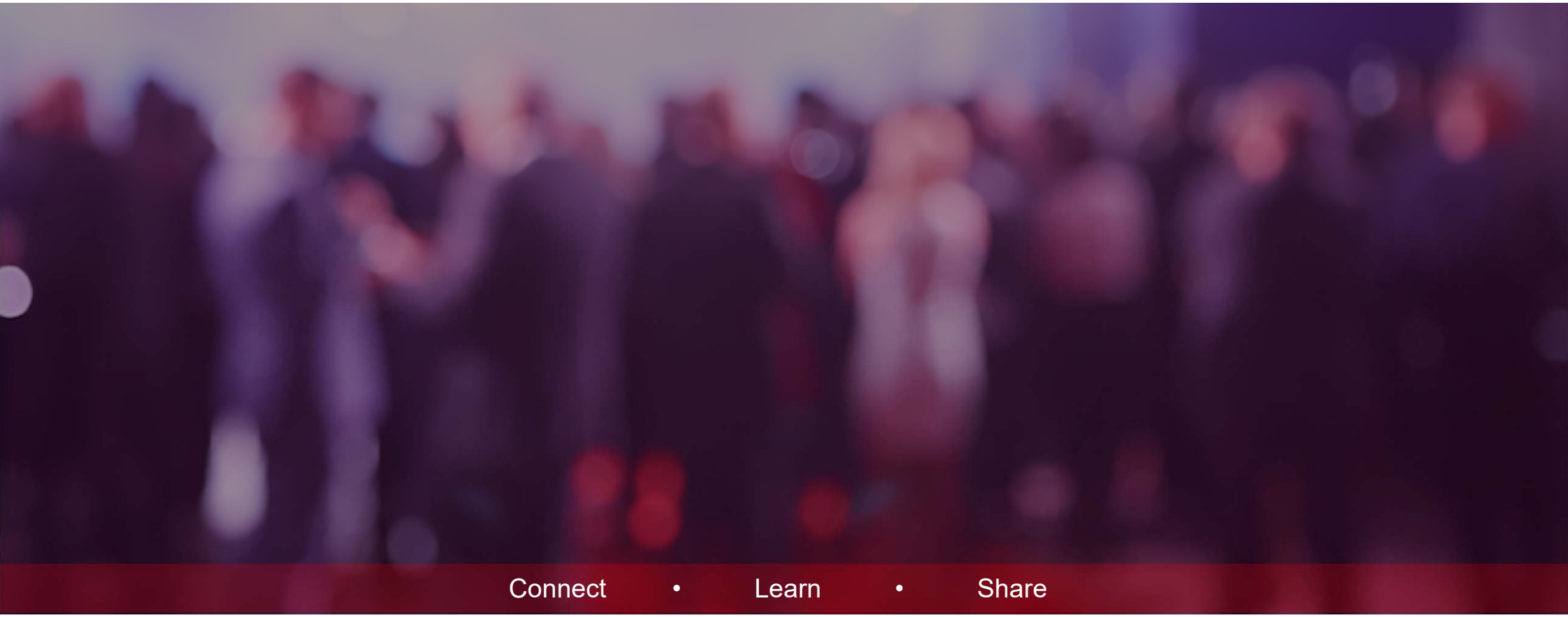


XILINX
ALL PROGRAMMABLE™

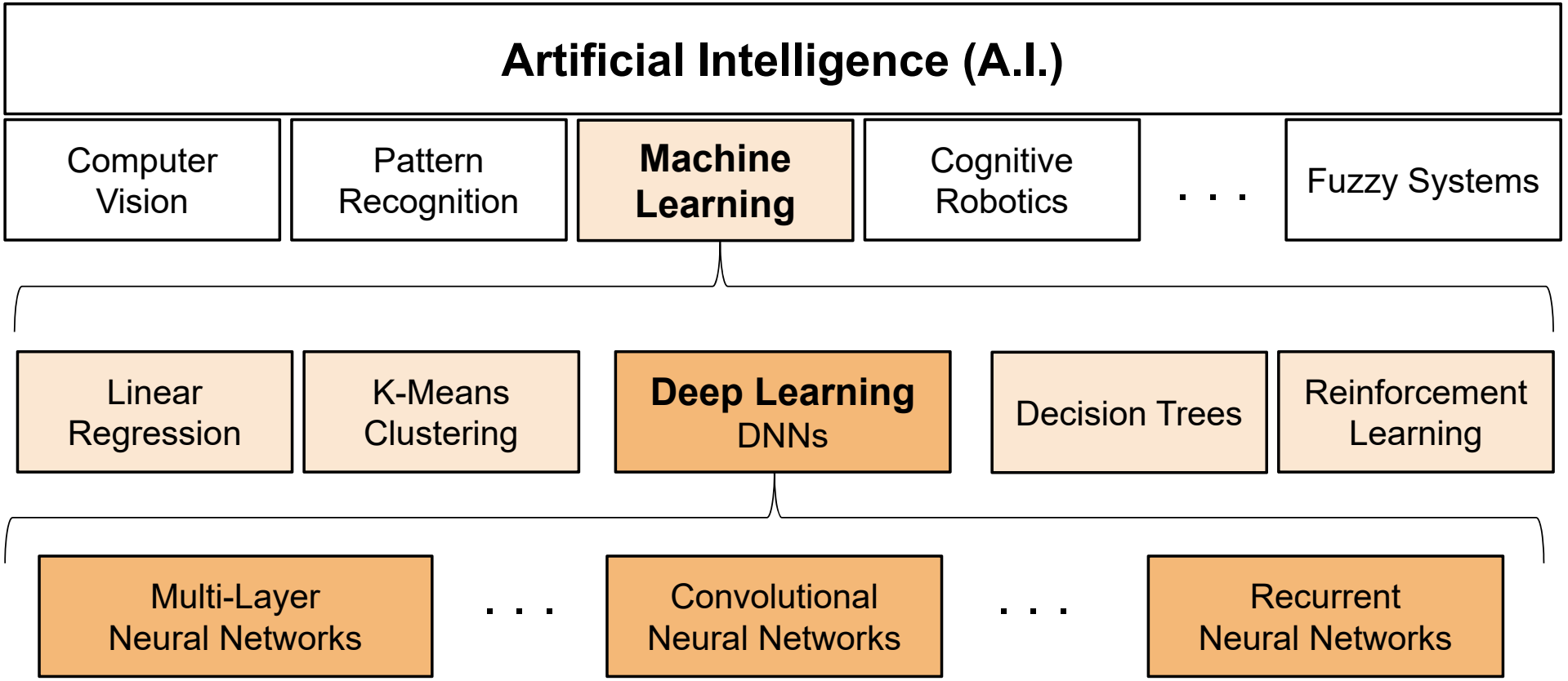
Exploration and Tradeoffs of Different Kernels in FPGA Deep Learning Applications

Elliott Delaye, Principal Engineer
Xilinx Inc., San Jose, CA

In this presentation

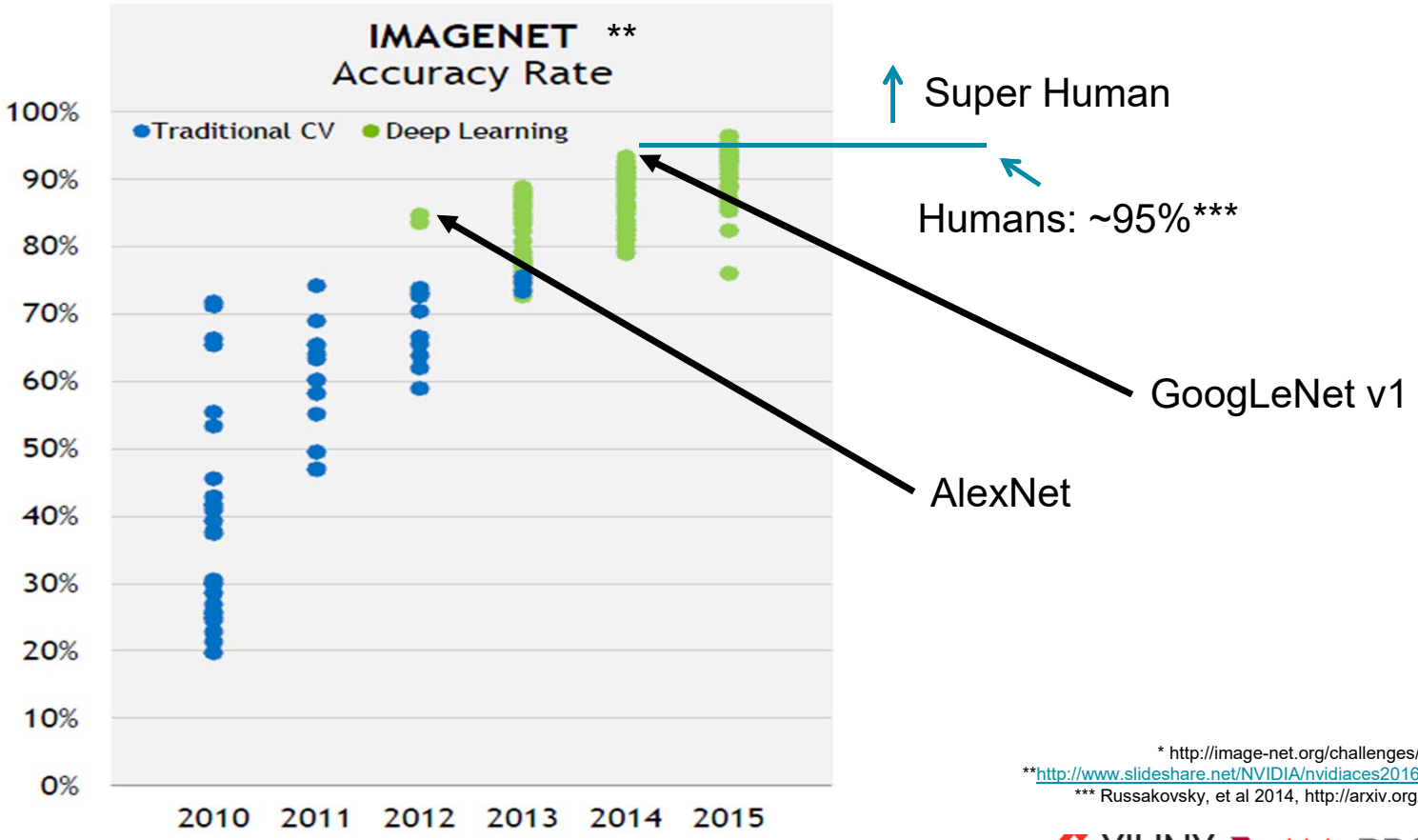
- Background
- Deep Learning - Algorithms, Compute Complexity
- Architecture Evaluations
- FPGA Implementations
- Memory Architectures
- Compute Array Choices
- Network Compilation and Quantization

A.I., Machine Learning, and Deep Learning



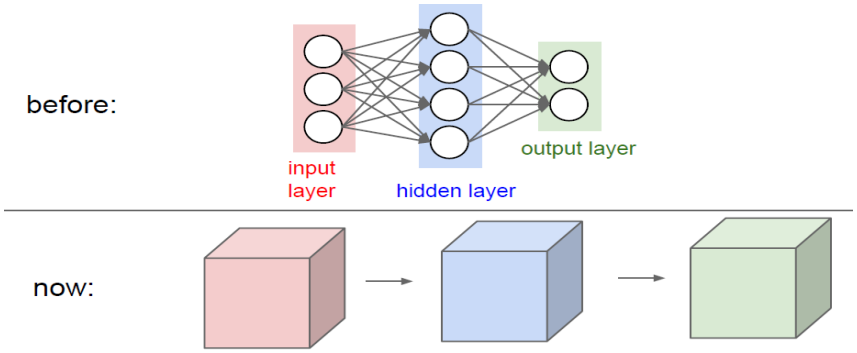
2012 - Breakthrough in Automatic Image Classification(Top 5)

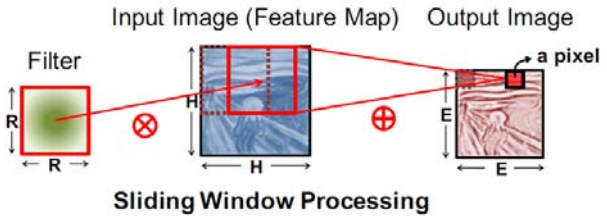
Image-Net Large-Scale Visual Recognition Challenge (ILSVRC*)

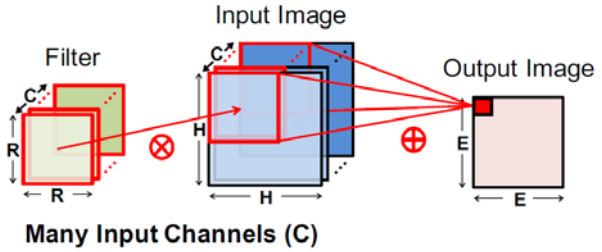


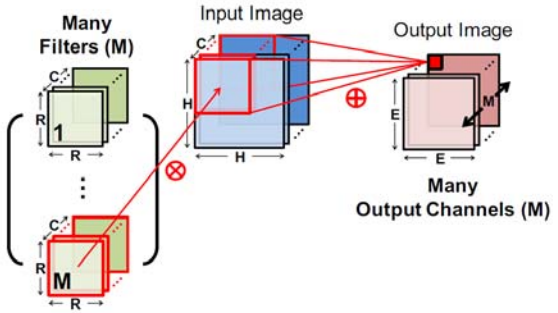
* http://image-net.org/challenges/LSVRC/
**http://www.slideshare.net/NVIDIA/nvidiaces2016-press-conference, pg 10
*** Russakovsky, et al 2014, http://arxiv.org/pdf/1409.0575.pdf

Convolutional Neural Networks - CNN



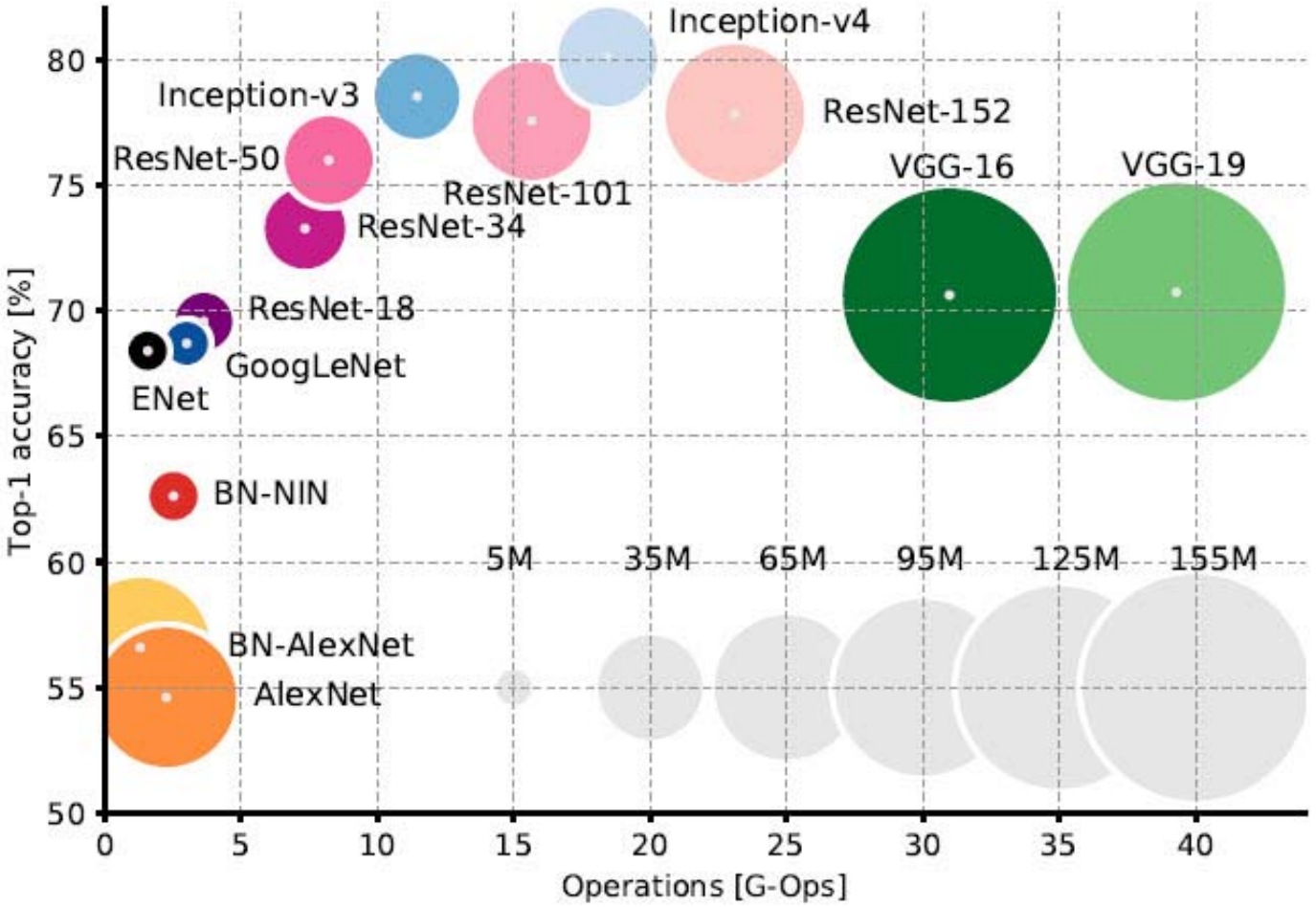






- **Use Convolution Function to compute next layer – Dot product and Accumulation**
- **Pooling or subsampling layer to reduce features**
- **Convolution Implementation: Direct, Matrix Multiply, FFT, Winograd Convolution**

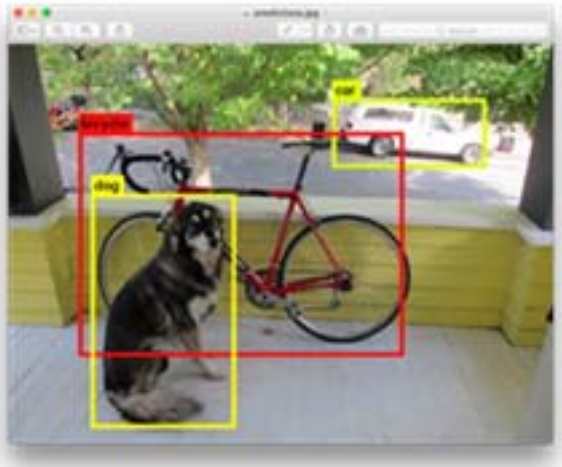
Network Complexity by Operations and Weights Sizes



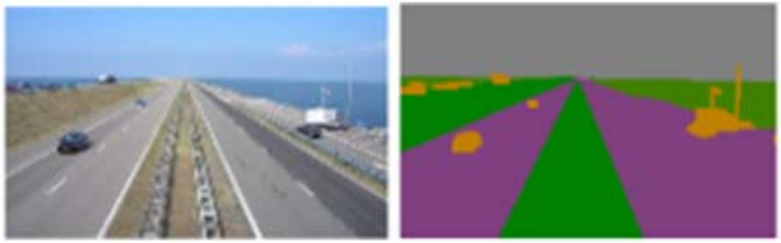
An Analysis of Deep Neural Network Models for Practical Applications, Canziani et al.

Classification, Object Detection, Segmentation

Classification

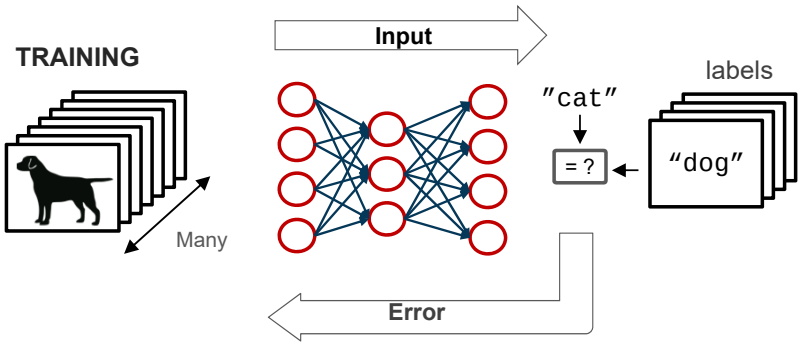


Object Detection



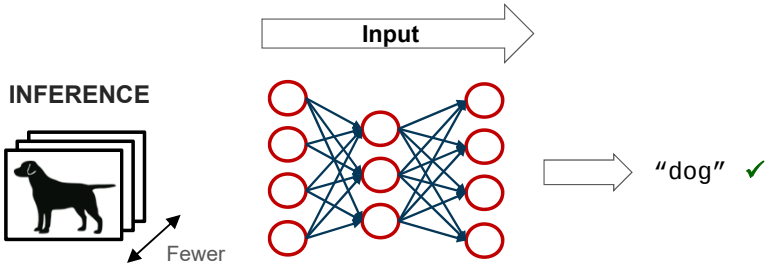
Segmentation

Training and Inference in Deep Learning



Training: Process for machine to "learn" and optimize model from data

FPGA Focus



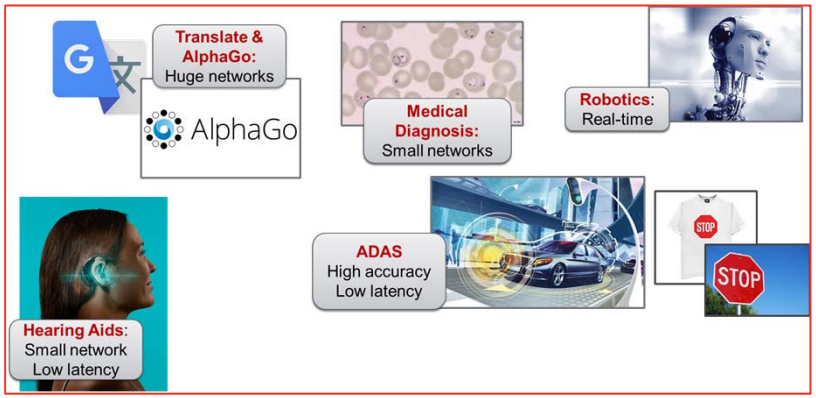
Inference: Using trained models to predict/estimate outcomes from new observations in efficient deployments

https://arxiv.org/pdf/1510.00149v5.pdf

Deep Learning Challenges – Solution Flexibility is Key

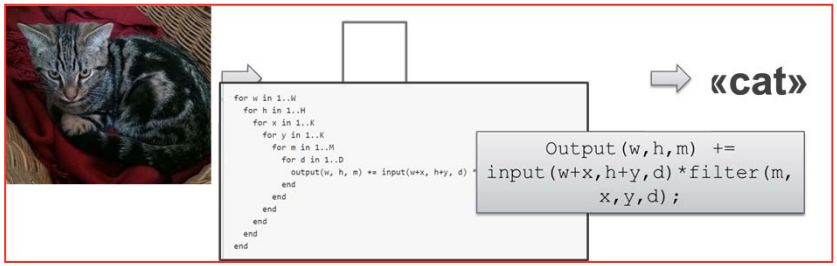
Challenge 1:Different use cases require different networks & different figures of merits

Speed, latency, energy, accuracy



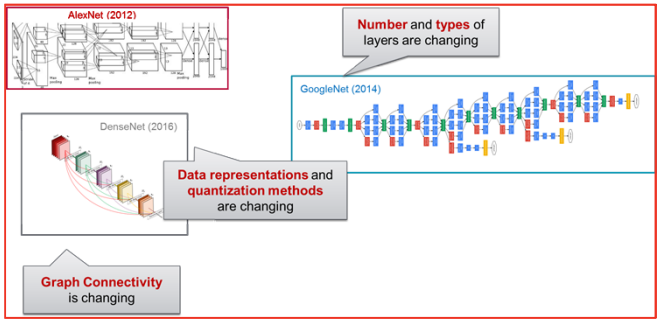
Challenge 2: billions of multiply-accumulate ops & tens of megabytes of parameter data

Sea of operators, custom math & memory hierarchy



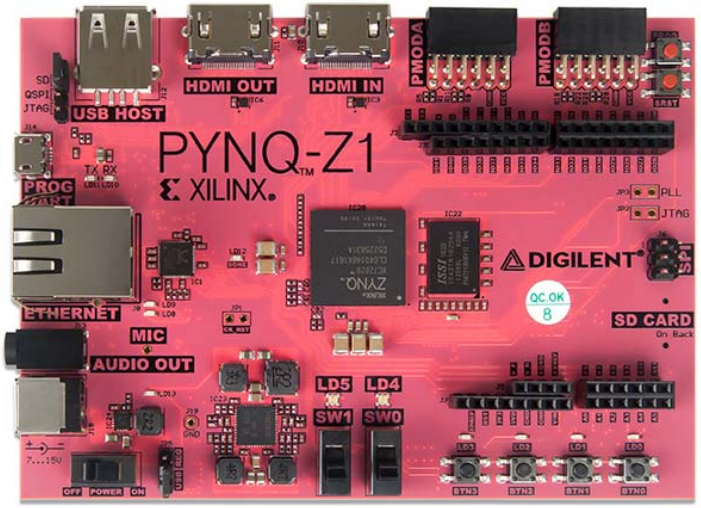
Challenge 3: Continuous stream of new algorithms

Hardening architectures is risky



Scale of Devices

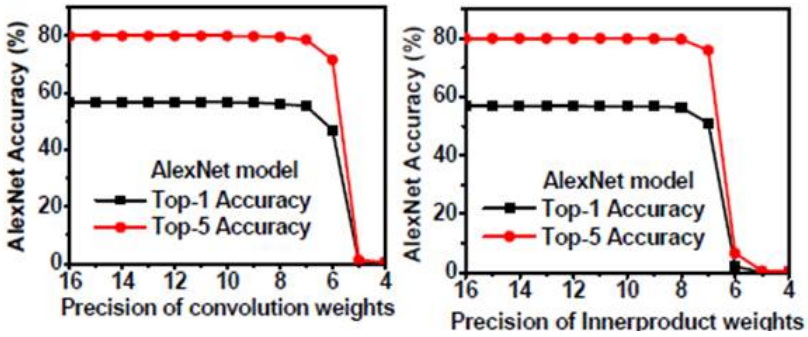
- Embedded/Low Cost:
  - Zynq (PYNQ, Zedboard, Snickerdoodle, Zybo)
  - Snickerdoodle: Zynq 7010 – Dual Core ARM
    - 28K LUTs, 80 DSPs, 262KB RAM
  - PYNQ: Zynq 7020 – Dual Core ARM
    - 85K LUTs, 220 DSPs, 612KB RAM
- Large High Performance:
  - Virtex Ultrascale+ (VCU1525, Amazon F1, etc.)
  - Virtex Ultrascale+ VU13P
    - 1.7M LUTs, 3.4M FFs, 45MB RAM
    - 12228 DSPs (153x Larger than Zynq 7010!)
  - AWS EC2: f1.16xlarge using VU9P
    - 54720 DSPs across 8 FPGAs



FPGA Precision Optimization - Inference

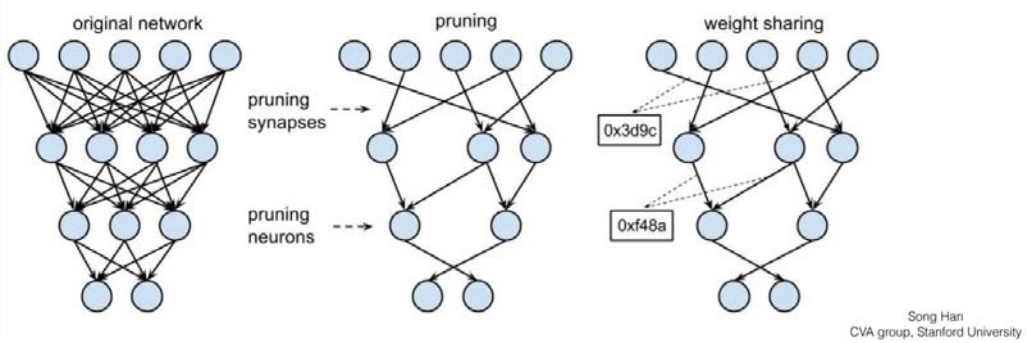
- Inference moving towards lower precision
- Inference with Integer Quantization
  - Fixed-Point Sufficient For Deployment (INT16, INT8)
  - No Significant Loss in Accuracy (< 1%)
- Energy Efficiency
  - >10x Energy Efficiency OPs/J (INT8 vs FP32)
  - 4x Memory Energy Efficiency Tx/J (INT8 vs FP32)
- Reduced memory bandwidth for same throughput
- Reduced model size
  - Easier to fit entire activation data on-chip
- Pruning reduces complexity further

Reduced Precision



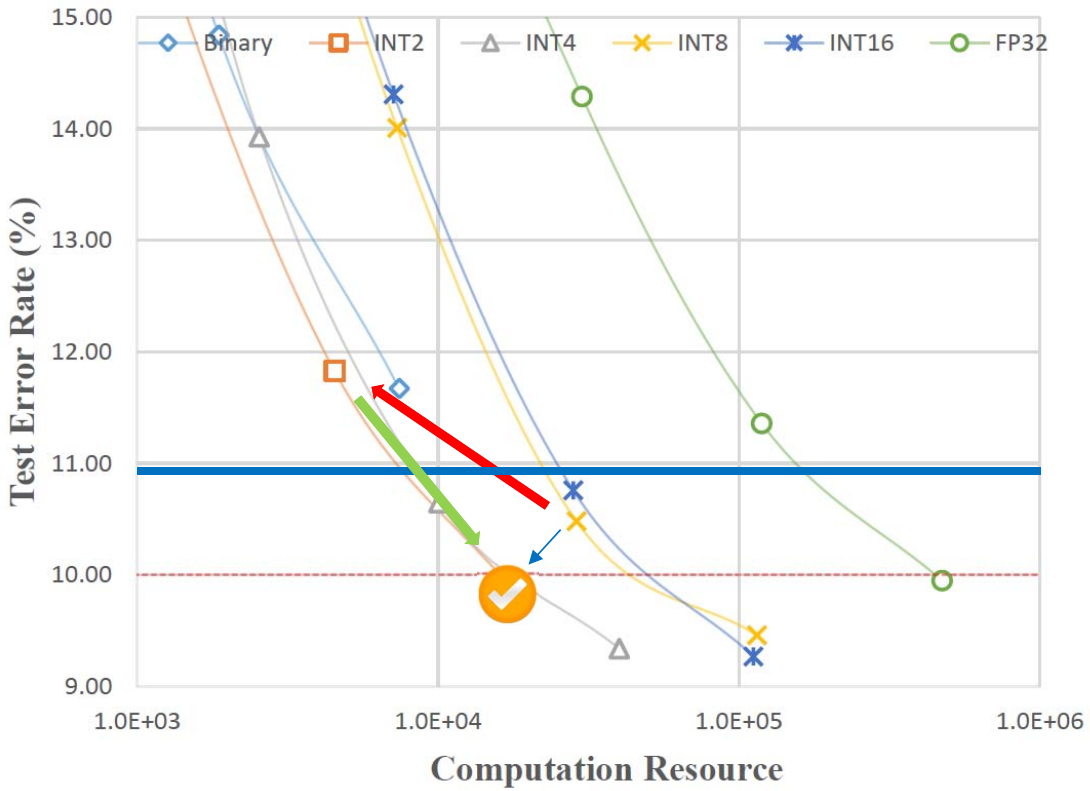
Naveen Suda et. al, International Symposium on Field-Programmable Gate Arrays, 2016

Model Pruning

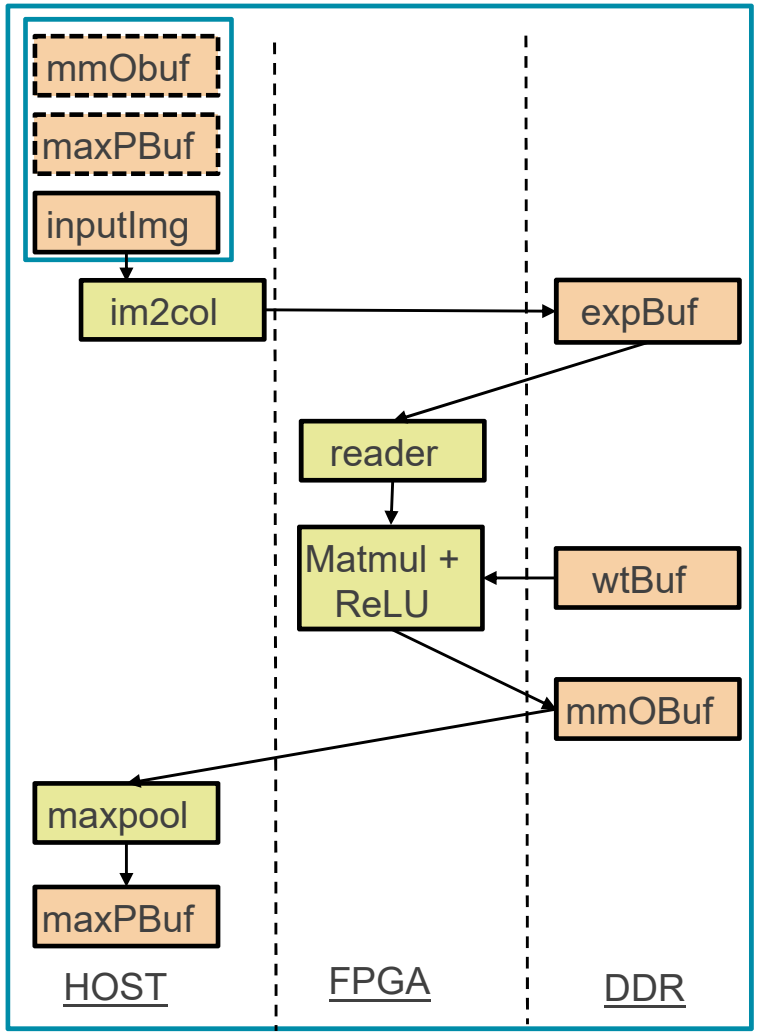


Do we loose Accuracy?
Compensating Quantization with Network Retraining

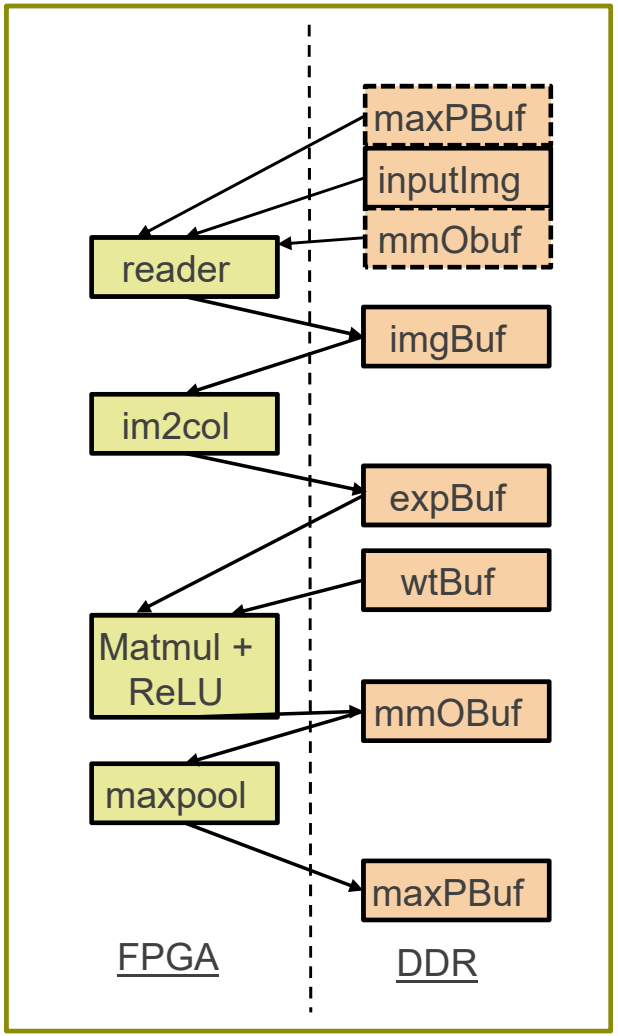
- Just reducing precision, reduce hardware cost & increases error
- Recuperate accuracy by retraining & increasing network size



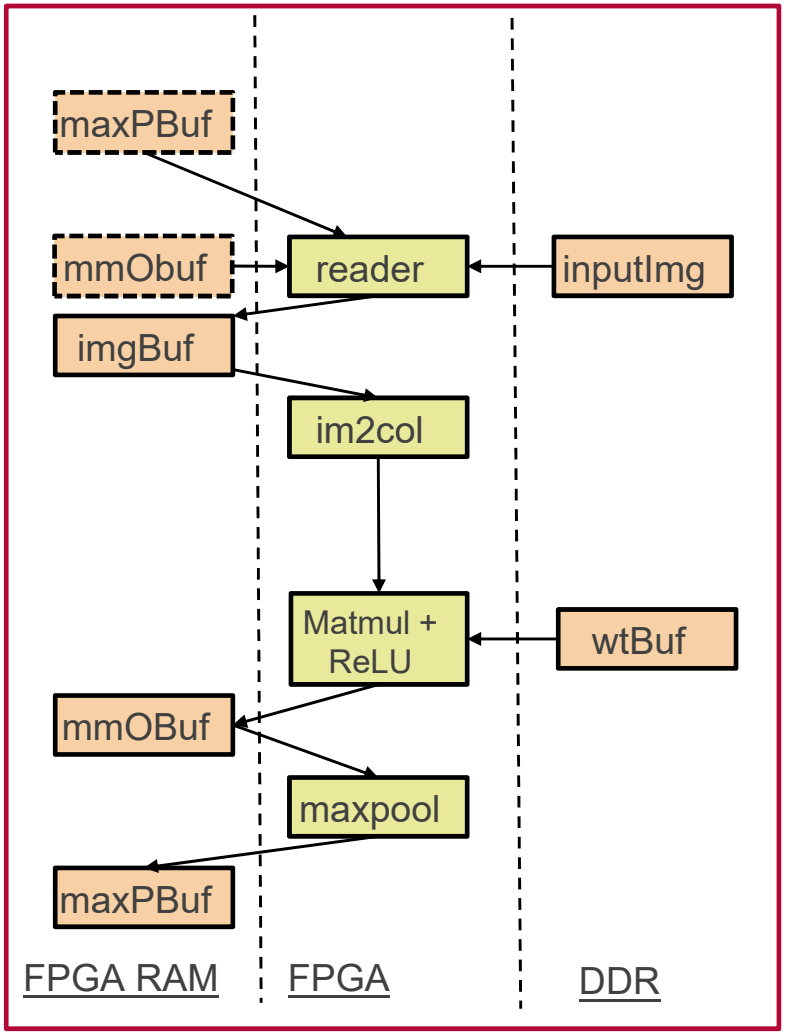
Modes of execution – Memory Architectures



Single Step

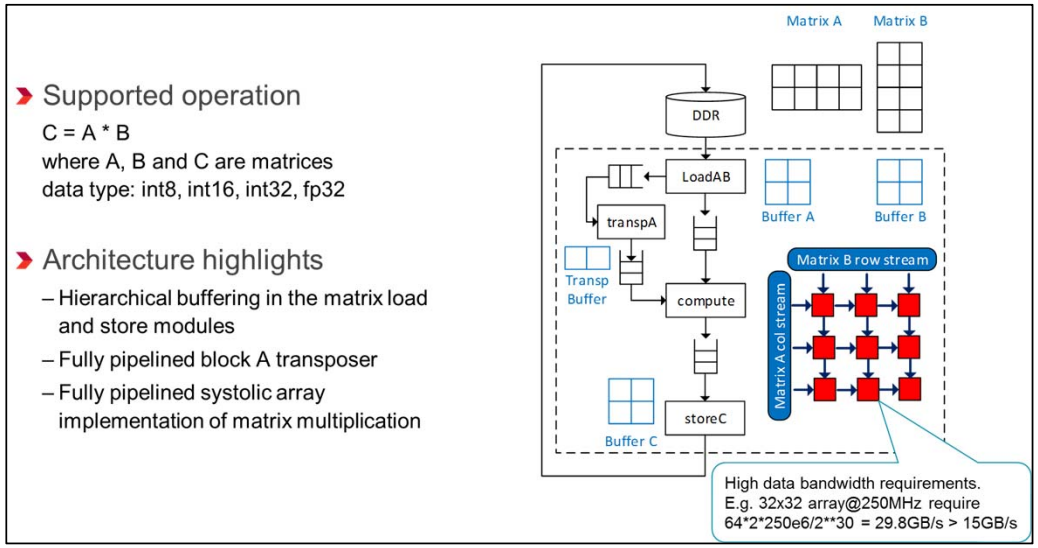


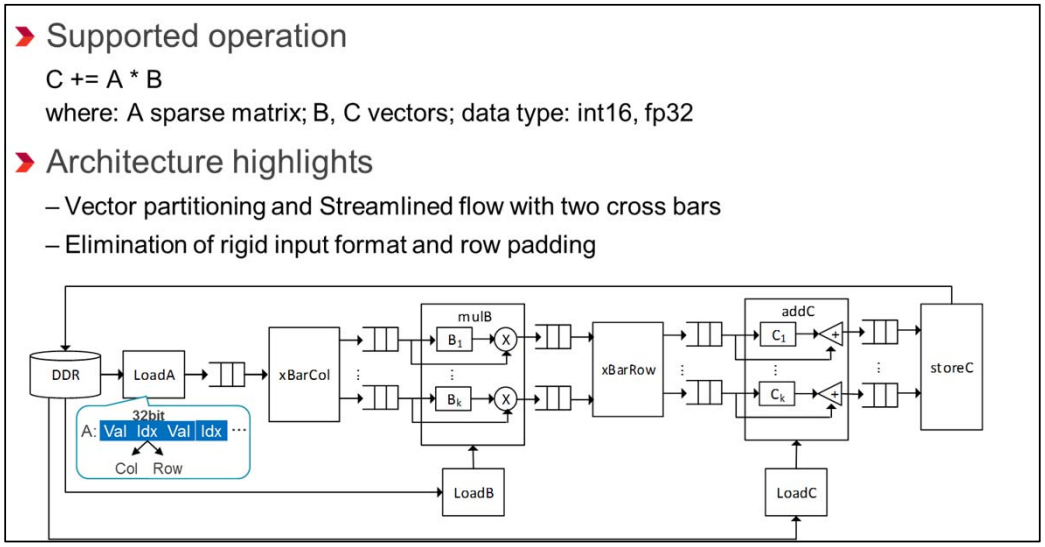
Multi-step



Fully Automatic

Xilinx GEMX – Dense and sparse fixed point GEMM for MLP



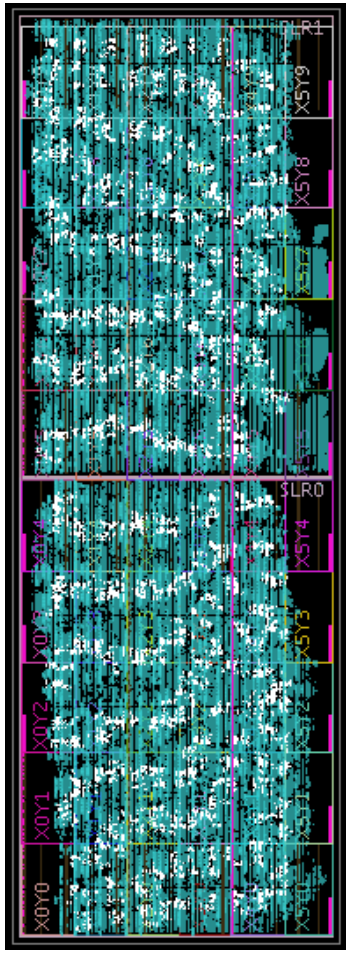


- Dense GEMM
- int8, int16
- Per block performance 400 GOP/s
- Works with arbitrary size matrix

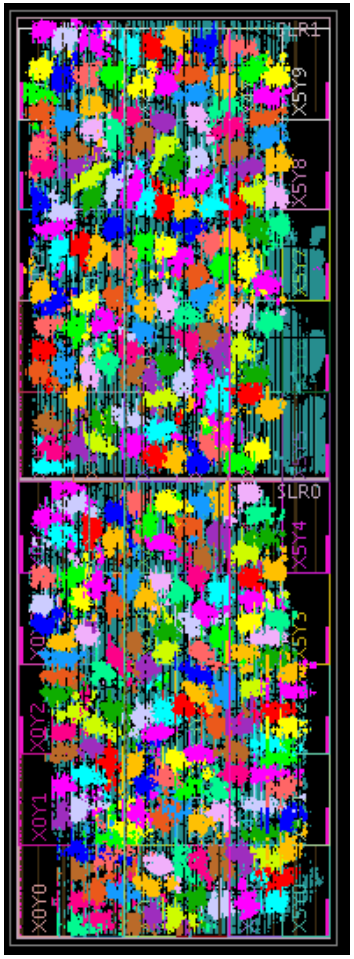
- Sparse Matrix Vector
- int8, int16
- Solve significantly large problems
- Performance a function of NNZ

https://github.com/xilinx/gemx

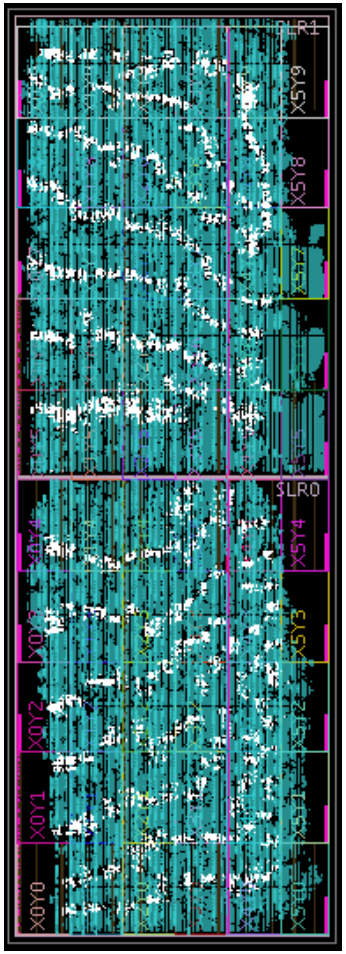
GEMM Array – Visualizing Systolic Arrays (4096 DSPs)



Array Columns



Array Nodes



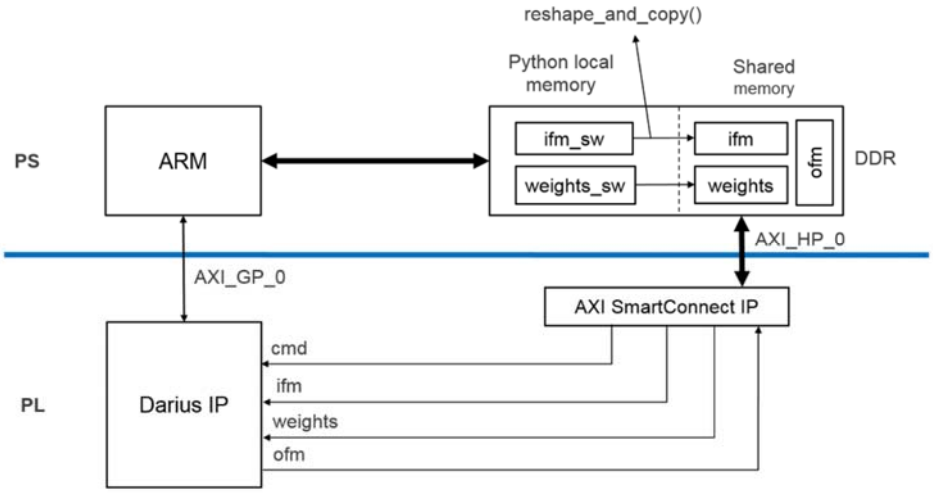
Array Rows

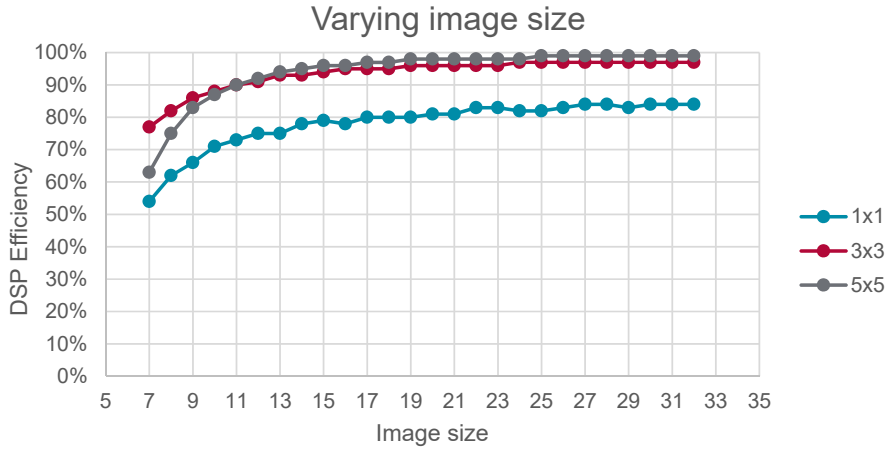
Xilinx KU115 Device
Two Kernels, One per SLR

Darius – Convolution IP

- Designed to scale
  - As small as 64 DSPs
- Direct access to ARM for control
- Shared DDR for data transfers
- Key Feature: Data Movement
  - Designed for limited on-chip memory
- Key Feature: Systolic Array
  - Use hardened DSP carry chain

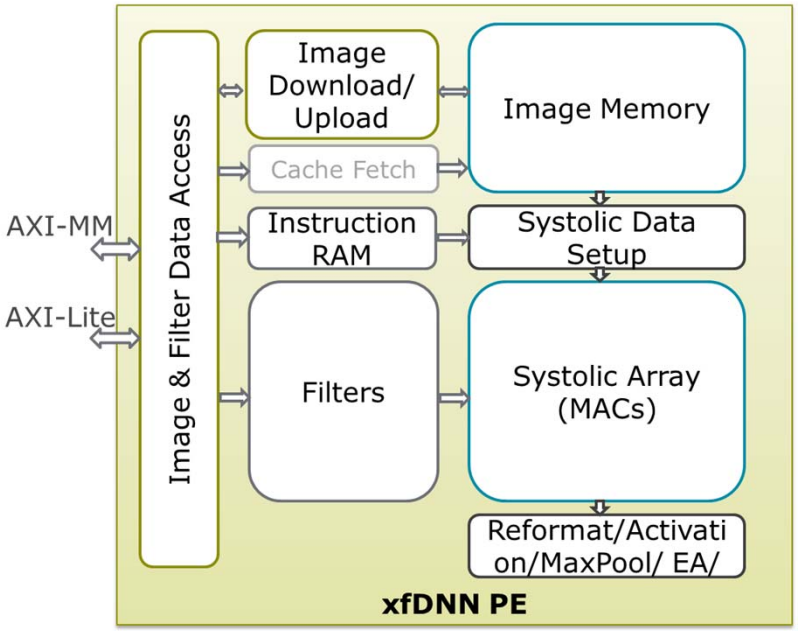
- Provided as part of DAC 2018 contest





https://github.com/xilinx/PYNQ-DL

Xilinx CNN Programmable Overlay - xDNN



Features		Description	
Supported Operations	Convolution	Kernel Sizes	1x1, 3x3, 5x5, 7x7
		Strides	1-8
		Padding	<= Kernel Size
		Activation	ReLU
		Bias	Per Channel
		Scaling	Scale & Shift Value Per Channel
	Max Pooling	Kernel Sizes	2x2, 3x3, 4x4, 5x5, 6x6, 7x7
		Strides	1-7
		Padding	<= Kernel Size
	Avg Pooling	Kernel Sizes	2x2, 3x3, 4x4, 5x5, 6x6, 7x7
		Strides	1-7
		Padding	<= Kernel Size
	Element-Wise Add	Width & Height must match; Depth can mismatch.	
Miscellaneous	Data width	16-bit or 8-bit	
	DSP Array Size	Configurable (14x32, 28x32, 56x32 for GoogleNet/ResNet)	
	Target Clock Speed	XDNN core: <= 300MHz DSP Systolic Array: 500MHz-600MHz	
	Networks Supported	GoogLeNet v1, ResNet, SqueezeNet etc	

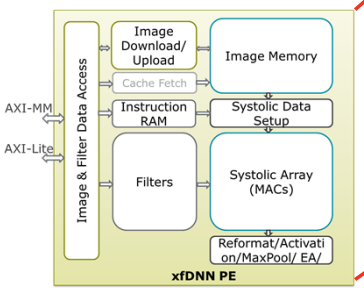
- Programmable Feature-set
- Tensor Level Instructions
- < 7ms latency for ResNet50
- 500+MHz DSP Freq (VU9P-2 speedgrade)

- Supports GoogleNet V1, ResNet50, YoloV2, Classification, Segmentation and Object Detection
- Full toolchain for Python/Caffe/Tensorflow/MxNet
- Available on Amazon F1

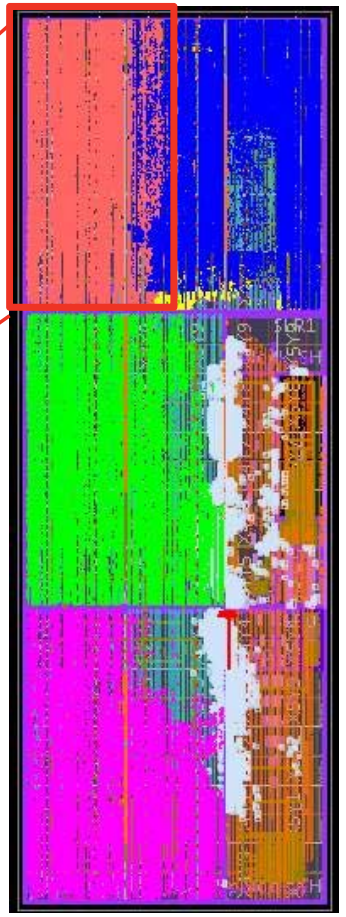
https://github.com/xilinx/ML-Development-Stack-From-Xilinx

Customization Flexibility

- Flexible Processing Engines to be optimized for different Problems –latency vs throughput
- Scalable Implementations to meet performance requirement – Add PEs for more performance
- Each PE can run different CNN – Mix and match object detection with deep classification
- Enable Inline ML processing with other application

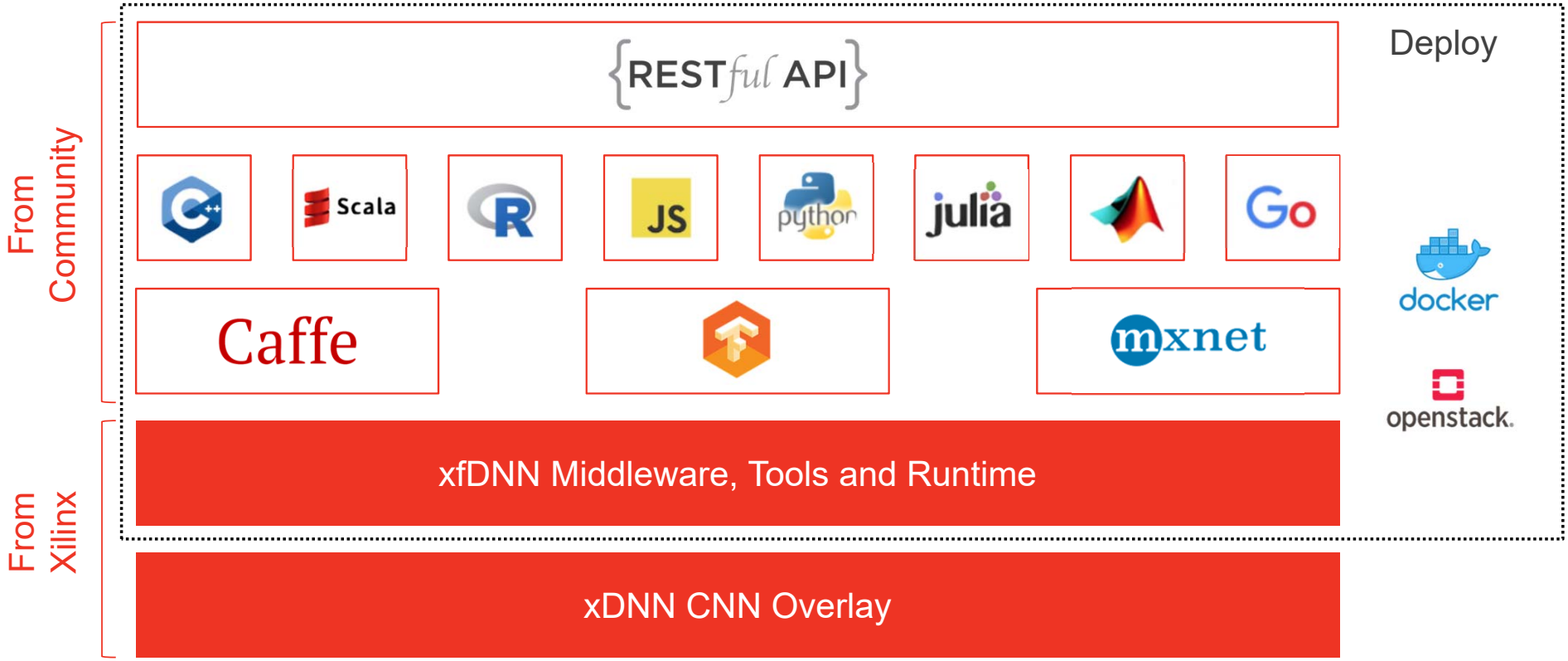


28x32 PE



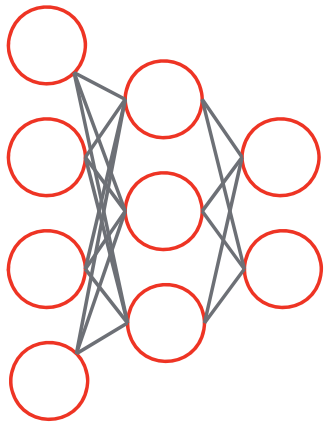
PE Array	#DSPs	Cache	16 bit GOP/s	8 bit GOP/s	Advantage
28x32	896	4MB	896	1,792	Optimized for Throughput
56x32	1792	5 MB	1,702.4	3,404.8	50% less latency
56x32	1792	8MB	1,612.8	3,225.6	50% less latency, larger CNN networks

Seamless Deployment with Open Source Software



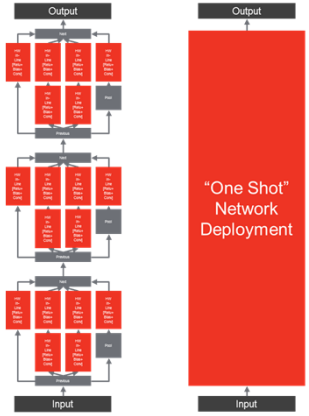
Network Compilation

Graph Compiler



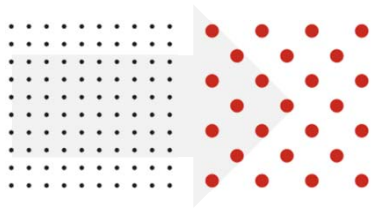
- Build network graphs from Frameworks
- Optimizes for Inference

Network Optimization



- Fused Layer Optimization
- On-Chip Memory Streaming
- Internal Scheduling

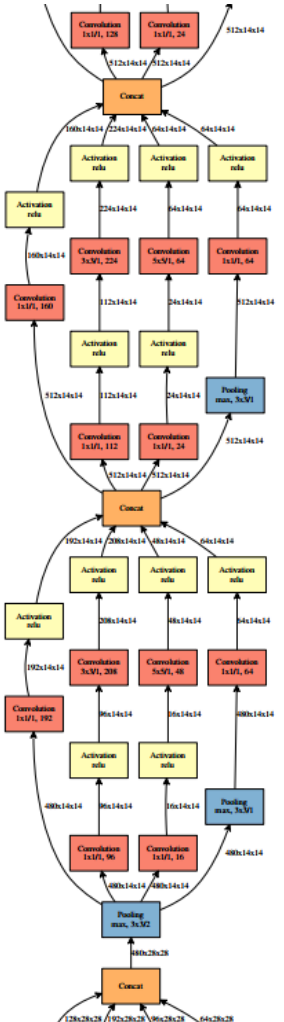
Quantizer



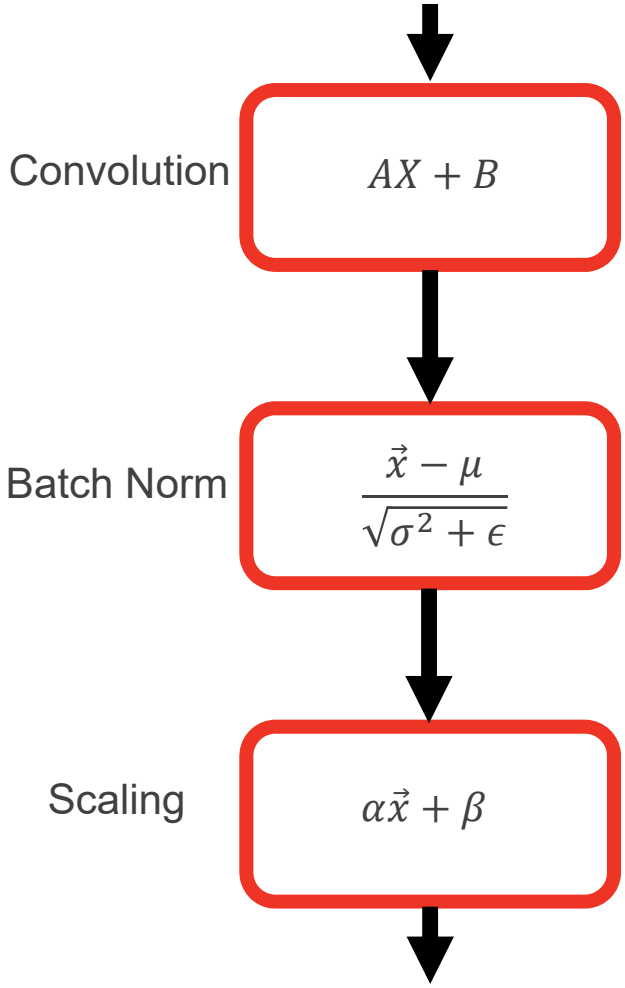
- Deploy pre-trained floating point models on 16bit, 8bit and lower
- Maintains accuracy without retraining
- Easy and Fast

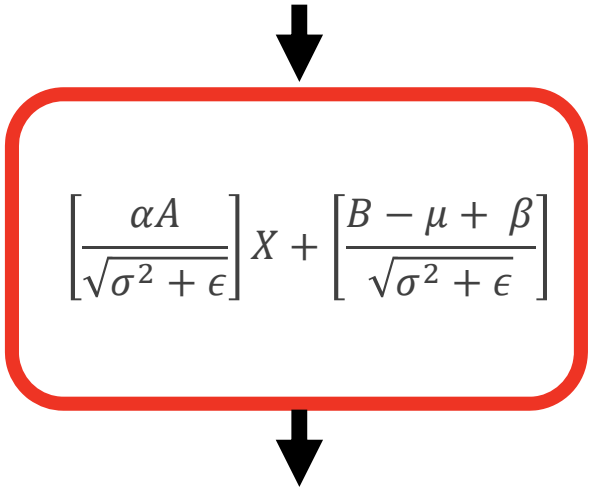
Graph Compilation

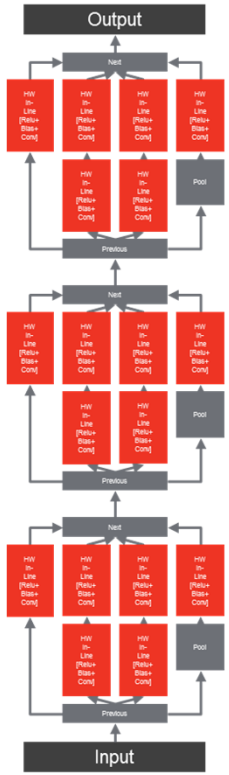
- Graph comes first from a Framework
  - TensorFlow/Caffe/MxNet/etc.
- Cleanup to produce unified dataflow graph
- Basic optimizations, node merging
- Memory optimization
- DDR static vs dynamic scheduling
- Partitioning, Parallelism



Merging Layers: Convolution + BN + Scaling

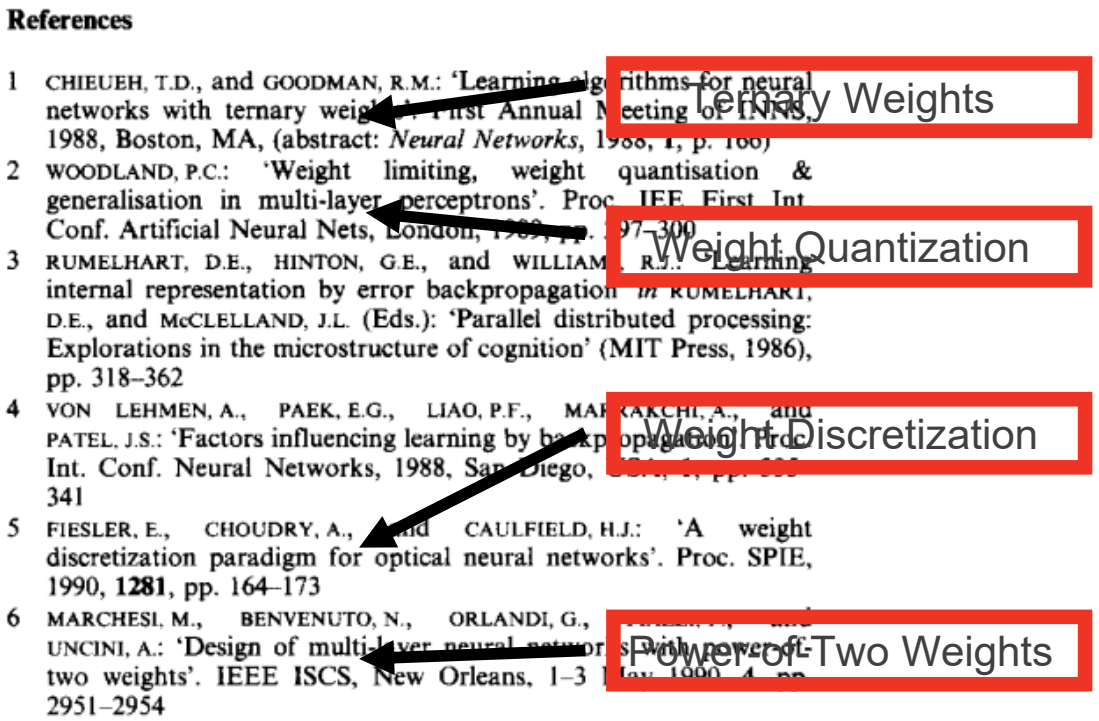






Blast from the Past - Quantization

- References from research published in Electronics Letters, 1994



References

1 CHIEUEH, T.D., and GOODMAN, R.M.: 'Learning algorithms for neural networks with ternary weights'. First Annual Meeting of INNS, 1988, Boston, MA, (abstract: *Neural Networks*, 1988, **1**, p. 166)

2 WOODLAND, P.C.: 'Weight limiting, weight quantisation & generalisation in multi-layer perceptrons'. Proc. IEE First Int. Conf. Artificial Neural Nets, London, 1989, pp. 297–300

3 RUMELHART, D.E., HINTON, G.E., and WILLIAMS, R.J.: 'Learning internal representation by error backpropagation' *in* RUMELHART, D.E., and McCLELLAND, J.L. (Eds.): 'Parallel distributed processing: Explorations in the microstructure of cognition' (MIT Press, 1986), pp. 318–362

4 VON LEHMEN, A., PAEK, E.G., LIAO, P.F., MARRAKCHI, A., and PATEL, J.S.: 'Factors influencing learning by backpropagation'. Proc. Int. Conf. Neural Networks, 1988, San Diego, CA, pp. 335–341

5 FIESLER, E., CHOUDRY, A., and CAULFIELD, H.J.: 'A weight discretization paradigm for optical neural networks'. Proc. SPIE, 1990, **1281**, pp. 164–173

6 MARCHESI, M., BENVENUTO, N., ORLANDI, G., [illegible] UNCINI, A.: 'Design of multi-layer neural networks with powers-of-two weights'. IEEE ISCS, New Orleans, 1–3 May 1990, 4, pp. 2951–2954

Ternary Weights

Weight Quantization

Weight Discretization

Power of Two Weights

Integer-weight neural nets, Khan et al., Electronics Letters – July 1994

Conclusions

- FPGA demonstrate proven compute capabilities
- Design architectures must balance flexibility and performance
- Memory hierarchy and data movers to fully utilize the FPGA
- Mix of specific and generic compute arrays
- Network Compiler bridges gap between User code and FPGA
- Quantization choices based on training options