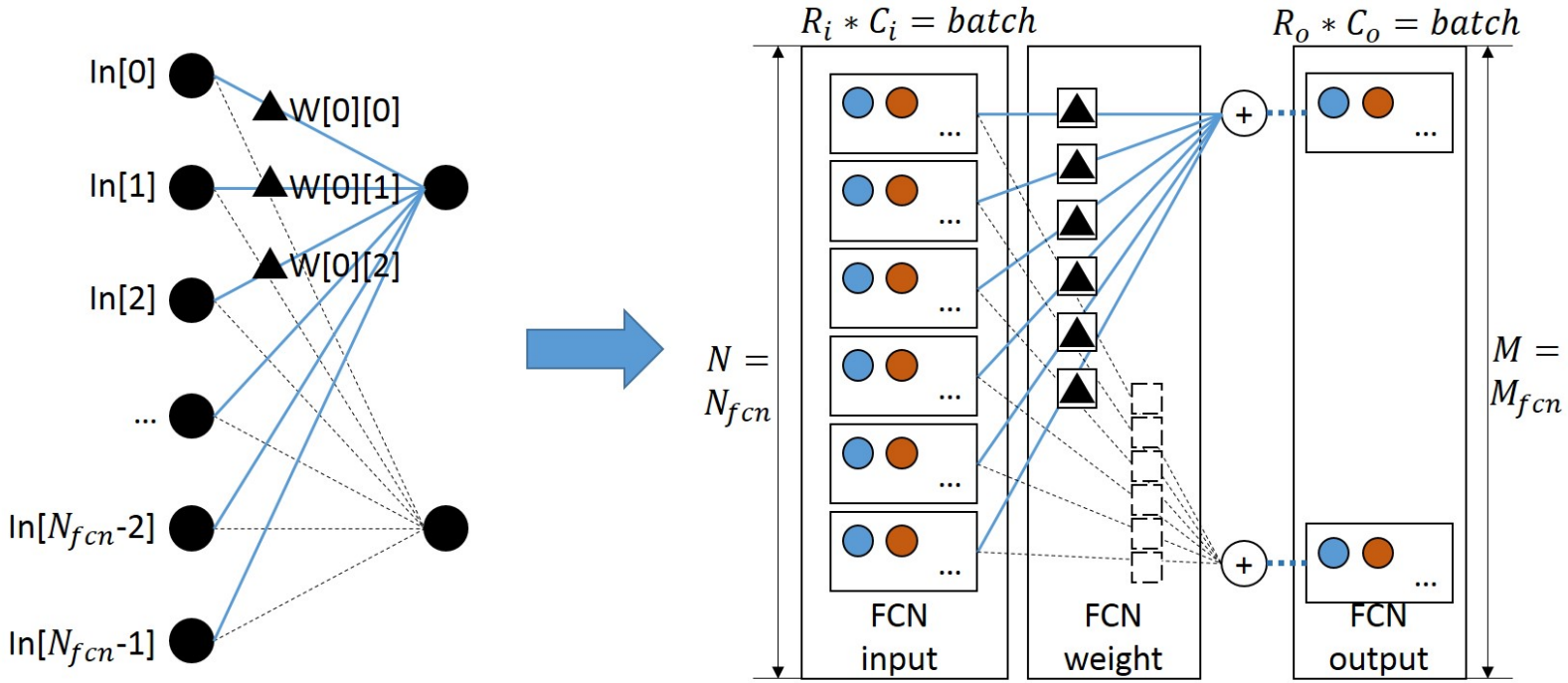


Figure: Input-major mapping with $Ker = 1$

Uniformed CNN Representation

Input-Major Mapping

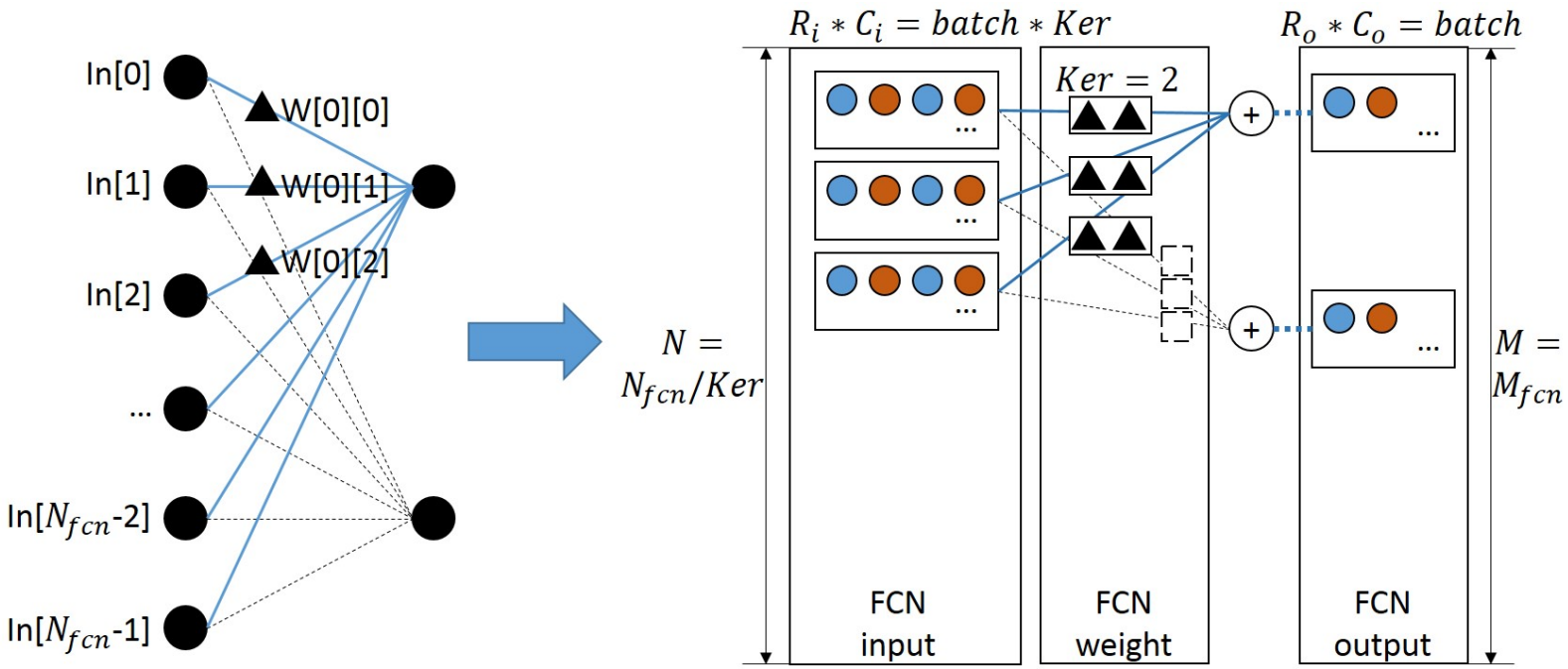


Figure: Input-major mapping with $Ker = 2$

Uniformed CNN Representation

Weight-Major Mapping

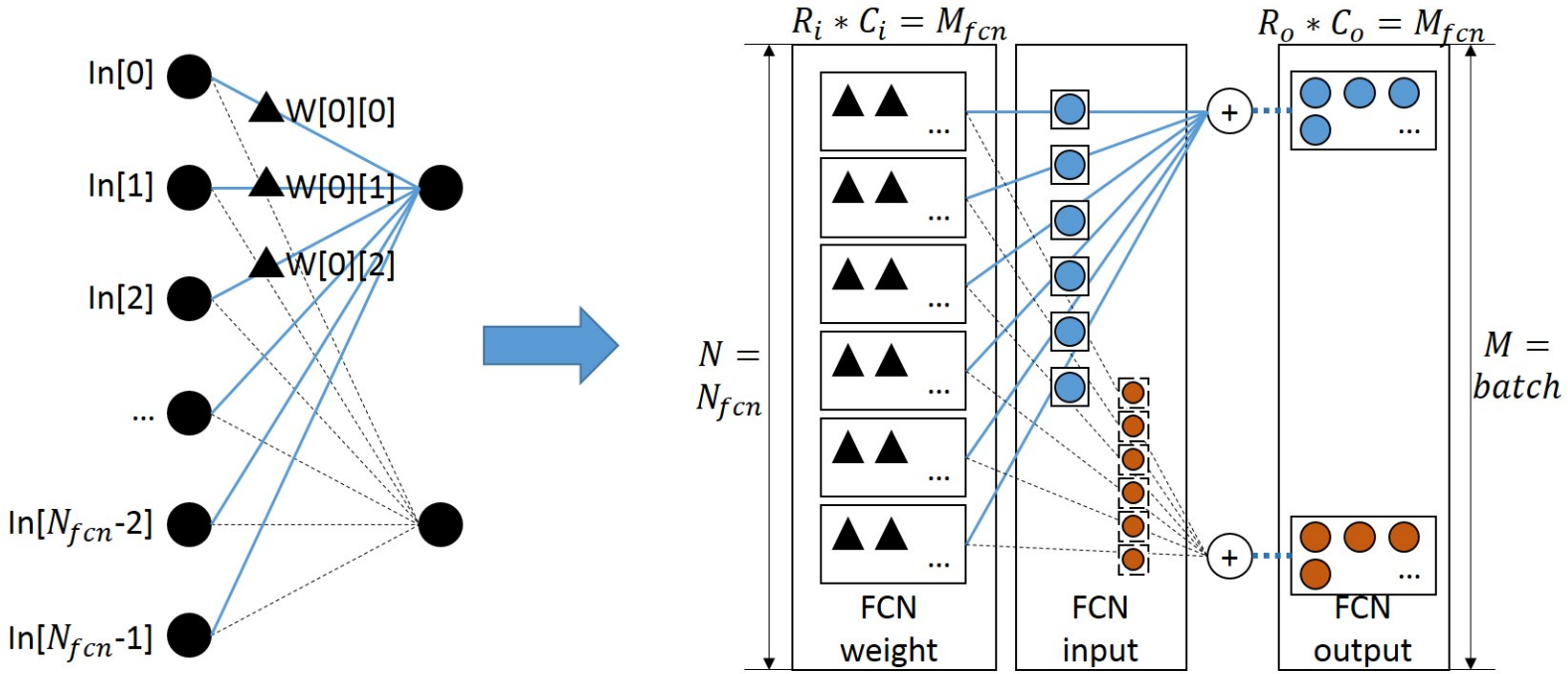


Figure: Weight-major mapping with $Ker = 1$

Uniformed CNN Representation

Weight-Major Mapping

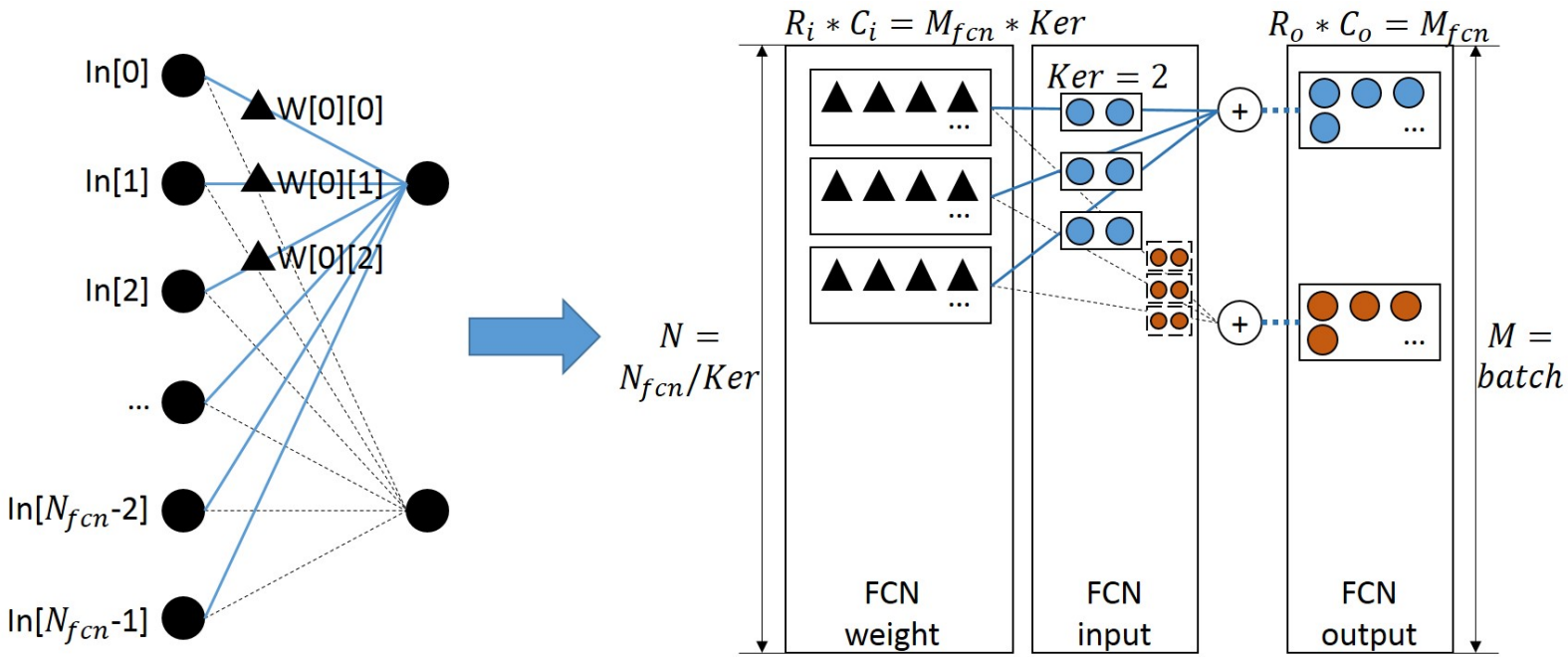


Figure: Weight-major mapping with $Ker = 2$

	Uniformed	Conv	FCN-Input	FCN-Weight
Input FM#	N	N_{conv}	N_{fcn}/ker	n_{fcn}/ker
Input FM Size	$R_i \cdot C_i$	$R_{conv}^{in} \cdot C_{conv}^{in}$	$batch \cdot ker$	$M_{fcn} \cdot ker$
Output FM#	M	M_{conv}	M_{fcn}	$batch$
Output FM Size	$R_o \cdot C_o$	$R_{conv}^{out} \cdot C_{conv}^{out}$	$batch$	M_{fcn}
Kernel Size	$K_1 \cdot K_2$	$K_1 \cdot K_2$	ker	ker
Stride	$S_1 \cdot S_2$	$S_1 \cdot S_2$	ker	ker

Table: Uniformed representation parameters for CONV, FCN input-major mapping and FCN weight-major mapping

Caffeine Design

System Overview

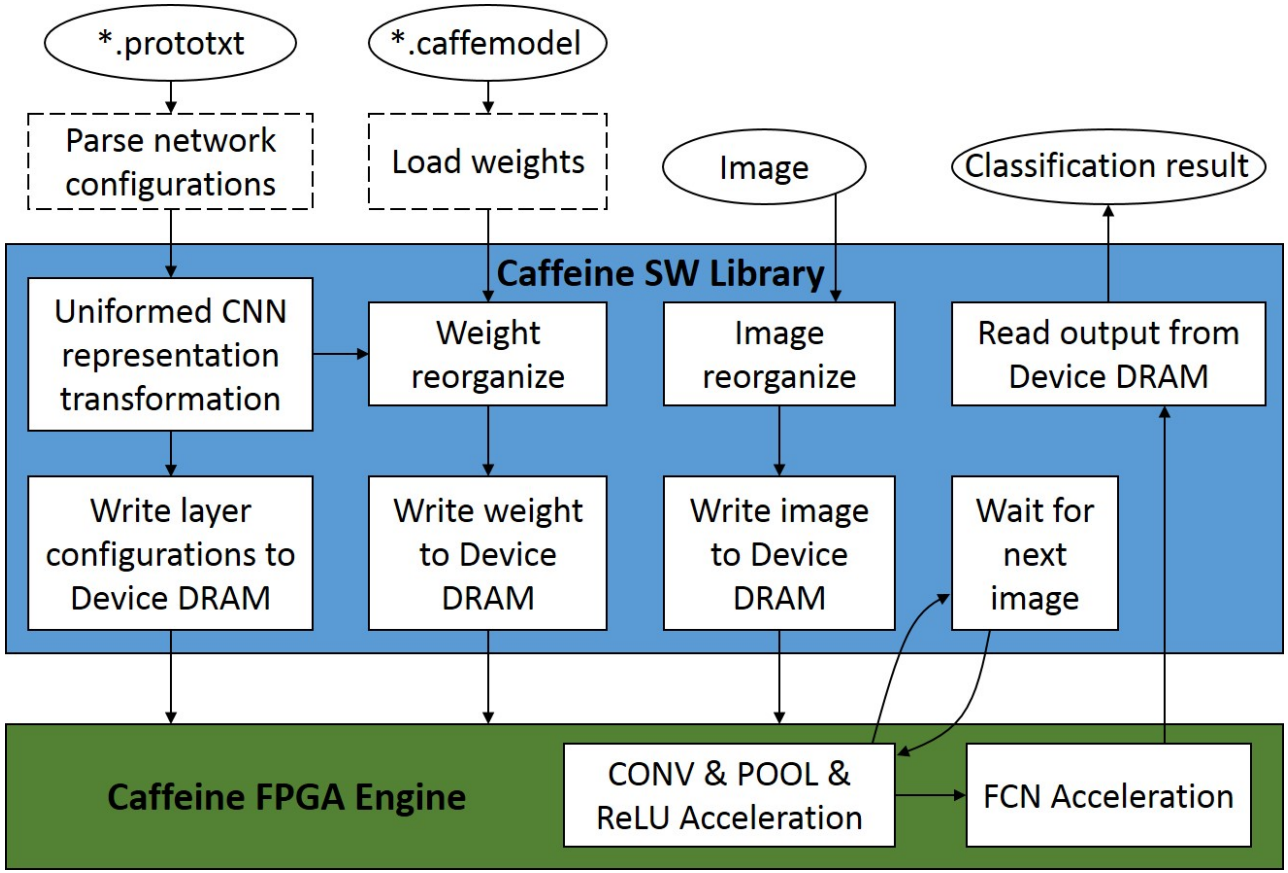


Figure: Caffe-Caffeine integration

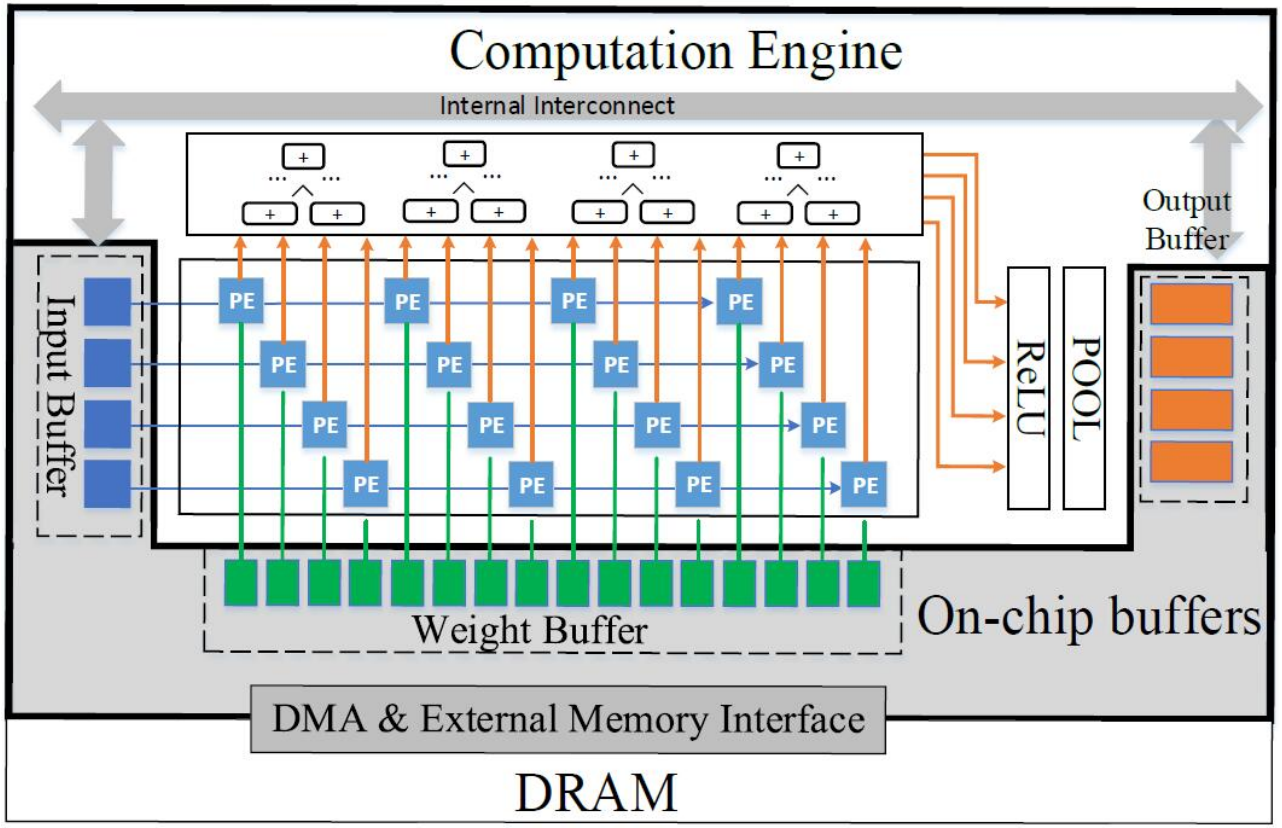


Figure: Scalable accelerator architecture design

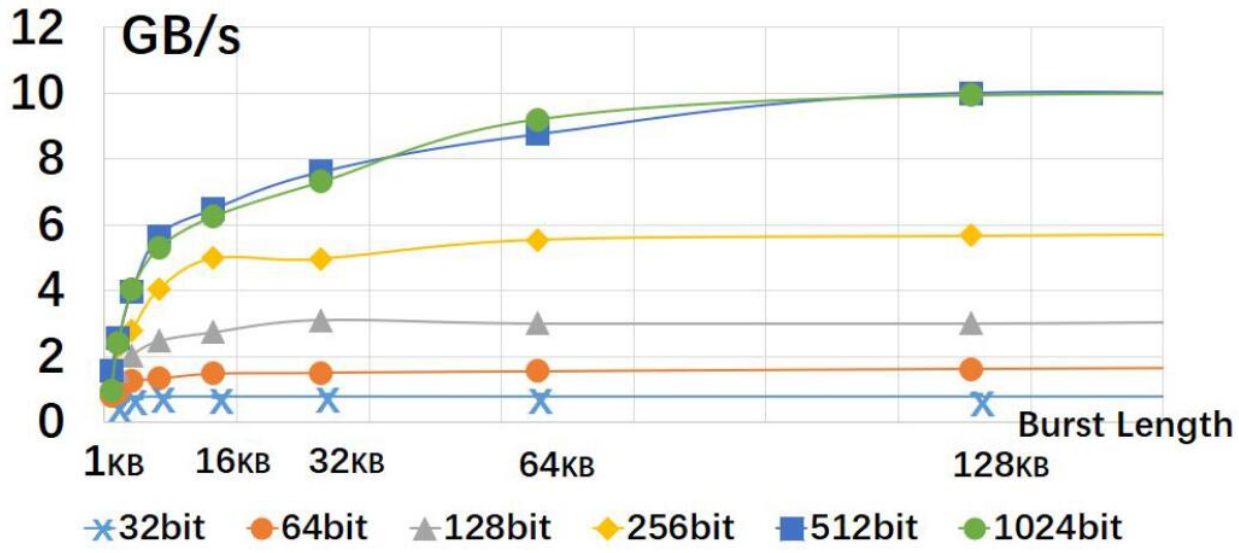


Figure: Effective FPGA DRAM bandwidth

- Effective of FPGA bandwidth goes up with the increase of burst length, and finally flatten
- Limited burst length greatly degrade actual bandwidth performance

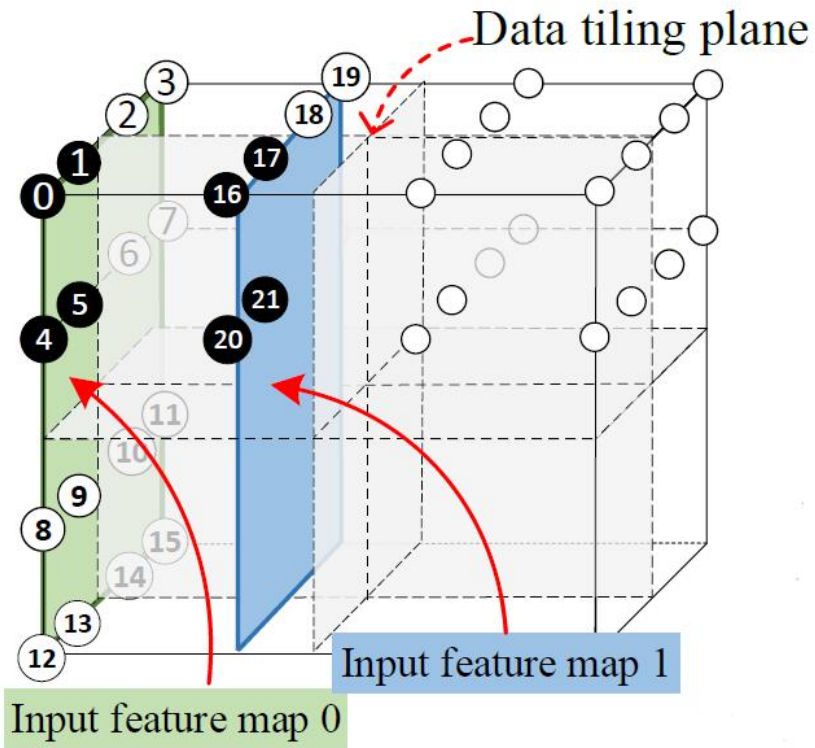


Figure: A logic 3D data layout

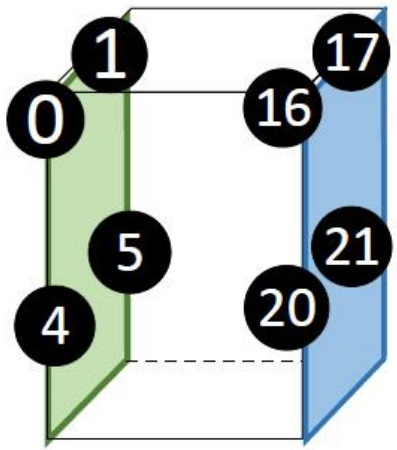


Figure: A piece of data tile

Caffeine Design
Bandwidth Optimization

Data	0	1	...	4	5	...	...	16	17	...	20	21
DRAM Addr.	x0	x4	...	x10	x14	...	...	x40	x44	...	x50	x54

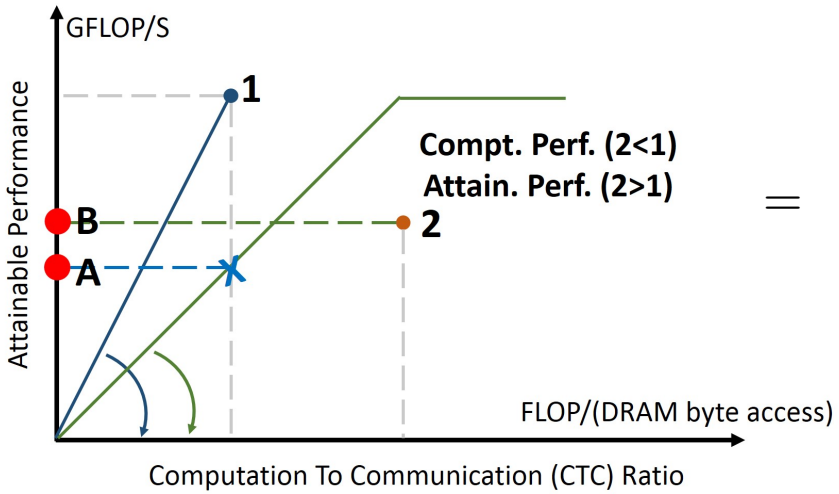
Data	0	16	1	17	4	20	5	21	...	...	...	...
DRAM Addr.	x0	x4	x8	xc	x10	x14	x18	x1c	...	...	...	...

Figure: Optimization of data layout in DRAM space

- Move data for an entire tile to a continuous space for improving burst length and bit-length
- Interleave data for different BRAM banks for reducing bank read/write conflicts

Roofline Model

Original Model



$$CTC\ Ratio = \frac{total\ number\ of\ operations}{total\ amount\ of\ DRAM\ access}$$

$$DRAM\_Access = \alpha_{in} \cdot \beta_{in} + \alpha_{weight} \cdot \beta_{weight} + \alpha_{out} \cdot \beta_{out} \quad (1)$$

- α: number of data transfer times for input/weight/output data
- β: size of input/weight/output data tile

Figure: Effective FPGA DRAM bandwidth

- Original model ignores the fact that different data volumes in each tile have different burst length and effective bandwidth

Roofline Model

Revised Model

$$
\begin{aligned}
DRAM\_Access &= \gamma_{in} \cdot \alpha_{in} \cdot \beta_{in} \\
&+ \gamma_{weight} \cdot \alpha_{weight} \cdot \beta_{weight} \\
&+ \gamma_{out} \cdot \alpha_{out} \cdot \beta_{out} \qquad (2)
\end{aligned}
$$

$$
\gamma = max\_bandwidth / f(\beta) \qquad (3)
$$

$f(\beta)$ is the effective function between bandwidth and burst length

Roofline Model

Revised Model

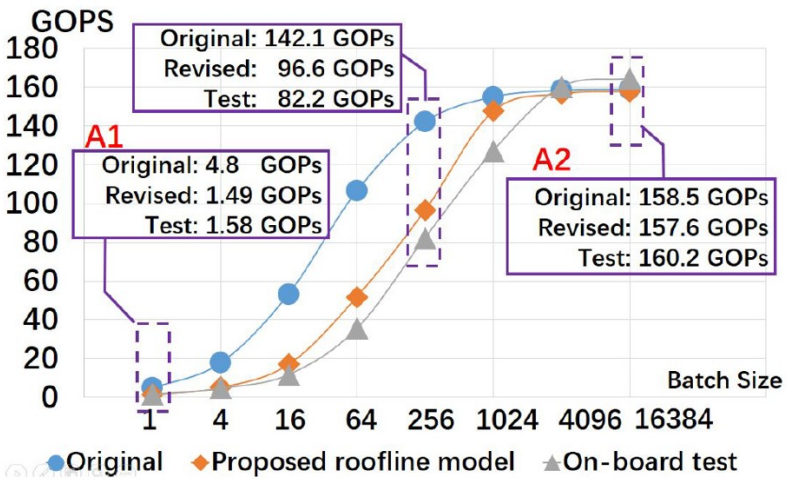


Figure: Comparison of original, revised model and on-board test result with **input-major mapping**

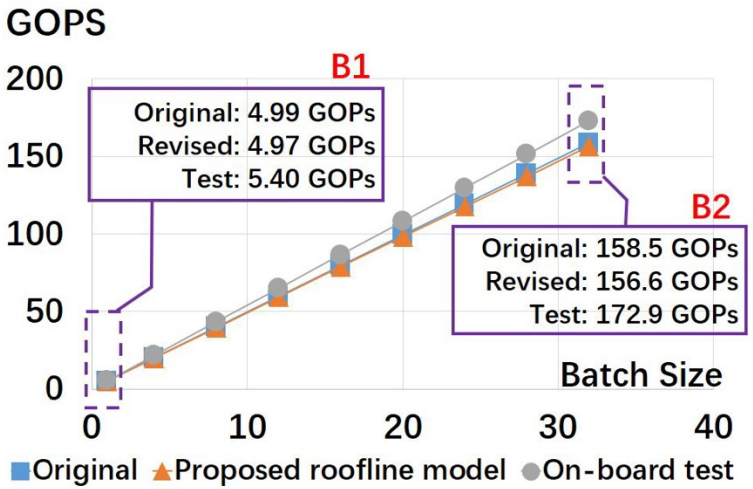
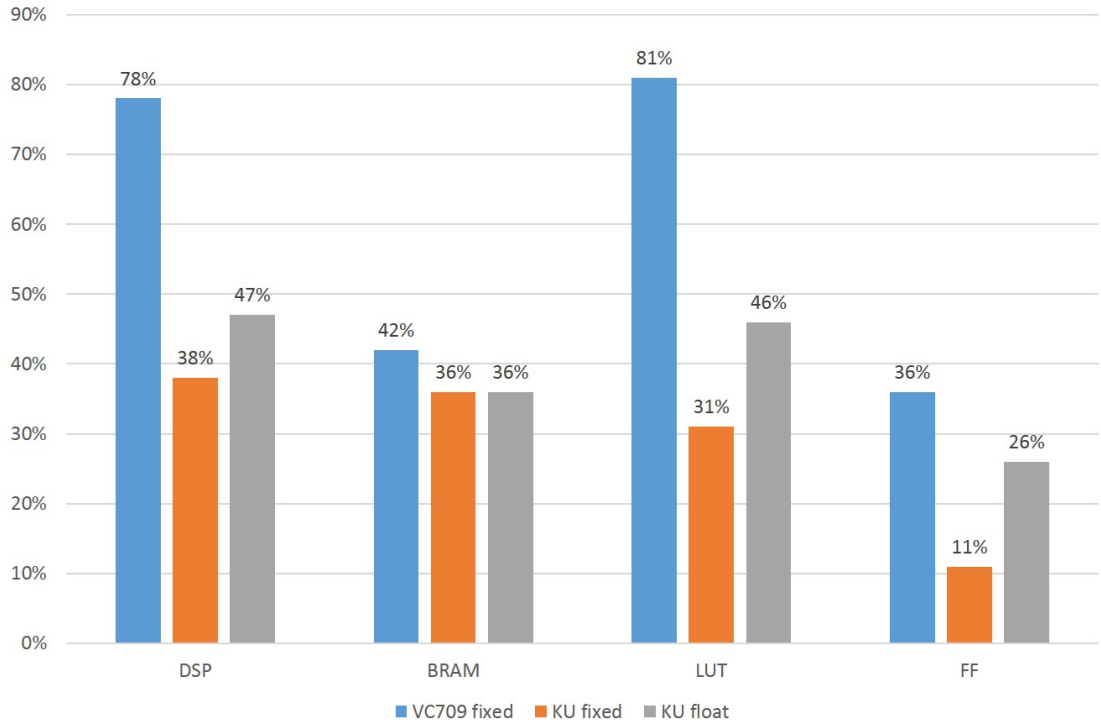


Figure: Comparison of original, revised model and on-board test result with **weight-major mapping**

- Revised model is more accurate than original model
- Weight-major mapping is better than input-major mapping in small batch size, which is required for real-time inference phase

Experiment and Result

Resource Utilization



	DSP	BRAM	LUT	FF	Freq.
VC709 fixed	2833(78%)	1248(42%)	3E5(81%)	3E5(36%)	150MHz
KU fixed	1058(38%)	782(36%)	1E5(31%)	8E4(11%)	200MHz
KU float	1314(47%)	798(36%)	2E5(46%)	2E5(26%)	200MHz

Table: FPGA resource utilization of Caffeine

Experiment and Result

Comparison with CPU/GPU

Platforms	CPU	CPU+GPU	CPU+FPGA	
Device	E5-2609	K40	KU60	VX690T
Technology	22nm	28nm	20nm	28nm
Freq.	1.9GHz	1GHz	200MHz	150MHz
Power(W)	150	250	25	26
Latency (ms/image)	733.7	15.3	101.15	65.13
Speedup	1x	48x	7.3x	9.7x
J per image	110	3.8	2.5	1.69
Energy Efficiency	1x	28.7x	**43.5x**	**65x**

Table: Comparison with CPU/GPU platforms

Experiment and Result

Comparison with CPU/GPU

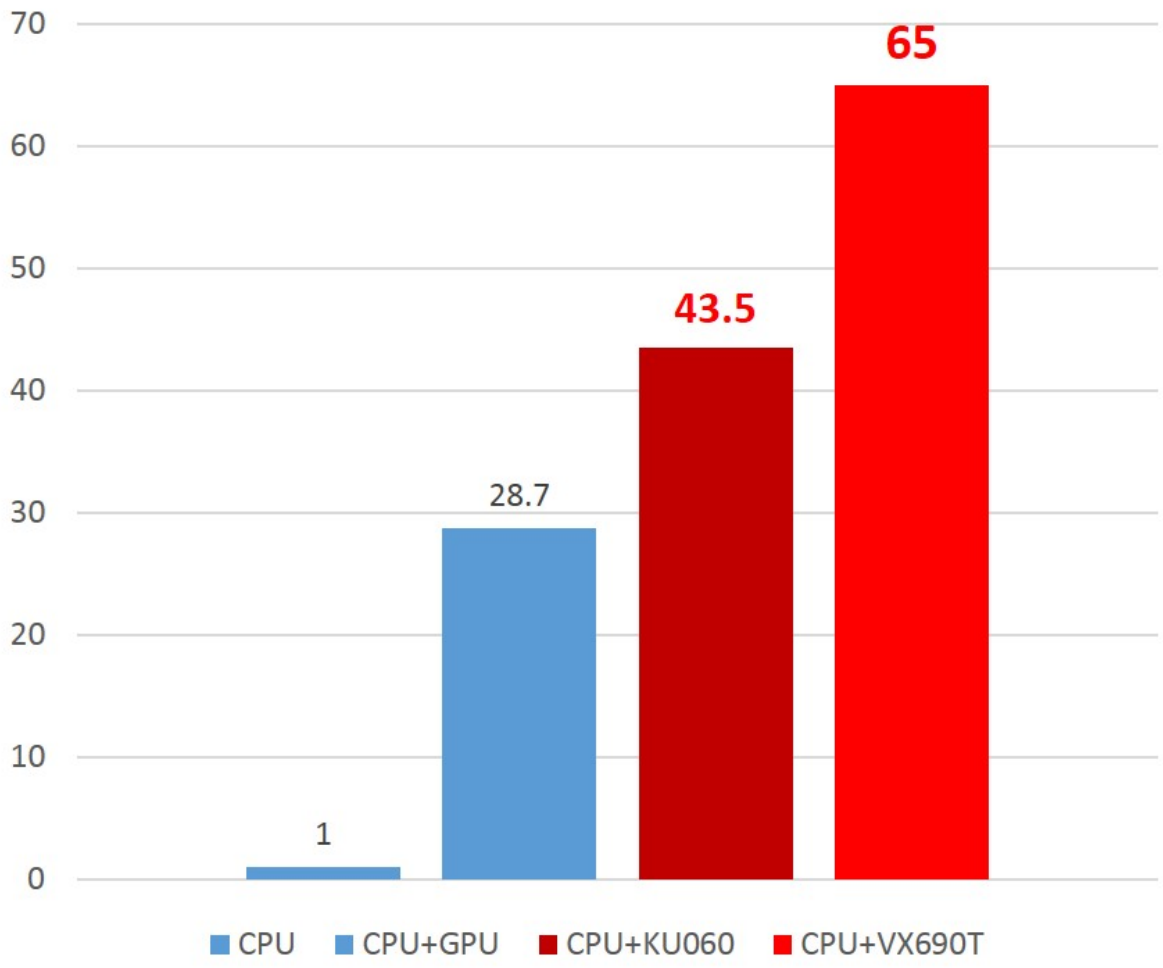


Figure: Comparison with CPU/GPU platforms

Experiment and Result
Comparison with Prior Work

	Prior Works			This Work	
CNN models	AlexNet	VGG			
Device	Virtex7 485T	Zynq XC7Z045	Stratix-V GSD8	Ultrascale KU060	Virtex7 690T
Precision	float 32bit	fixed 16bit	fixed 16bit	fixed 16bit	fixed 16bit
Numbers of DSP	2240	780	1963	1058	2833
CONV (peak) GOPS	83.8	254.8	-	365	636
CONV (overall) GOPS	61.6	187.8	136.5	310	488
FCN (overall) GOPS	-	1.2	-	173	170
CONV+FCN GOPS	-	137	117.8	266	354

Table: Comparison with other FPGA work

Experiment and Result

Comparison with Prior Work

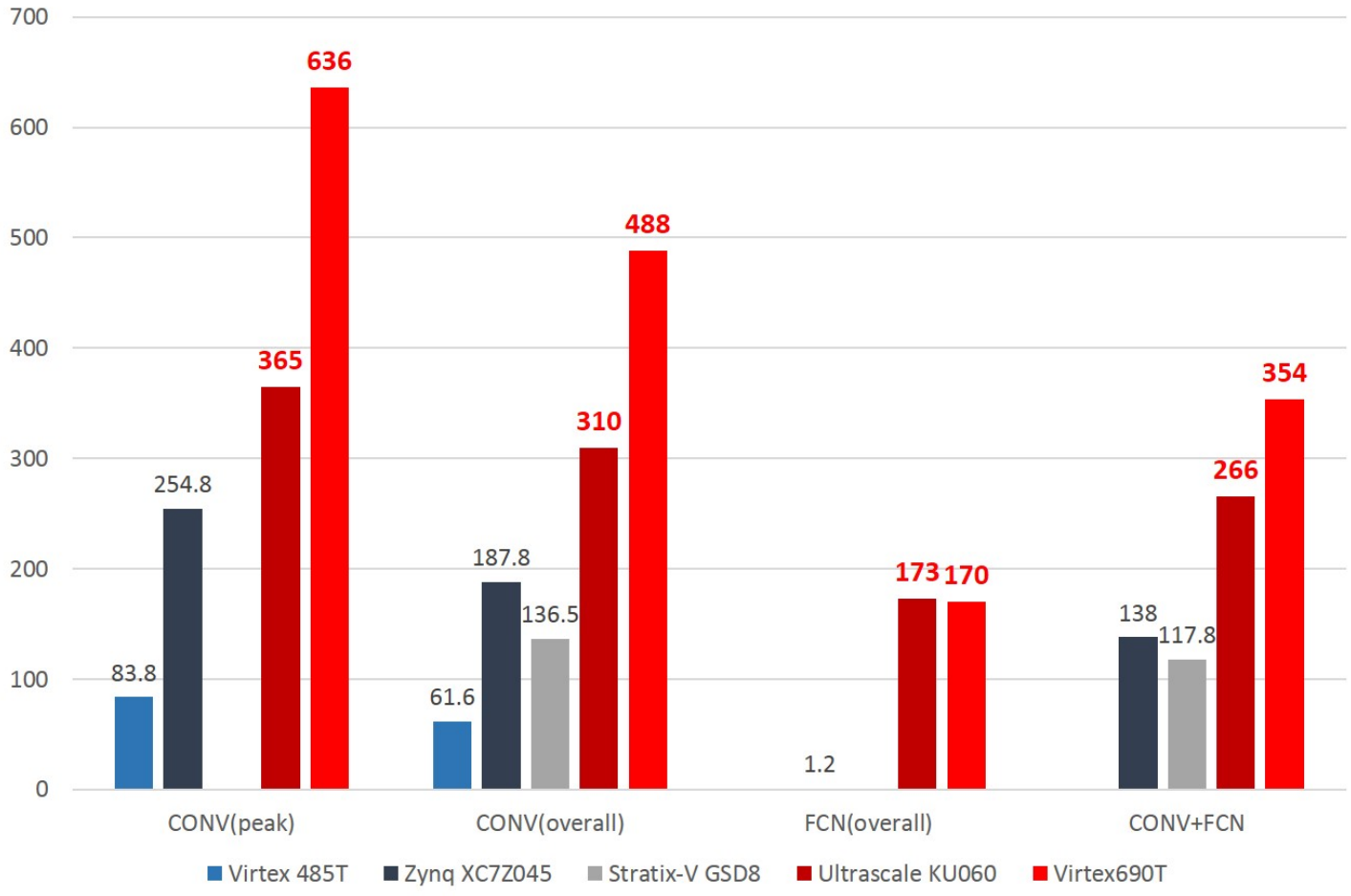


Figure: Comparison with other FPGA work

Conclusion

Contribution

- Proposed a **uniformed** convolutional MM **representation** for CNN layers
- Designed and implemented **Caffeine**

Result

- Achieved **365** GOPS on KU060 and **636** GOPS on VC707
- Achieved **7.3x** and **43.5x** performance and energy gains over a 12-core CPU and **1.5x** better energy-efficiency over GPU on KU060

THANK YOU

Q & A?