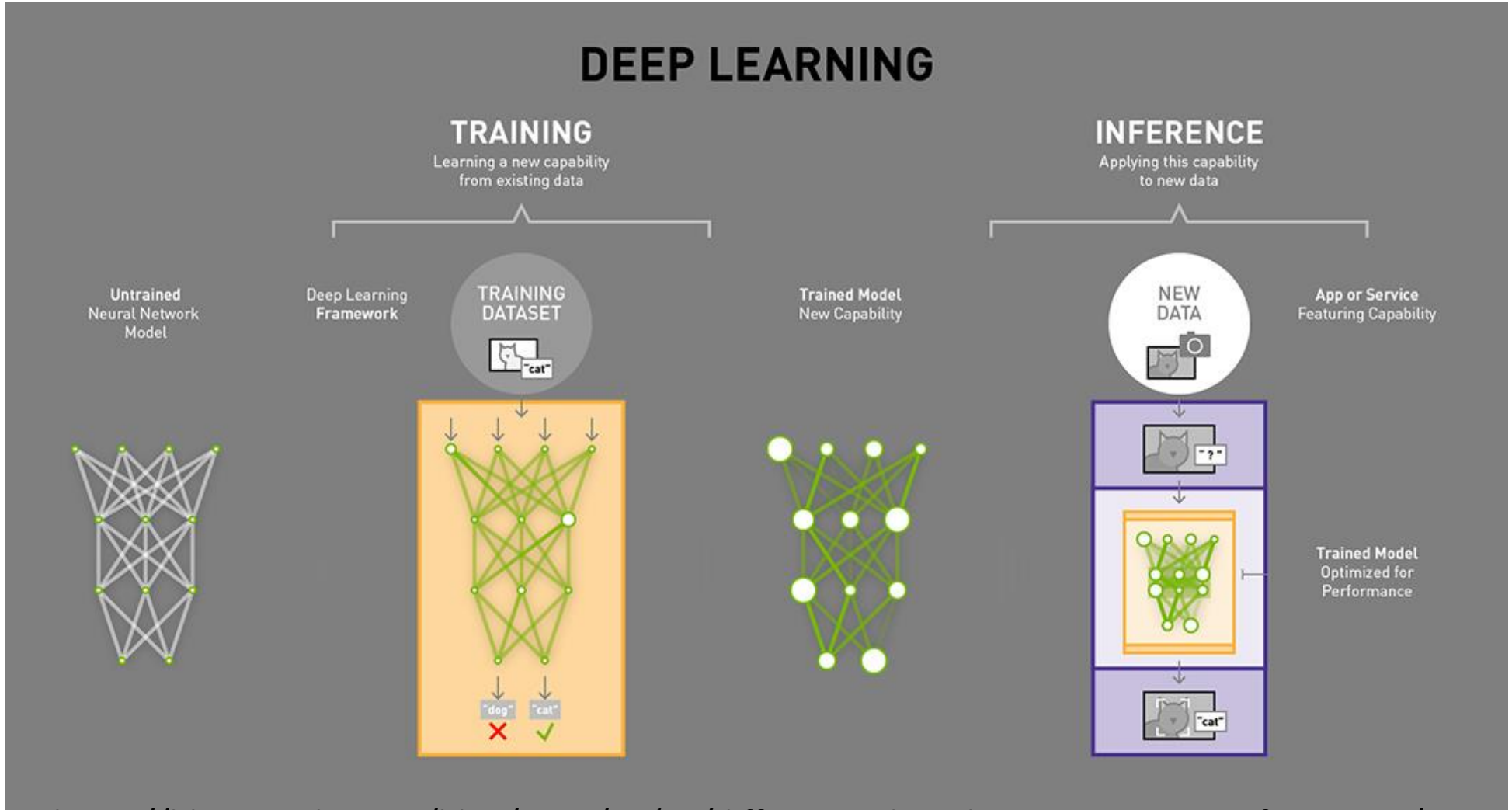
Cognitive computing at the edge

Paolo Meloni, PhD

DIEE- Università degli Studi di Cagliari

Deep Learning at the edge

- Challenges
- Efficient implementation techniques
  - Algorithms
  - Platforms
  - Tools

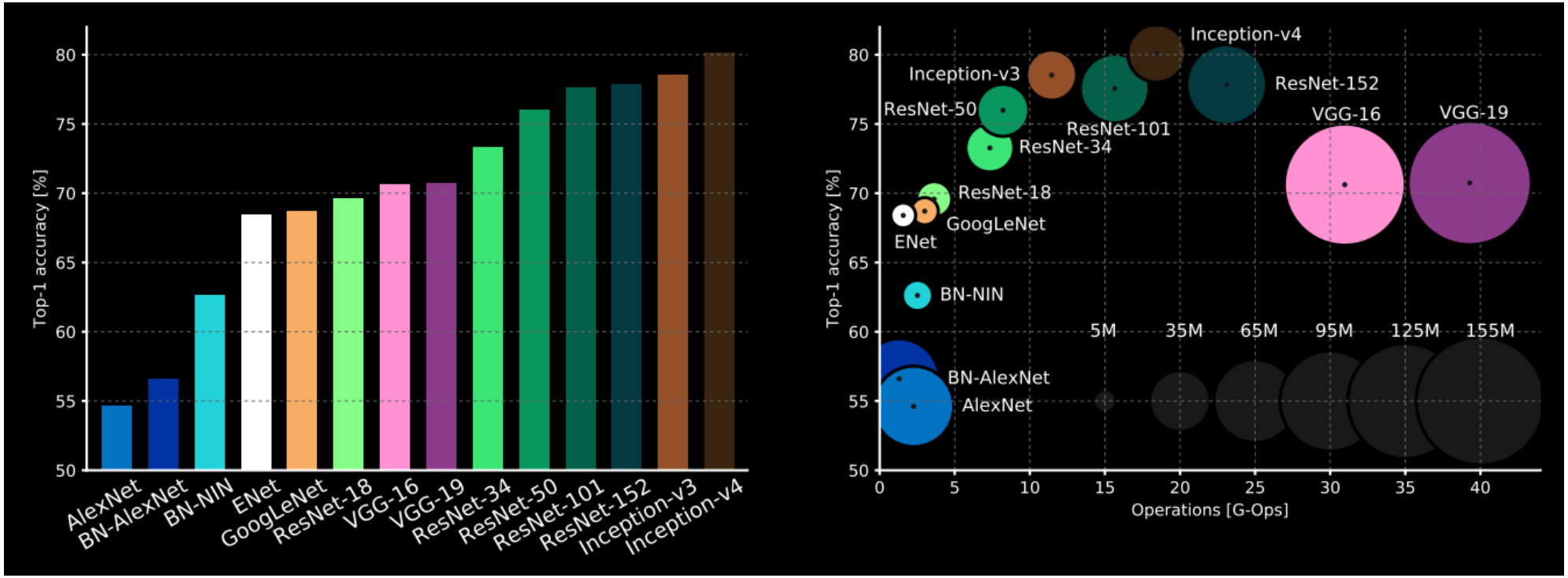


https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/

Why edge computing?

- Latency
- Resilience
- Privacy

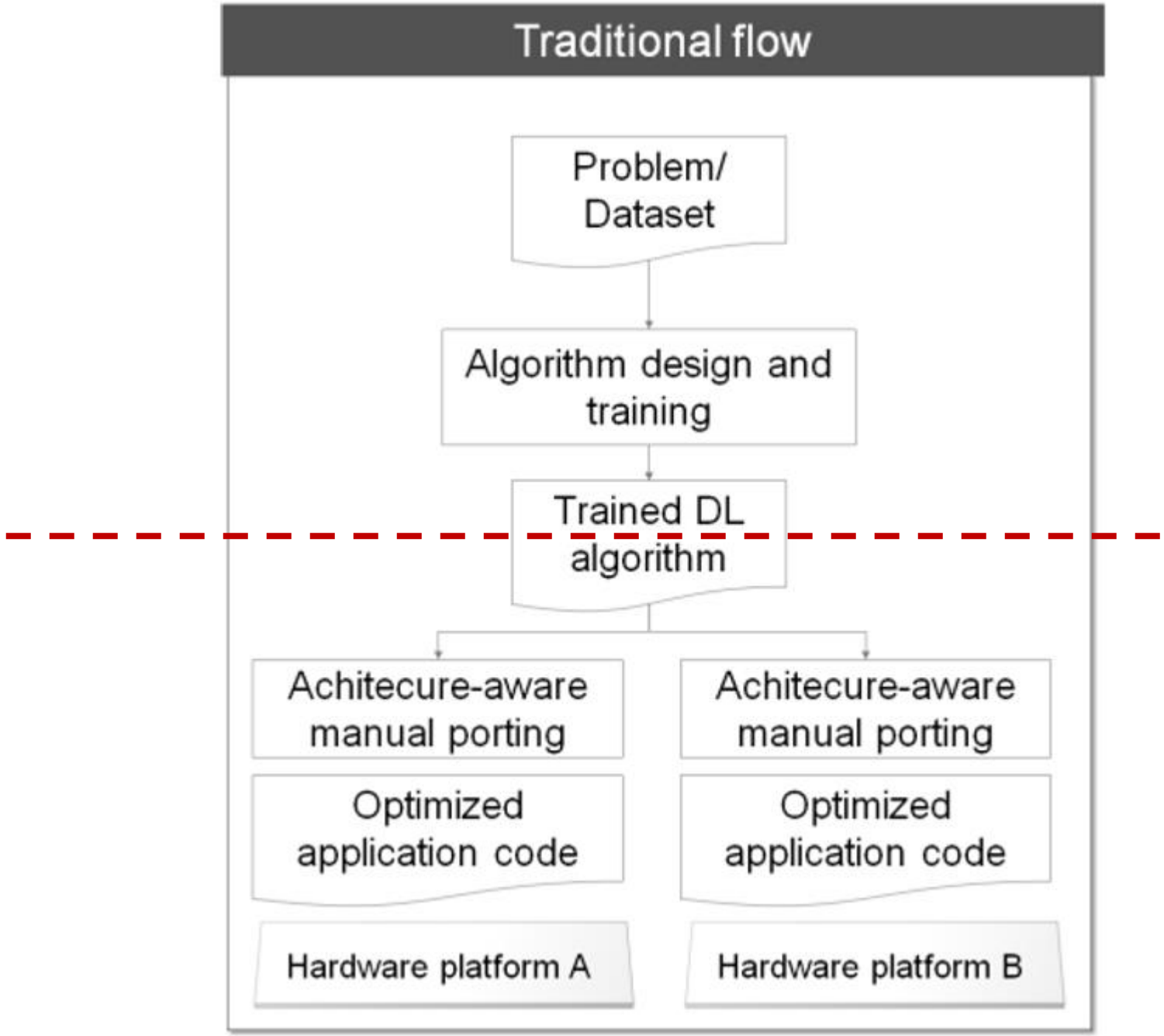
Challenges: Model size and complexity



An Analysis of Deep Neural Network Models for Practical Applications A. Canziani, A. Paszke, E. Culurciello, 2016

Enormous computational and memory requirements
e.g., VGG-19: 140 million floating-point parameters to classify a single image

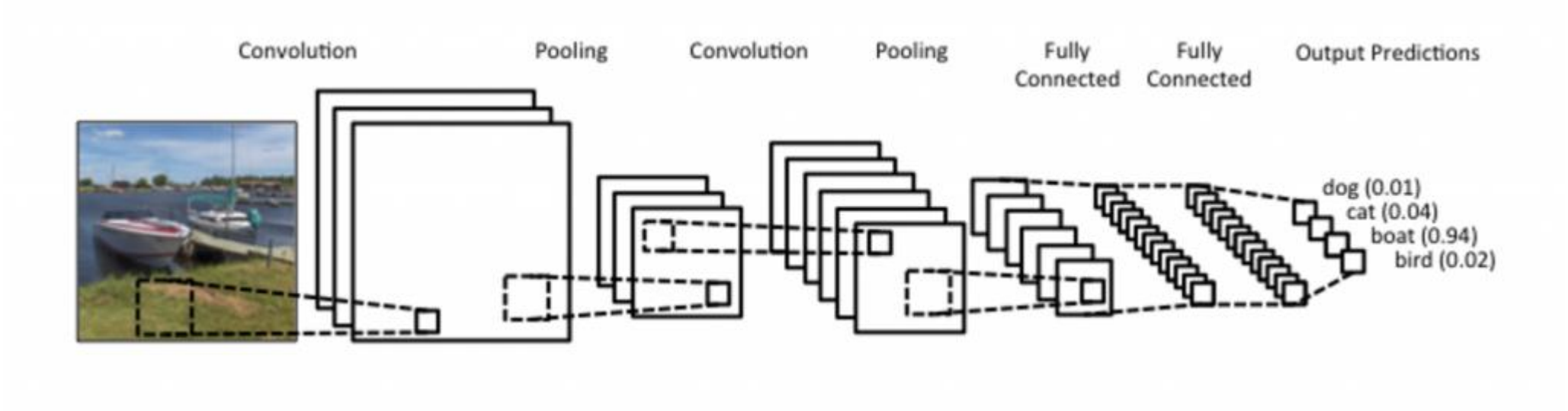
The DL dichotomy

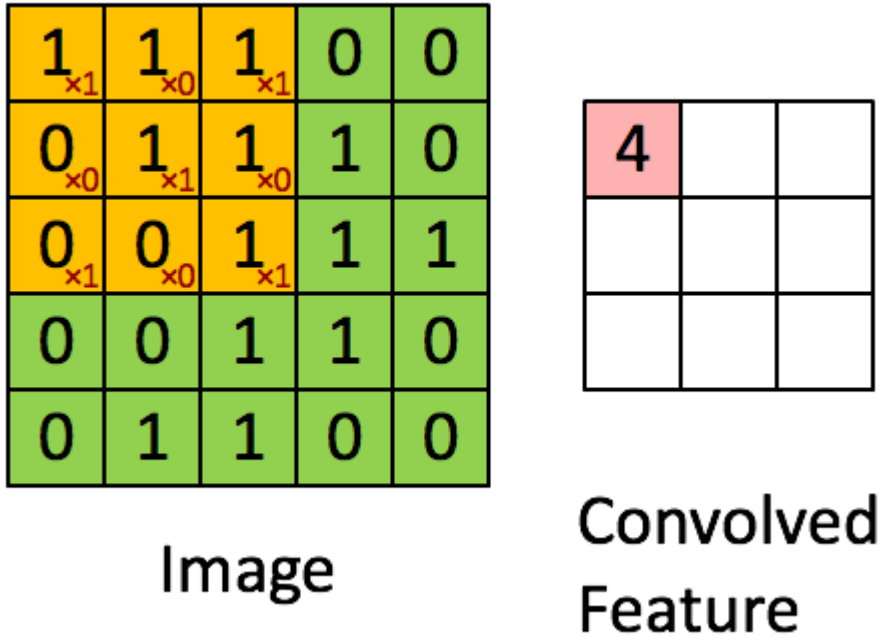


Efficient implementation techniques

- Algorithms
- Platforms
- Tools

Algorithms: example – Convolutional Neural Network





Convolution with 3×3 Filter. Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

#Synapses in Human Brain

50 Trillion Synapses

1000 Trillion Synapses

500 Trillion Synapses

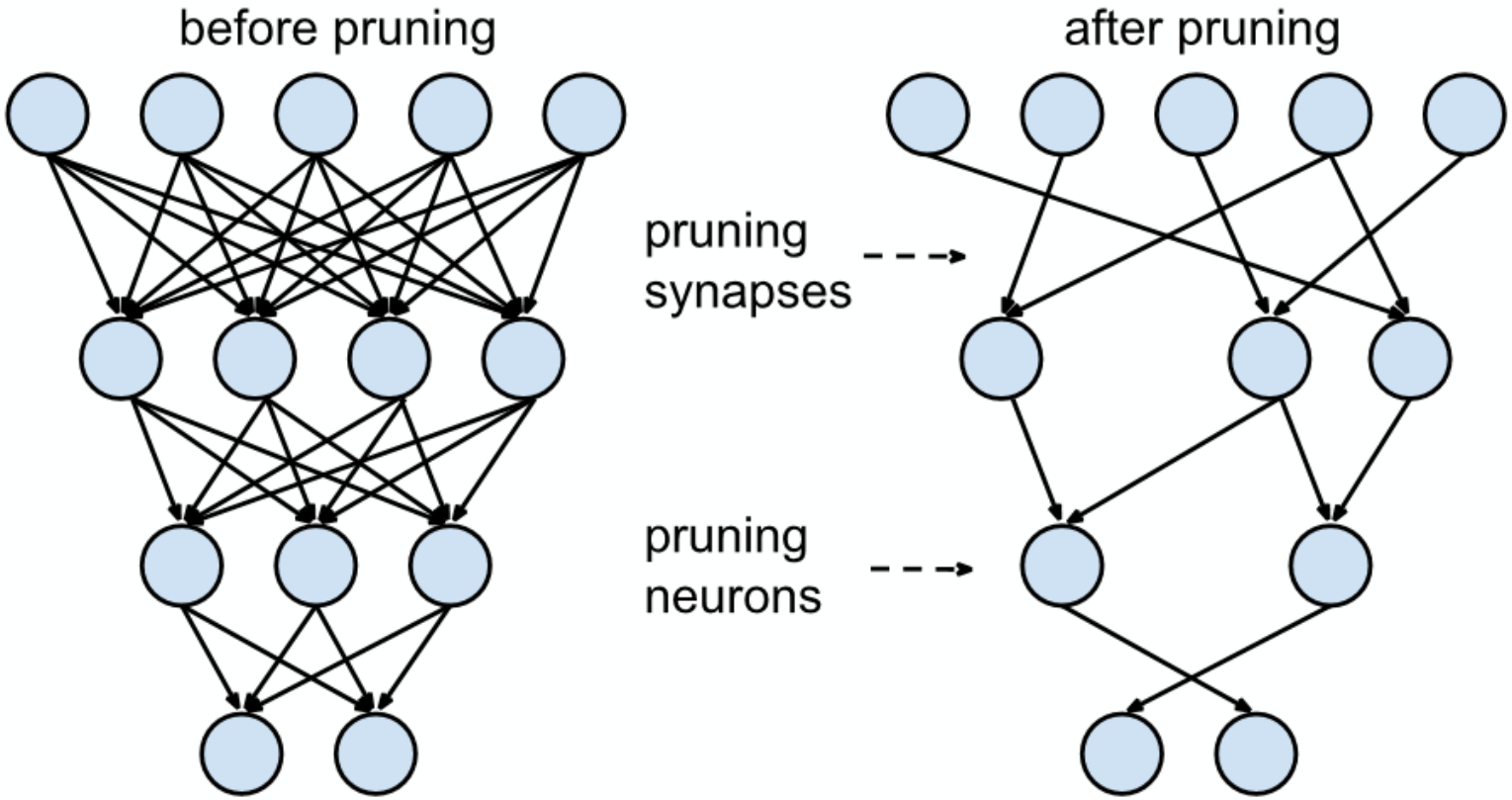
Newborn

1 year old

Adolescent

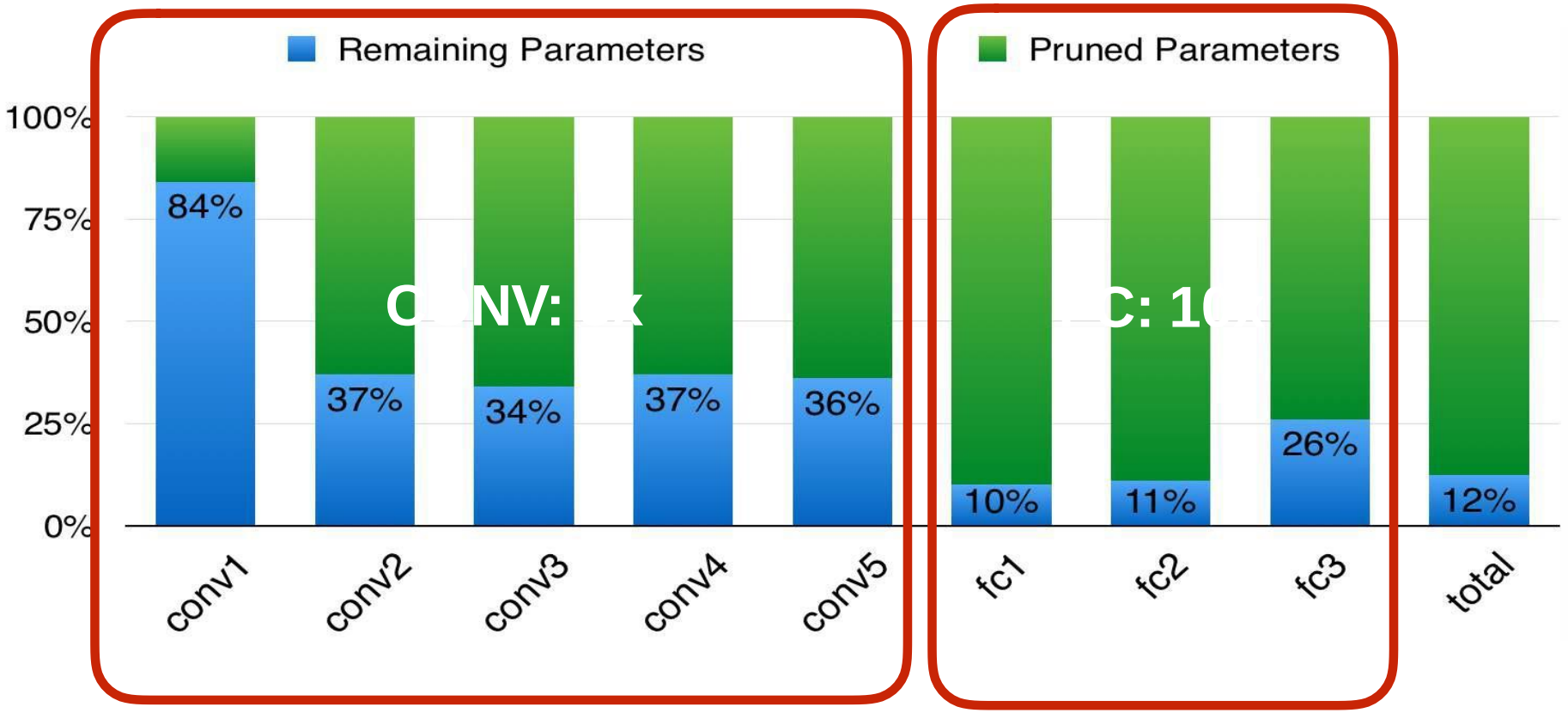
Christopher A Walsh. Peter Huttenlocher (1931-2013). Nature, 502(7470):172–172, 2013.

Pruning Neural Networks



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS'15

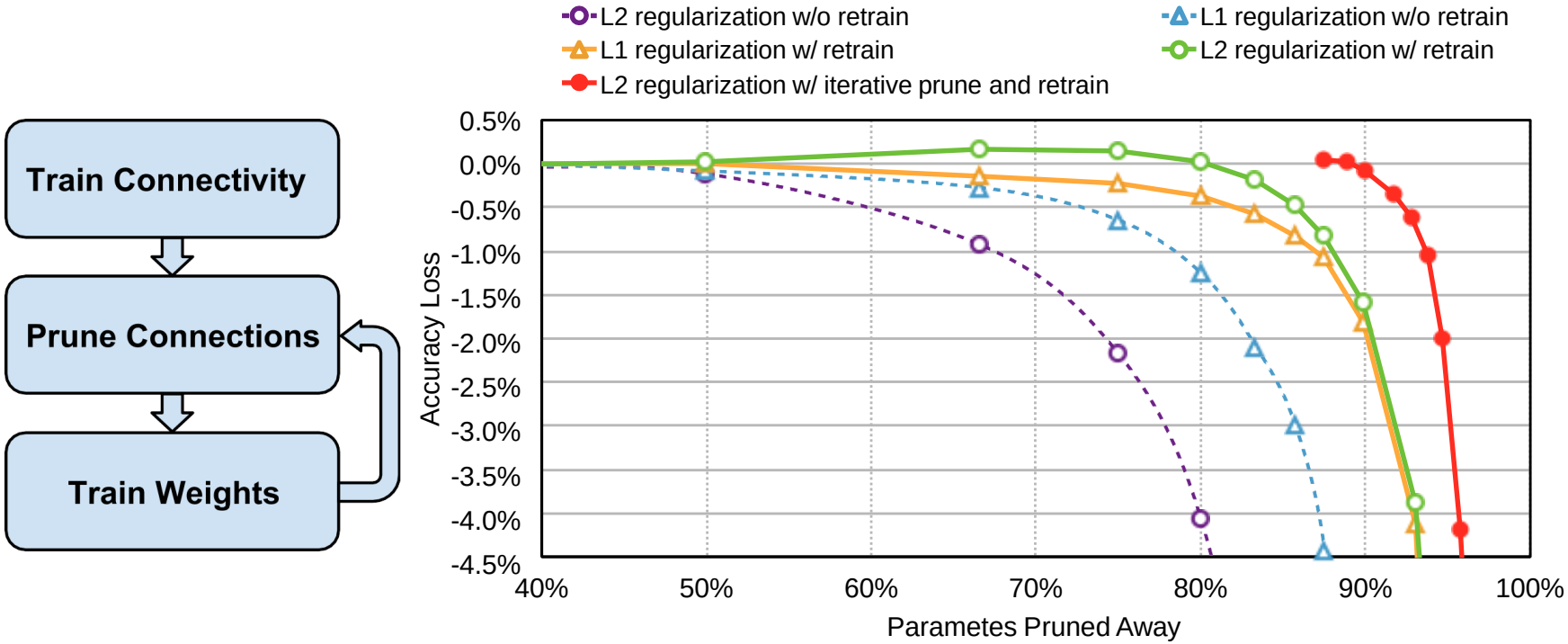
AlexNet



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

[Han16]

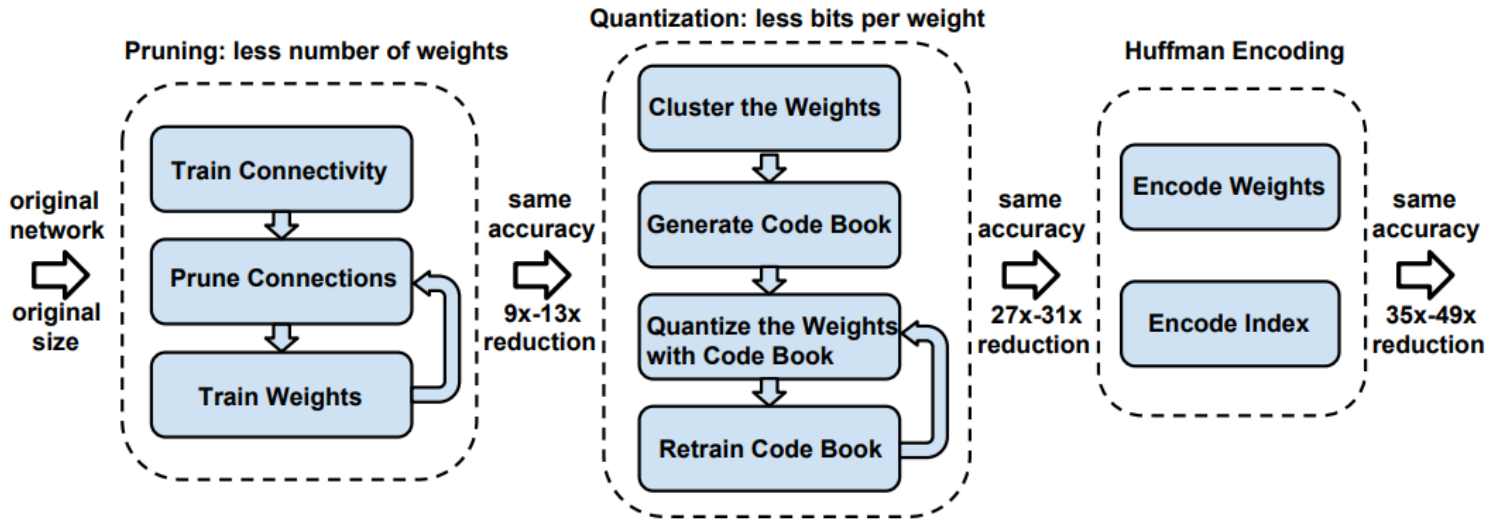
Retrain to Recover Accuracy



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

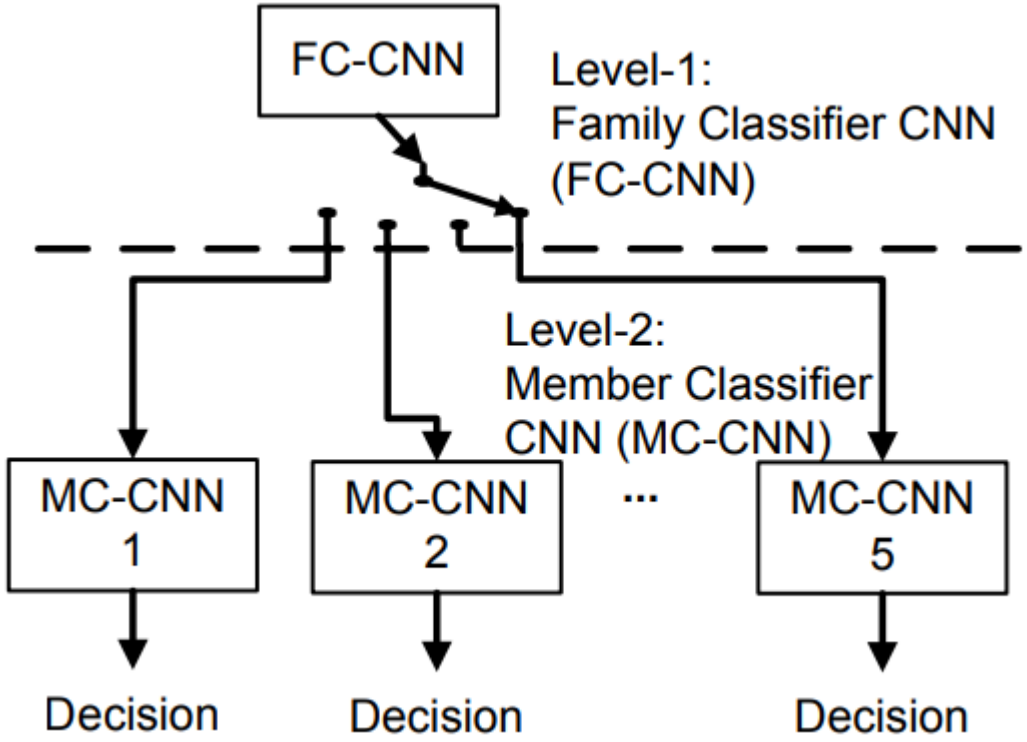
[Han16]

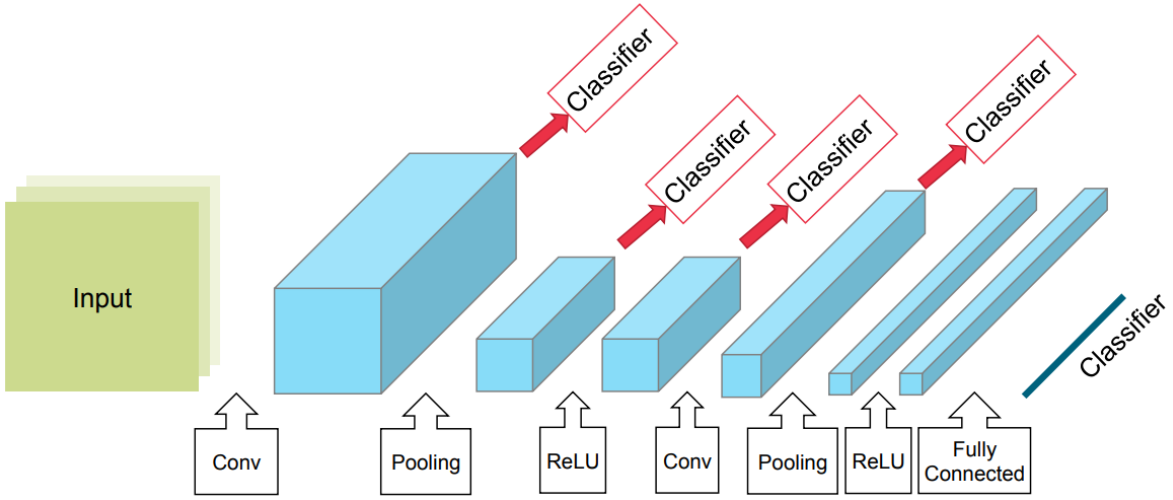
Quantization and Compression

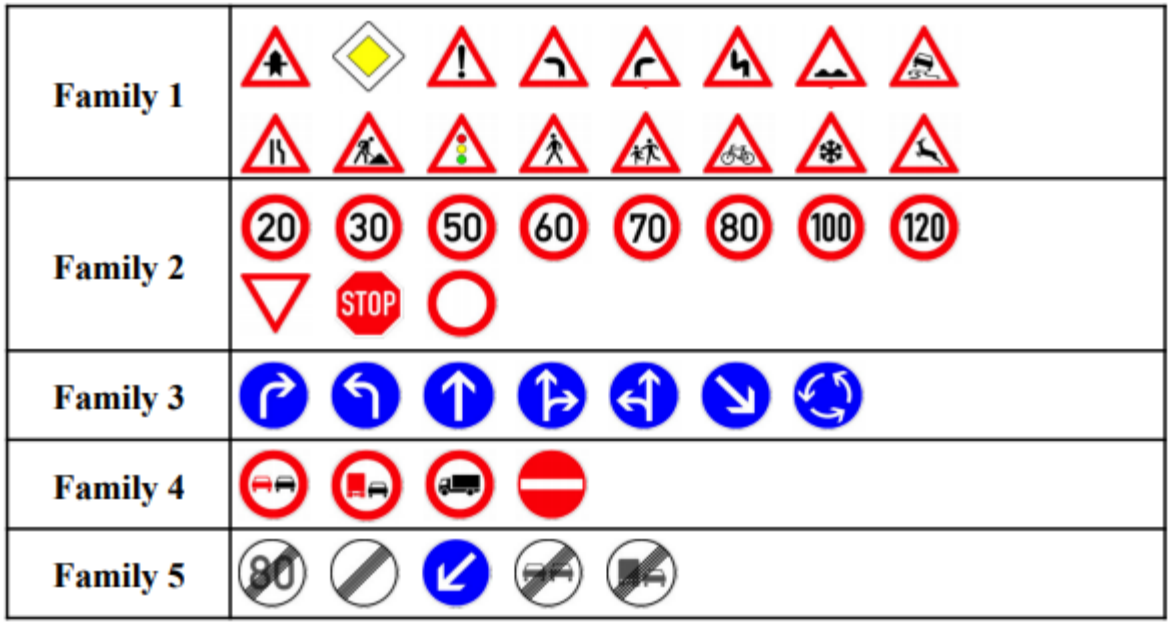


Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	**40**×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39**×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35**×
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49**×

Hierarchical CNN







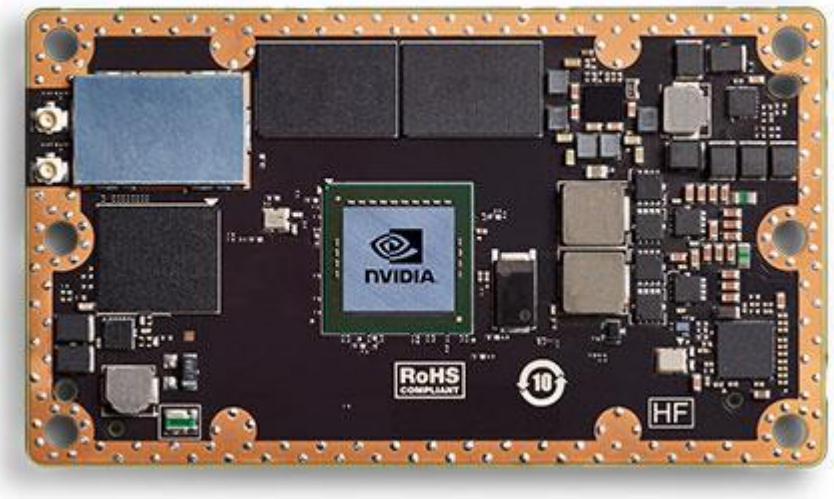
- 5+1 different CNNs for classification
- lower complexity than one-vs-all classifier

https://ip.cadence.com/uploads/presentations/1345PM_ENNS_v10_Samer_Hijazi.pdf

Platforms

- GPU
- FPGA
- Specialized processing elements

GPUs – NVIDIA



- >90% utilization of processing elements
- Good floating point support
- Good tool support CUDNN
- Good performance on vanilla CNNs

Network: AlexNet	Batch Size	Tegra X1 (FP32)	Tegra X1 (FP16)	Core i7 6700K (FP32)
Inference Performance	1	47 img/sec	67 img/sec	62 img/sec
Power		5.5 W	5.1 W	49.7 W
Performance/Watt		8.6 img/sec/W	13.1 img/sec/W	1.3 img/sec/W

https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

GPU - MALI

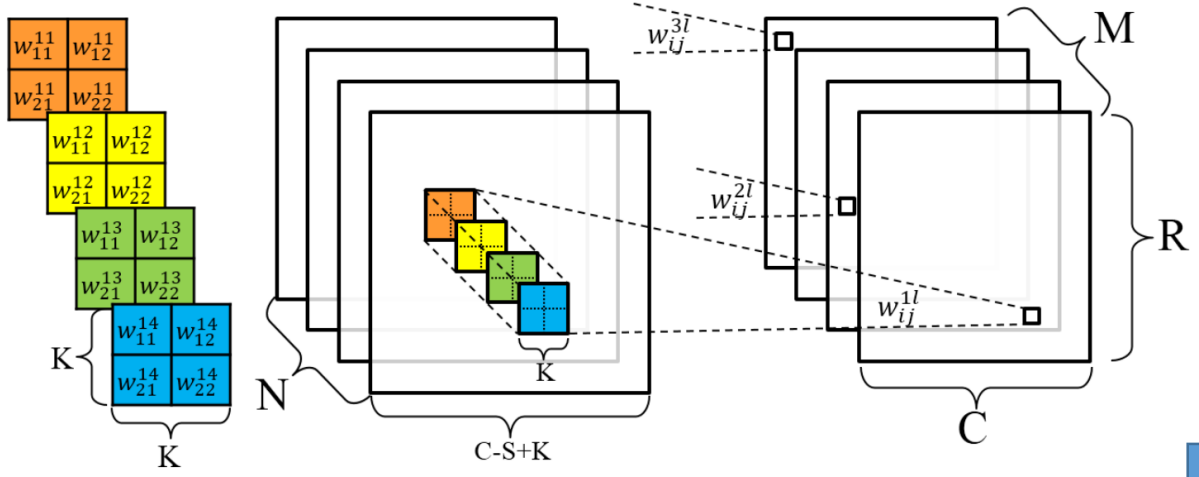
- CaffeNet (similar to AlexNet @around 10 FPS on mobile phone)

Table 1: inference speed of CaffeNet on ILSVRC 2012 and FOOD 101 datasets (Batch 12)

Image Dataset	Mean, inference time
Xiaomi Redmi Note 4: Mediatek MT6797 Helio X20 (ARM/Mali-T880 MP4)	
ILSVRC 2012	250.7 ms
FOOD 101	250.2 ms
Samsung S7 Edge: Exynos 8890 Octa (ARM/MALI-T880 MP 12)	
ILSVRC 2012	110 ms
FOOD 101	110 ms

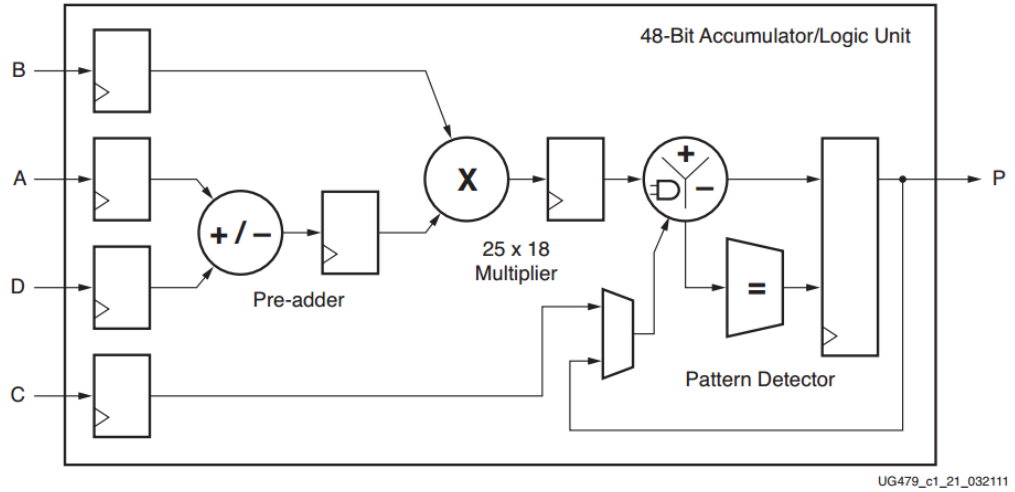
IRIDA Las publication -https://docs.wixstatic.com/ugd/4812cc_e3fae3418c8d4d67a05d199c4aadde19.pdf

Convolutional Neural Networks on FPGAs – CONV layer to DSPs



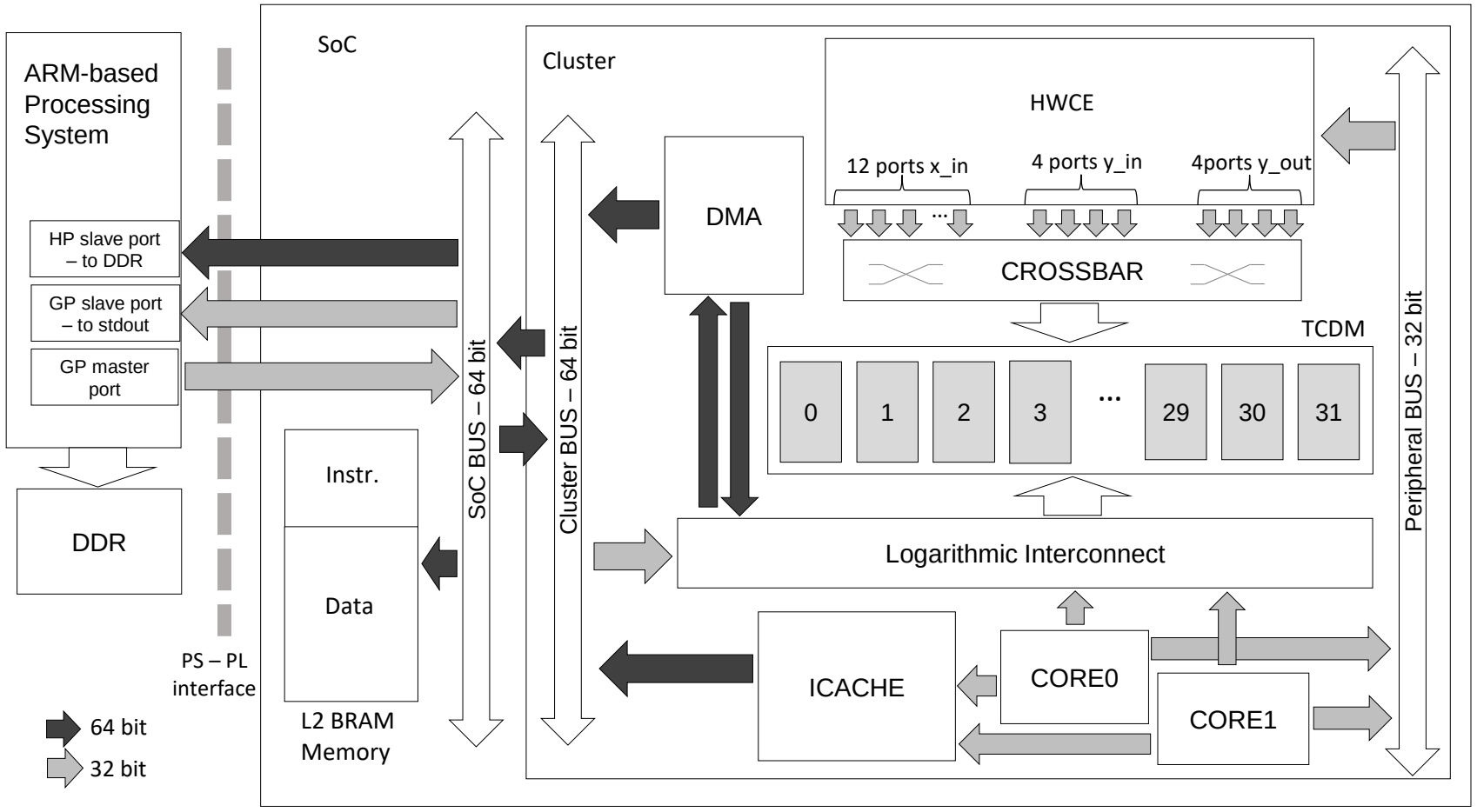
[Zhang et al., 2015]

```
for i in range(0, nb_out_feat):
    for j in range(0, nb_in_feat):
        for s in range(0, feat_height):
            for t in range(0, feat_width):
                Conv= Conv (X[j,s,t] * Wt[i,j])
                //sum of product
            Y[i,s,t] += conv
```



- Multiply-and-accumulate intensive
- High level of parallelism
- Maps well on FPGA MAC primitives (Xilinx DSP48 slices)

CNN acceleration on FPGAs: Neuraghe

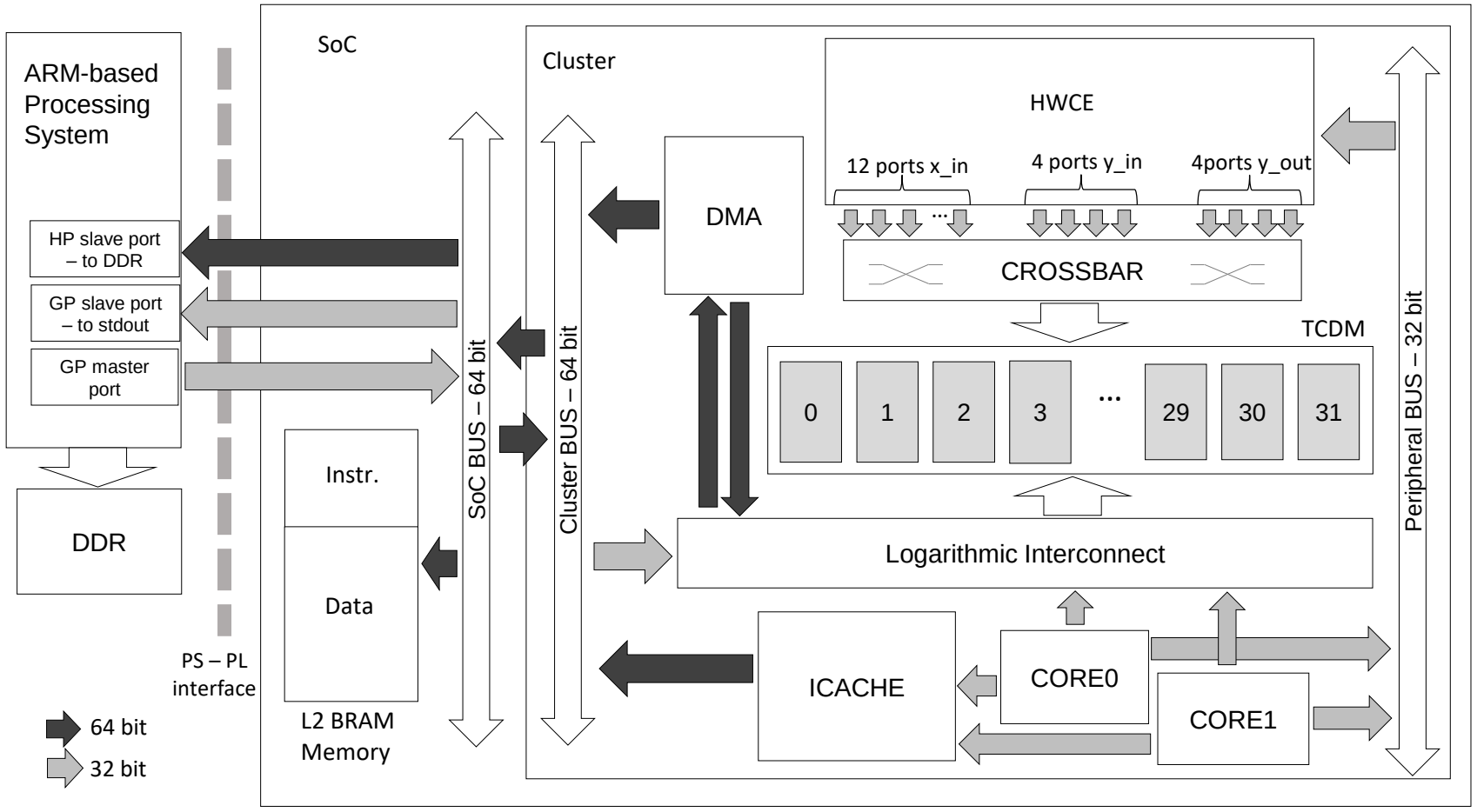


P. Meloni, G. Deriu, F. Conti, I. Loi, L. Raffo and L. Benini, "A high-efficiency runtime reconfigurable IP for CNN acceleration on a mid-range all-programmable SoC," *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, 2016, pp. 1-8.

Design principles

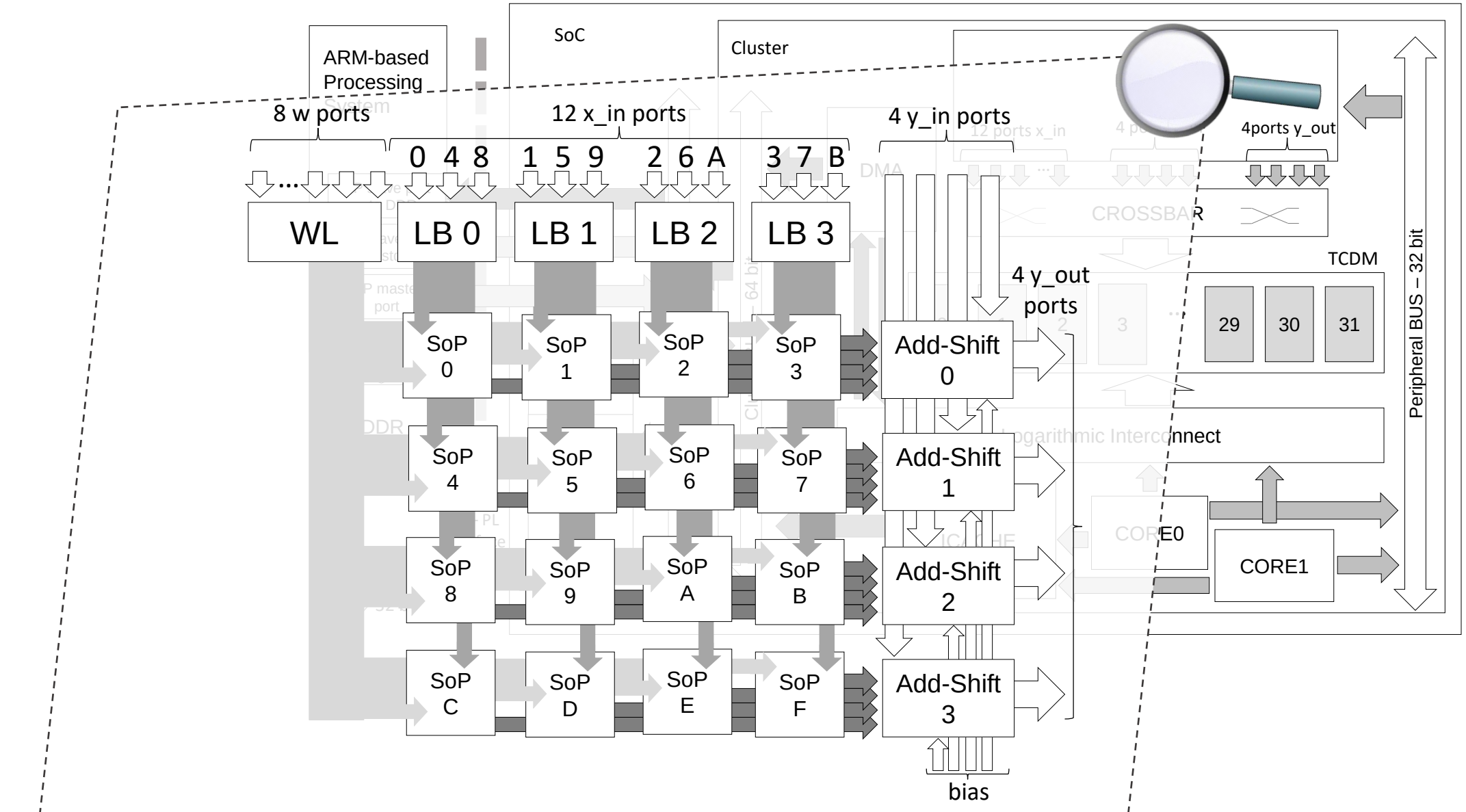
- Exploit parallelism
  - Use as many DSP as possible (Reference SoC – Zynq Z7045)
- Support dynamic reconfiguration
  - Adapt to multiple conv layers
    - Different kernel sizes
    - Different strides
- Reduce I/O bottleneck
  - Careful convolution scheduling
- All-programmable SoCs
  - Exploit ARM for housekeeping and other CNN layers

Architectural template

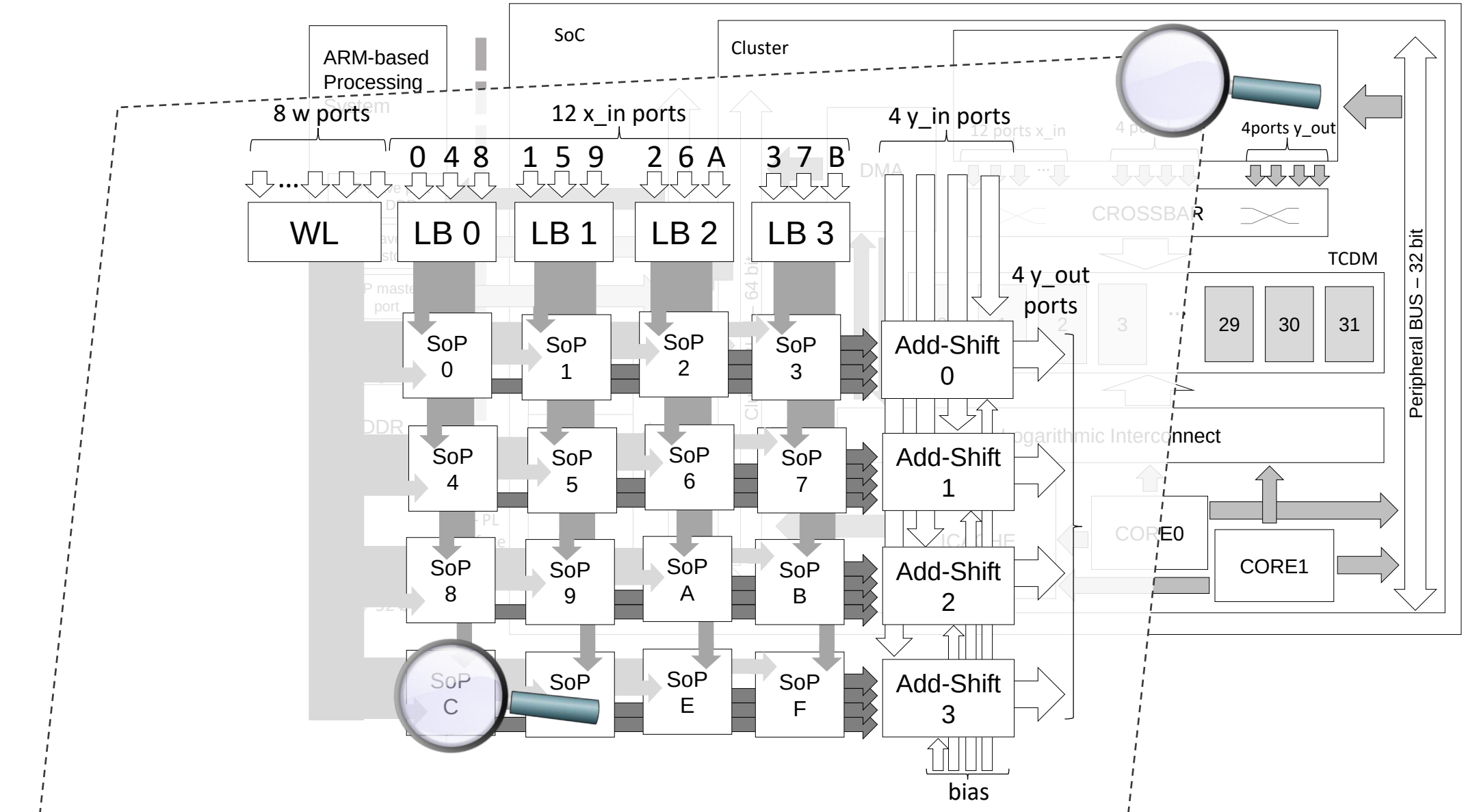


F. Conti et al. PULP: A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision.
Journal of Signal Processing Systems, 2015.

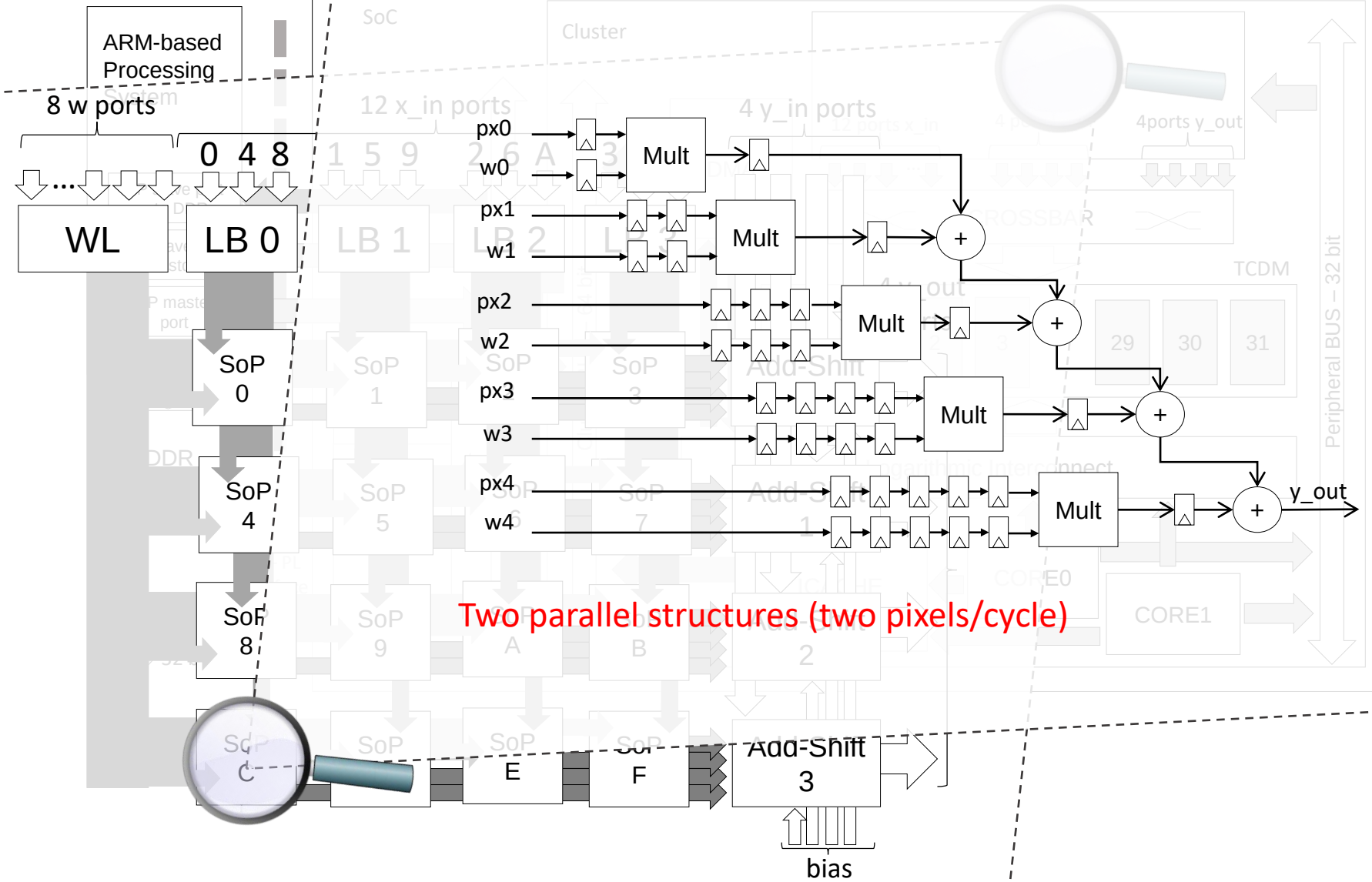
Architectural template



Architectural template



Architectural template



Dynamic reconfiguration support

- Support 5x5 and 3x3 layers
- SoP modules sized to suit both kernel sizes (27 MACs)

  a) 3x3 filters on 3 input features b) 5x5 filters on 1 input feature (25 MACs)

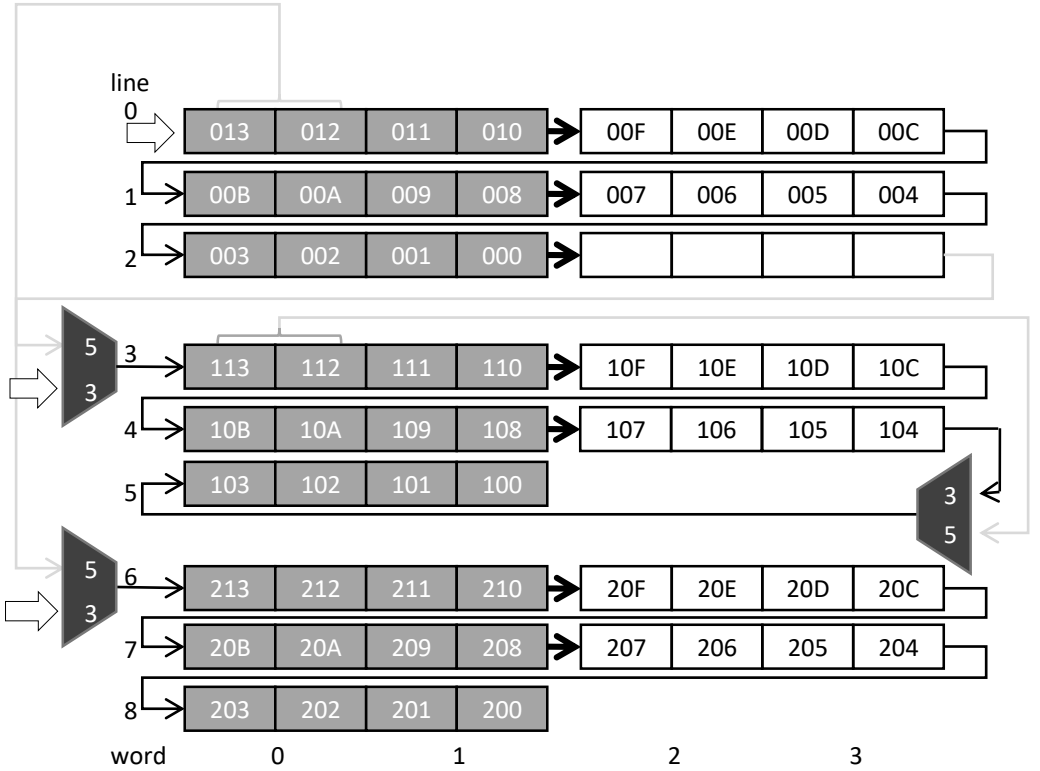
- Line buffer implements adaptivity

a)3x3

	200	201	202	203	204	205	206	207
	100	101	102	103	104	105	106	107	20F
000	001	002	003	004	005	006	007	10F	217
008	009	00A	00B	00C	00D	00E	00F	117	21F
010	011	012	013	014	015	016	017	11F	227
018	019	01A	01B	01C	01D	01E	01F	127	22F
020	021	022	023	024	025	026	027	12F	
028	029	02A	02B	02C	02D	02E	02F		

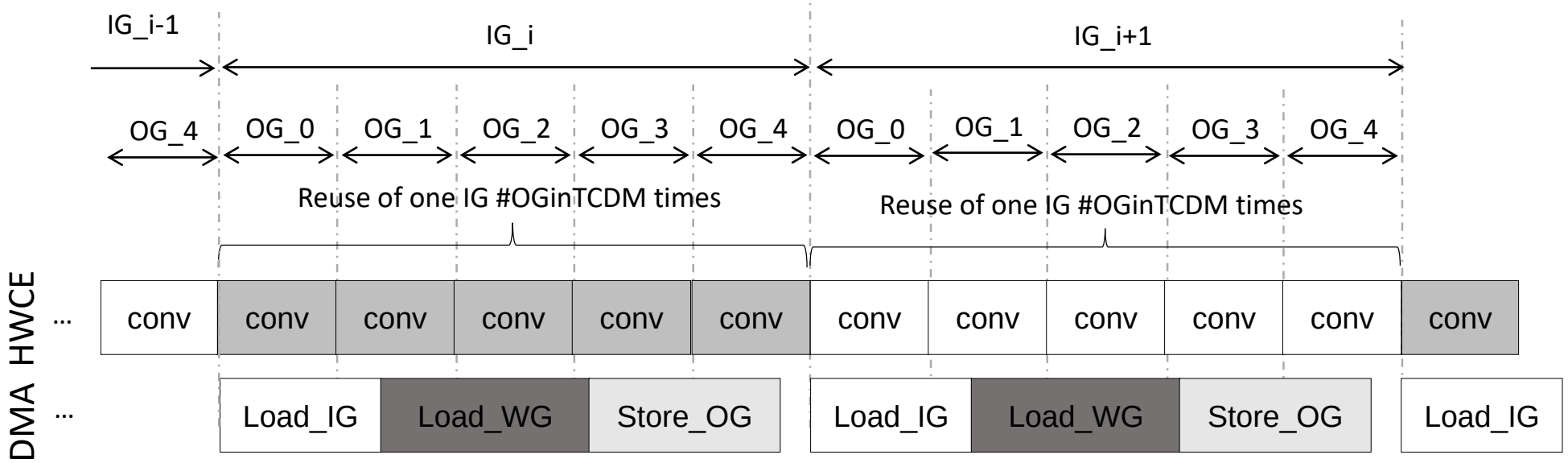
b) 5x5

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F



Convolution scheduling

IG (Input group): 4/12 input features simultaneously loaded by HWCE
OG (Output group): 4 output feature contributions simultaneously produced by HWCE



- Load an IG
- Reuse it! Compute its contributions to several output groups
- Overlap communication and computation (double buffering) – Load next IG and weights during conv
- Iterate until all the contributions to the OGs are accumulated
- Use DMA idle slots to output OGs when complete

Results – Hardware implementation evaluation

Parameter	Value
Slow clock frequency	75 MHz
HWCE clock frequency	150 MHz
Peak performance	129.6 GMAC/s (260 GOPS/s)
I/O bandwidth	16 B/cycle
Line buffer size	128 word
Pixel data precision	16 bit

Resource	DSP	BRAM	LUTs as logic	LUTs as SRegs	Regs
Used	874	192	86047	19989	97585
Available	900	545	218600	218600	437200
Utilization	97.1%	32.2%	39.4%	28.4%	22.3%

Reference SoC – Xilinx Zynq Z7045

Other Works on FPGAs

- Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks, Zhang et al [Zhang 2015]
  - High Level Synthesis approach (loop unrolling and pipelining selected after DSE)
- Going Deeper with Embedded FPGA Platform for Convolutional Neural Network, Qiu et al. [Qiu2016]
  - Model compression (SVD), convolution AND dense layers

	Zhang 2015	Qiu 2016
Platform	Virtex7 VX485t	Zynq XC7Z045
Clock (MHz)	100	150
Quantization	32-bit float	16-bit fixed
Logic Utilization	186K (61%)	183K (84%)
DSP Utilization	2240 (80%)	780 (89%)
BRAM Utilization	1024 (50%)	486 (87%)
Total GOP in Network	1.33	30.8
Performance (GOP/s)	61.6	137.0
Power (W)	18.6	9.6
Energy Efficiency (GOP/J)	3.3	14.2

Binarized Neural Network

- Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs, Zhao et al. [ZhaoFPGA2017]
- FINN: A Framework for Fast, Scalable Binarized Neural Network Inference, Umuroglu et al.

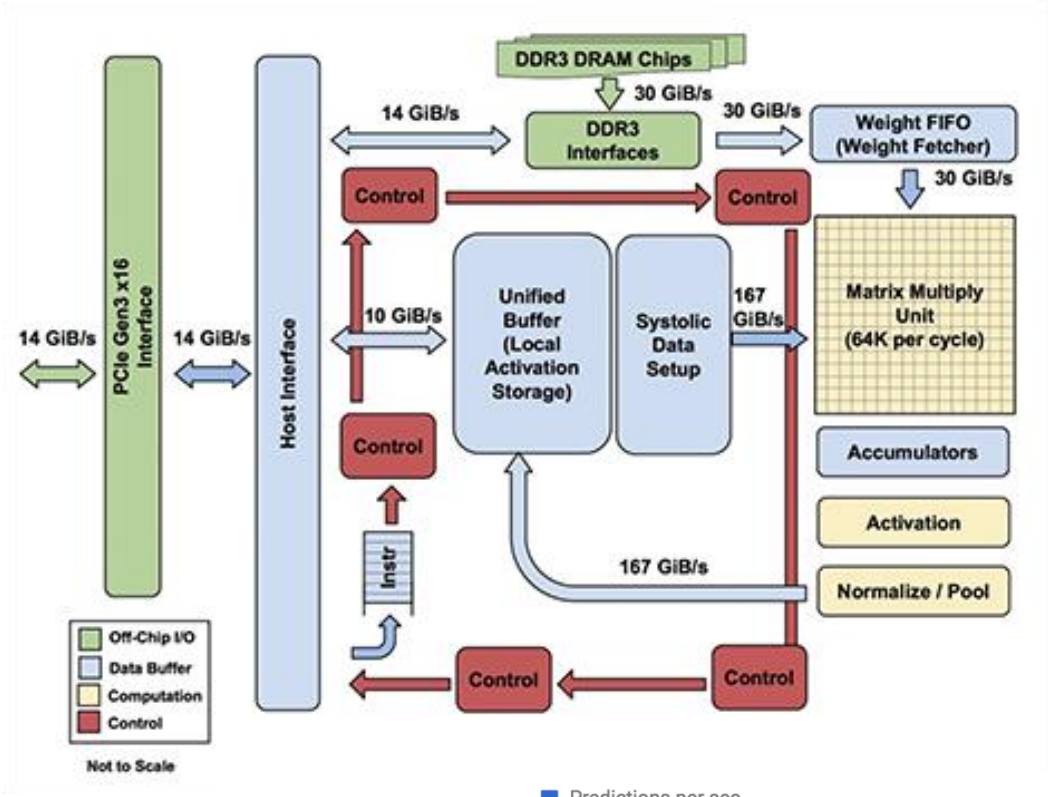
Custom processing platforms

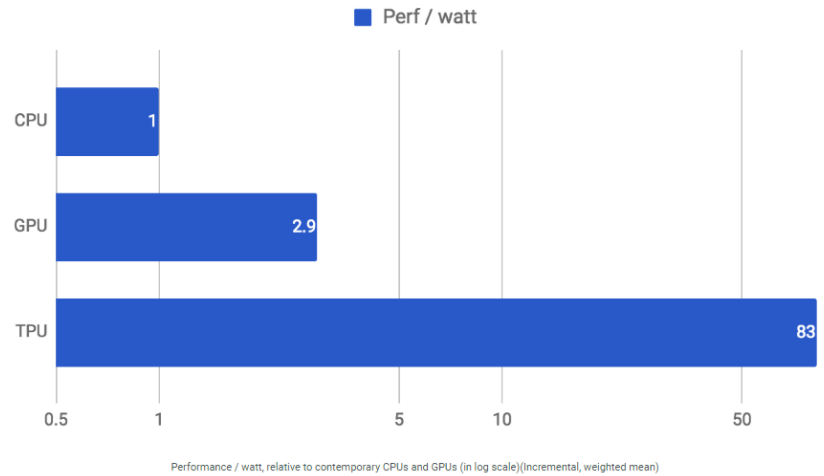
- People who are really serious about software should make their own hardware.

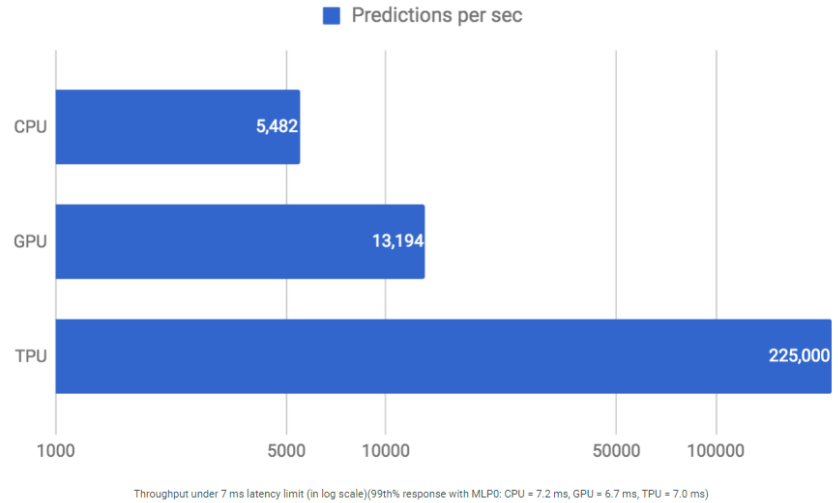
Alan Kay (1982)

Google's Tensor Processing Unit (TPU)

- The TPU includes the following computational resources:
  - **Matrix Multiplier Unit (MXU)**: 65,536 8-bit multiply-and-add units for matrix operations
  - **Unified Buffer (UB)**: 24MB of SRAM that work as registers
  - **Activation Unit (AU)**: Hardwired activation functions
- Controlled by a dozen high-level instructions for neural network inference.







Other ASIC accelerators/PEs

Publication	Throughput [GOPS]	En.Eff. [GOPS/W]	Supply [V]	Area Effic. [GOPS/MGE]
Neuflow [Pham2012]	320	490	1.0	17
EIE [Moons2016]	102	2600	0.5 - 1.1	64
Eyeriss [Chen2016]	84	160	0.8 - 1.2	46
NINEX [Park2016]	569	1800	1.2	51
k-Brain [Park2015]	411	1930	1.2	109
Origami [Cavigelli2015]	196/74	437/803	1.2/0.8	90/34
YodaNN [Andri2016]	1510/55	9800/61200	1.2/0.6	1135/41

Courtesy of prof. Luca Benini - Plenty of Room at the Bottom? Micropower Deep Learning for Cognitive Cyberphysical Systems

Tools

TensorFlow	Google Brain, 2015 (rewritten DistBelief)
Theano	University of Montréal, 2009
Keras	François Chollet, 2015 (now at Google)
Torch	Facebook AI Research, Twitter, Google DeepMind
Caffe	Berkeley Vision and Learning Center (BVLC), 2013

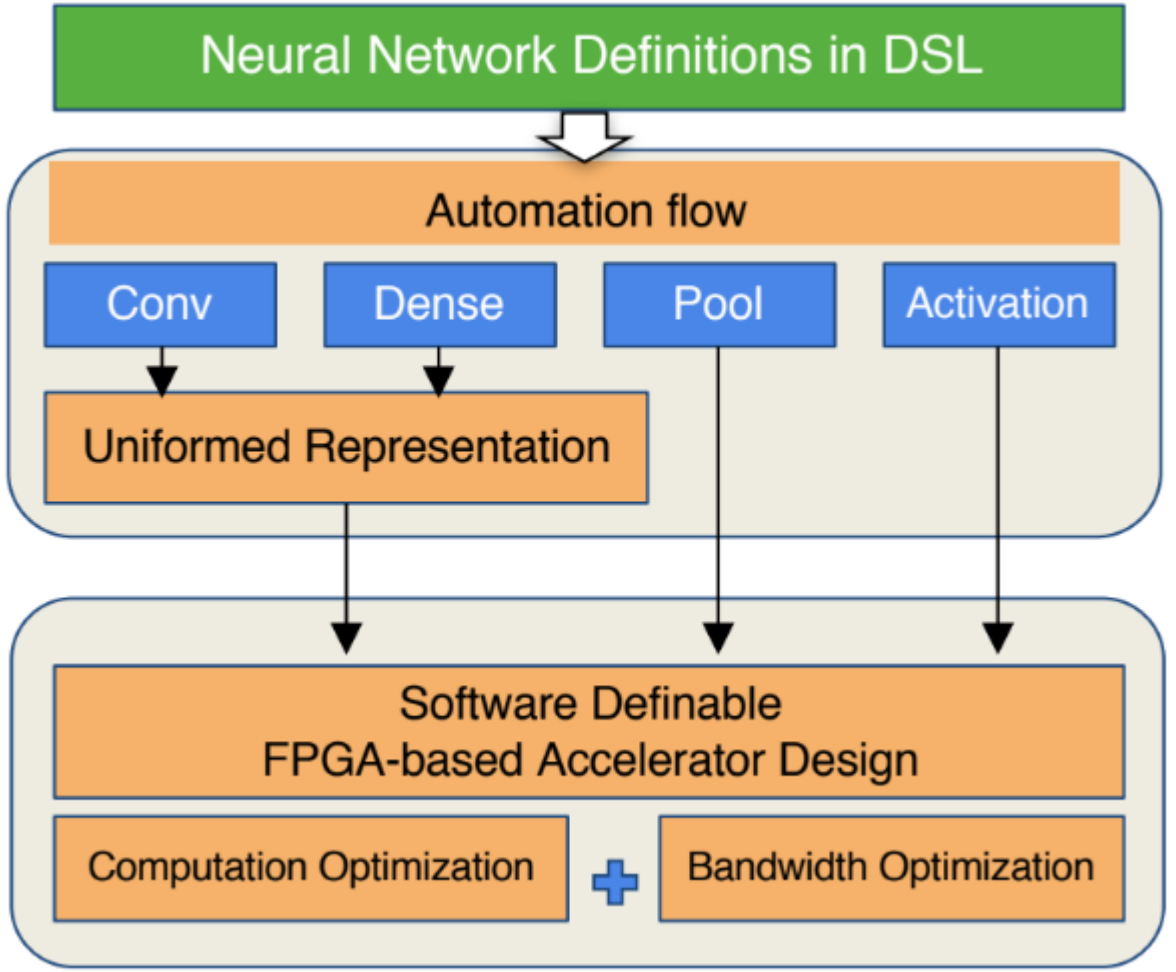
```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 0
    decay_mult: 0
  }
  convolution_param {
    num_output: 64
    kernel_size: 3
    pad: 1
   }
}
```

```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "fc8"
  bottom: "label"
  top: "loss"
}
```

- Platform-specific Implementation tools
- The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.
  - highly tuned implementations fo forward and backward convolution, pooling, normalization, and activation layers.
  - part of the NVIDIA Deep Learning SDK.
- Xilinx Deep Neural Network (xfDNN) library is highly optimized for building deep learning inference applications. Designed for maximum compute efficiency at 16-bit and 8-bit integer data types.

Holistic flows

- Caffeine: Towards Uniformed Representation and Acceleration for Deep Convolutional Neural Networks, Zhang et al. ICCAD'16, Nov 2016



Coming soon - ALOHA