From Model to FPGA: Software-Hardware Co-Design for Efficient Neural Network Acceleration

Kaiyuan Guo[1,2], Lingzhi Sui[1], Jiantao Qiu[2], Song Yao[1], Song Han[1,3], Yu Wang[1,2], Huazhong Yang[1]

[1] DeePhi Technology [2] Tsinghua University, [3] Stanford University

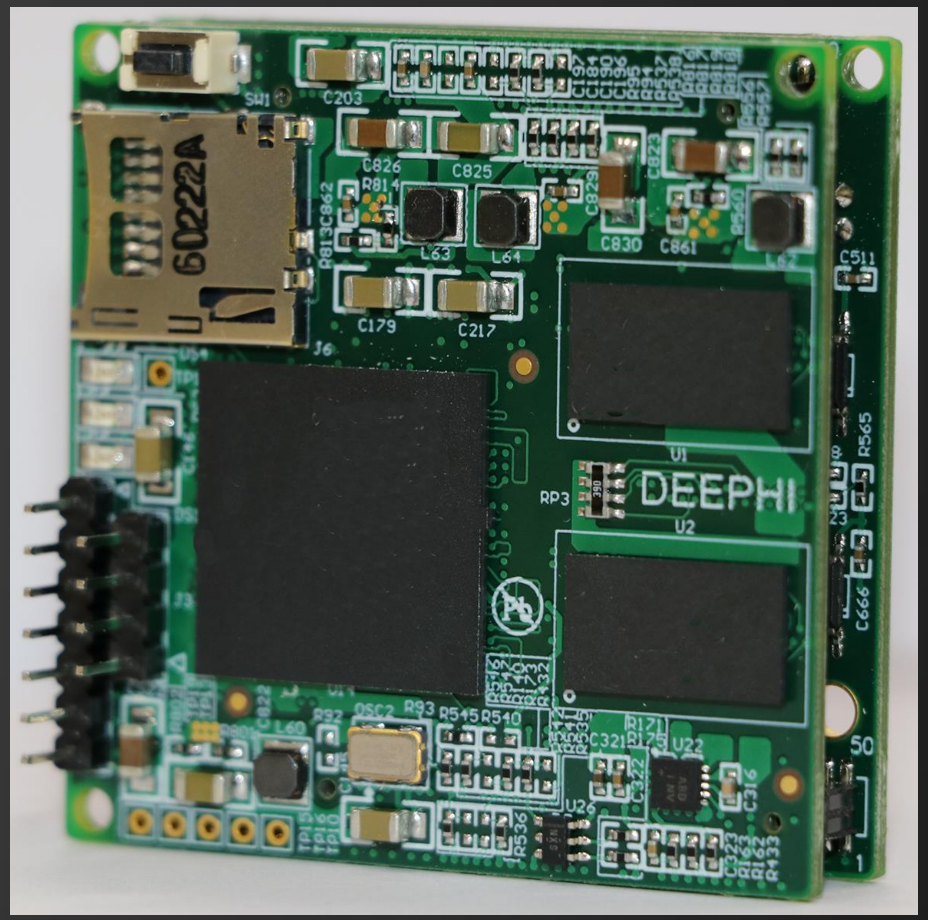
Acknowledgement: Dongliang Xie and DeePhi Engineering Team

DeePhi Tech

- *Discovering the philosophy behind deep learning computing*
- Founded by **Song Yao**, Yu Wang, and Song Han in March 2016

- FPGA-based solution provider for deep learning

Algorithm Development → Compression + Compilation →



✓ Automatic compilation tool chain + mini board/IP
✓ Architecture for CNN and RNN-LSTM
✓ Supporting detection, tracking, object/speech recognition, translation, and etc.

Overview

- **New Platform Expected for Deep Learning**
- **Trend in Neural Network Design**
- **Platform Selection**
- **Overall Flow**
- **Model Compression: Useful in Real-World Networks**
- **Activation Quantization: 8 Bits Are Enough**
- **Aristotle: Architecture for CNN Acceleration**
- **Descartes: Architecture for Sparse LSTM Acceleration**
- **Conclusion**

New Platform Expected for Deep Learning

Drone

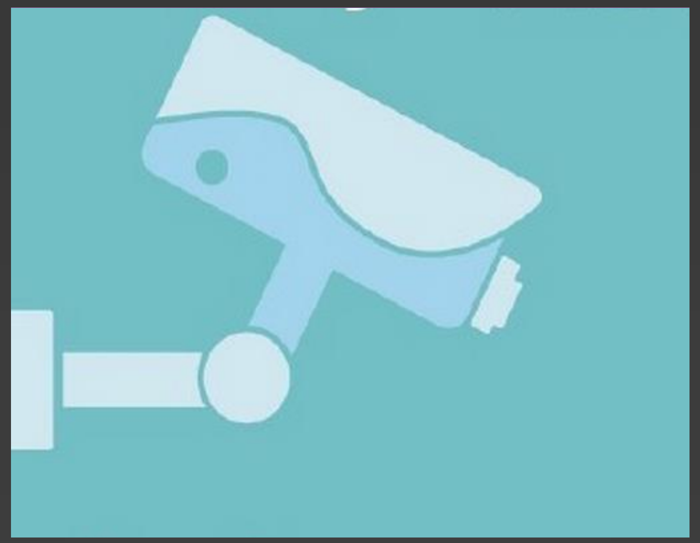
Client

Requirements

Real-time object recognition

Limitation

Battery capacity



Video Surveilliance

Edge

Requirements

Real-time video analysis

Limitation

High maintenance cost

Speech Recognition

Cloud

Requirements

Low latency

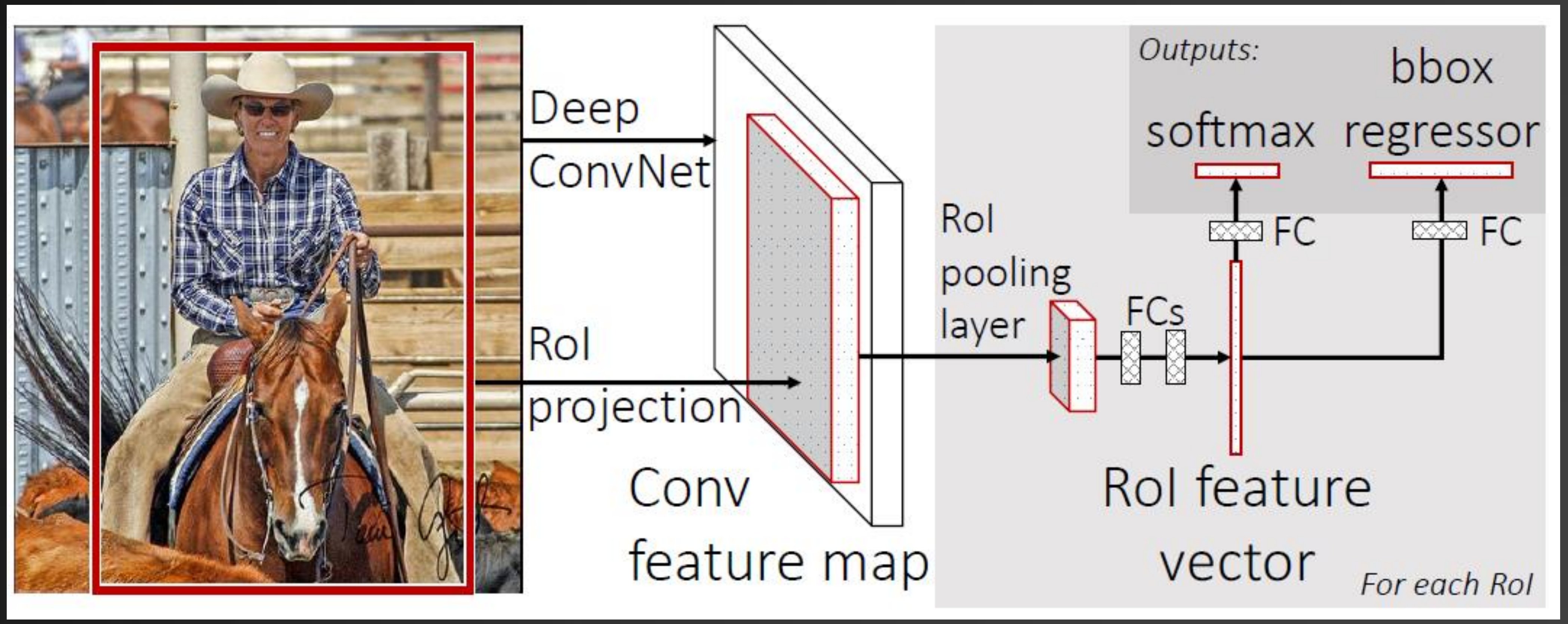
Limitation

High maintenance/cooling cost

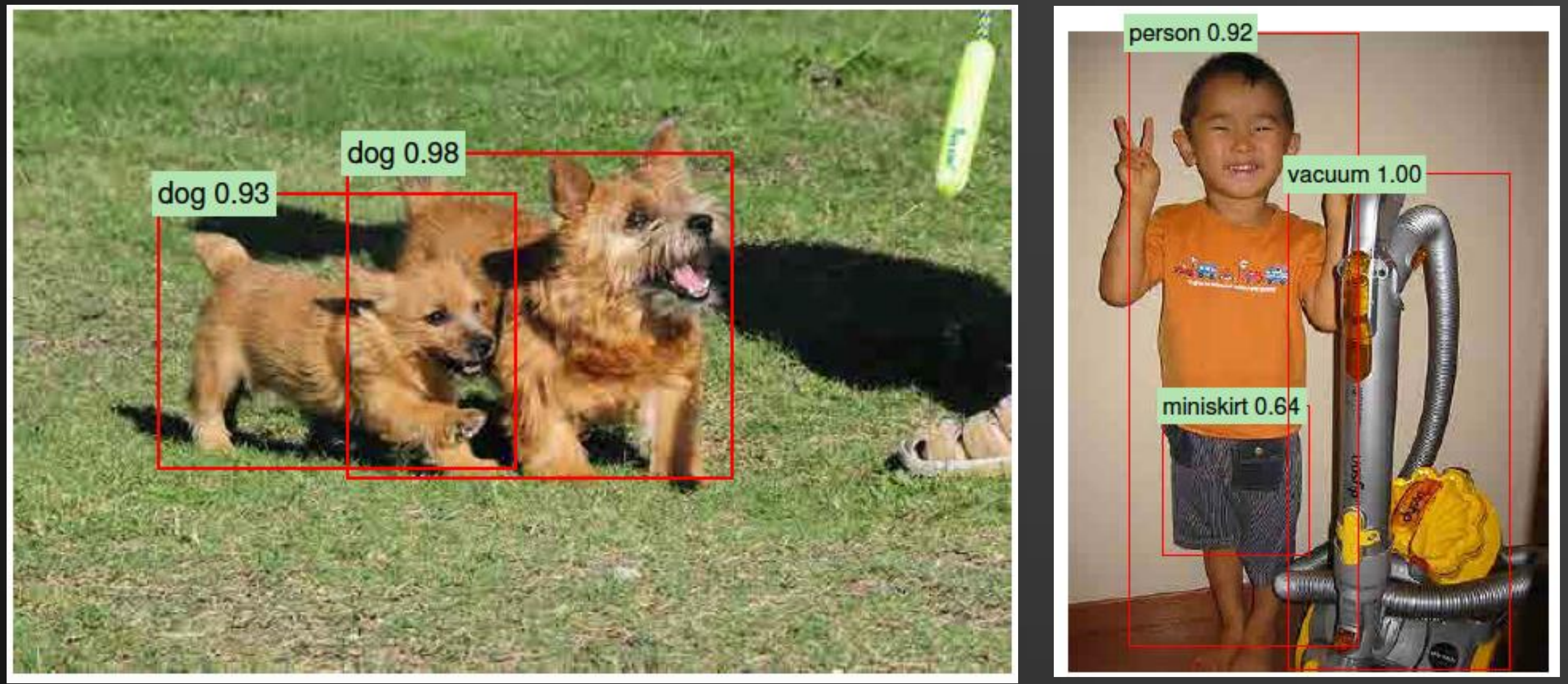
Low-power high-performance platform for deep learning is urgently needed

Trend in Neural Network Design

- CNN for Object Recognition

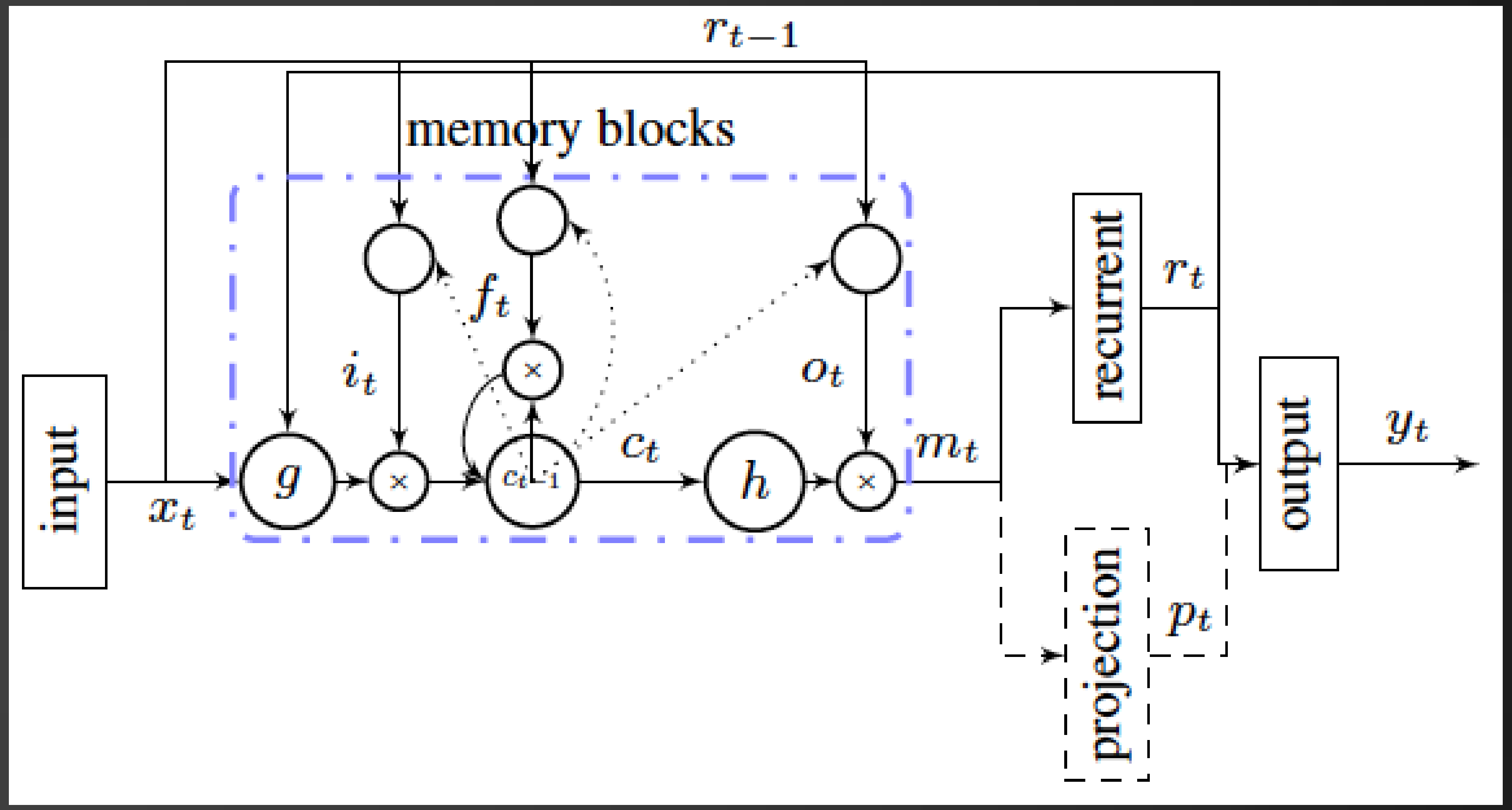


Source: Ross Girshick, “Fast R-CNN”



Source: Ross Girshick et al., “R-CNN”

- RNN-LSTM for Speech Recognition

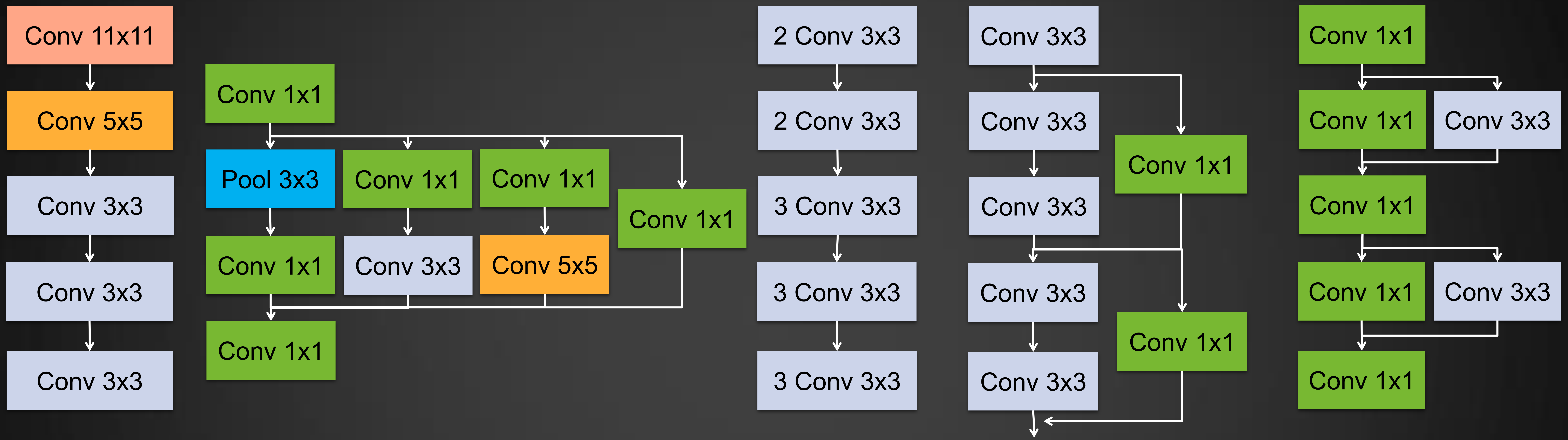


Source: Hasim Sak et al.,” Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition”

Frameworks for different applications have not been unified

Trend in Neural Network Design

- CNN: Smaller and Slimmer



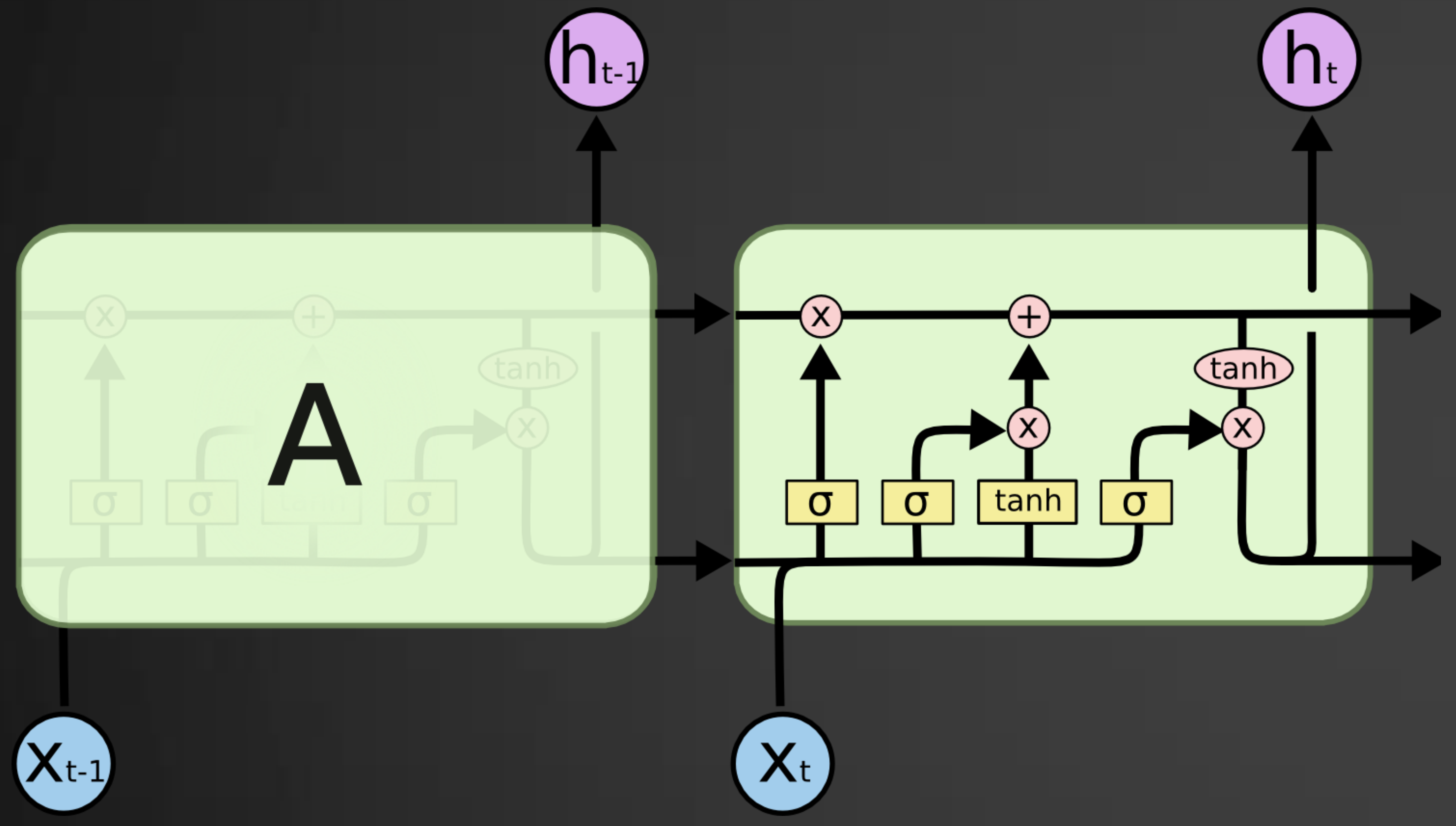
2012	2014	2014	2015	2016
AlexNet	GoogLeNet	VGG16	ResNet	SqueezeNet
84.7%	90.6%	90%	96.4%	84.7%

- Smaller: One convolution kernel has fewer computations
- Slimmer: fewer channels, fewer computations, less parallelism

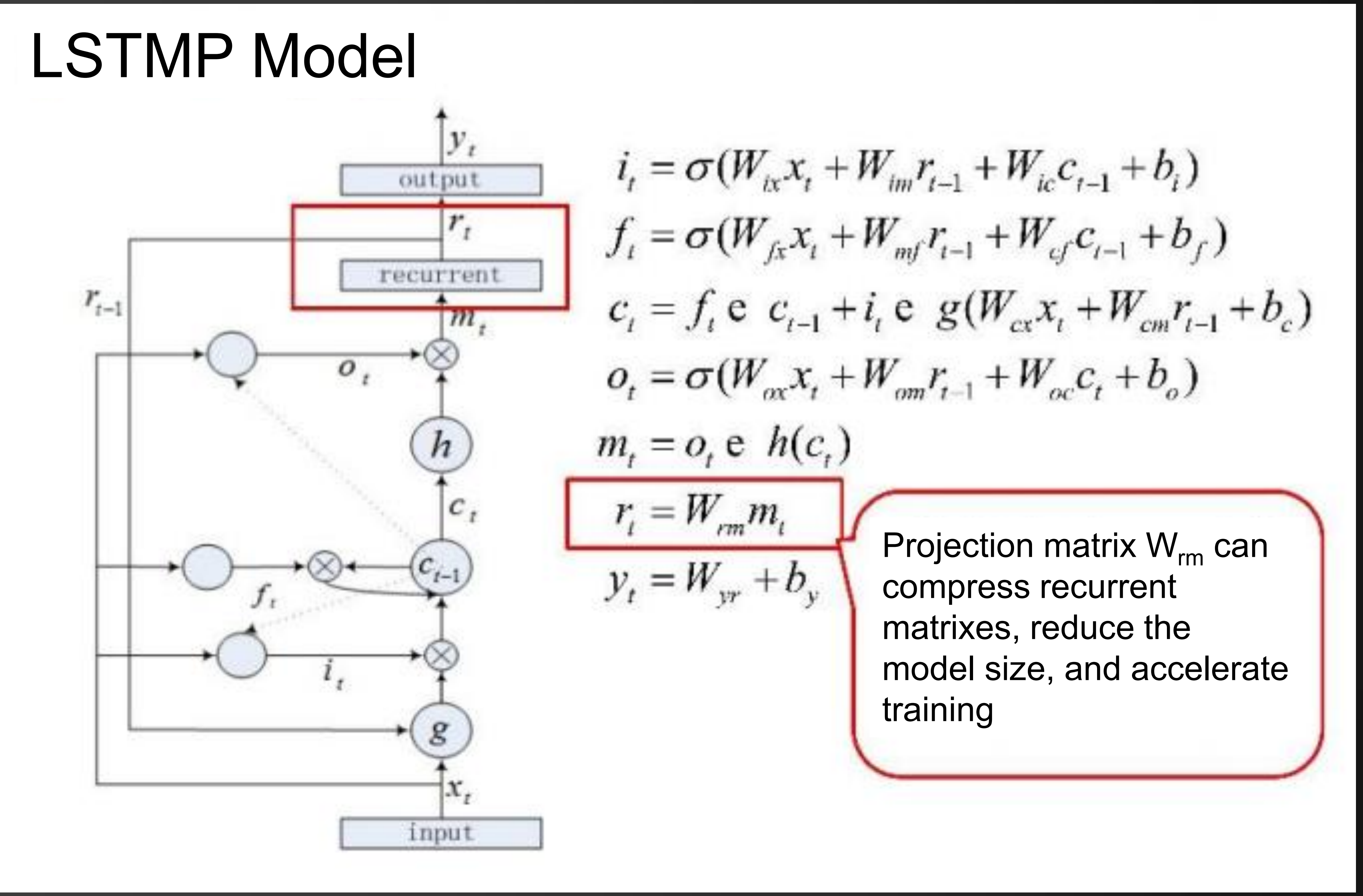
A CNN accelerator should perform better with small Conv kernels and low parallelism

Trend in Neural Network Design

- RNN-LSTM: Larger and Deeper
  - Max dimension: 128 → 256 → 512 → 1024 → 2048 -> 4096
  - Number of LSTM layers: 1 → 3 → 5



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/



Source: Lei Jia et al., Baidu

- Larger model size, higher bandwidth requirement
- An RNN-LSTM accelerator should overcome the bandwidth problem

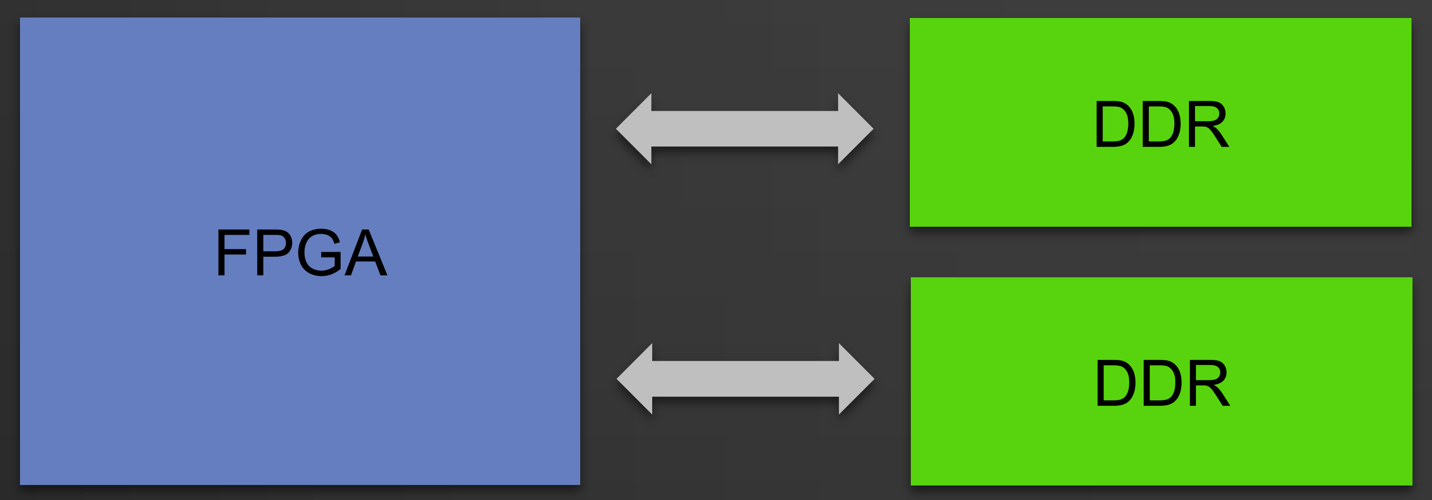
FPGA is good for inference applications

- CPU: Not enough energy efficiency
- GPU: Extremely efficient in training, not enough efficiency in inference (batch size = 1)
- DSP: Not enough performance with high cache miss rate
- ASIC has high NRE: No clear huge market yet
- ASIC has long time-to-market but neural networks are in evolution
- FPGA
  - Acceptable power and performance
  - Supports customized architecture
  - High on-chip memory bandwidth
  - Relatively short time to market
  - High reliability

FPGA-based deep learning accelerators meet most products' requirements

Software-Hardware Co-Design is Necessary

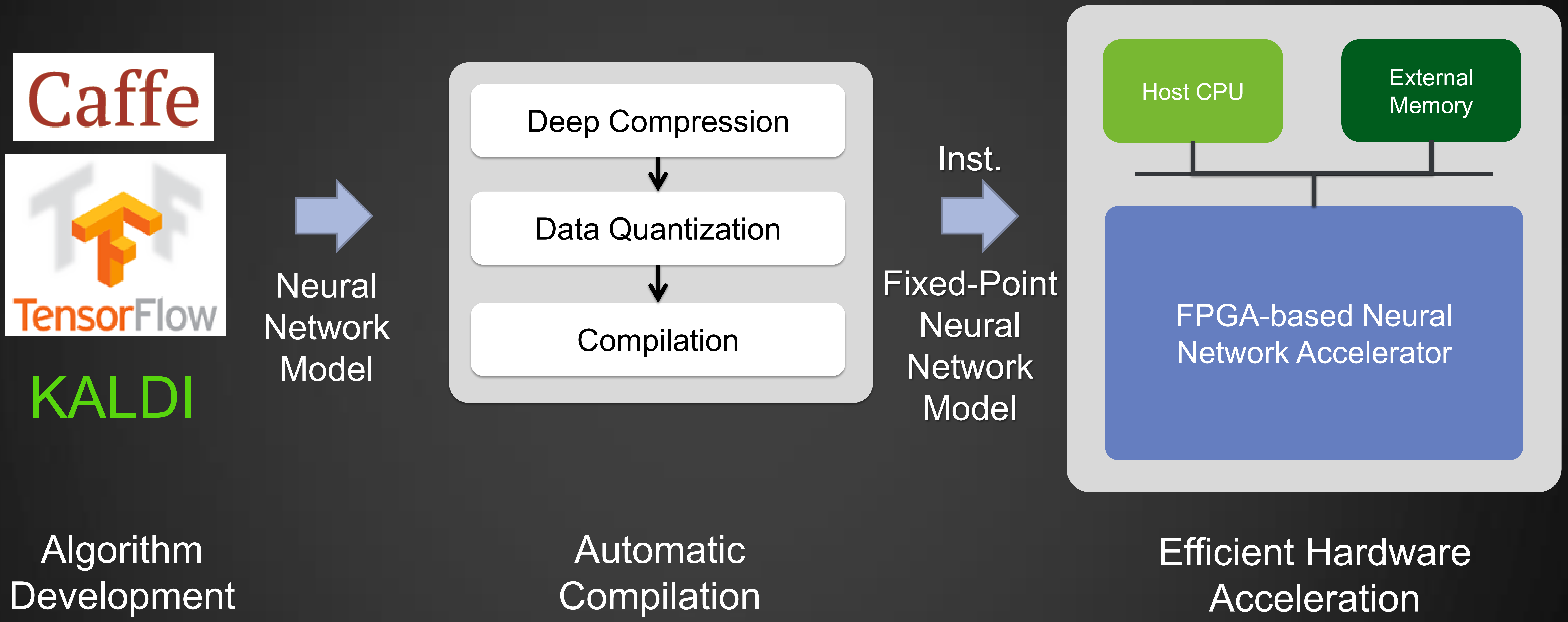
- Great redundancy in neural networks
  - VGG16 network can be compressed from 550MB to 11.3MB
  - FPGA has limited BRAM and DDR bandwidth
- Different neural network has different computation pattern
  - CNN: Frequent data reuse, dense
  - DNN/RNN/LSTM: No data reuse, sparse
  - Different architectures must adapt to different neural network
- Neural networks are in evolution
  - Architecture must adapts to new algorithms



Limitations of FPGA platform

- Limited BRAM size
- Limited DDR Bandwidth

Overall Flow



Algorithm engineers can simply run the compiler tool to implement FPGA acceleration

Overall Flow

Traditional FPGA-based Acceleration Faced Two Major Problem

- Long development period
  - Hand coded: 2 – 3 months
  - OpenCL and HLS: 1 month
- Insufficient performance and energy efficiency

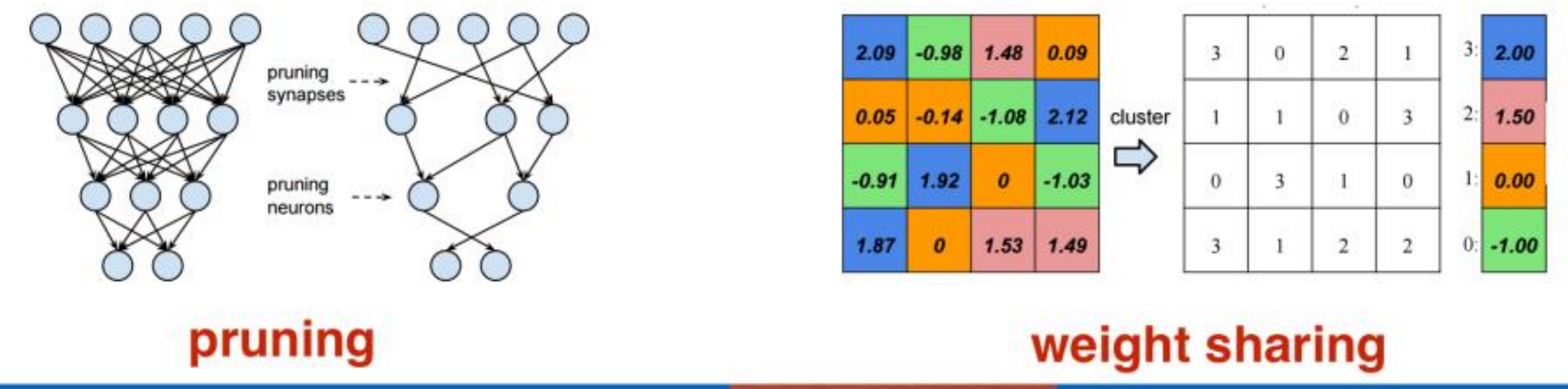
DeePhi's workflow solves the two problems in FPGA acceleration

- Compiler + Architecture instead of OpenCL
  - Algorithm designer need to know nothing about hardware
  - Generates instructions instead of RTL code
  - Compilation in 1 minute
- Much higher performance and energy efficiency
  - Hand-coded IP core and efficient architecture design

Model Compression: Useful in Real-World Netoworks

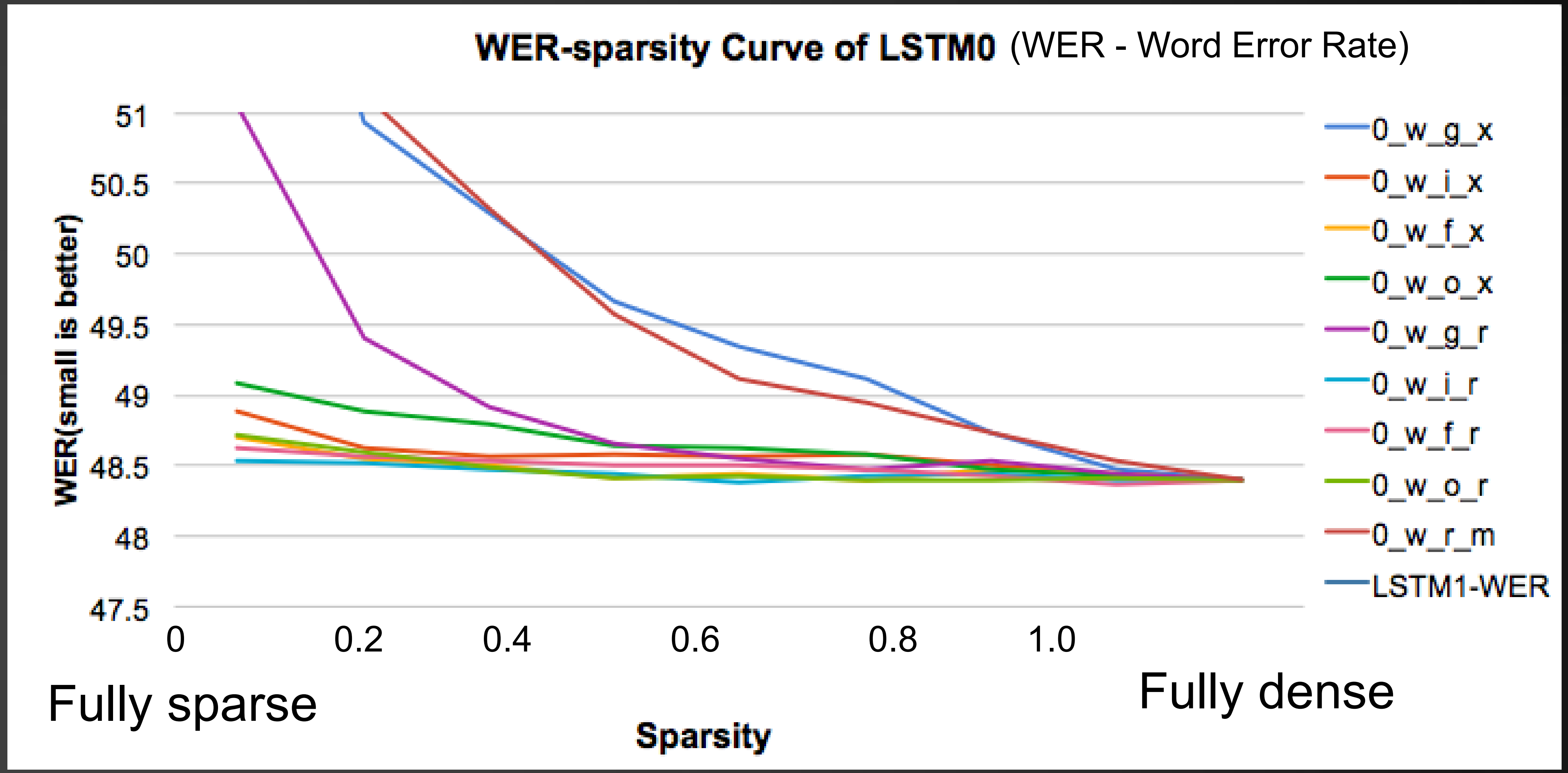
- Deep Compression: Useful for RNN-LSTM and FC layers in CNN

- **Small DNN models are critical.**



Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
AlexNet	240MB	→ 6.9MB	**35x**	80.27%	→ 80.30%
VGGNet	550MB	→ 11.3MB	**49x**	88.68%	→ 89.09%
GoogleNet	28MB	→ 2.8MB	**10x**	88.90%	→ 88.92%
SqueezeNet	4.8MB	→ 0.47MB	**10x**	80.32%	→ 80.35%

Source: Song Han et al., Stanford University



Different gate in LSTM has different sensitivity

With re-training, we can achieve:

- < 10% sparsity for real-world FC layers in CNN
- ~ 15% sparsity for real-world LSTMs
- 4 bit weight quantization with no accuracy loss

Deep Compression is useful in real-world neural networks and can save a great deal of computations and bandwidth demands

Activation Quantization: 8 Bits Are Enough

- Image classification on ILSVRC 2012

		FP32	FIXED-16		FIXED-8	
		ORIGINAL	RAW	RE-TRAIN	RAW	RE-TRAIN
VGG16	Top-1	65.77%	65.78%	67.84%	65.58%	67.72%
	Top-5	86.64%	86.65%	88.19%	86.38%	88.06%
GoogLeNet	Top-1	68.60%	68.70%	68.70%	62.75%	62.75%
	Top-5	88.65%	88.45%	88.45%	85.70%	85.70%
SqueezeNet	Top-1	58.69%	58.69%	58.69%	57.27%	57.27%
	Top-5	81.37%	81.35%	81.36%	80.32%	80.35%

- Object detection on PASCAL VOC 2007
  - R-FCN: < 2% mAP loss without re-training using 8-bit quantization
  - YOLO: < 1% mAP loss without re-training using 8-bit quantization

Activation Quantization: 8 Bits Are Enough

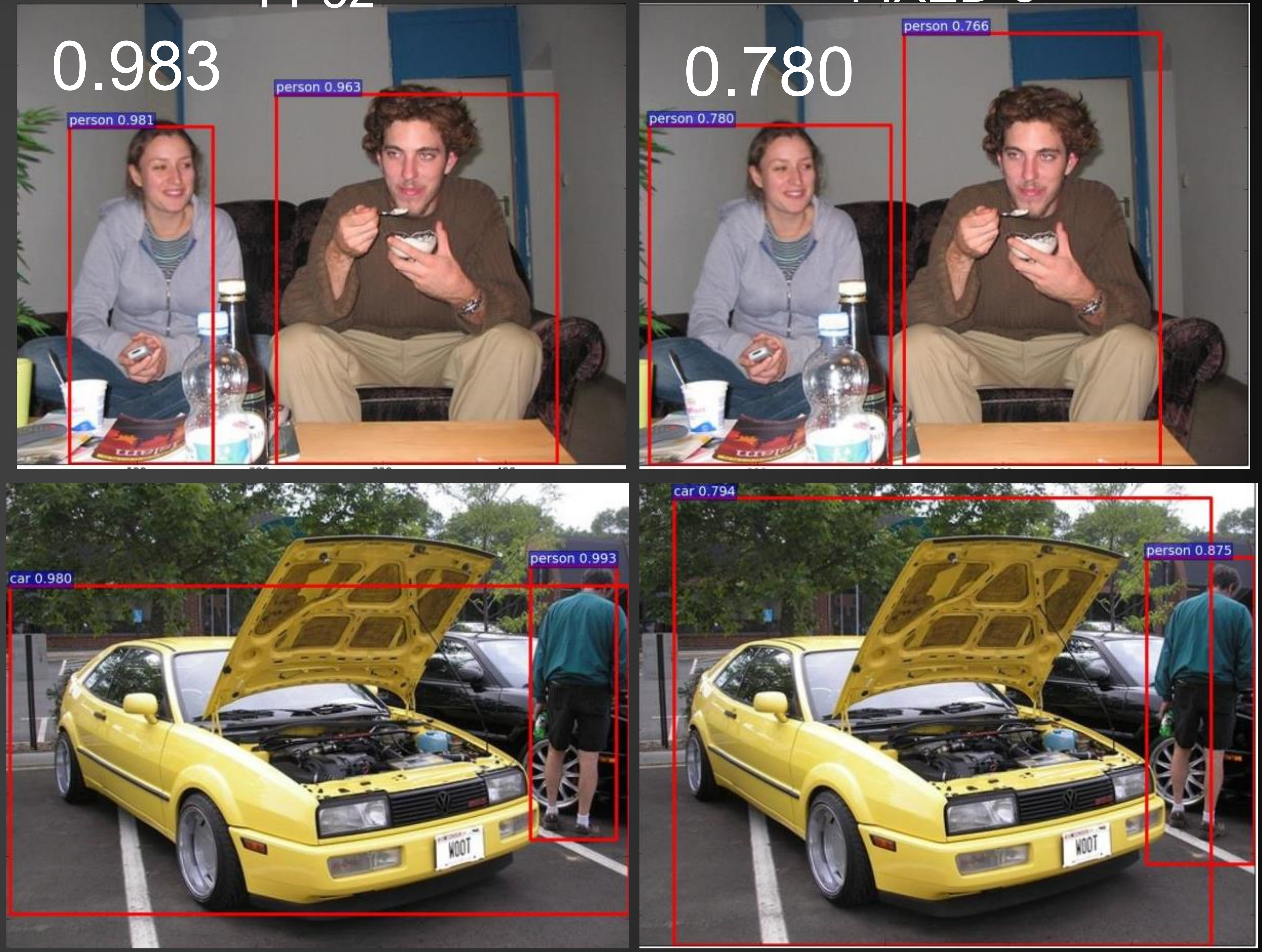
- Image classification: Results comparison

GoogLeNet		SqueezeNet		VGG16	
FP32	FIXED-8	FP32	FIXED-8	FP32	FIXED-8
Shetland Sheepdog	Shetland Sheepdog	Shetland Sheepdog	Shetland Sheepdog	Shetland Sheepdog	Shetland Sheepdog
Collie	Collie	Collie	Collie	Collie	Collie
Borzoi	Borzoi	Border collie	Papillon	Borzoi	Borzoi
Afghan hound	Pomeranian	Afghan hound	Border collie	Afghan hound	Papillon
Pomeranian	Afghan hound	Papillon	Pomeranian	Papillon	Australian terrier

- Most differences are in low-priority guesses

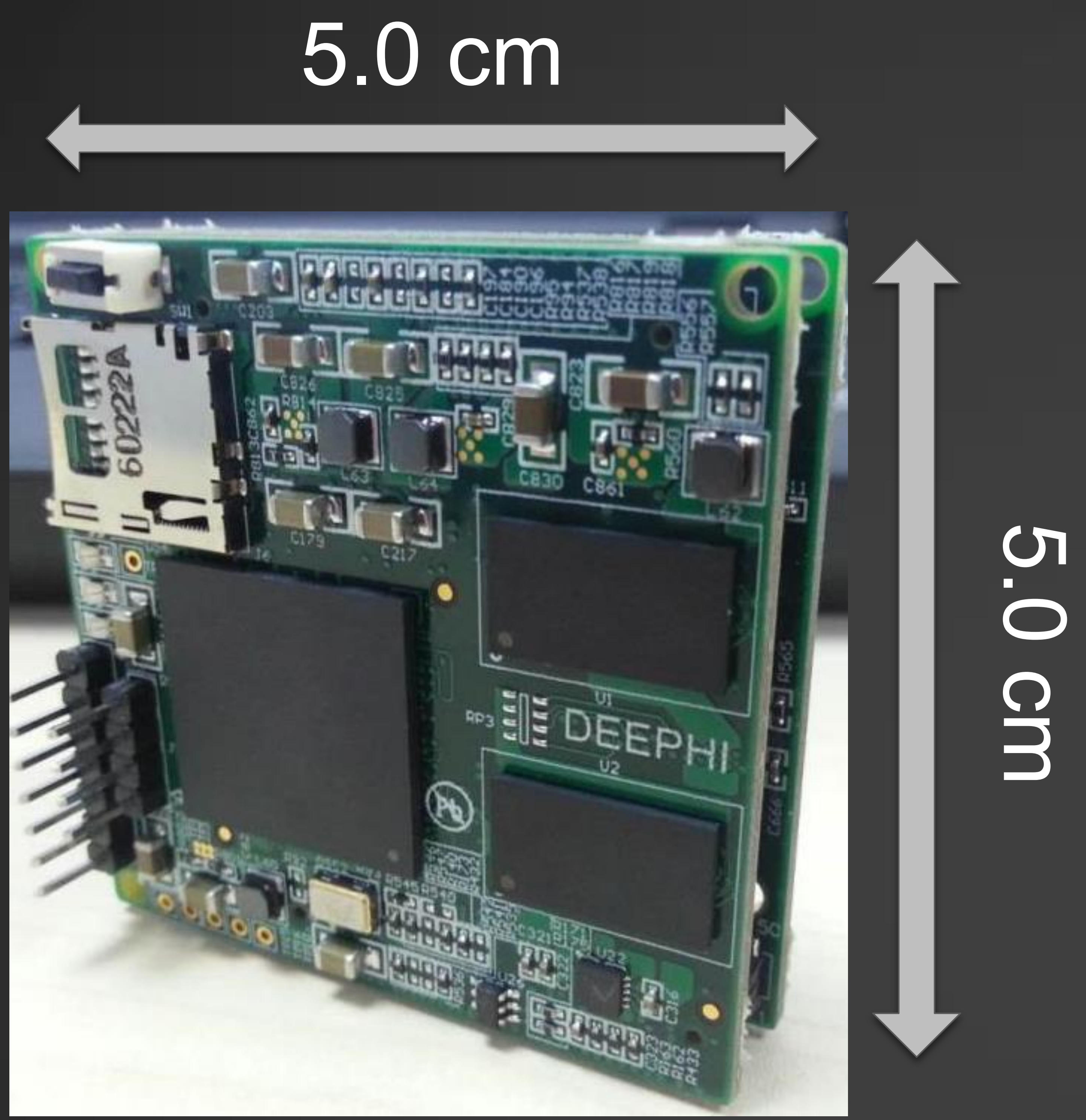
- Object detection: Results comparison
  - SqueezeNet + R-FCN

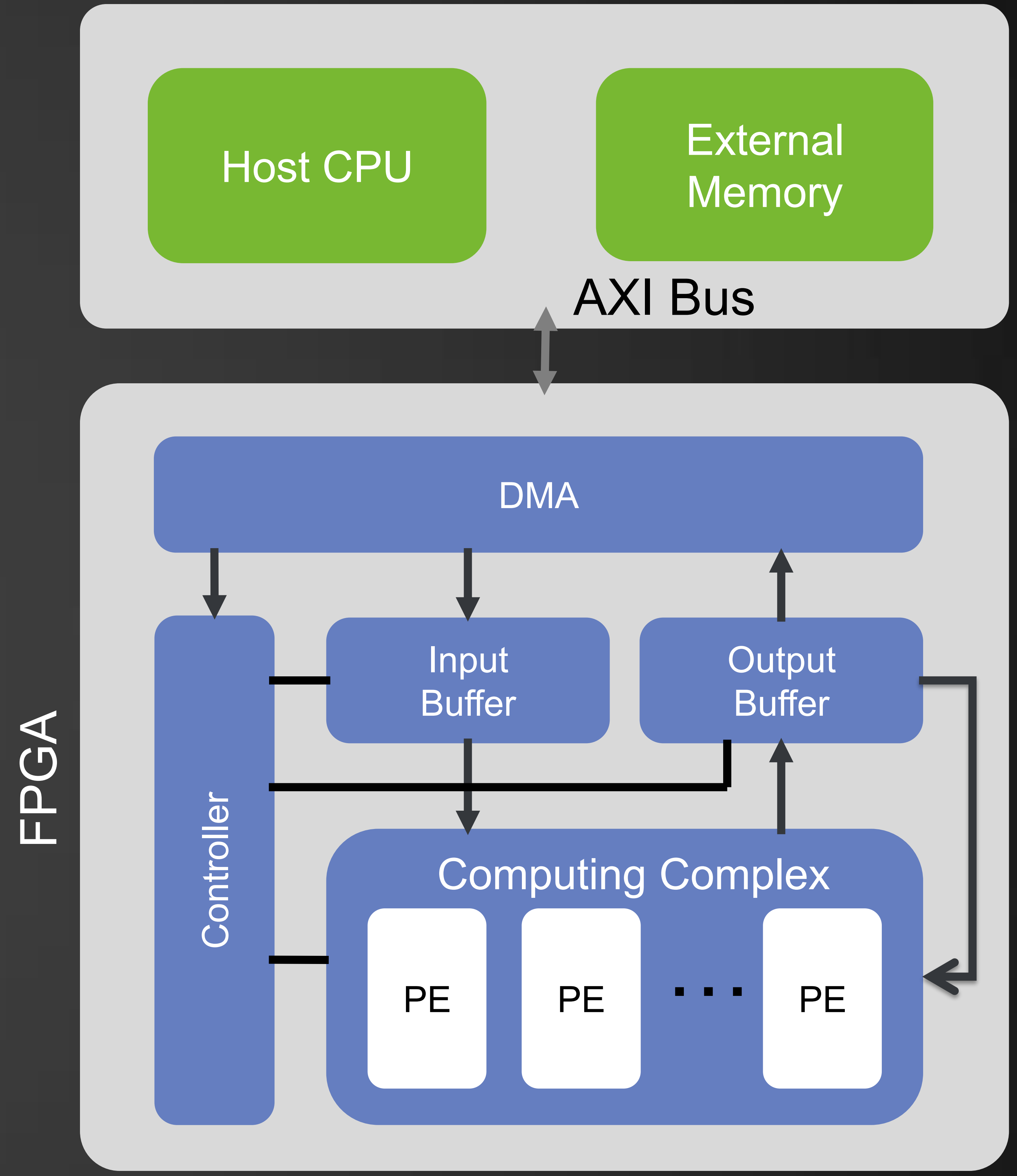
FP32 FIXED-8



- Similar proposal results with lower confidence

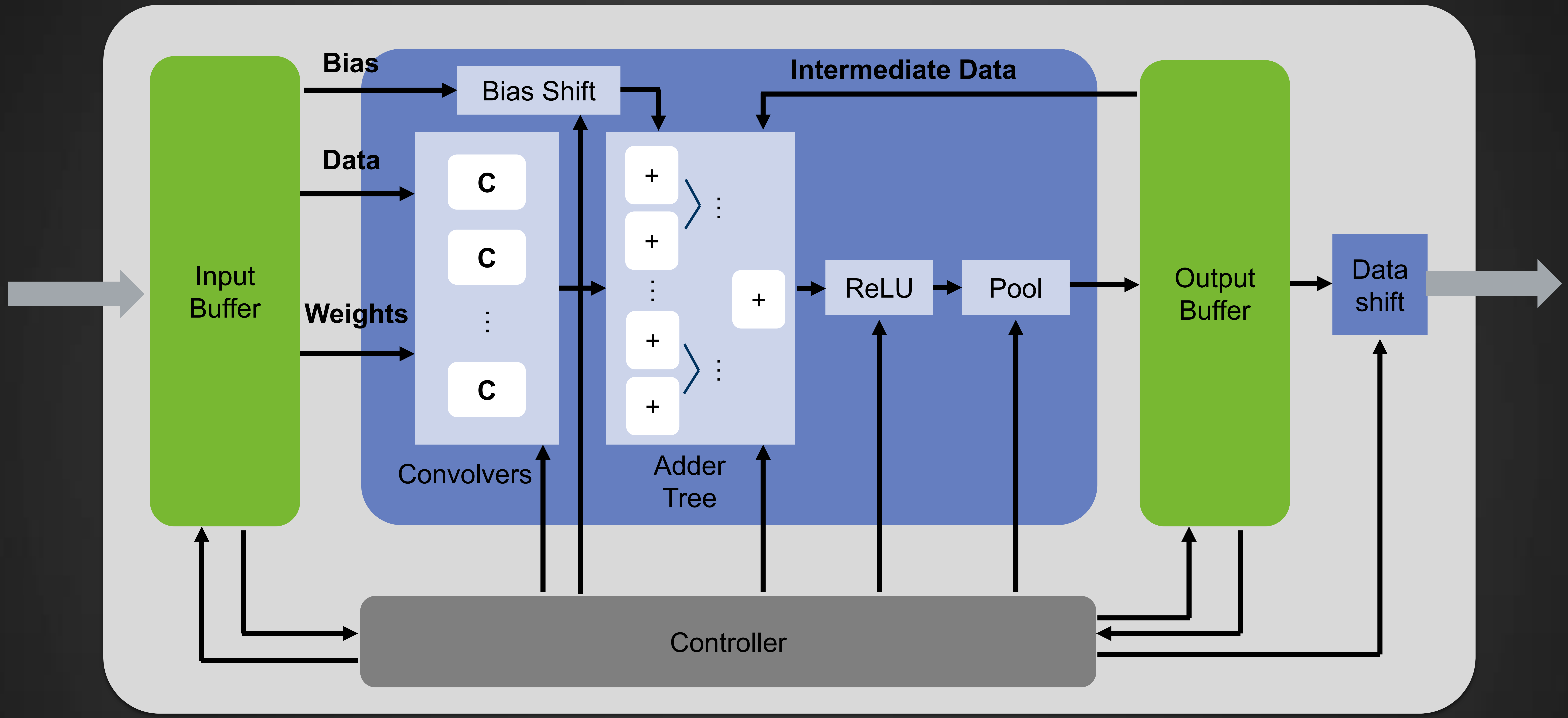
Aristotle: Architecture for CNN Acceleration





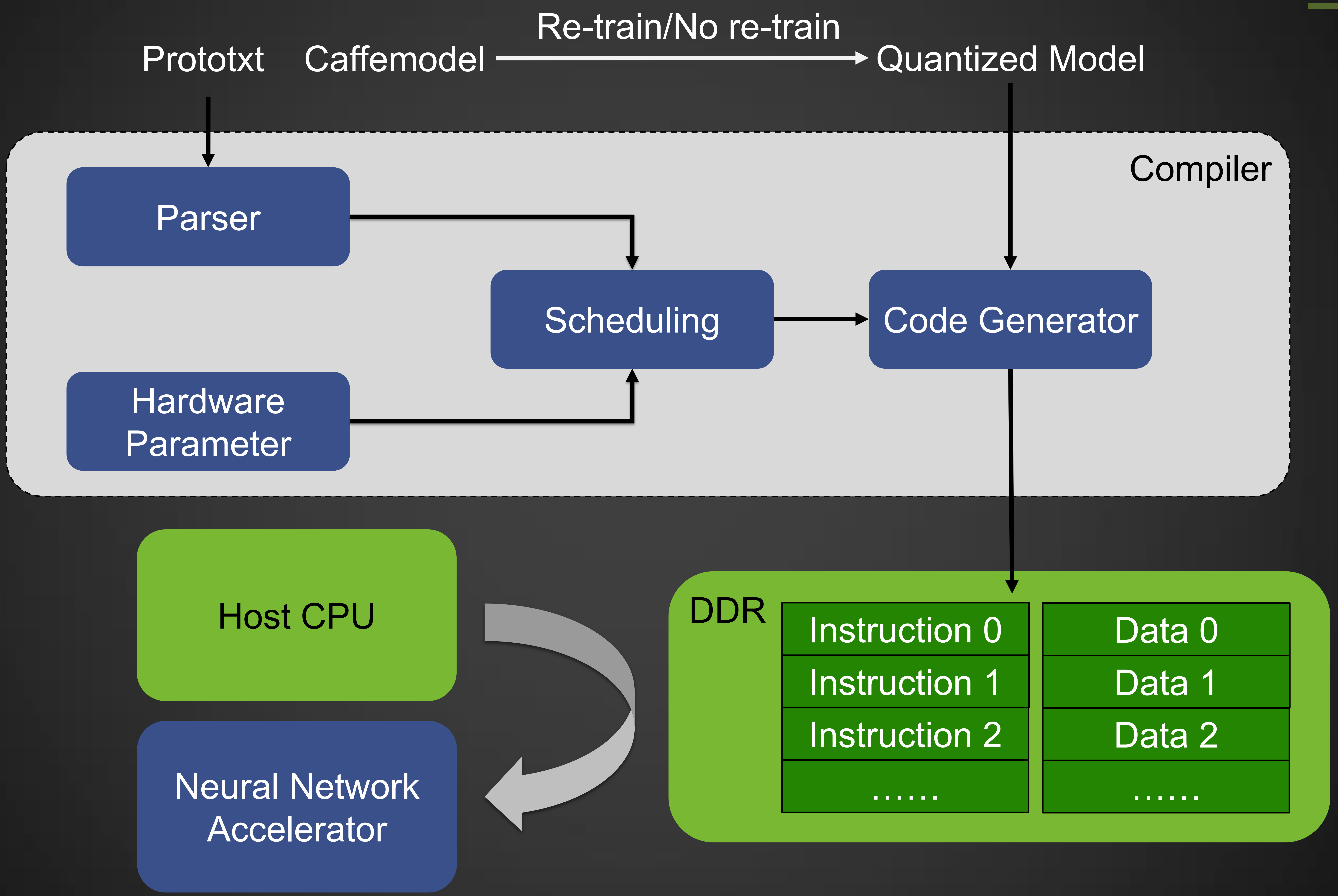
- Based on Zynq 7000 Series FPGA
- Optimized for 3x3 Conv kernels
- Supports different Conv stride sizes
- <u>Scalable</u> design (1PE, 2PE, 4PE, 12PE) on Zynq 7010/7020/7030/7045
- Supports mainstream deep learning object framework: R-FCN, YOLO, and etc

Aristotle: Processing Element Architecture



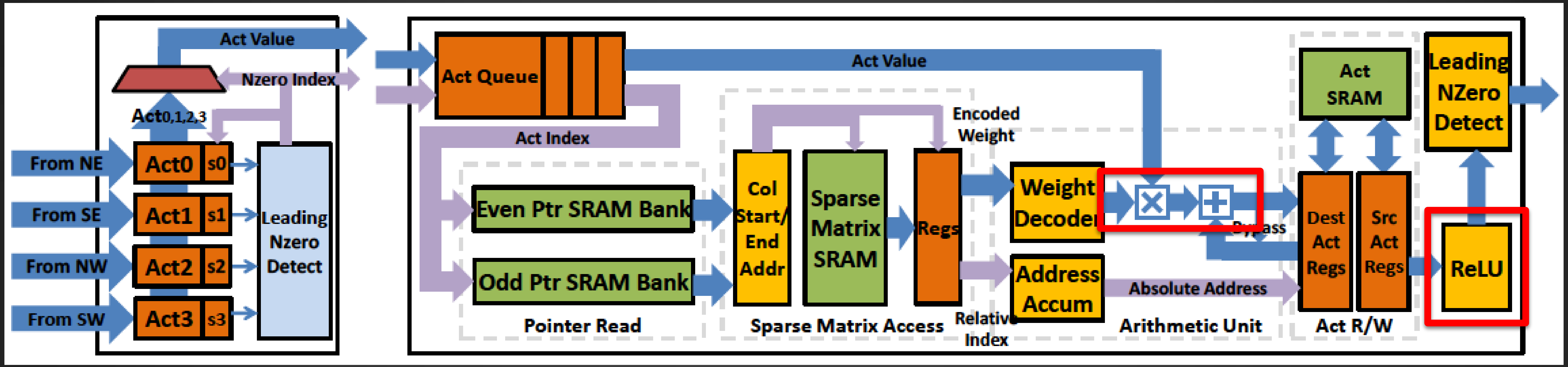
- Integrate convolvers, adder tree, non-linearity, and pooling units into one PE
- Fully pipeline without intermediate data load/store
- Supports dynamic-precision quantization

From Model to Instructions



Descartes: Architecture for Sparse LSTM Acceleration

- EIE (Efficient Inference Engine): Extremely efficient, but not for FPGA
  - Designed by Song Han et al. from Stanford University and published on ISCA 2016
  - 102 GOPS@600 mW, 800MHz

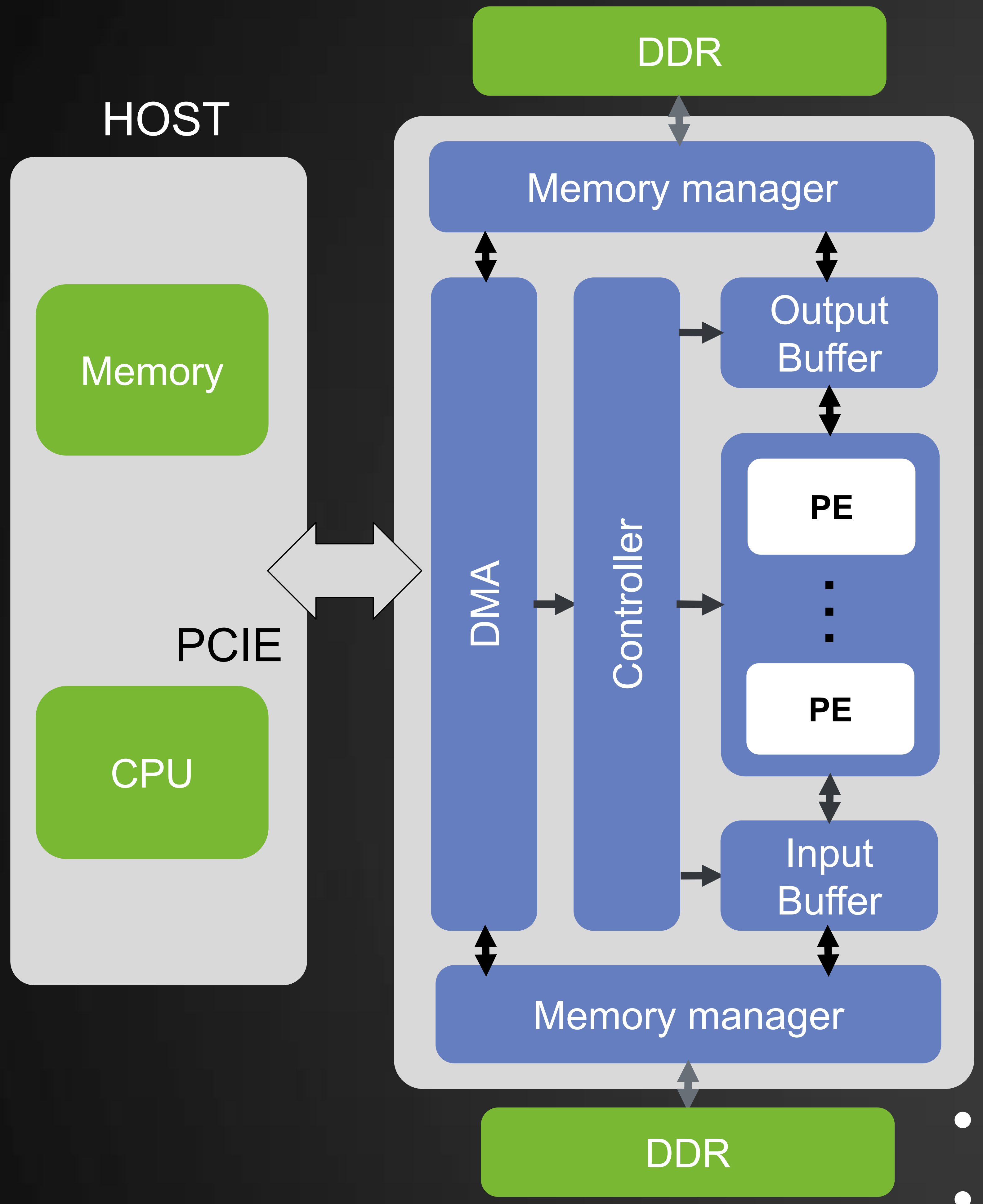


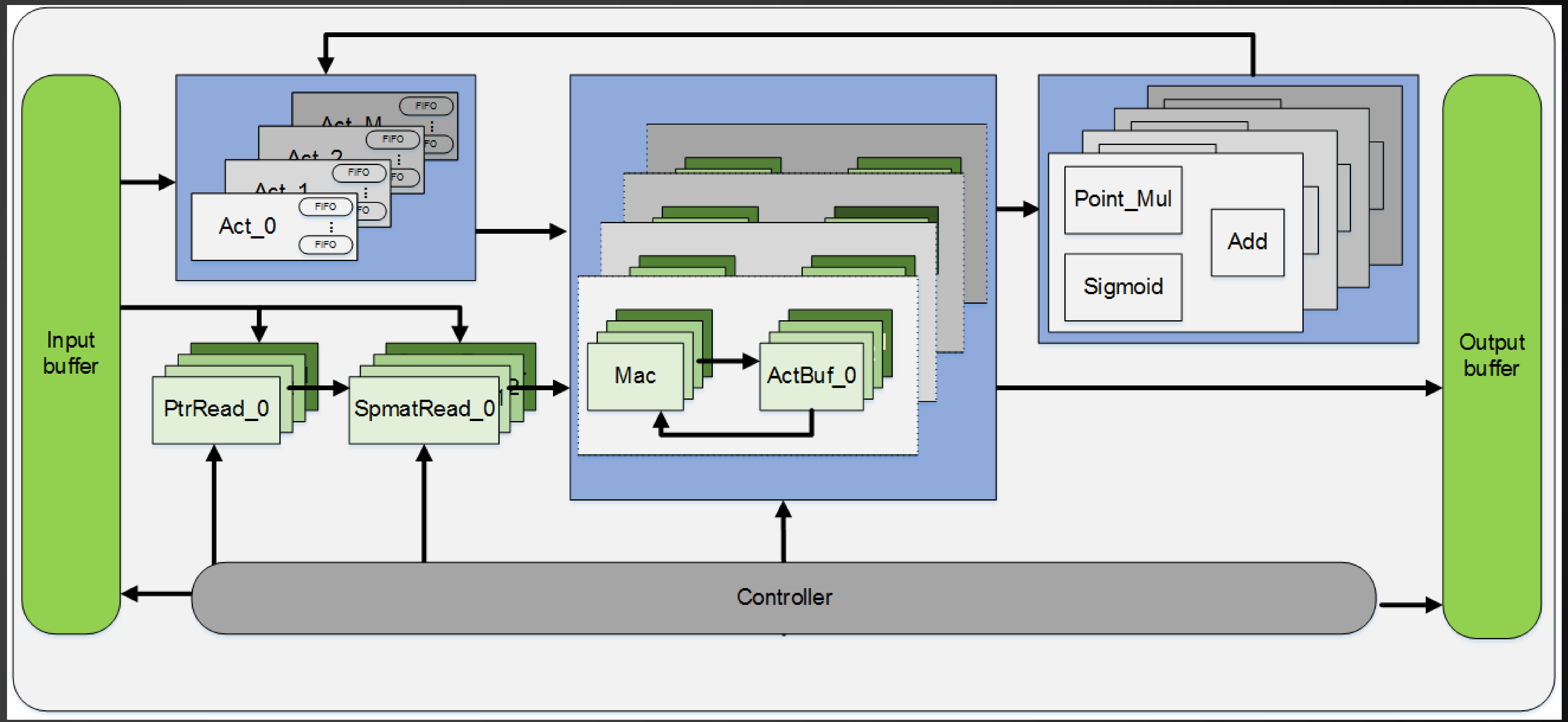
EIE chip (64PE)	Xilinx KU060	Xilinx KU115
• 10.13 MB SRAM	• 4.75 MB BRAM	• 9.49MB BRAM
• 64 Multiplier	• 2760 DSP	• 5520 DSP
• 800MHz	• 250-300MHz	• 250-300MHz

- FPGA has significantly more computing units but strictly limited on-chip memory
- LSTM cannot utilize activation sparsity

FPGA

Descartes: Architecture for Sparse LSTM Acceleration

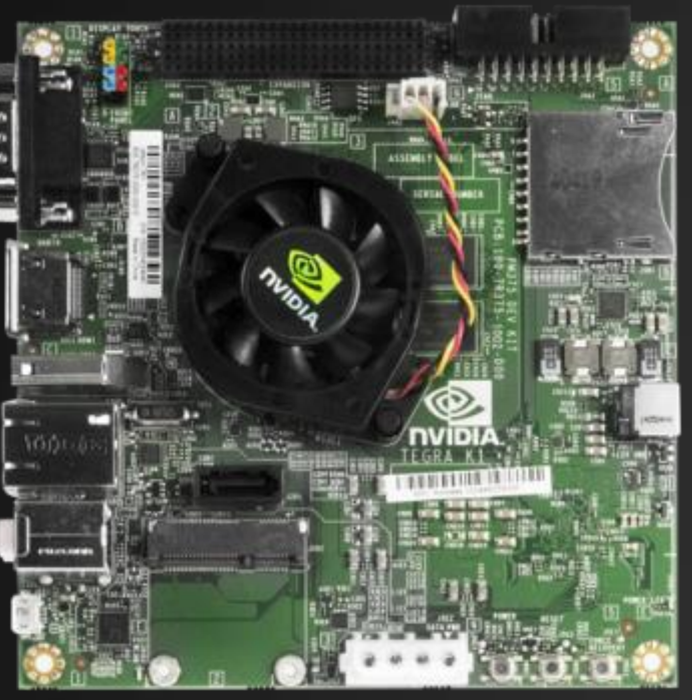




- Designed for LSTM: Supports any matrix size and layer number
- Supports any sparsity
- Considers scheduling and non-linear functions in LSTM
- Scalable design (16/32/64 PEs for each thread)
- Two modes: Batch (high throughput) / No Batch (low latency)

Evaluation: Platform and Benchmark for CNN

- Platform Comparison



- Nvidia Tegra K1 SoC
  - 28 nm
  - ARM Cortex-A15 CPU
  - Kepler GPU 192 Cores
  - Caffe with CuDNN

- Xilinx Zynq 7000 Series
  - 28nm
  - 85k/125k/350k logic cells (7020/30/45)
  - 220/400/900 DSP (7020/30/45)
  - 4.9/9.3/19.1Mb BRAM (7020/30/45)

- Benchmark

2 x 64 Conv → 2 x 128 Conv → 3 x 256 Conv → 3 x 512 Conv → 3 x 512 Conv

VGG16

Image classification

30.68 Gop 13 Conv layers

16 Conv → 32 Conv → 64 Conv → 128 Conv → 256 Conv → 512 Conv → 3 x 1024 Conv

YOLO Tiny

General object detection

5.54 Gop, 9 Conv layers

2 x 16 Conv → 2 x 32 Conv → 48 Conv → 2 x 64 Conv → 80 Conv → 128 Conv

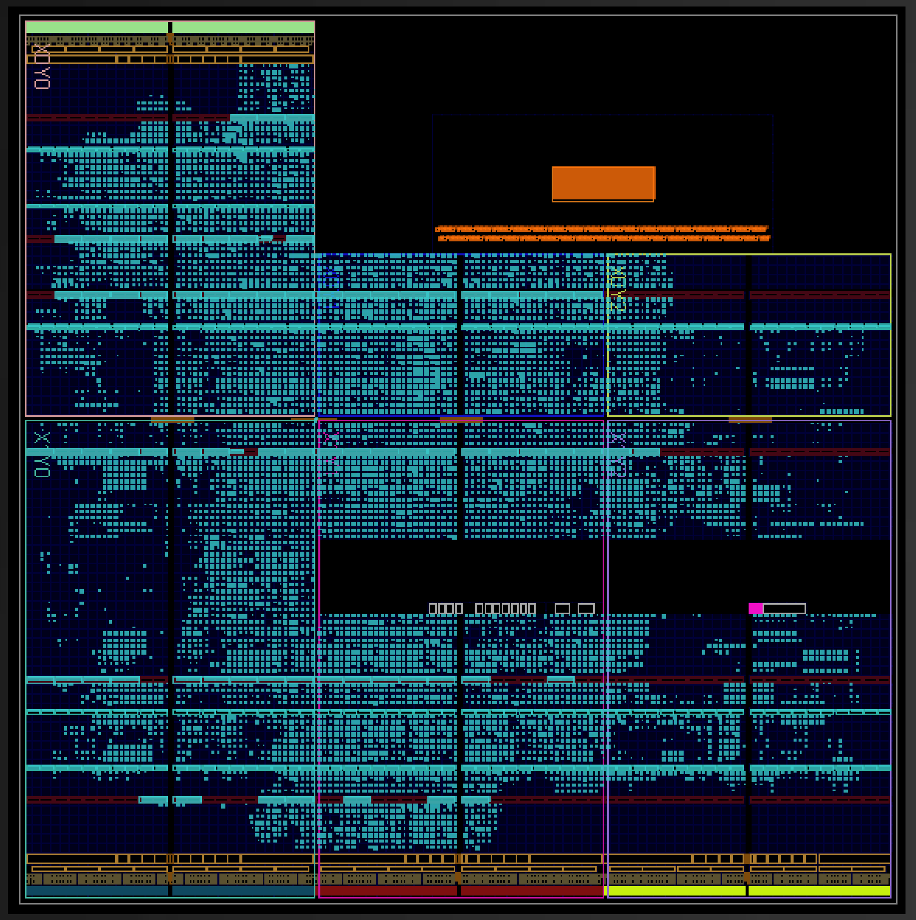
Customized Network

Face alignment

104.6 Mop, 9 Conv layers

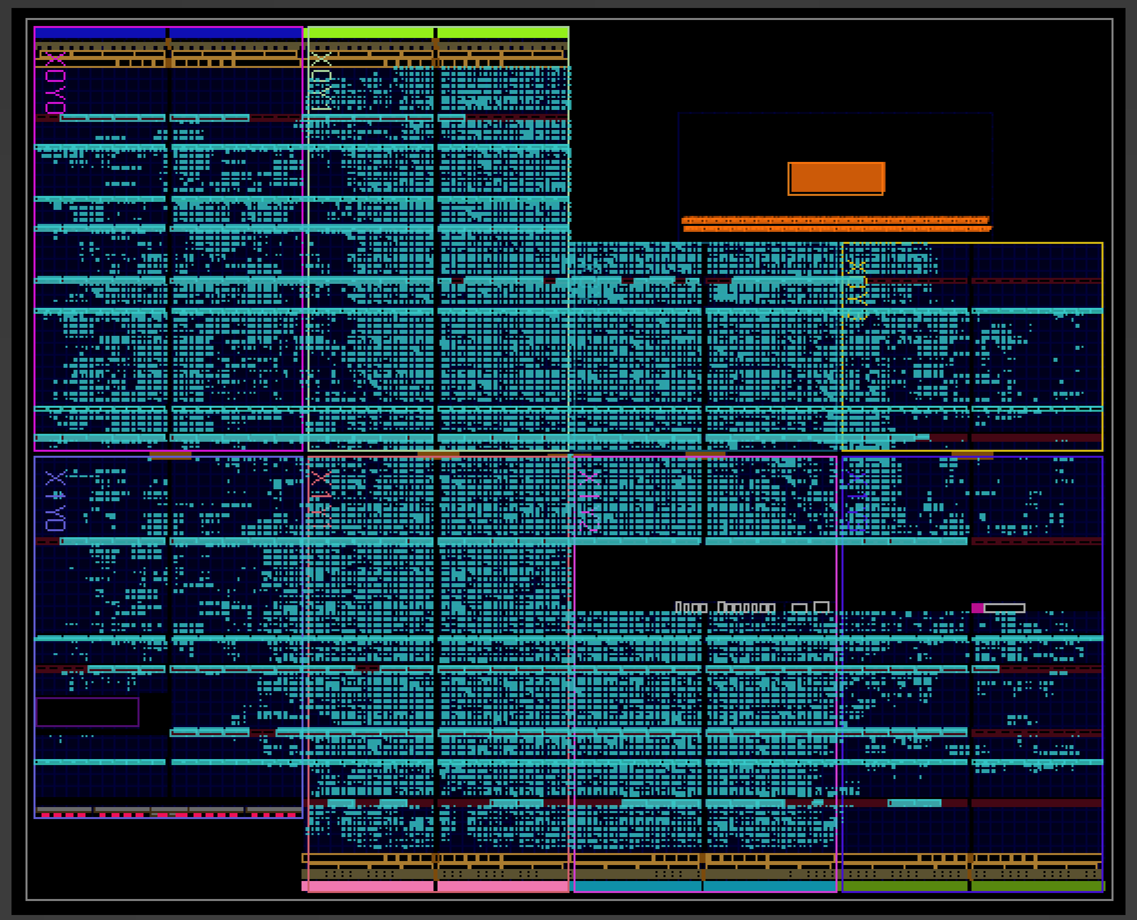
Evaluation: Resource Utilization with Aristotle Architecture

- Zynq 7020
- Zynq 7030
- Zynq 7045



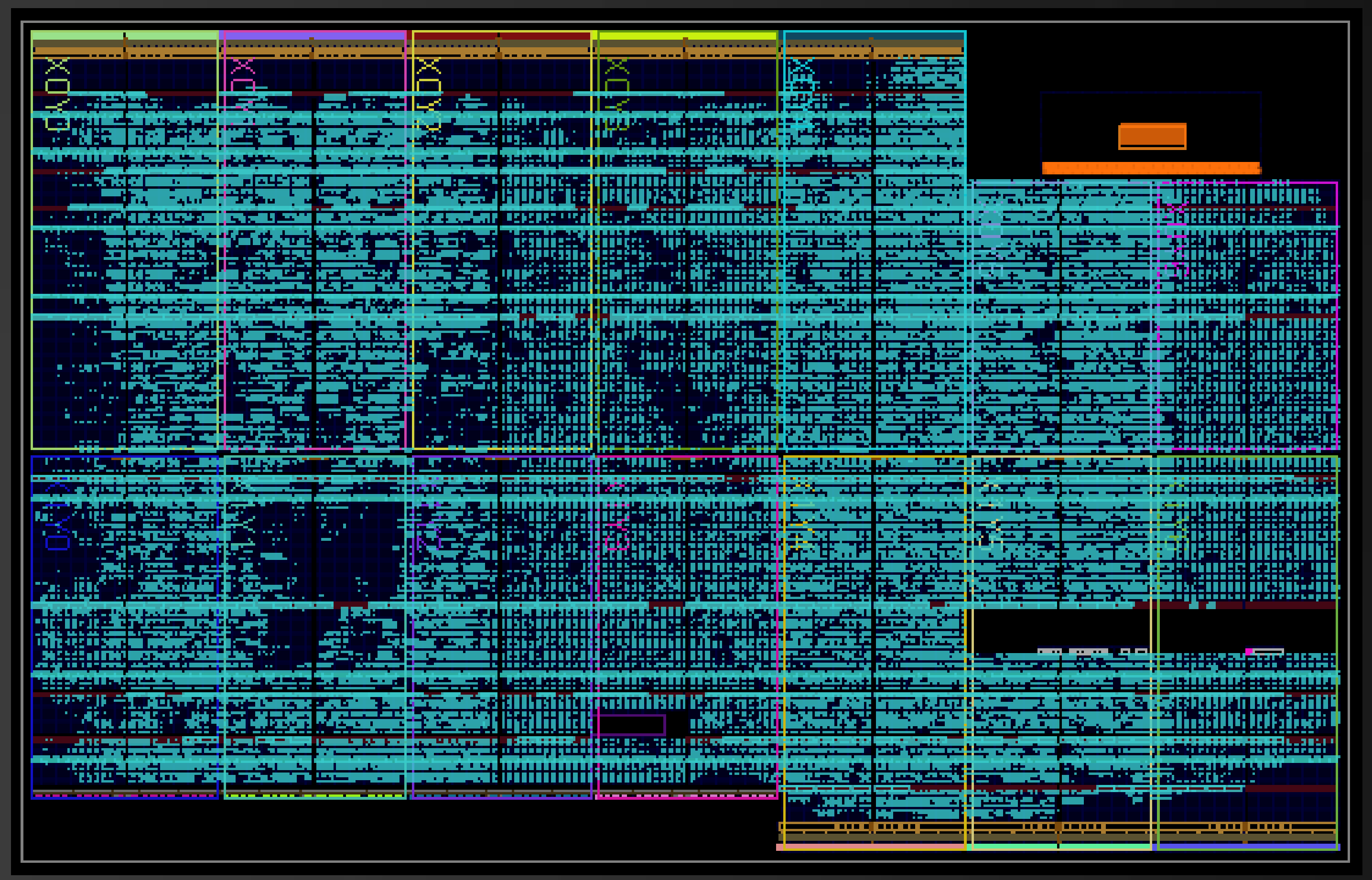
	LUT	FF	BRAM	DSP
Total	53200	106400	140	220
Used	27761	26600	75	220
Ratio	52%	22%	54%	100%

2 Processing elements
Peak performance: 86.4GOPS@150MHz



	LUT	FF	BRAM	DSP
Total	78600	157200	265	400
Used	43118	34097	203	400
Ratio	55%	22%	77%	100%

4 Processing elements
Peak performance: 172.8GOPS@150MHz



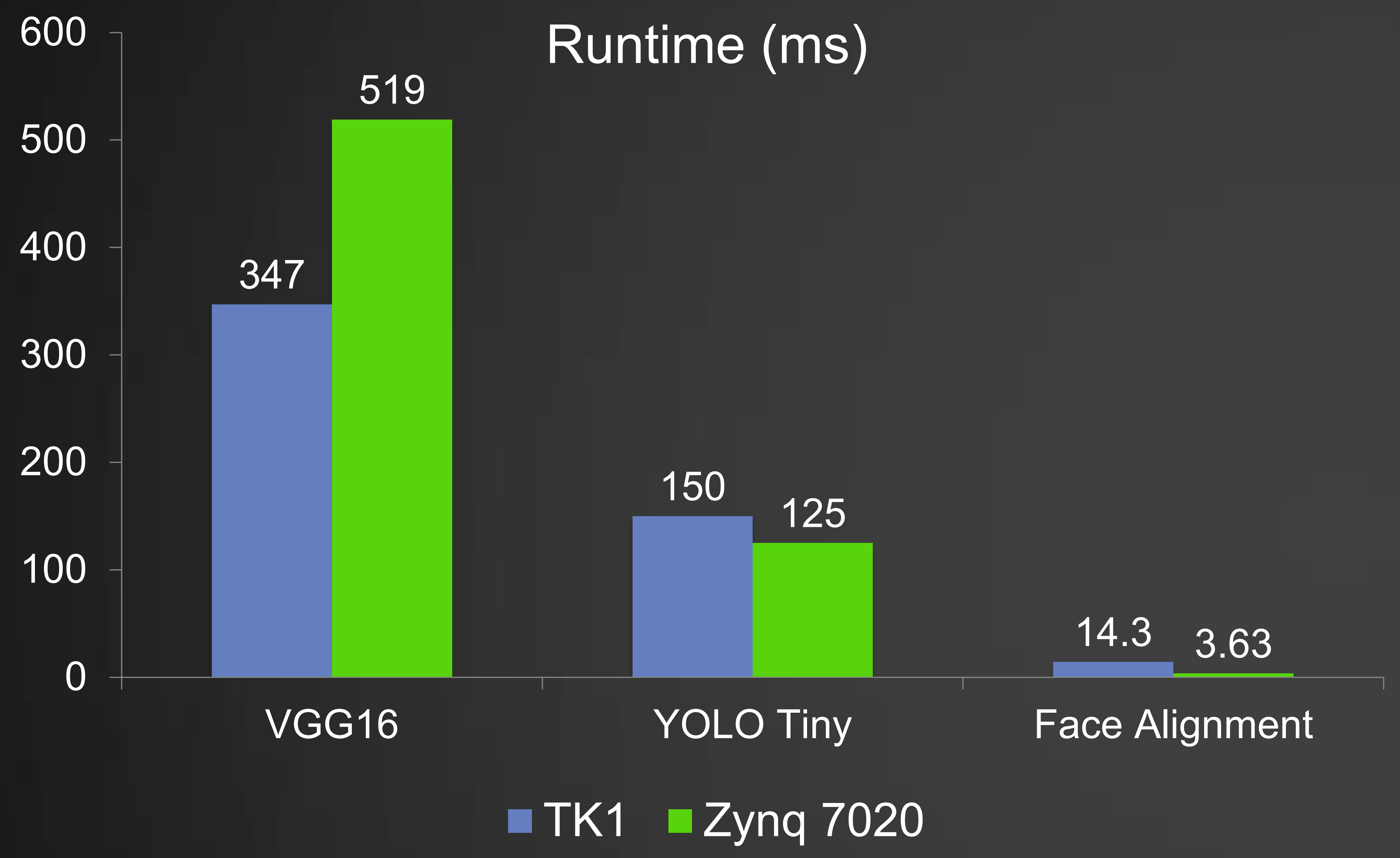
	LUT	FF	BRAM	DSP
Total	218600	437200	545	900
Used	139385	85172	390.5	900
Ratio	64%	19%	72%	100%

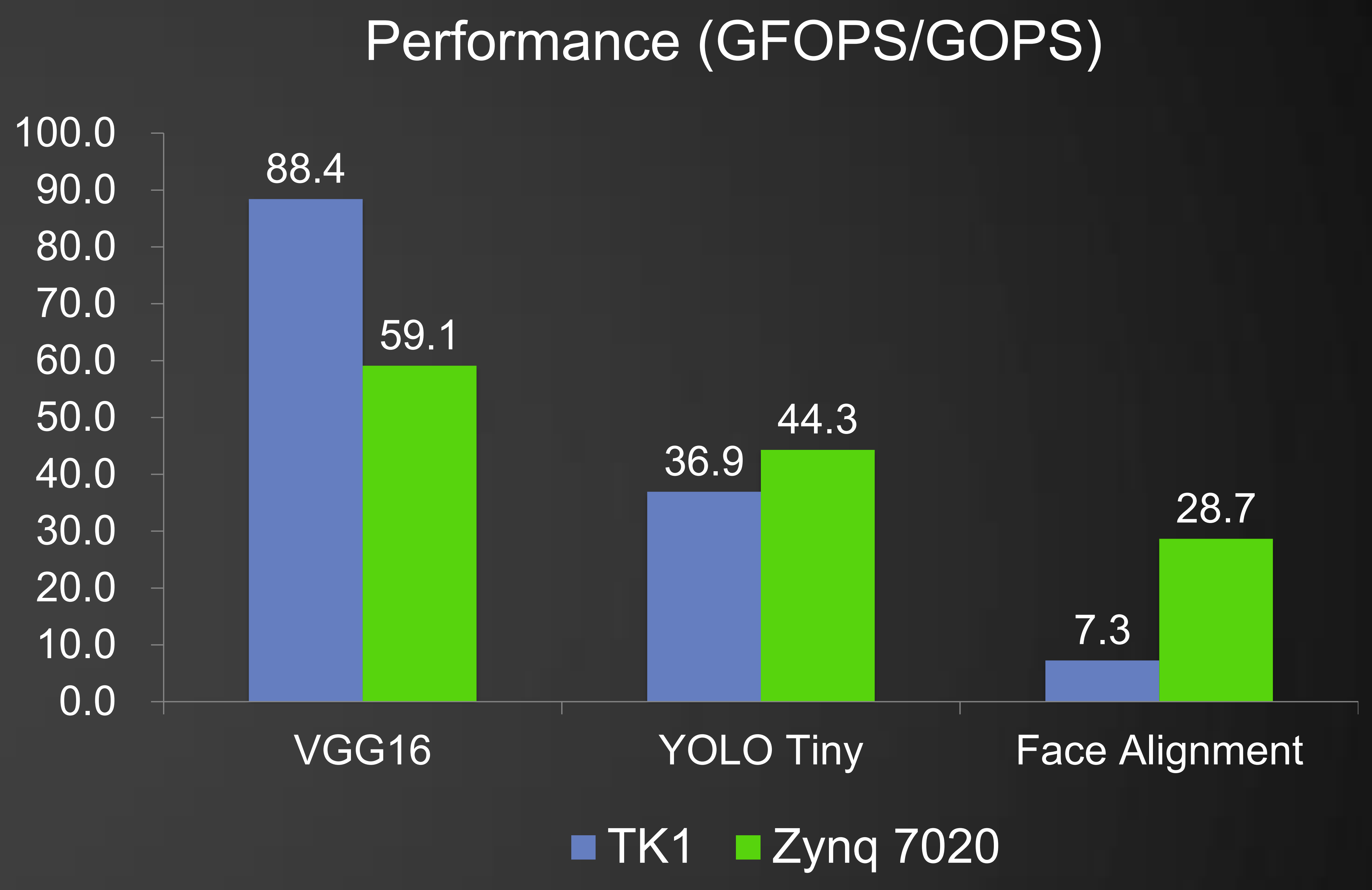
12 Processing elements
Peak performance: 518.4GOPS@150MHz

- Tegra K1 GPU - Peak performance : 326 GFOPS

Evaluation: Performance of Aristotle Architecture

- Runtime and performance*[1] on TK1 and Zynq 7020





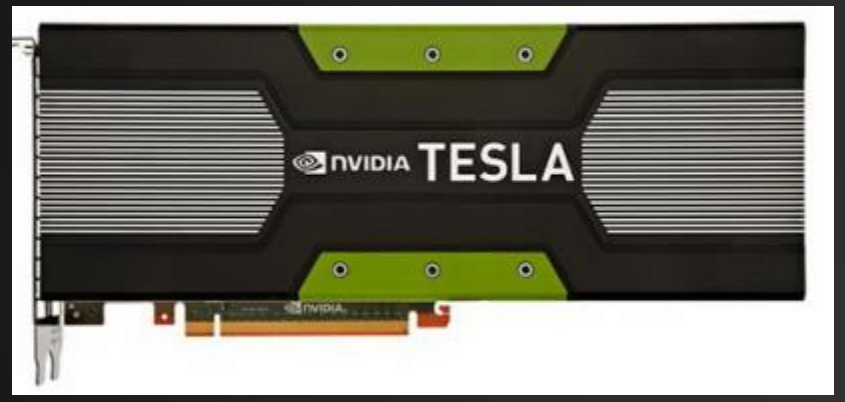
- Aristotle architecture performs better when network is small but has limited peak performance
- Zynq 7020 consumes 20% - 30% power of TK1 and costs less of TK1

- 1.78x higher performance on Zynq 7030 compared with Zynq 7020
- 4.94x higher performance on Zynq 7045 compared with Zynq 7020

*[1] All results are measured with batch_size = 1

Evaluation: Platform and Benchmark for LSTM

- Platform Comparison

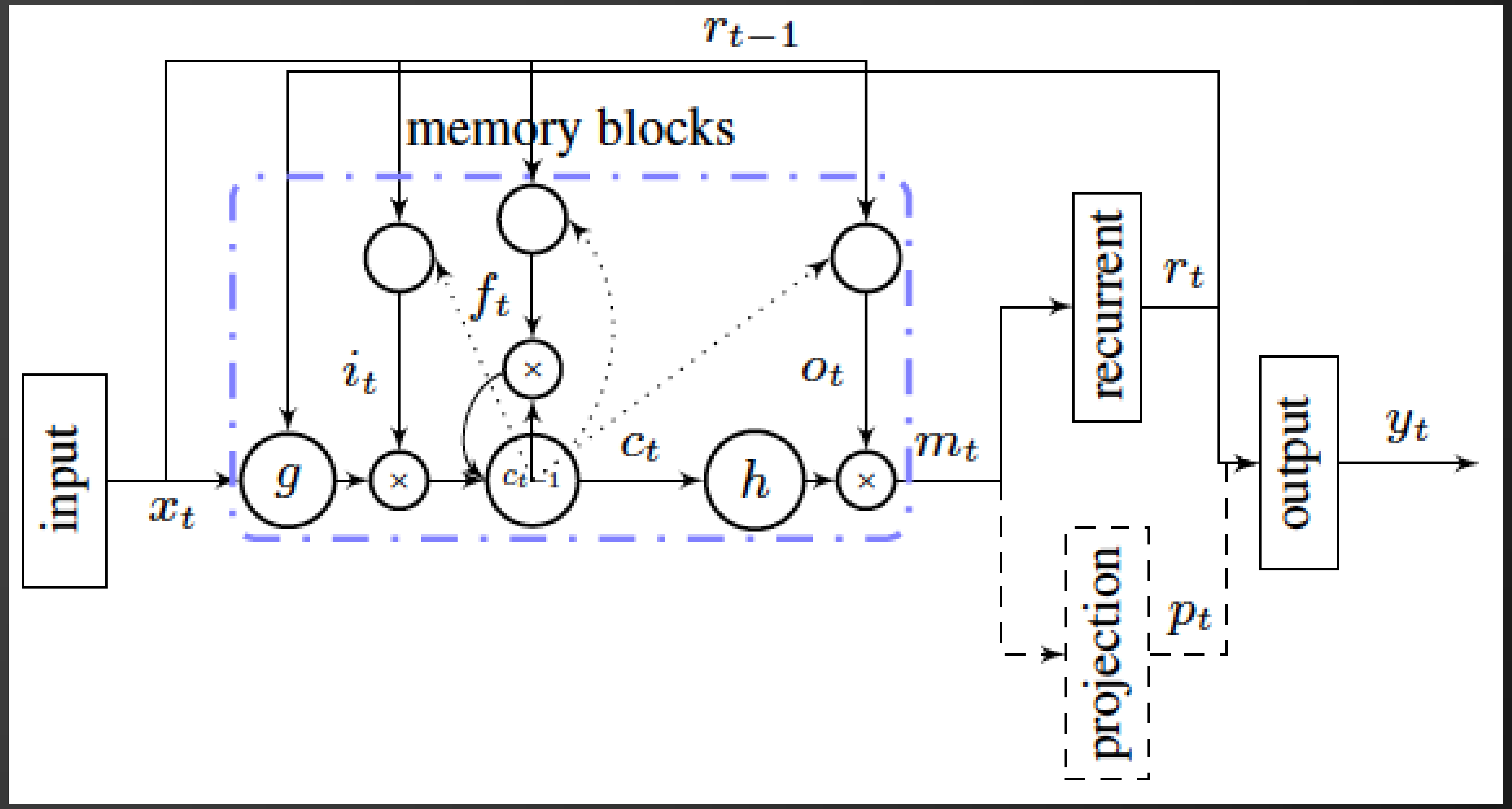


Nvidia K40 GPU
- 28nm
- 2880 CUDA Cores
- 810MHz / 875MHz
- 12GB GDDR5

Kintex Ultrascale Series
- 20nm
- 4.75/9.49MB BRAM (KU060/115)
- 2760/5520 DSP (KU060/115)
- 300MHz

- Benchmark: Real-world LSTM for Speech Recognition

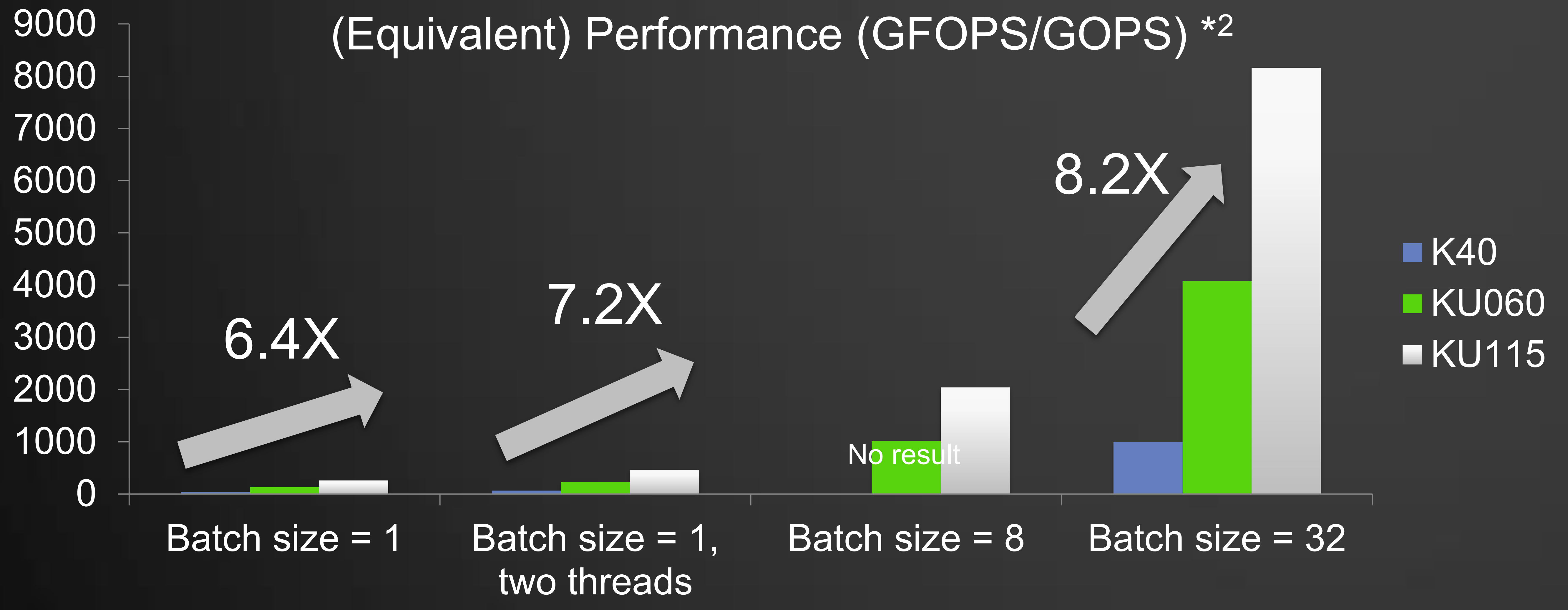
  - Max matrix size: 4096*1536
  - Consider scheduling of multiple matrixes
  - Consider non-linear functions
  - 100 frames per second



Evaluation: Performance and Resource Utilization of Descartes Architecture

- Performance Comparison

Platform	GPU K40*1	FPGA KU060	FPGA KU115
Dense or Sparse	Dense	Sparse (10% sparsity)	
Frequency	810/875 MHz	300 MHz	
Precision	FP32	FIXED-4 to FIXED-16	
Threads to be Supported	Not limited	2 (Separate) / 32 (Batch)	
Peak Performance	4.29 TFOPS	4.8 TOPS*3	9.6 TOPS*4
Real Power	235W	30 – 35W	45 – 50W



*1 Results on K40 GPU were provided by DeePhi's partners
*2 Generally, real performance is 85%-90% of peak performance with Descartes architecture
*3 480GOPS for dense LSTM *4 960 GOPS for dense LSTM

- Resource Utilization
  - KU060

	LUT	FF	BRAM	DSP
Total	331680	663360	1080	2760
Used	298875	446655	1011	1505
Ratio	90%	67%	94%	55%

  - KU115

	LUT	FF	BRAM	DSP
Total	663360	1326720	2160	5520
Used	563403	848990	1155	2529
Ratio	85%	64%	54%	46%

Conclusion

- DeePhi: Making deployment of deep learning algorithms simple and efficient
  - Automatic compilation tool
    - Deep compression
    - Activation quantization
    - Compiler
  - Aristotle: Architecture for CNN acceleration
  - Descartes: Architecture for sparse LSTM acceleration

Evaluation boards will be shipped in Oct 2016
Apply for test at partner@deephi.tech

New architecture for CNN revealed in Q4 2016

Demo

Live demo at Poster Session

Thank You!

Song Yao
Founder & CEO
songyao@deephi.tech

About us
– www.deephi.com

Collaborate with us
– partner@deephi.tech

Join us
– dream@deephi.tech