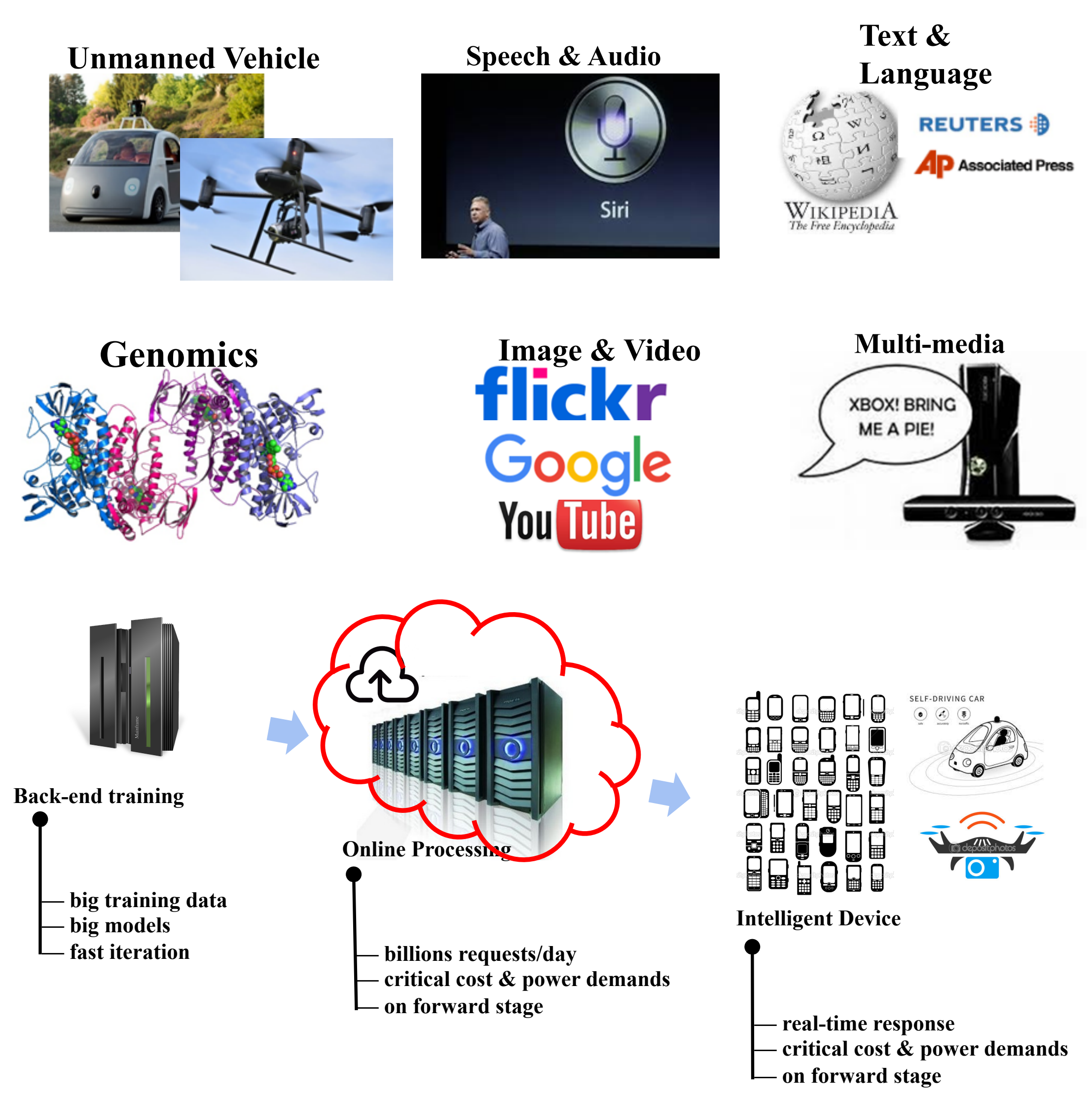
Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks

Chen Zhang, Zhenman Fang, **Peipei Zhou**, Peichen Pan, Jason Cong

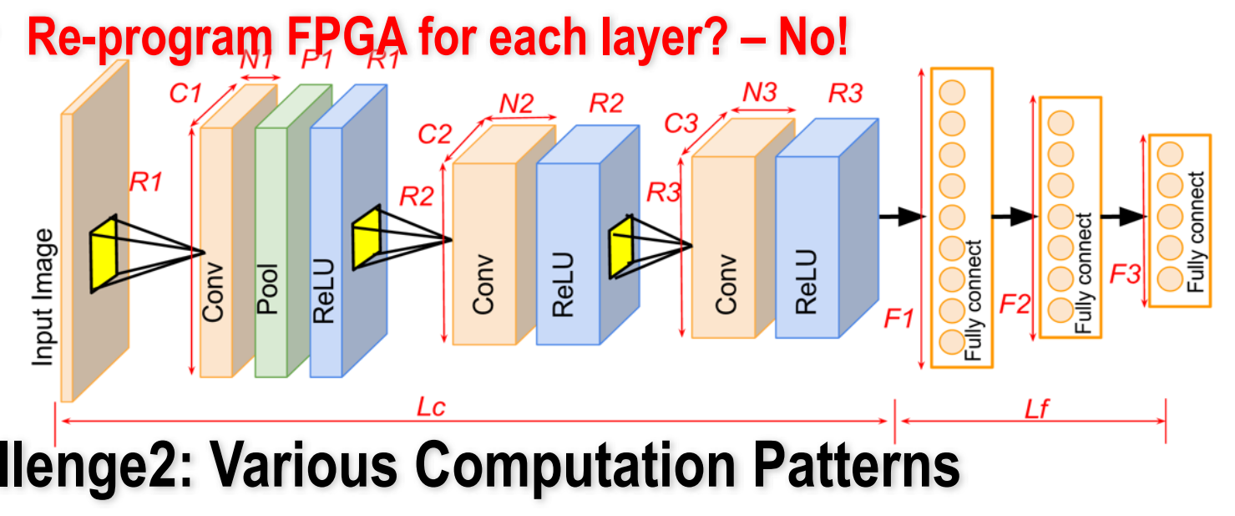
Center for Energy-effcient Computing and Applications, Peking University, Beijing, China
Center for Domain-Specific Computing, University of California, Los Angeles, US
Falcon-computing Solutions Inc., Los Angeles, US

Deep learning & applications



Challenge for CNN/DNN accelerator

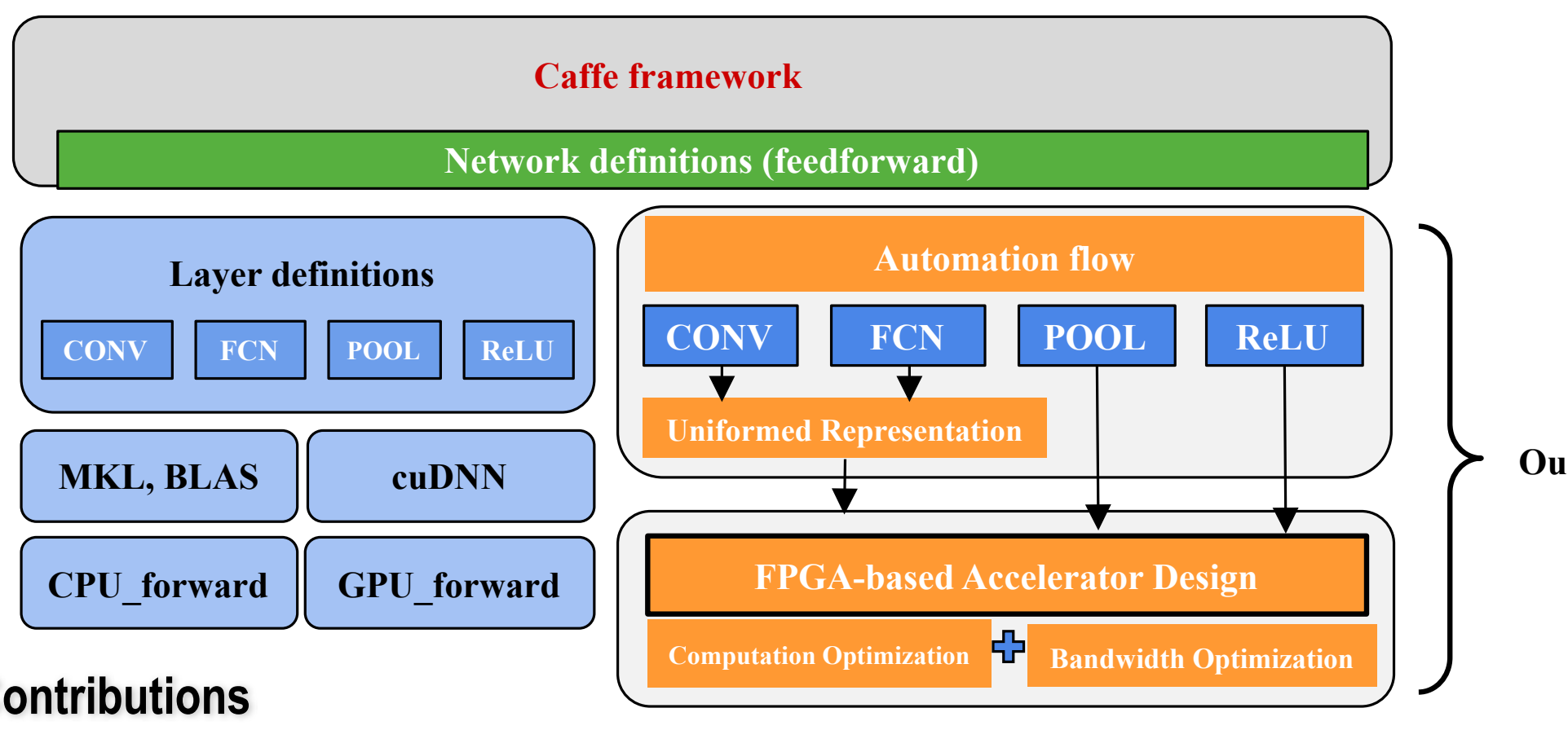
- Challenge1: Programmability
  - Network definitions (& layer parameters) change according to applications
    - CNNs are getting deeper, more precision achieved with deeper network
    - Network parameters vary across layers, many user definable parameters
  - Hardware has limited programmability
    - Re-synthesize bitstream for each network? – Not preferred
    - Re-program FPGA for each layer? – No!



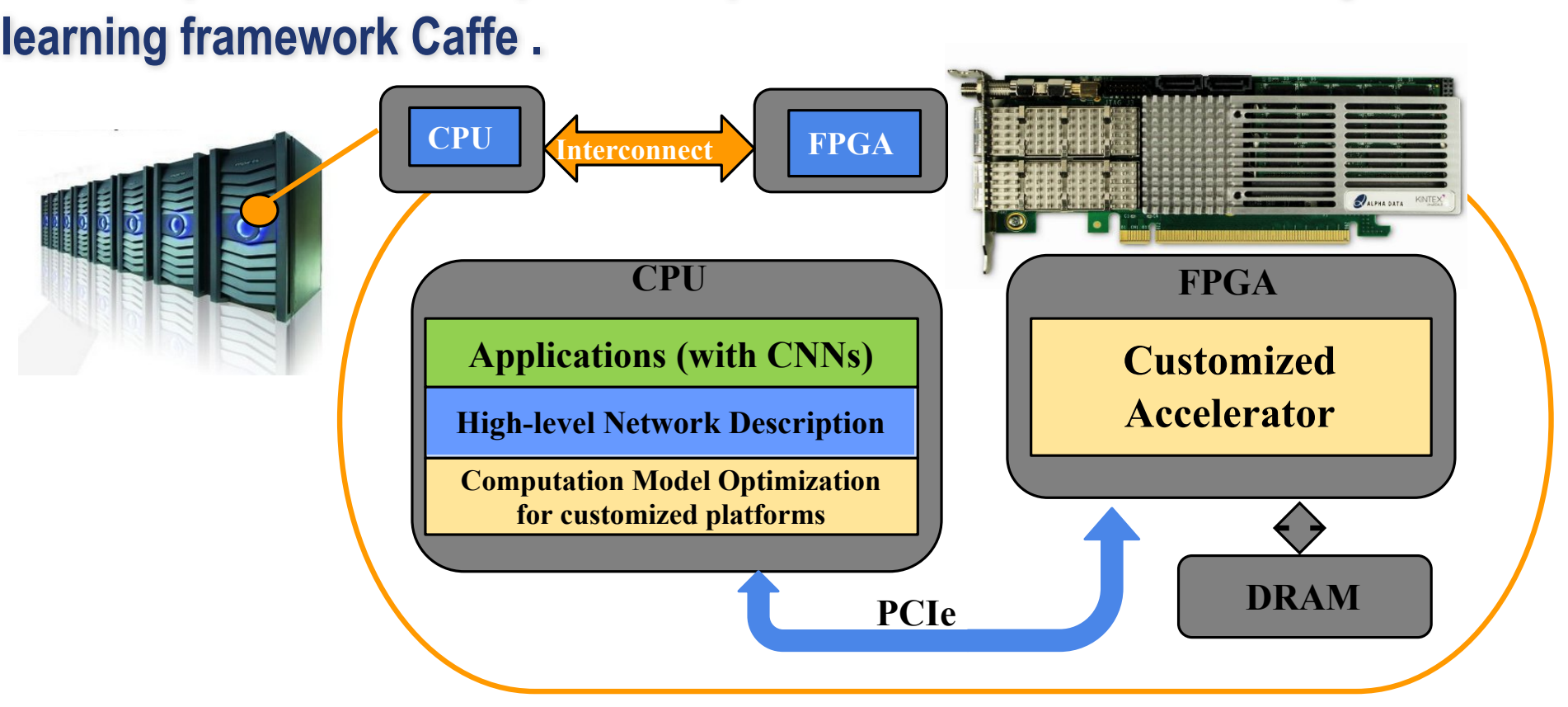
- Challenge2: Various Computation Patterns
  - Convolution layer (CONV) & Fully connected layer (FCN)
    - COV-- Compute Intensive, FCN-- Communication Intensive
  - Hardware has limited bandwidth resource
    - Convolution layer – extensive previous study
    - Fully Connected layer -- How to maximize bandwidth utilization?
    - CONV + FCN How to re-use hardware for both kernels?

	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%**

Caffeine: SW/HW Co-designed FPGA-based deep learning library

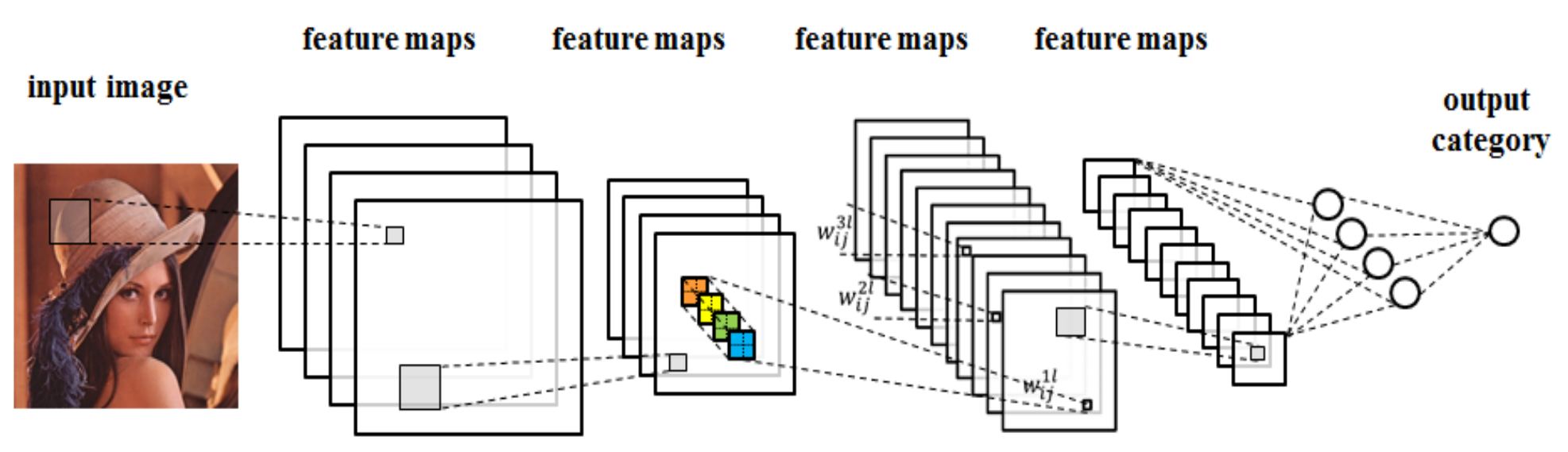


- Contributions
  - A uniformed convolutional MM representation adapted for efficient FPGA acceleration of both CONV and FCN layers in CNN/DNN.
  - A HW/SW co-designed efficient and reusable CNN/DNN engine Caffeine, where the FPGA accelerator maximizes the computing and bandwidth resource utilization.
  - The first published attempt to incorporate FPGAs into the industry-standard deep learning framework Caffe .


CPU
Interconnect
FPGA
CPU
Applications (with CNNs)
High-level Network Description
Computation Model Optimization for customized platforms
FPGA
Customized Accelerator
PCIe
DRAM

Convolution layer & Fully Connected Layer

- Convolution

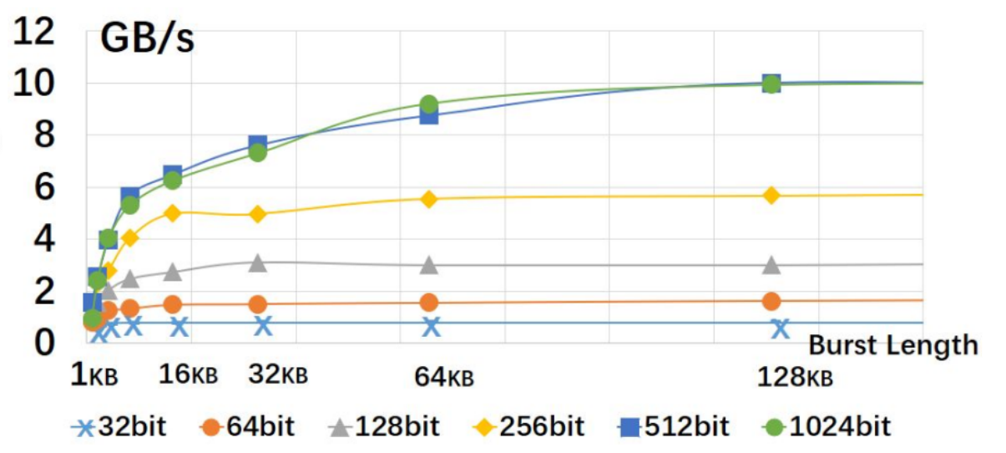


- Fully Connected Layer: Communication Intensive

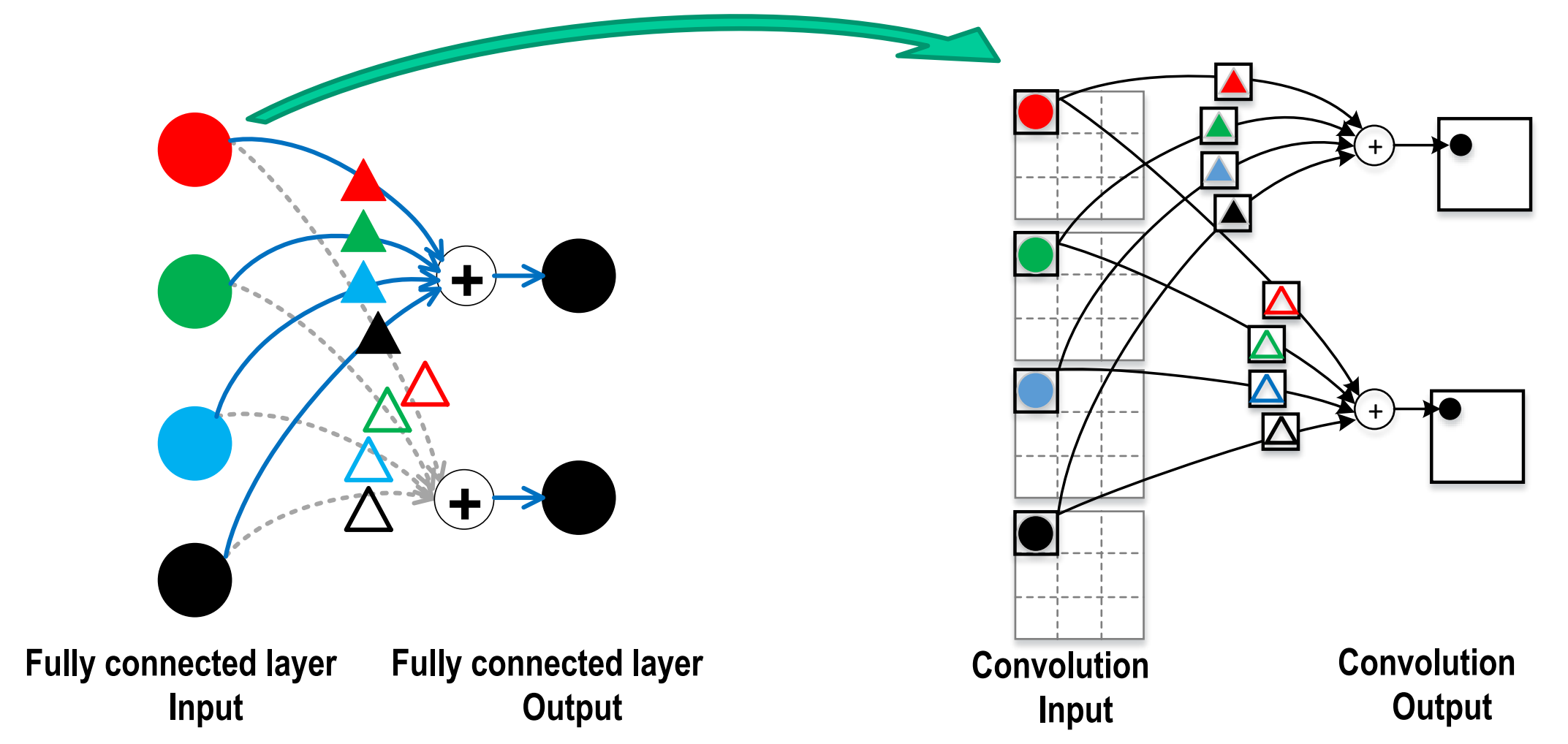
input	output	row	col	kernel	stride	Input + output + weight size (MB)	Mega Floating Operations	Comp/Data Ratio
64	64	224	224	3	1	6.16	3700	600.3

input	output	-	-	-	-	Input + output + weight size (MB)	Mega Floating Operations	Comp/Data Ratio
25088	4096	-	-	-	-	98	2100	21.4

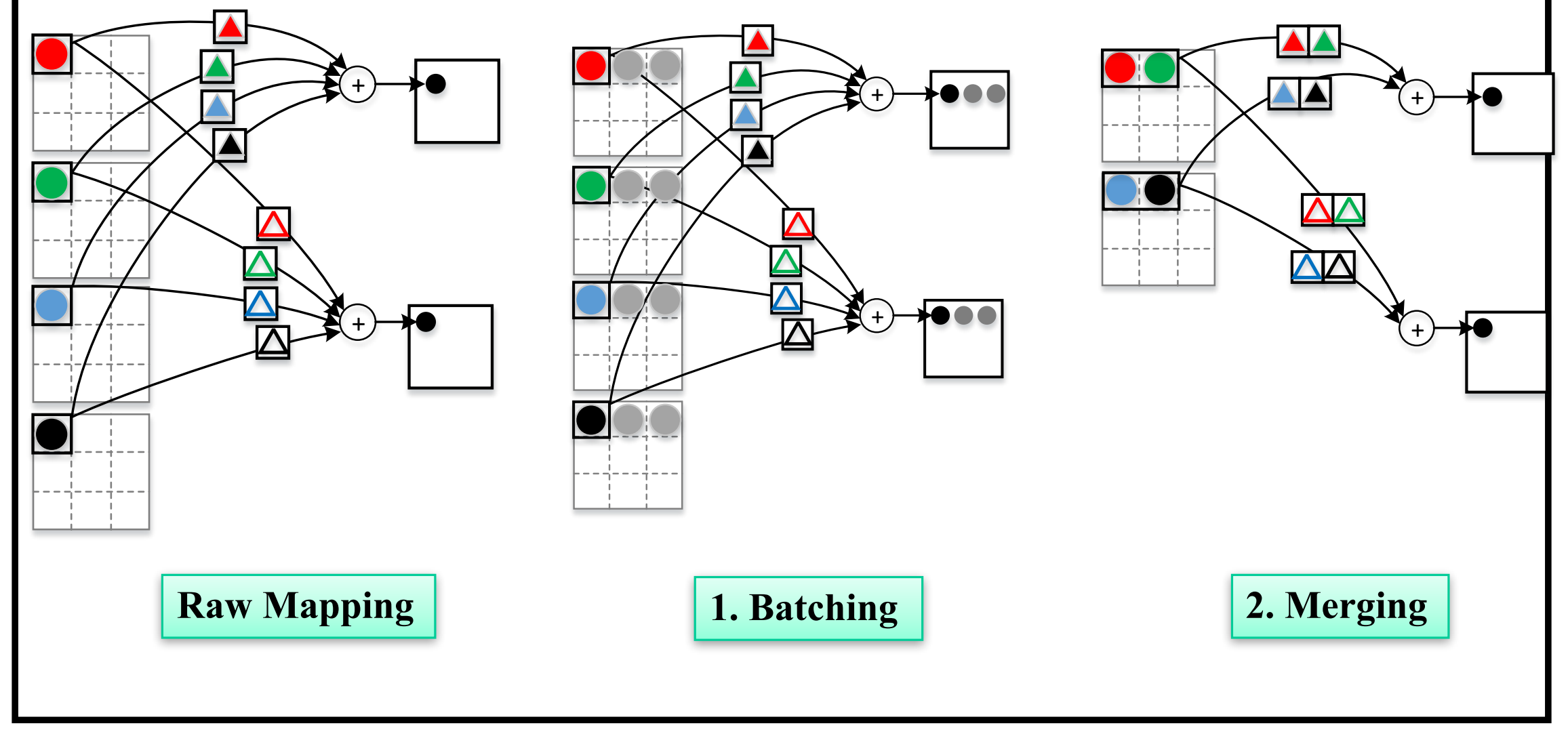
- FPGA bandwidth
  - Much smaller than GPU (10 vs 200)
  - Sensitive to burst length & bus bitwidth



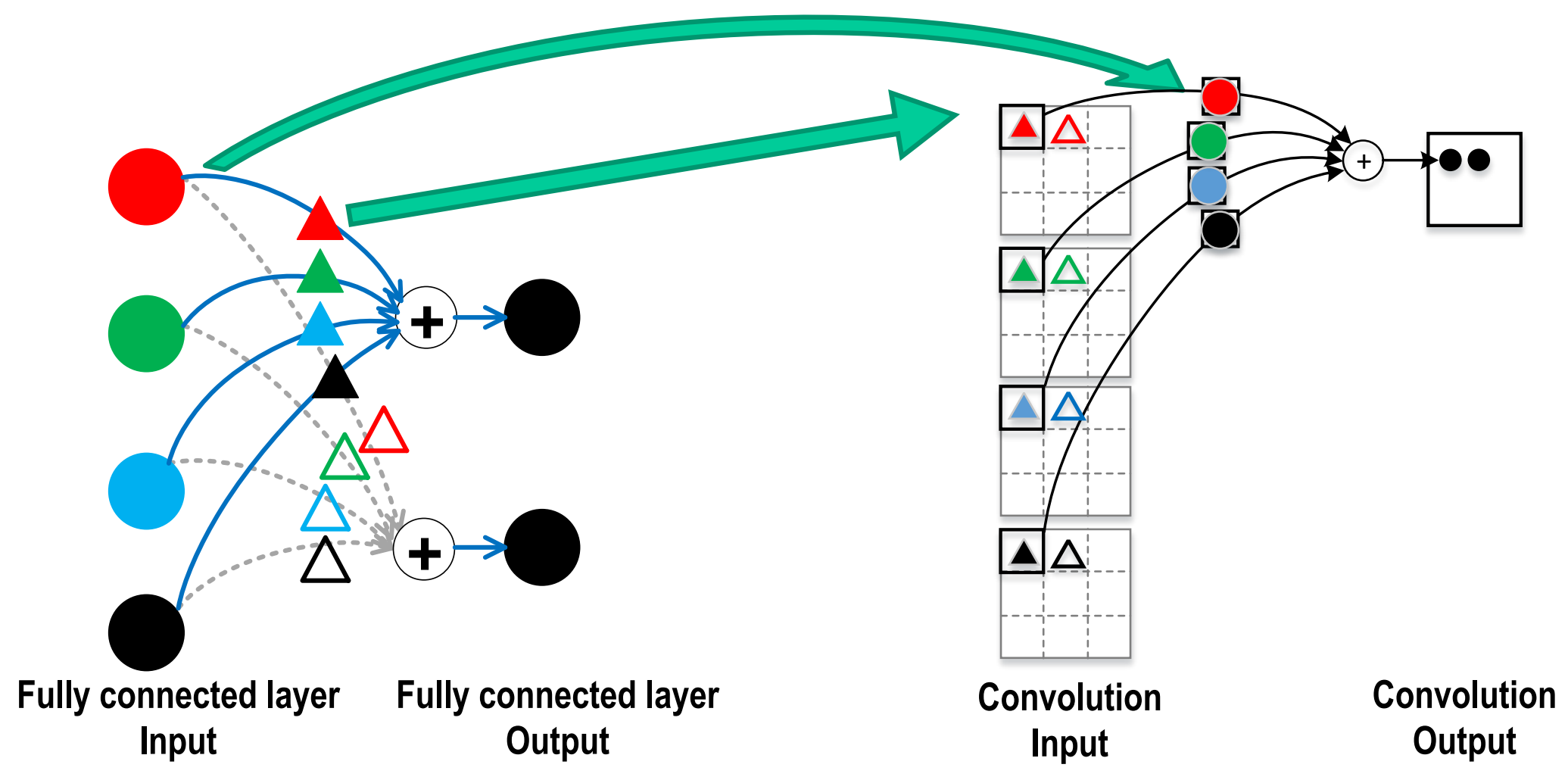
Input-Major Mapping



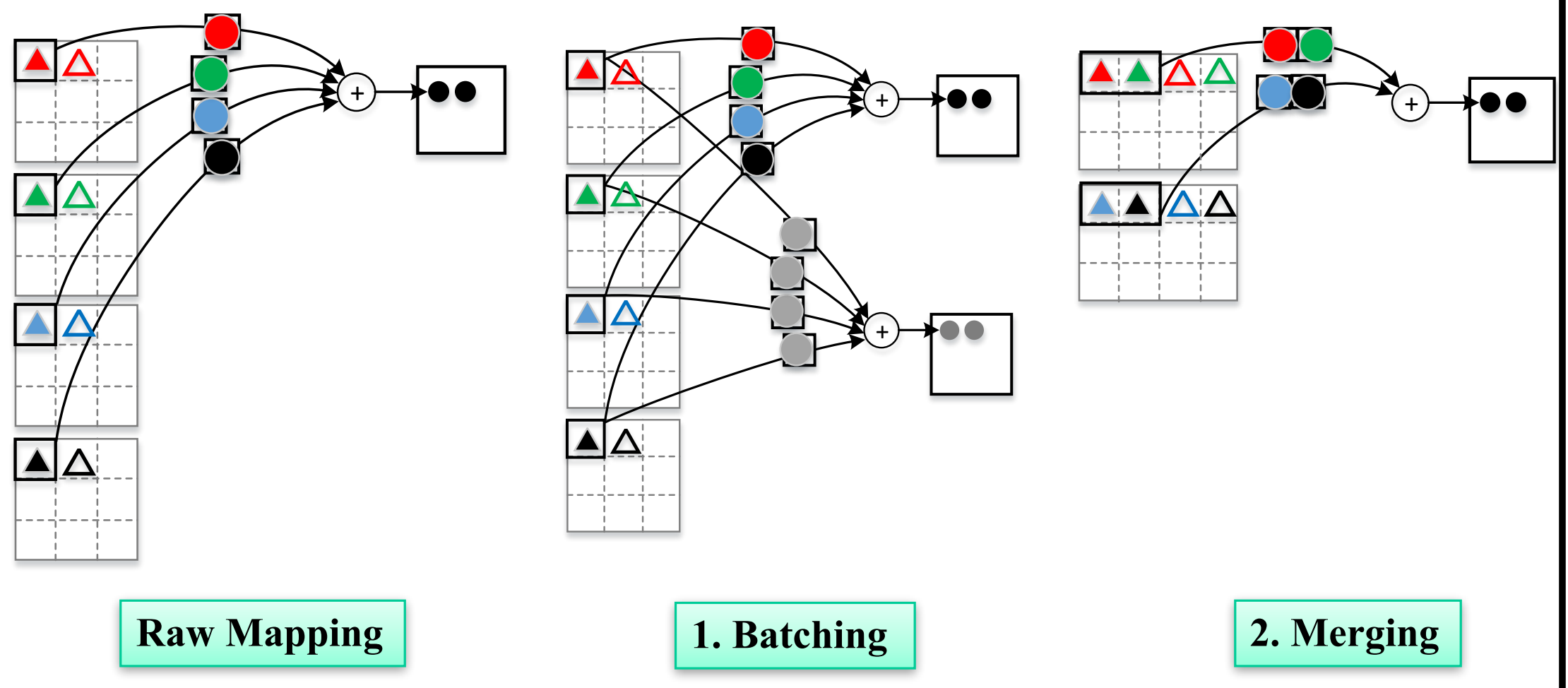
- Two ways of improving effective bandwidth



Weight-Major Mapping

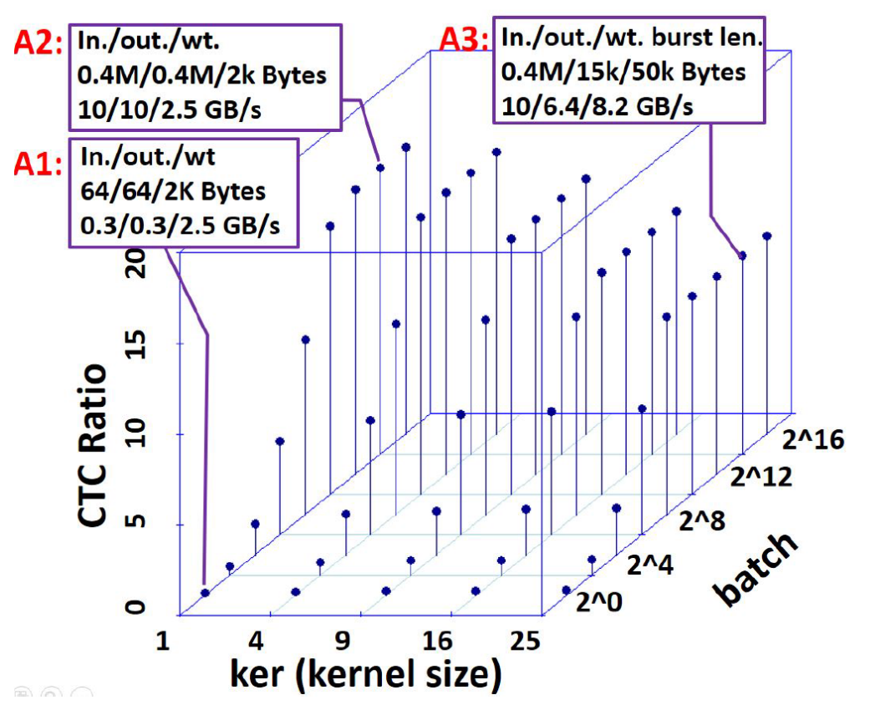


- Two ways of improving effective bandwidth

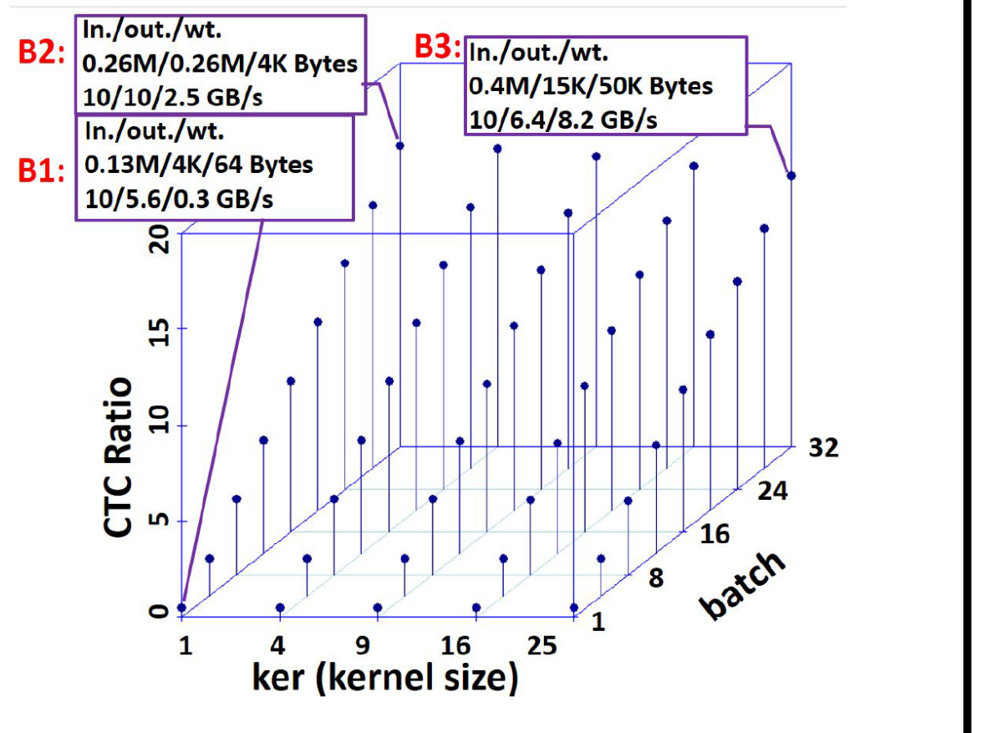


Bandwidth improvement

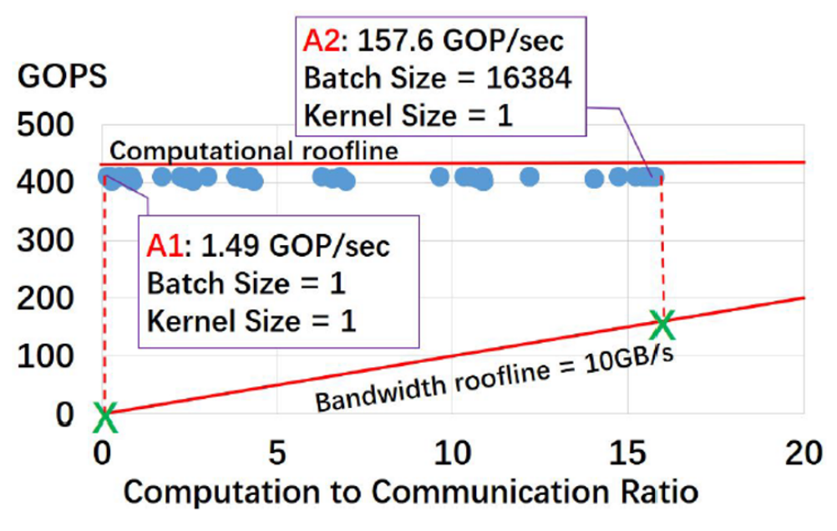
Method 1: Input-major Mapping

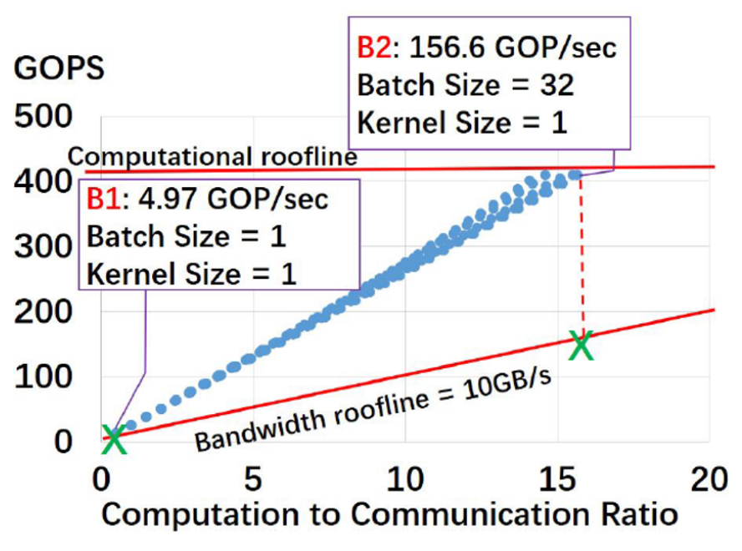


Method 2: Weight-major Mapping



- Design Space in Roofline Model





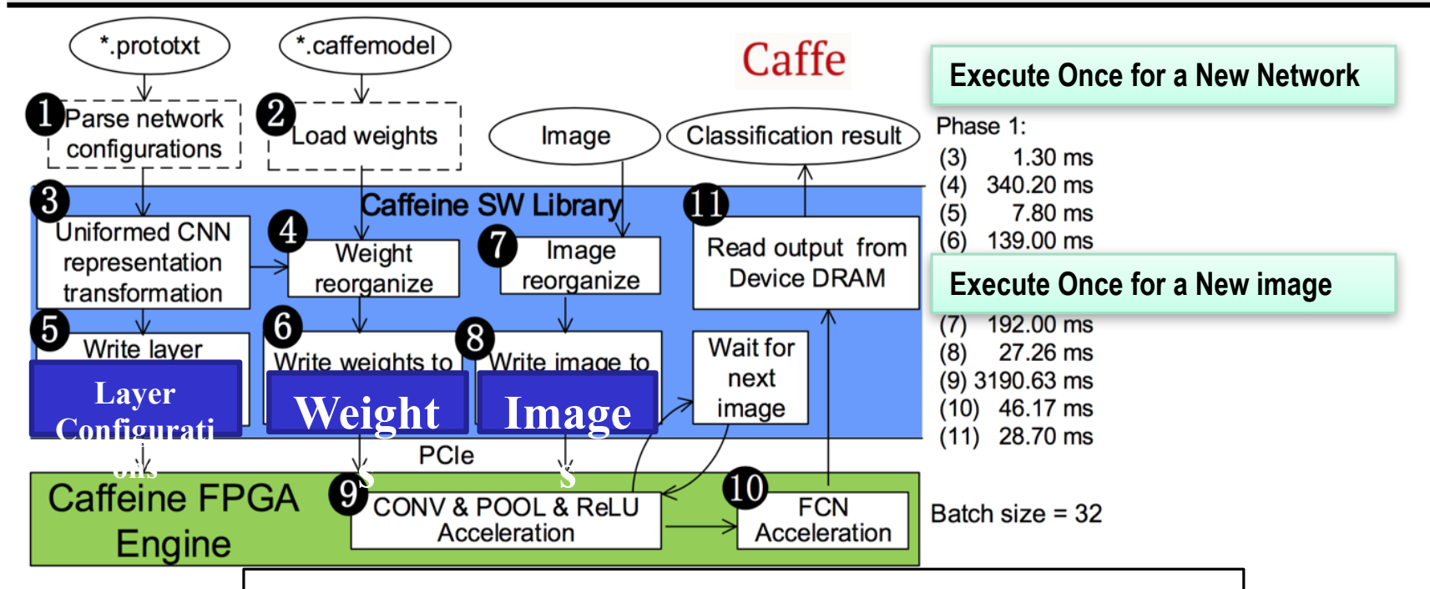
Automation flow from Caffe to FPGA

CNN Layer Definitions

input: "data"
input_dim: 3
input_dim: 224
input_dim: 224
	layers {
	bottom: "conv1_1“
	top: "conv1_2“
	name: "conv1_2“
	type: CONVOLUTION
	convolution_param {
	num_output: 64
	pad: 1
	kernel_size: 3 }}
		layers {
		bottom: "conv1_2"
		top: "conv1_2"
		name: "relu1_2"
		type: RELU
		}
			layers {
			bottom: "conv1_2"
			top: "pool1"
			name: "pool1"
			type: POOLING
			pooling_param {
			pool: MAX
			kernel_size: 2
			stride: 2 }

Customized Accelerator Instructions

Type	input	output	row	col	kernel	stride	pad	ReLU	POOL	size	stride
CONV/FCN	3	64	224	224	3	1	1	1	1	2	2



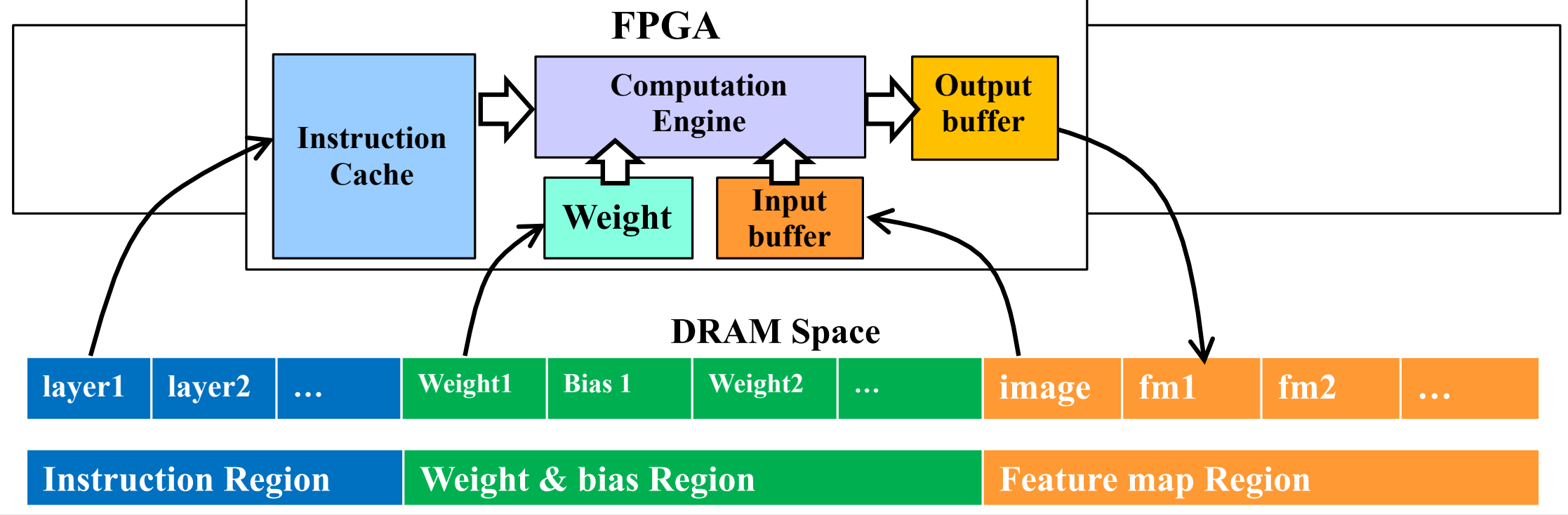
Execute Once for a New Network

Phase 1:
(3) 1.30 ms
(4) 340.20 ms
(5) 7.80 ms
(6) 139.00 ms

Execute Once for a New image

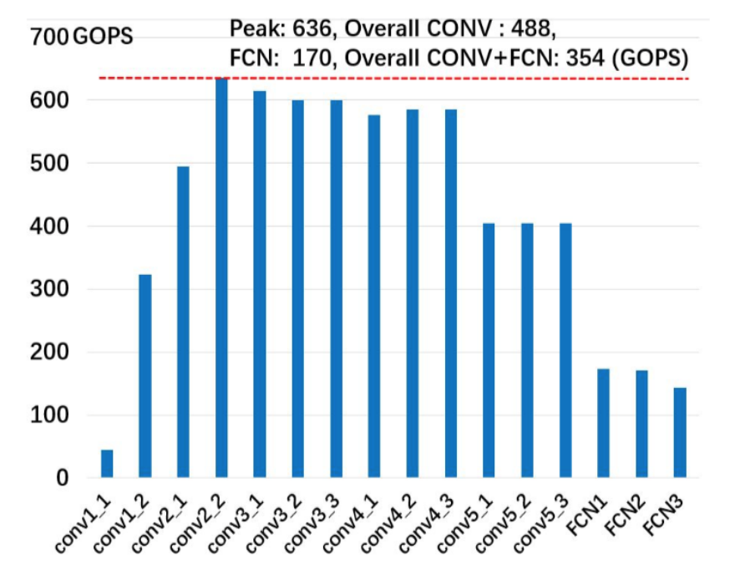
(7) 192.00 ms
(8) 27.26 ms
(9) 3190.63 ms
(10) 46.17 ms
(11) 28.70 ms

Batch size = 32

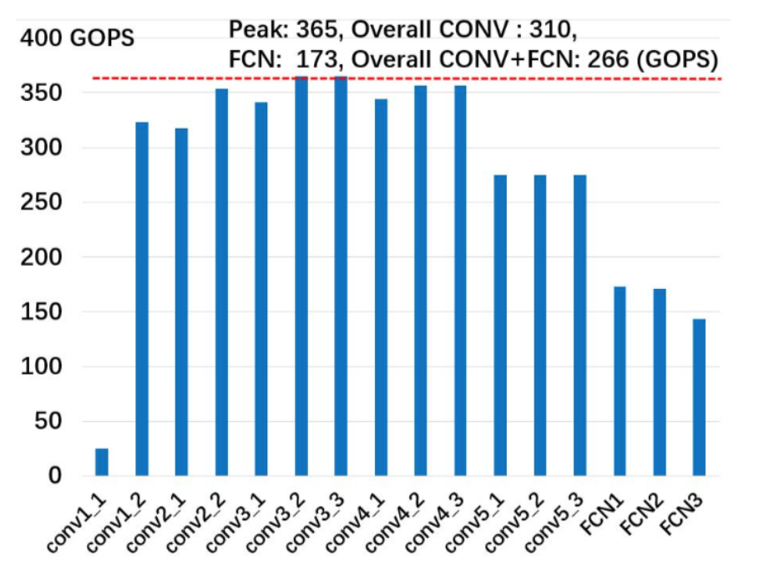


Results

- Portability to different board

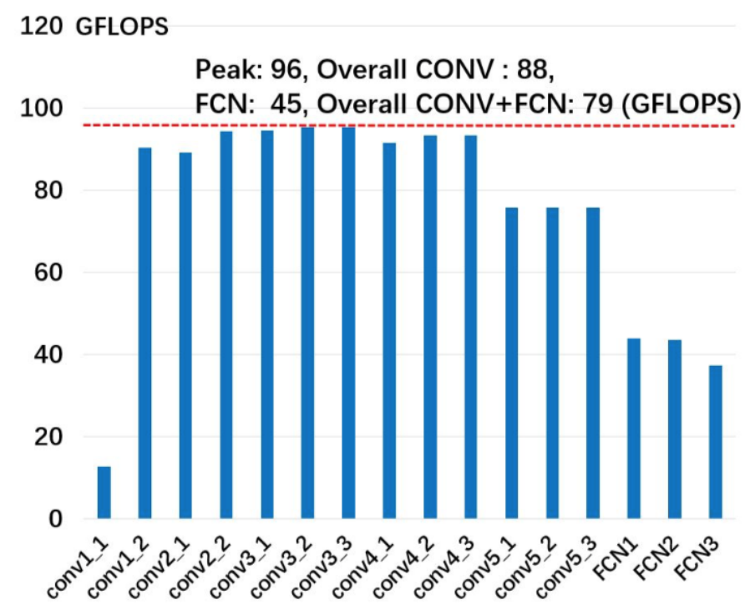


(a) VC709 VGG 16-bit fixed-point

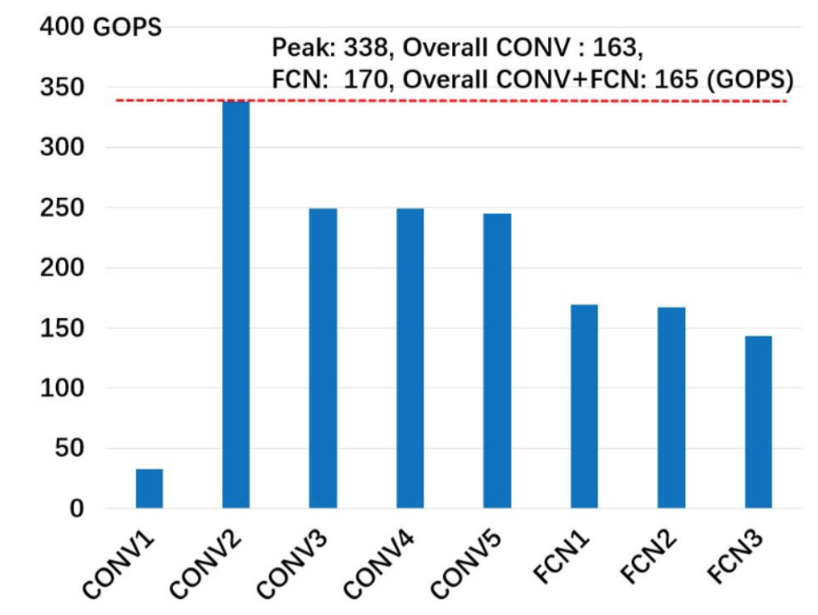


(b) KU VGG 16-bit fixed-point

- Different data type



(c) KU VGG 32-bit floating-point



(d) KU AlexNet 16-bit fixed-point

Summary:

Performance: 7.3x speedup over Intel Xeon 12-core CPU

Energy: 43.5x over Intel Xeon, 1.5x over GPU