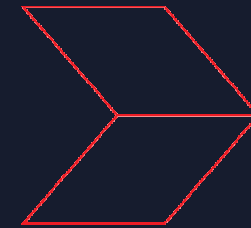# The Future of Machine Learning Acceleration

Quenton Hall
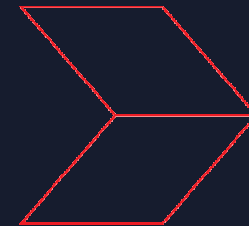
Avnet Field Applications Engineer / ML Specialist

Detroit | November 2018

Slide credit: Michaela Blott, Hot Chips 2018 Tutorial, "Overview of Deep Learning and Computer Architectures for Accelerating DNNs"

☲ XILINX.

>> 1

> **Neural Networks**

> **Computation & Memory Requirements**

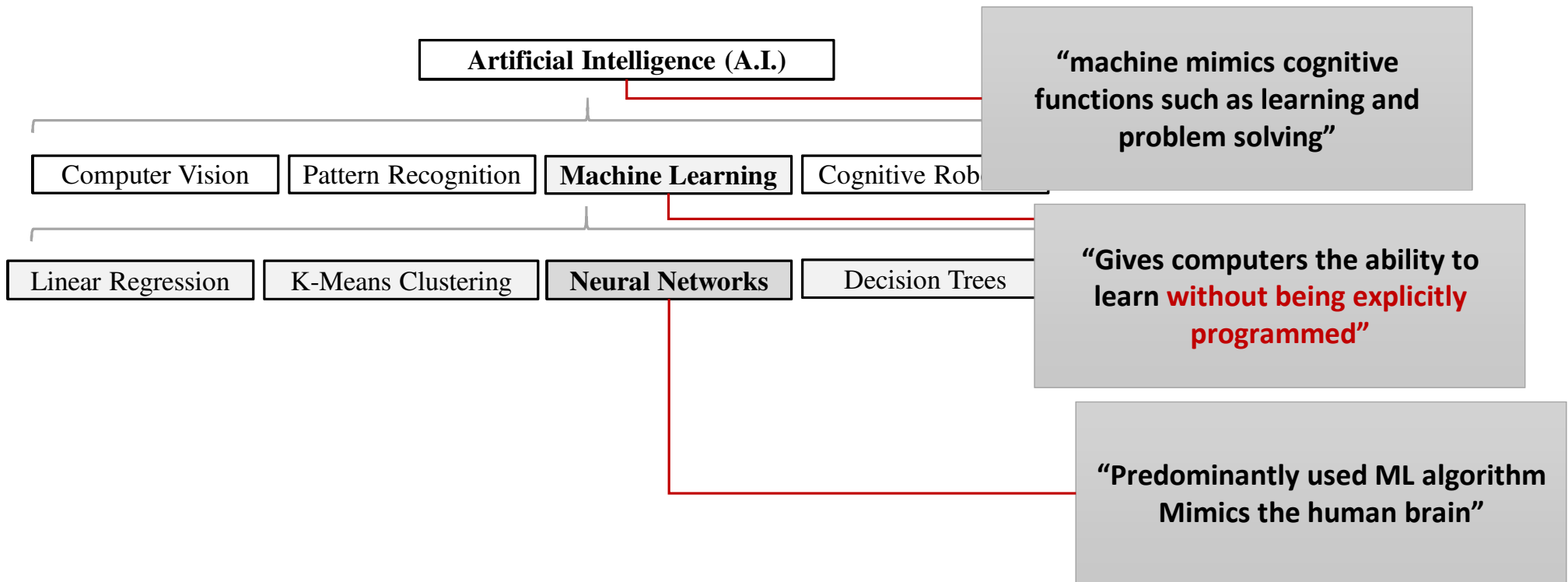> **Algorithmic Optimization Techniques**

> **Hardware Architectures**

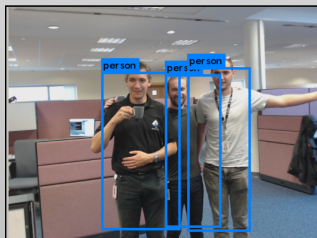**XILINX.**

Neural Networks

XILINX.

A.I. – Machine Learning - Neural Networks

```
                    ┌─────────────────────────────┐
                    │ Artificial Intelligence (A.I.) │
                    └─────────────────────────────┘
```

Computer Vision	Pattern Recognition	**Machine Learning**	Cognitive Rob...

Linear Regression	K-Means Clustering	**Neural Networks**	Decision Trees

"machine mimics cognitive functions such as learning and problem solving"

"Gives computers the ability to learn **without being explicitly programmed**"

"Predominantly used ML algorithm Mimics the human brain"

XILINX.

Increasing Range of Applications

Image Classification

Object Detection

Sedan

Semantic Segmentation

Computer Vision
CNNs

Speaker Diarization
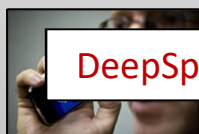
Speech Recognition

Speech Recognition
RNNs, LSTMs

Translation

Sentiment Analysis

Natural Language Processing
Sequence to sequence

Recommender

GamePlay

Many more emerging…

Others

XILINX

Popular Neural Networks

ResNet50, VGG, AlexNet, InceptionV3

Faster R-CNN, Yolo9000, YoloV2

Sedan

Mask-R-CNN, SSD

Computer Vision
CNNs

Image Classification

Object Detection

Semantic Segmentation

DeepSpeech2

eech Recognition
RNNs, LSTMs

Speaker Diarization

Speech Recognition

Seq2Seq, Transformer

Seq-CNN

Natural Language Processing
Sequence to sequence

Translation

Sentiment Analysis

NCF

MiniGo, DeepQ, A3C

Others

Recommender

**GamePlay**

>> 6

Adopted from MLPerf, Fathom, TDP

**XILINX**

Convolutional Neural Networks (CNNs)
from a computational point of view

> **CNNs are usually feed forward* computational graphs constructed from one or more layers**
>> Up to 1000s of layers

$w00$ $n0$

$i0$

$i1$ $n1$

$n2$

$w12$

Synapse with weight w_{ji}

Neuron n_i

> **Each layer consists of neurons n_i which are interconnected with synapses, associated with weights w_{ij}**

$$n0 = Act(w00*i0 + w10*i1)$$

> **Each neuron computes:**
>> Typically linear transform (dot-product of receptive field)
>> Followed by a non-linear "activation" function

Weights W0	Weights W1	Weights W2
Layer L0	Layer L1	Layer L2

Inputs

Outputs
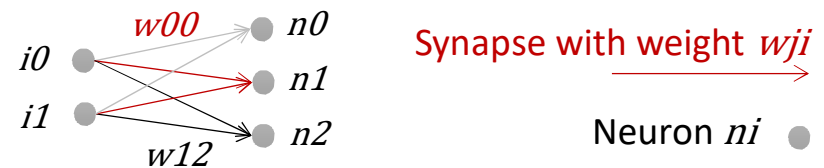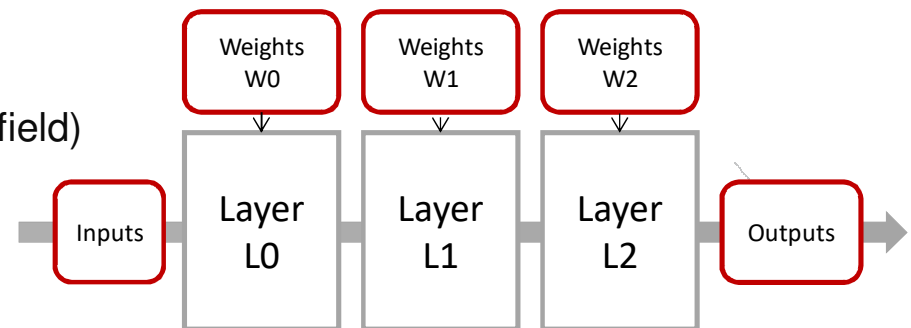
* With exception of RNNs

XILINX

Convolutional Neural Networks (CNNs)
Why are they so popular?

> **Requires little or no domain expertise**

> **NNs are a "universal approximation function"**

> **If you make it big enough and train it enough**
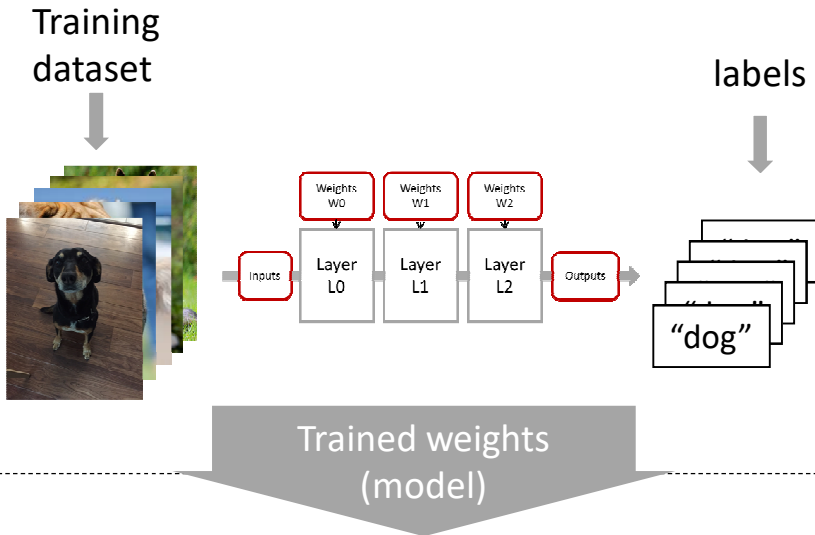>> Can outperform humans on specific tasks

> **Will increasingly replace other algorithms**
>> unless for example simple rules can describe the problem

> **Solve problems previously unsolved by computers**

> **And solve completely unsolved problems**

ILSVRC top-5 error on ImageNet
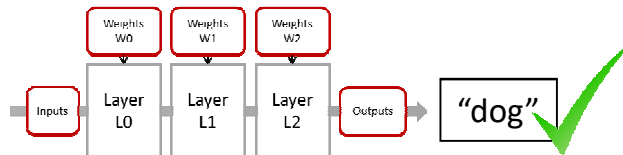
XILINX.
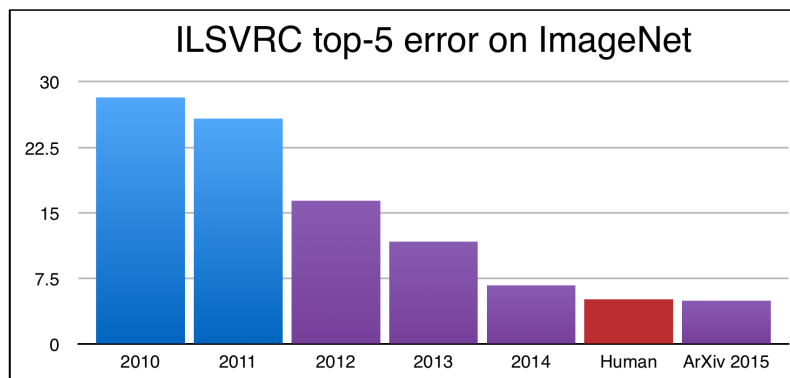
From Training to Inference

Training
dataset

labels

Training
Process for a machine to **_learn_** by optimizing models (weights) from labeled data.

Typically computed in the cloud

Trained weights
(model)
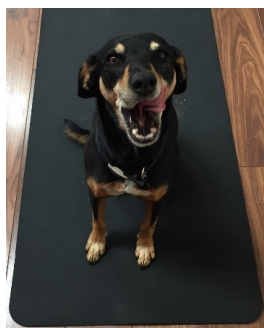
Inference
Using trained models to predict or estimate outcomes from new inputs.

Deployment at the edge

XILINX

Example: ResNet50
Forward Pass (Inference)

Input Image

(initialized)
Neural Network

Neural Network

| Weights | Weights | Weights |

Cat?
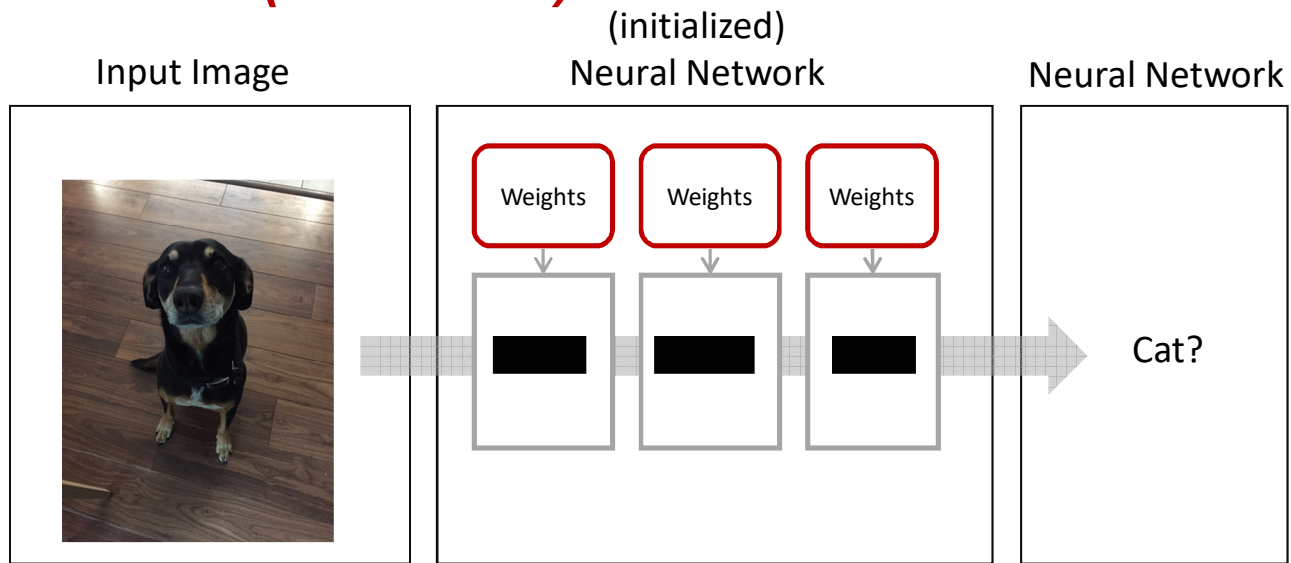
For ResNet50:

    70 Layers

    7.7 Billion operations

    25.5 MBytes of weight storage*

    10.1 MBytes for activations*

    Assuming int8

XILINX.

NNs in More Detail

feature extraction

classification

Convolutional Layers (CNV)

Pooling Layers (POOL)

Recurrent Layers (RL)

Fully Connected Layers

Activation & Batch Normalization

XILINX

Activation Functions

> **Implements the concept of "Firing"**
>> Non-linear so we can approximate more complex functions

> **Most popular for CNN: rectified linear unit (ReLU)****
>> Popular as it propagates gradients better than bounded and easy to compute
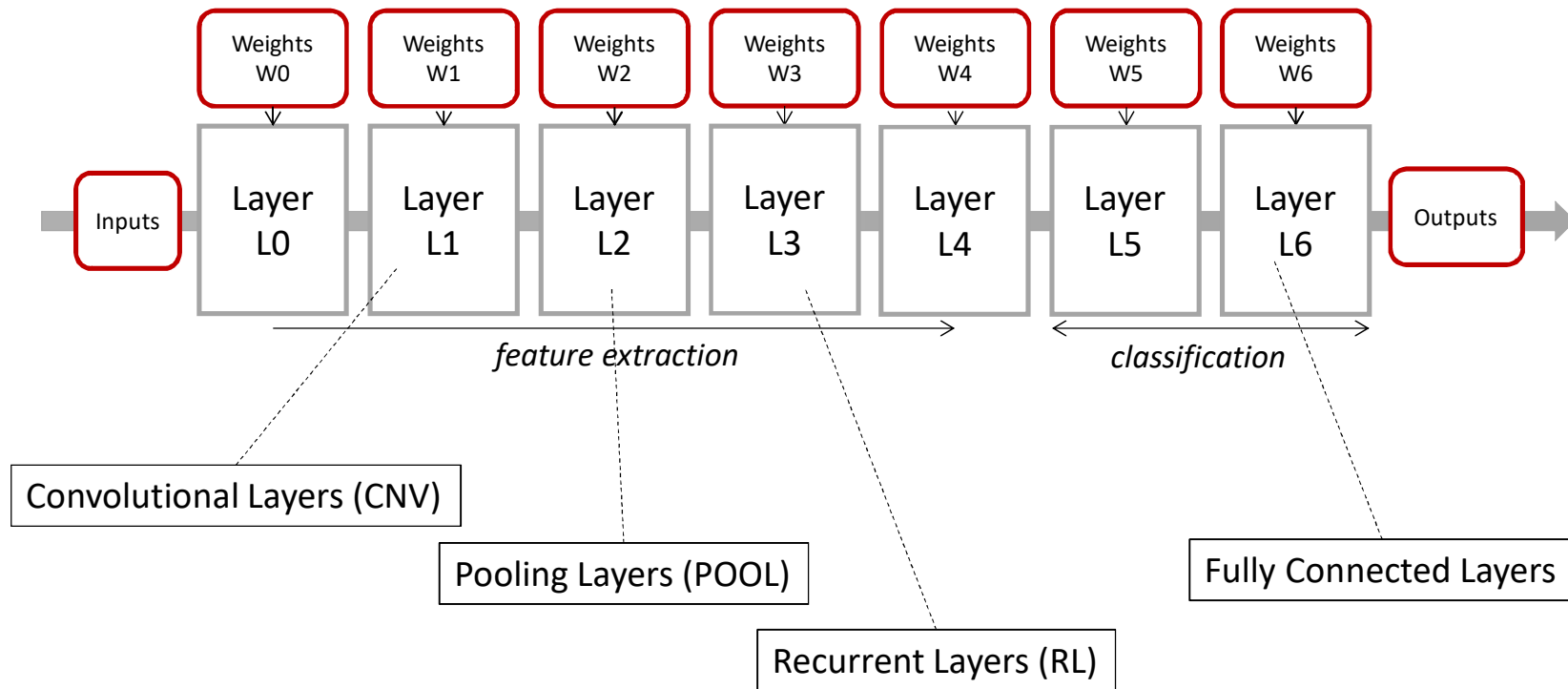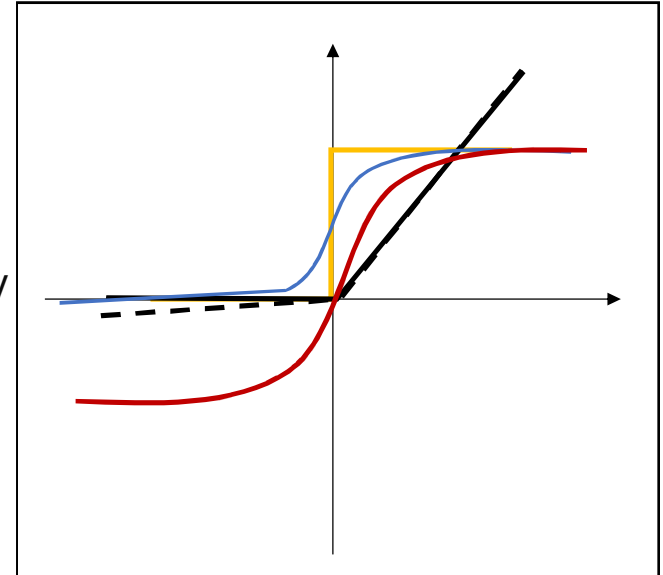>> However, recent work says as long as you have the proper initialization, it'll be fine even with bounded act. function*

> **Other common ones include: tanh, leaky ReLU, sigmoid, threshold functions for quantized neural networks**

> **Implementation:**
>> Support for special functions as well as some level of flexibility

*Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S.S. and Pennington "Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks." arXiv preprint arXiv:1806.05393 (2018).

**Nair, V. and Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

XILINX.

Batch Normalization

> **Normalizes the statistics of activation values across layers**

> **Significantly reduces the training time of networks, can improve accuracy and makes it less sensitive to initialization**

> **Compute:**
>> Lightweight at inference
>> Heavy duty during training
- – Subtract mean, divide by standard deviation to achieve zero-centered distribution with unit variance
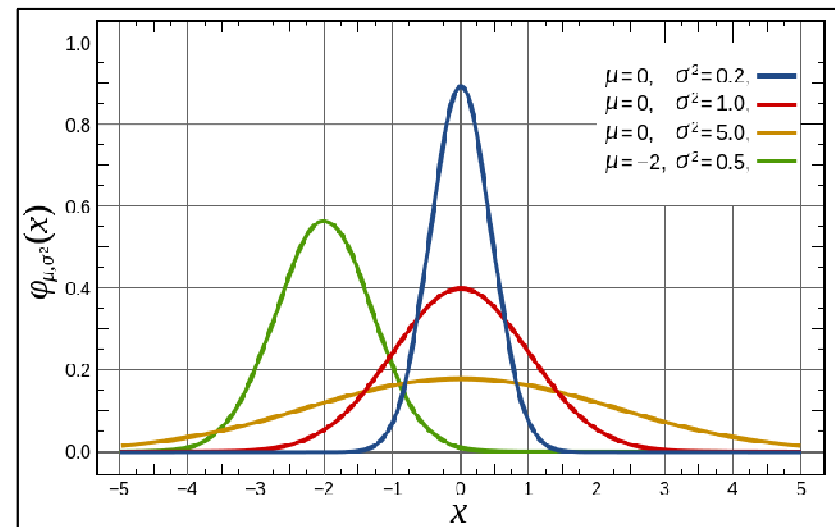
https://en.wikipedia.org/wiki/Normal_distribution

*Ioffe, S. and Szegedy, C., 2015. Batch normalization:
Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.*

XILINX

Fully Connected Layers
(aka inner product or dense layers)

> **Each input activation is connected to every output activation**
>> Receptive field encompasses the full input
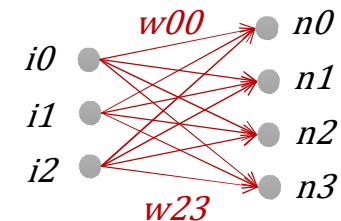
> **Can be written as a matrix-vector product with an element-wise non-linearity applied afterwards.**

> **Implementation Challenges**
>> Connectivity
>> High weight memory requirement: #IN * #OUT * BITS
>> Low arithmetic intensity assuming weights off-chip
   2 * #IN* #OUT / #IN * #OUT * BITS/8

$$\begin{bmatrix} i0\ i1\ i2 \end{bmatrix} \times \begin{bmatrix} W00\ W01\ W02\ W03 \\ W10\ W11\ W12\ W13 \\ W20\ W21\ W22\ W23 \end{bmatrix} = \begin{bmatrix} n0'\ n1'\ n2'\ n3' \end{bmatrix}$$

$$(n0\ n1\ n2\ n3) = Act(\ n0'\ n1'\ n2'\ n3')$$

MODEL	CONV WEIGHTS (M)	FC WEIGHTS (M)
ResNet50	23.454912	2.048
AlexNet	2.332704	58.621952
VGG16	14.710464	123.633664

IN: number of input channels
OUT: number of output channels
BITS: bit precision in data types

XILINX.

Convolutional Layers
Example 2D Convolution

> **Convolutions capture some kind of locality, spatial or temporal, that we know exists in the domain**

> **Receptive field of each neuron reduced**
>> Applying convolution to all images in the previous layer

> **Weights represent the filters used for convolutions**

filter

Input channels

input

output

Input channel 0

w00   w01
w10   w11

*

i00   i01   i02
i10   i11   i12
i20   i21   i22

=

n00   n01
n10   n11

Output channels

$n00 = Act(w00{*}i00 + w01{*}i01 + w10{*}i10 + w11{*}i11 + w00{*}i00 + w01{*}i01 + w10{*}i10 + w11{*}i11 + w00{*}i00 + w01{*}i01 + w10{*}i10 + w11{*}i11)$

w00   w01
w10   w11

Feature maps

n00   n01
n10   n11

 XILINX.
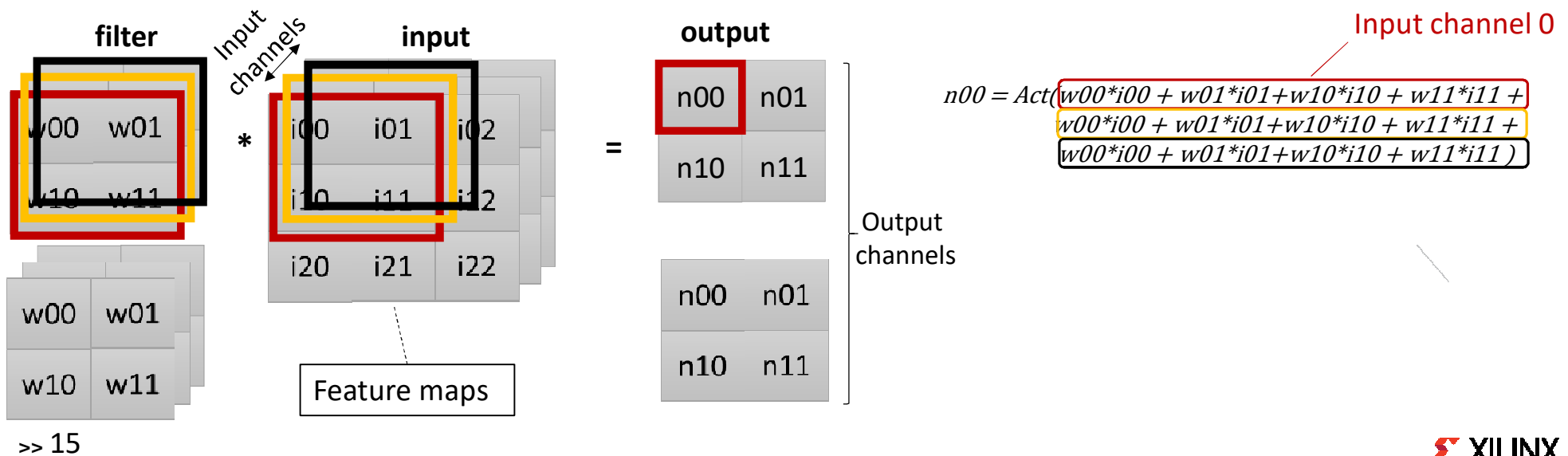
2D Convolutional Layers

> **Slide the window till one feature map is complete**
>> With a given stride size

Stride = 1

filter	input	output

filter

w00 w01
w10 w11

w00 w01
w10 w11

*

input

i00	i01	i02
i10	i11	i12
i20	i21	i22

=

output

n00	n01
n10	n11

n00	n01
n10	n11

XILINX.
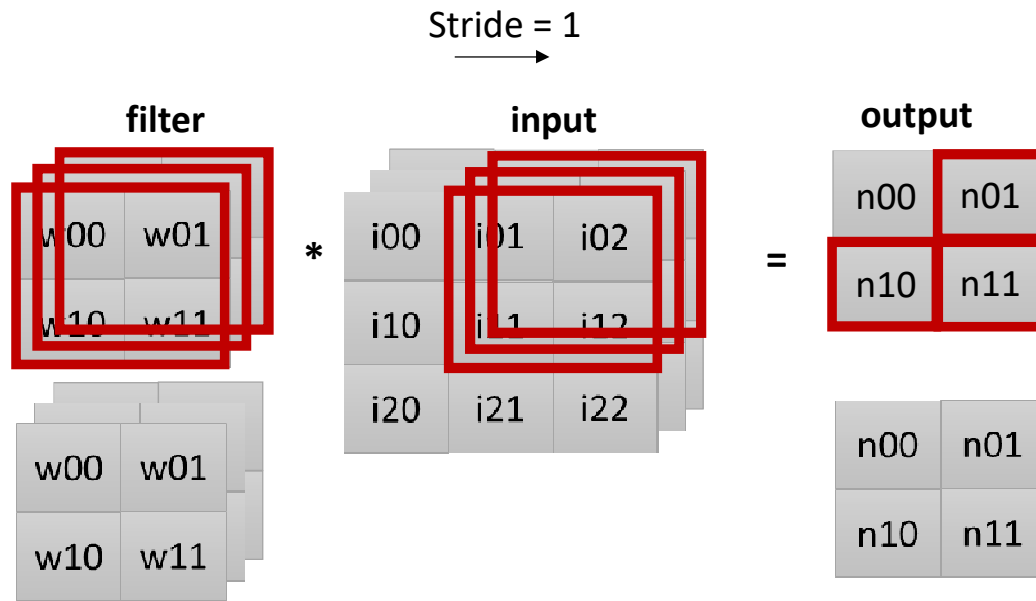
2D Convolutional Layers

> **Compute next channel**

XILINX.
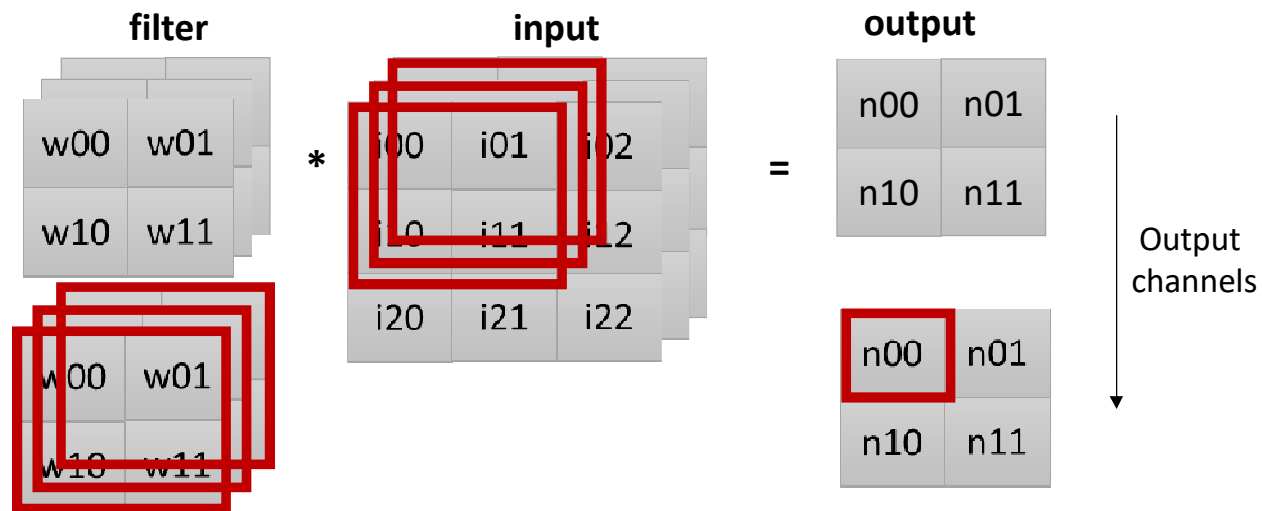
Convolutions
Challenges

> **Channel connectivity issue**
  >> Every input channel information broadcasts to every output channel

#IN_CH: i0, i1, i2, ..., in-1

#OUT_CH: n0, n1, ..., nm-1
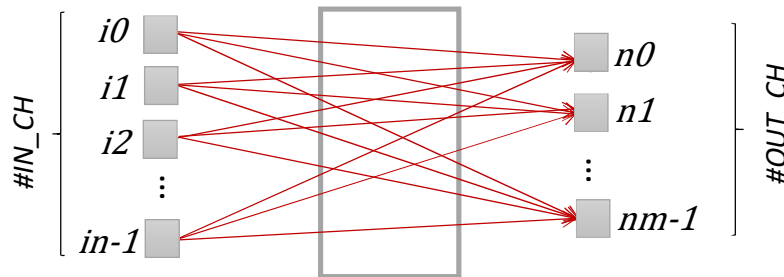
100s to 1000 channels

MODEL	CONV [GOPS]	FC [GOPS]
ResNet50	7.712	0.004
AlexNet	1.332	0.044
VGG16	30.693	0.247

> **Huge amounts of compute**
  >> Dense convolutions account for the majority of the compute

> **Novel (Non-Dense) Convolutions**
  >> Less spatial convolutions (1x1) (SqueezeNet's FireModules)
  >> Connectivity reduction between in and out channels (Shuffle, Shift layers)

=> Optimizations

 XILINX.

2D Convolutional Layers
1 input and 1 output channel

> Can be lowered to a matrix-matrix multiply using a Toeplitz Matrix

Convolution

filter

w00	w01
w10	w11

*

input

i00	i01	i02
i10	i11	i12
i20	i21	i22

=

output

n00	n01
n10	n11

$n00 = Act(w00*i00 + w01*i01 + w10*i10 + w11*i11)$

<=>

Toeplitz Matrix ("lowered image matrix")

Matrix Vector

filter

w00	w01	w10	w11

x

i00	i01	i10	i11
i01	i02	i11	i12
i10	i11	i20	i21
i11	i12	i21	i22

Duplication!

=

output

n00	n01	n10	n11

£ XILINX.
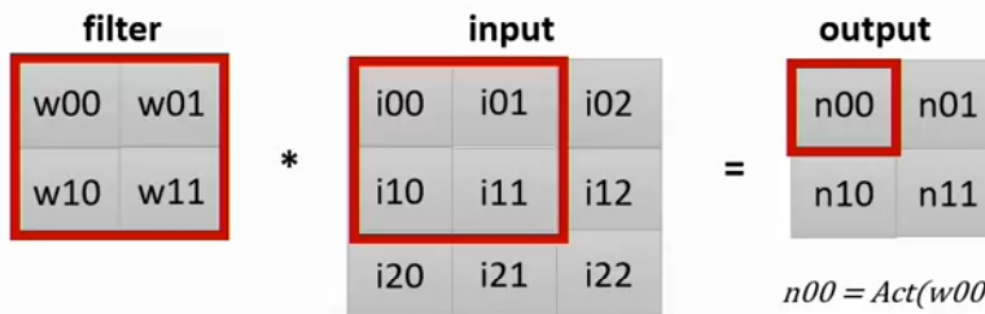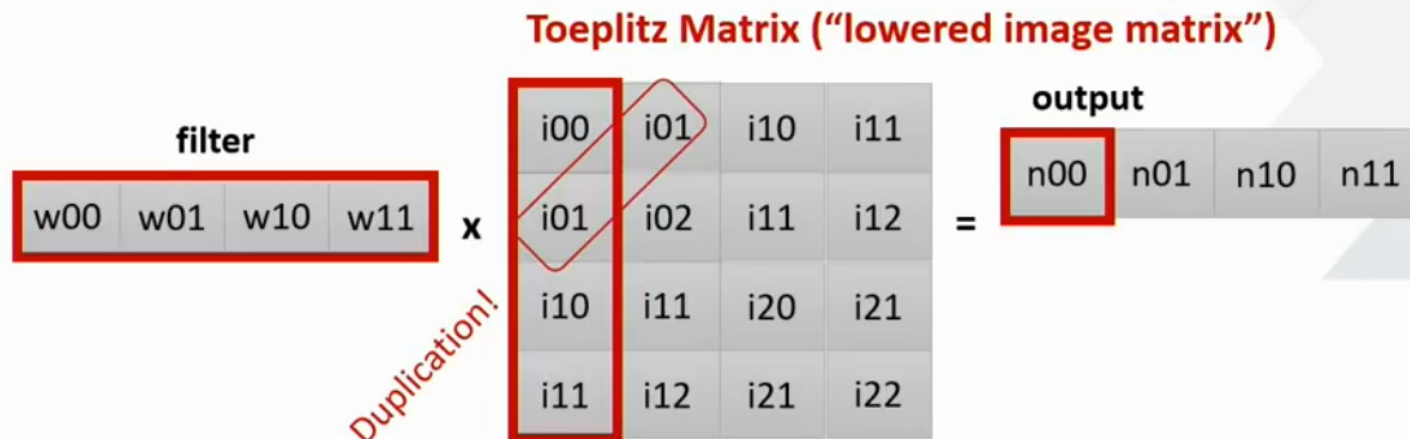
2D Convolutional Layers
3 input and 2 output channels

Convolution

filter

input

output

w00 w01

w10 w11

w00 w01

w10 w11

i00 i01 i02

i10 i11 i12

i20 i21 i22

n00 n01

n10 n11

n00 n01

n10 n11

*

=

<=>
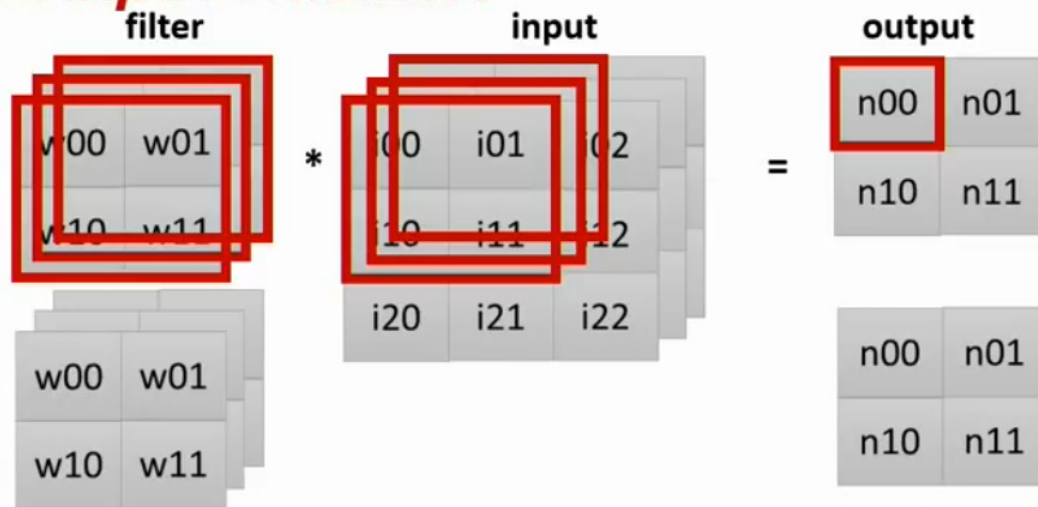
Matrix Matrix

Toeplitz Matrix

filters

Input channels

output

Output channel 0,1

IN_CH0 w11 ... w00 IN

w00 w01 w10 w11 ... w00 w01

i00 i01 i10 i11

i01 i02 i11 i12

i10 i11 i20 i21

i11 i12 i21 i22

IN_CH0

IN_CH2

Data duplication for taking advantage of linear algebra libraries such as OpenBLAS, cuBLAS, cuDNN

>> 25

XILINX
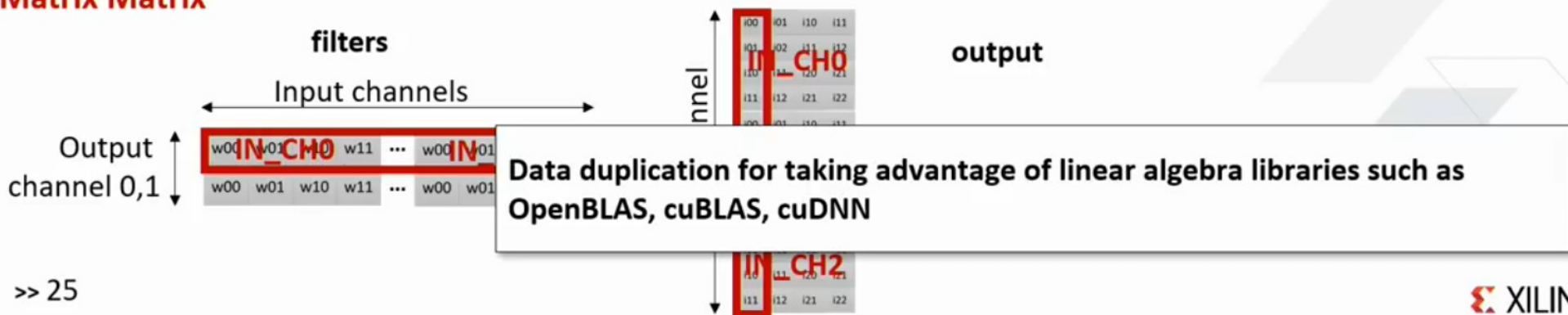
Convolutions
Challenges

> **Parallelization of compute across layers reduces memory bandwidth required for buffering activations in between layers**

> **Pyramid-shaped data dependency between activations across layers**

Alwani, M., Chen, H., Ferdman, M. and Milder, P., 2016, October. Fused-layer CNN accelerators. Micro 2016

Pooling Layer

> **Down-samplers of images**

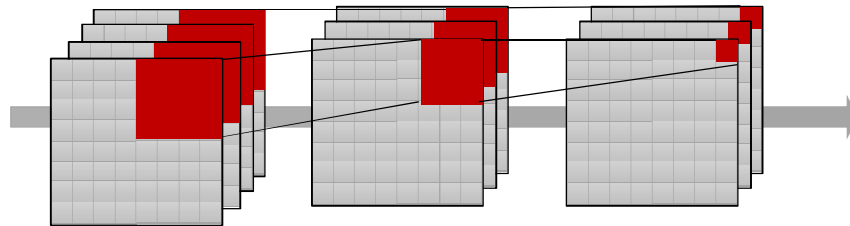> **Reduces compute in subsequent layers**

> **May use MAX or AVERAGE**

> **Compute:**
>> Low amount of compute
>> Potentially replaceable with larger strides in previous convolution

Max pool with 2x2 filters and stride of 2:

input   output

i00	i01	i02	i03
i10	i11	i12	i13
i20	i21	i22	i23
i30	i31	i32	i33

n00	n01
n10	n11

$n00 = Max(i00, i01, i10, i11)$

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for Simplicity: The all convolutional net.

XILINX.
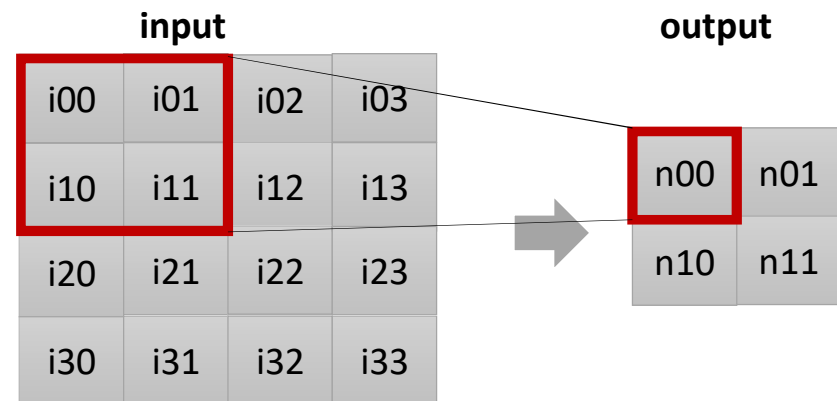
Recurrent Layer Types

> **Contain state for processing sequences**
>> For example needed in speech or optical character recognition
>> "Apocal???"

> **Uni-directional or bi-directional**
>> "I ???? You"

> **More sophisticated types to address the vanishing gradients problem for learning more than 5-10 timesteps**
>> GRU (gated recurrent unit)
>> LSTM (long short term memory)

Hopefully the AI Apocalypse won't happen during my lifetime.

XILINX
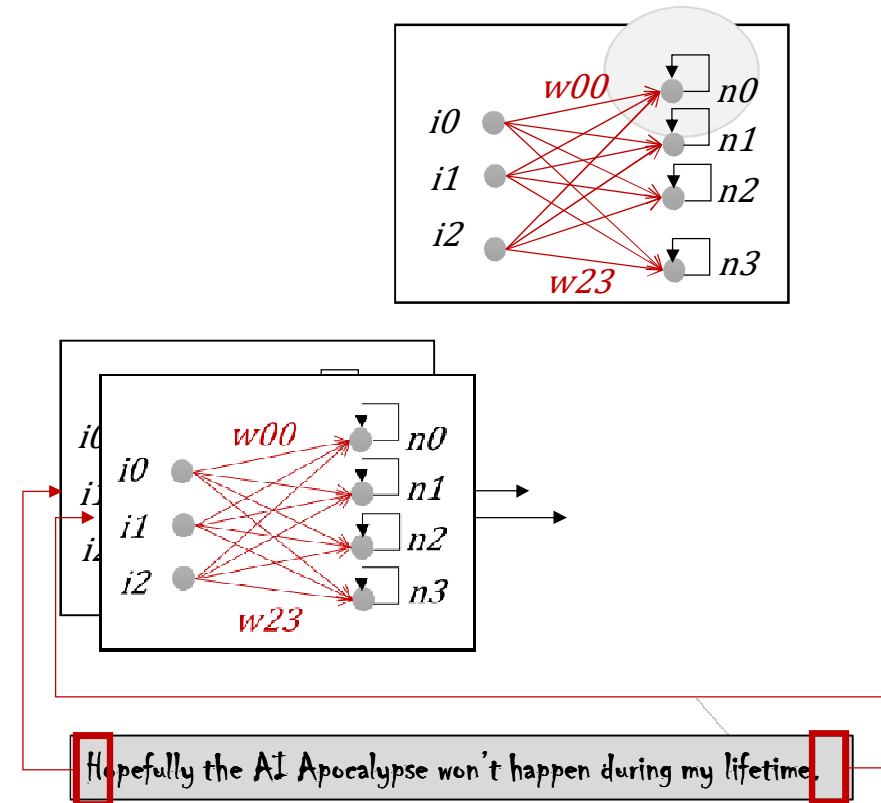
Recurrent Layers
Challenges in Additional Data Dependencies

> **Input sequence**
>> Unlike batch, additional data dependencies between inputs of the same sequence and state

> **Bi-directional NNs**
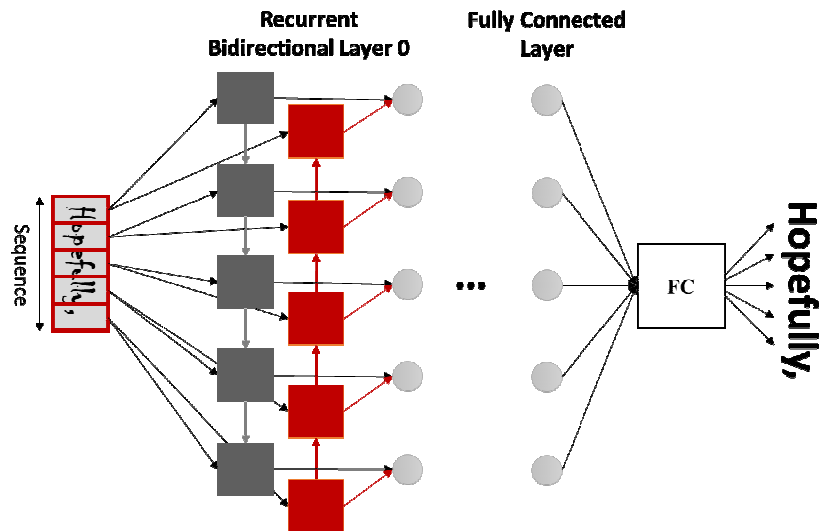>> Full sequence needs to be completed before the next layer

XILINX.

Meta-Layers

> **Residual layers (ResNets *)**
>> Introduced to make larger networks more trainable
>> Better gradient propagation through skip connections during training
>> Plus 1x1 convolutions to reduce dimensionality and save compute

| CNV 3x3, 64, Relu |
| CNV 3x3, 64, Relu |

| CNV 1x1, 64, Relu |
| CNV 3x3, 64, Relu |
| CNV 1x1, 256, Relu |

Elementwise addition
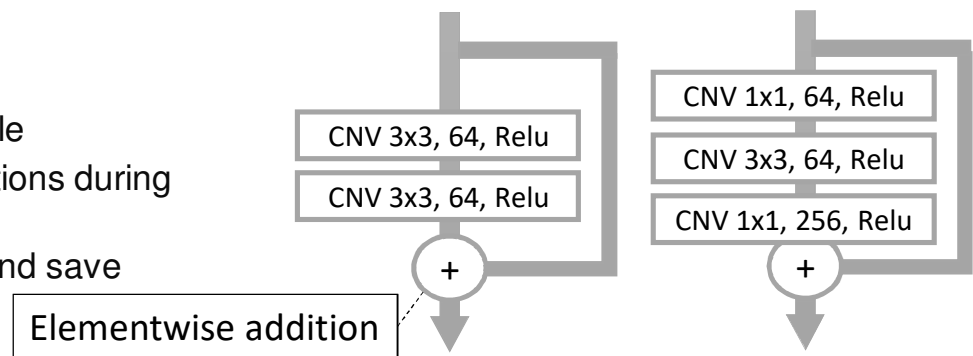
> **Inception layers (GoogleNet**)**
>> Huge variation in spatial features => combining different size convolutions in one layer
>> Plus 1x1 convolutions to reduce dimensionality and save compute
>> Later on additional factorization to reduce compute
>>> – 3x3 = 1x3 and 3x1

CNV 1x1

| CNV 1x1 | CNV 1x1 | CNV 3x3 |
| CNV 3x3 | CNV 5x5 | CNV 1x1 |

cc

Concatenation
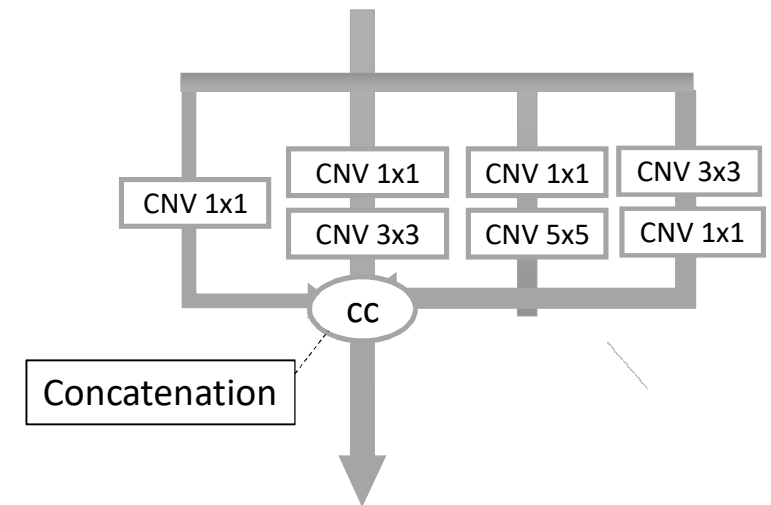
> **Many more…**

> **Implementation: support for non-linear topologies!**

*He, K., Zhang, X., Ren, S. and Sun, J. "Deep residual learning for image recognition." CVPR' 2016.
** Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. "Rethinking the inception architecture for computer vision." CVPR' 2016.

XILINX.

Computation & Memory Requirements

Compute and Memory Requirements
Architecture Neutral, Per Layer

Memory Requirements:
$$A_{total} = \sum A_i$$
$$W_{total} = \sum W_i$$

Compute Requirements:
$$O_{total} = \sum O_i$$

Layer i

Compute Elements
Number of Operations : O_i

Memory Elements
Weights W_i
Activations A_i

Weights

Compute

Activations
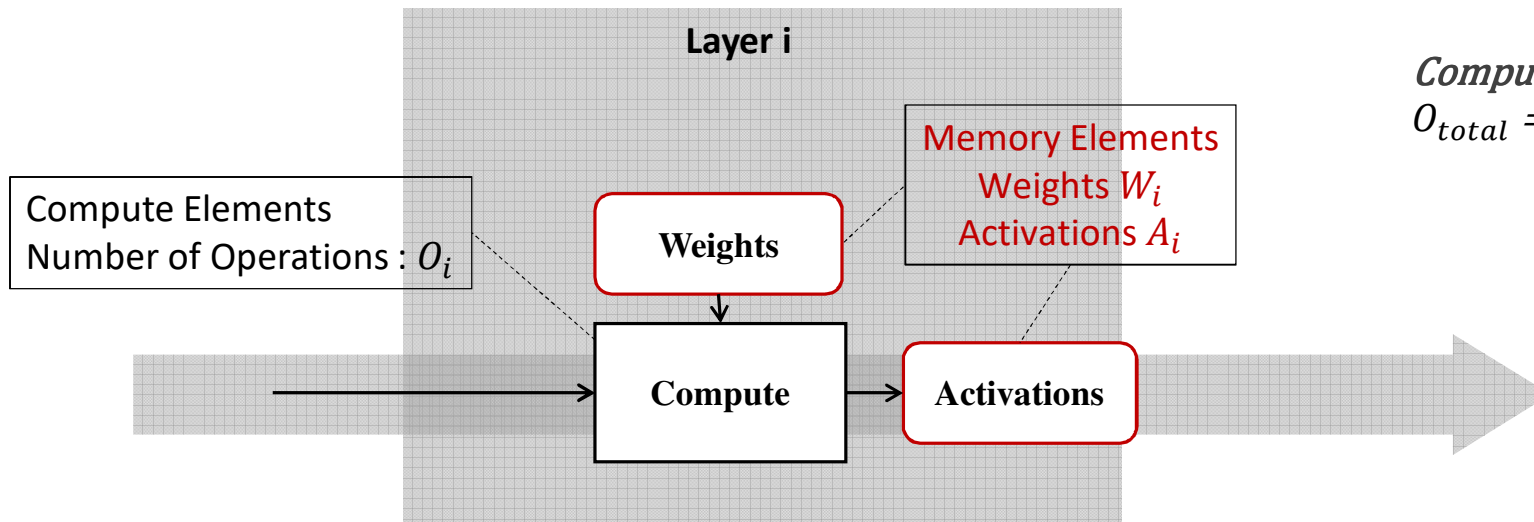
IN, IN_CH:	number of inputs and input channels
OUT, OUT_CH:	number of outputs and output channels
F_DIM, FM_DIM:	filter and feature map dimensions (assumed square)
BATCH:	batch size
BITS:	bit precision in data types
GATES:	number of gates in RNNs:
STATES:	worst case
SEQ:	sequence length
HID:	hidden size (state + output from each state)
DIRS:	1 for unidirectional and 2 for bidirectional RNN

XILINX

Inference Compute and Memory
Across a Spectrum of Neural Networks

*architecture independent
**1 image forward
*** batch = 1
**** int8

Spectrum of Neural Networks

GOPS and MBytes respectively

**Inference (1 input)
GOPS**

— — average

**Inference (1 input)
MBytes**

— — average

L3 Cache size Processor

MLP — **ImageNet Classification CNNs** — **Object Detection**

LFC, SFC, VGG16, AlexNet, GoogleNet, Inception v.3, Resnet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, MobileNet, YoloV, TinyYolo, Yolo, FCN, Net50, SSD

Networks

1. **Inference is hard**
2. **Huge Variation in Compute and Memory Requirements, even within subgroups**
3. **Models typically don't fit into cache**

XILINX.

Rooflines*

Williams, S., Waterman, A. and Patterson, D., 2009.
Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM

Arithmetic Intensity
Across a Spectrum of Neural Networks

> Memory requirement for weights, activations are beyond typically available on-chip memory

> This yields low arithmetic intensity
>> For example for inference, assuming weights off-chip and naïve implementation, majority of networks is below 6OPS:Byte
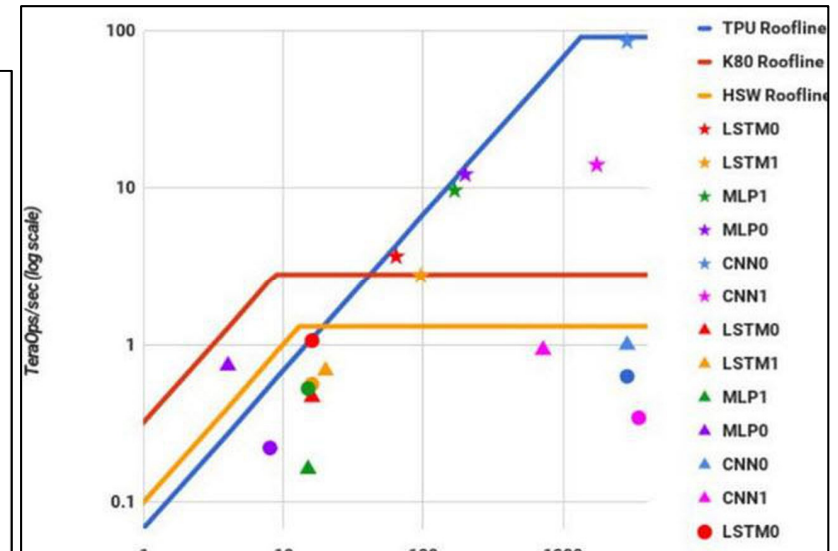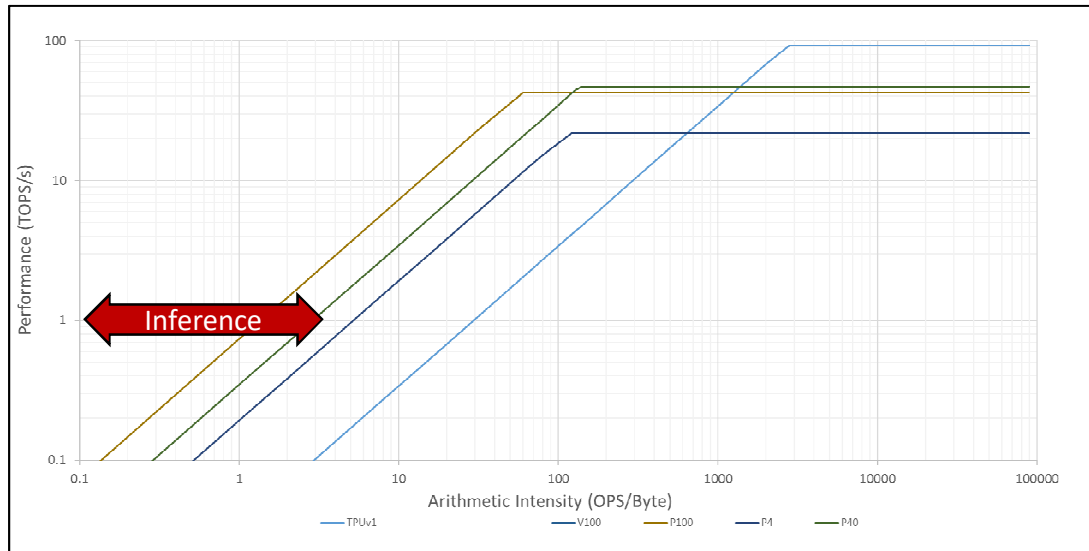
Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A. and Boyle, R., 2017, June. In-datacenter performance analysis of a tensor processing unit. ISCA'2017

XILINX

In Summary: CNNs are associated with…

> **Significant amounts of memory and computation**

> **Huge variation between topologies and within them**

> **Fast changing algorithms**

> **Special functions, non-linear topologies**

> **However, incredibly parallel!**
>> For convolutions: filter dimensions, feature map dimensions, input & output channels, batches, layers, and even precisions (discussed later)
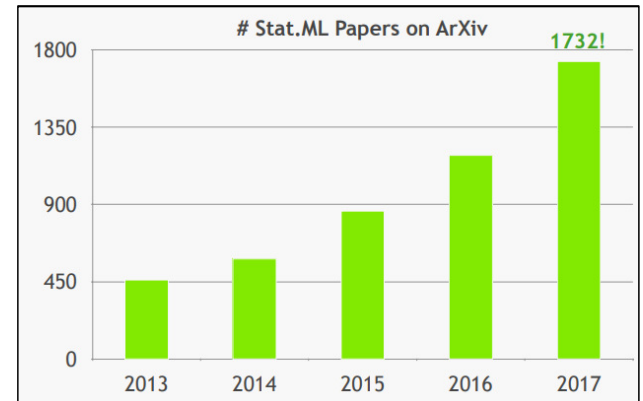
Adopted from Ce Zhang, ETH Zurich, Systems Group Retreat

XILINX

Architectural Challenges/ Pain Points

DRA

NN Inference/ Training Accelerator

DMA

Weight Buffer

Activation Buffering

Compute Array

Partial Sums

Activation Functions/ Pooling...

Input samples

Results

Weight & activation fetching: bandwidth throttles performance

Power consumption for embedded

Latency in real-time processing

Huge amount of memory spilling into DRAM And variations

Huge amount of compute and variation- Limited scalability with new technology nodes

Requires algorithmic & architectural innovation

XILINX.

Algorithmic Optimization Techniques

XILINX.

> 33

Optimization Techniques

Loop transformations to minimize memory access*

Pruning

Compression

Winograd, Strassen and FFT

Novel layer types (squeeze, shuffle, shift)

Numerical Representations & Reducing Precision

DRAM

Weight & activation fetching: bandwidth throttles performance

NN Inference/ Training Accelerator

DMA

Weight Buffer

Power consumption for embedded

Latency in real-time processing

Huge amount of memory spilling into DRAM

Input samples

Input & Activation Buffering

Compute Array

Results

Huge amount of compute - Limited scalability with new technology nodes

Partial Sums

Activation Functions/ Pooling...

*Chen, Y.H., Krishna, T., Emer, J.S. and Sze, V., 2017. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits, 52(1), pp.127-13

XILINX.

Example: Reducing Bit-Precision

> **Linear reduction in memory footprint**
>> Reduces weight fetching memory bandwidth
>> NN model may even stay on-chip

> **Reducing precision shrinks inherent arithmetic cost in both ASICs and FPGAs**
>> Instantiate **100x** more compute within the same fabric and thereby scale performance

Precision	Modelsize [MB] (ResNet50)
1b	3.2
8b	25.5
32b	102.5
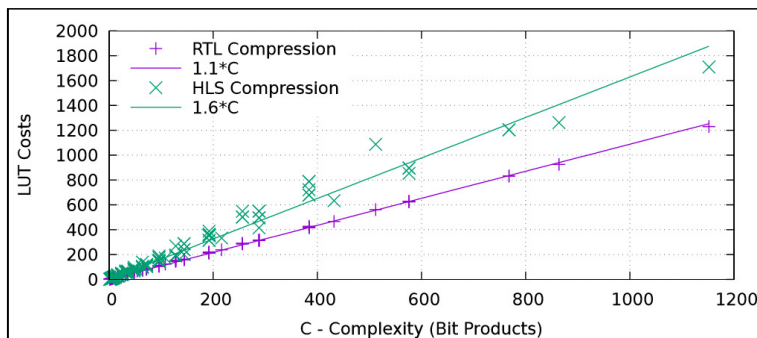
C= size of accumulator * size of weight * size of activation
(to appear in ACM TRETS SE on DL, FINN-R)

>> 35

Reducing Precision provides Performance Scalability
Example: ResNet50, ResNet152 and TinyYolo

Theoretical Peak Performance for a VU13P with different Precision Operations
Assumptions: Application can fill device to 90% (fully parallelizable) 710MHz

RP reduces model size=> to stay on-chip
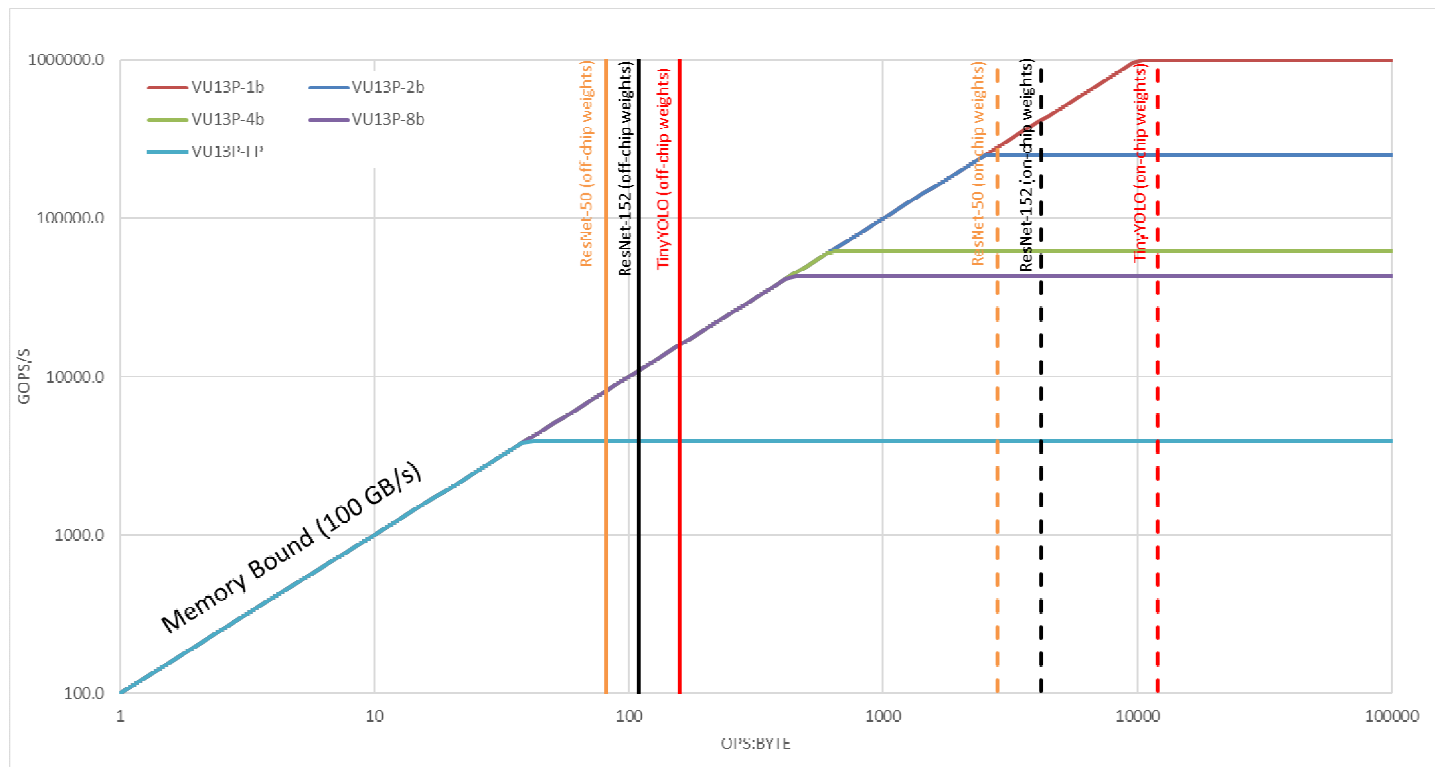
RP scales compute performance

Reducing Precision Inherently Saves Power

FPGA:

LSTM - Test Error vs Power(W)

ASIC:

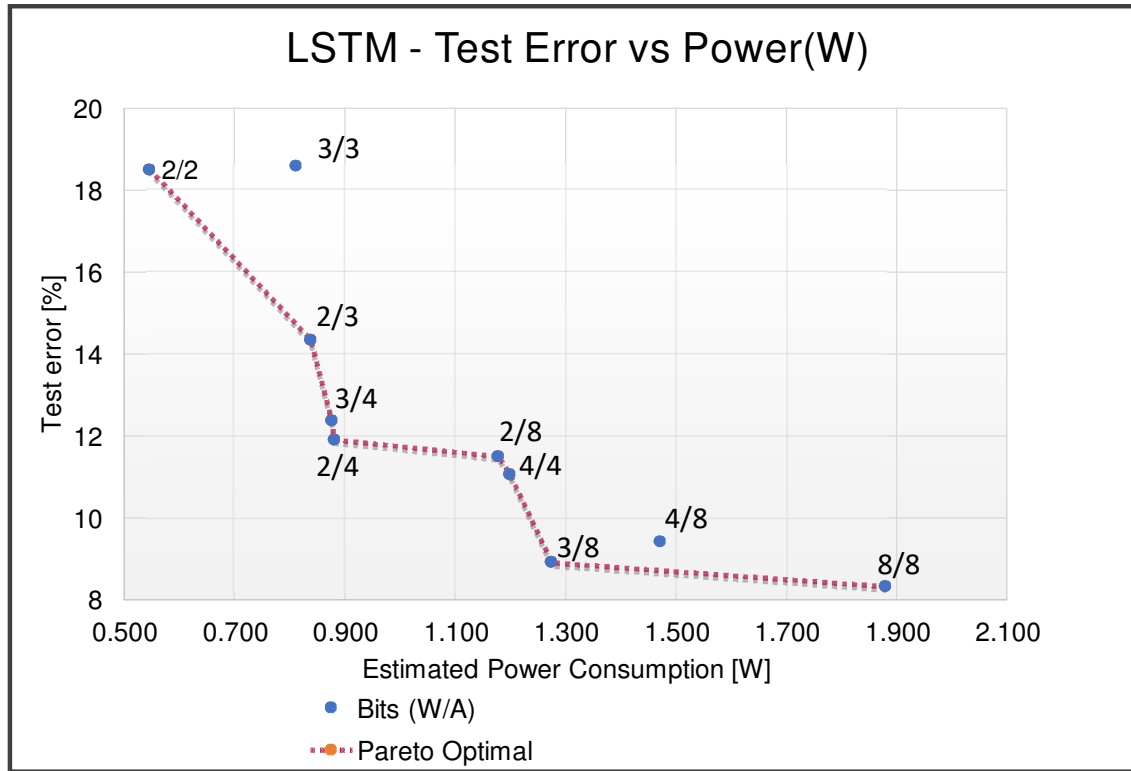| Operation:	Energy (pJ)	Relative Energy Cost
8b Add	0.03	
16b Add	0.05	
32b Add	0.1	
16b FP Add	0.4	
32b FP Add	0.9	
8b Mult	0.2	
32b Mult	3.1	
16b FP Mult	1.1	
32b FP Mult	3.7	
32b SRAM Read (8KB)	5	
32b DRAM Read	640	

Source: Bill Dally (Stanford), Cadence Embedded Neural Network Summit, February 1, 2017

Target Device ZU7EV ● Ambient temperature: 25 °C ● 12.5% of toggle rate ● 0.5 of Static Probability ● Power reported for PL accelerated block only

Rybalkin, V., Pappalardo, A., Ghaffar, M.M., Gambardella, G., Wehn, N. and Blott, M. "FINN-L: Library Extensions and Design Trade-off Analysis for Variable Precision LSTM Networks on FPGAs."

XILINX.

RPNNs: Closing the Accuracy Gap

Top-5 Error (ImageNet)

Float point improvements are slowing down
Reduced precision highly competitive and rapidly improving
BNNs and TNNs are still rapidly improving <10% top5

Error (%) — Publication Date

BNN   CNN   Reduced Precision
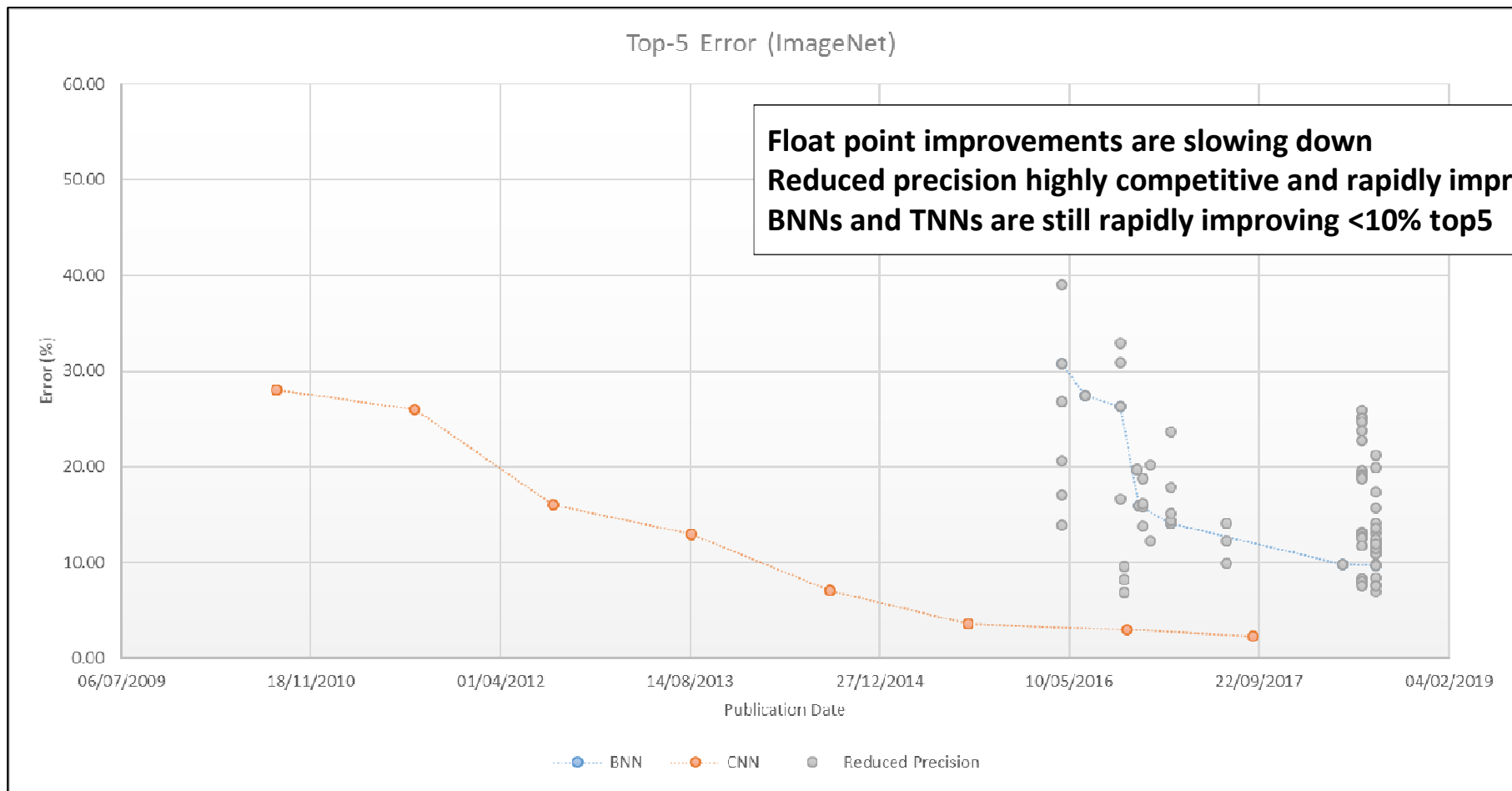
Latest numbers: Dongqing Zhang∗, Jiaolong Yang∗, Dongqiangzi Ye∗, and Gang Hua
"LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks"

Design Space Trade-Offs

IMAGENET CLASSIFICATION TOP5% VS COMPUTE COST F(LUT,DSP)

Legend: ◆ 1b weights ▲ 2b weights ✕ 5bit weights ● 8bit weights ✳ FP weights ■ minifloat + ResNet-50 – Syq

Resnet18
8b/8b
Compute Cost 286
Error 10.68%

Resnet50
2b/8b
Compute Cost 127
Error 9.86%

Pareto-optimal solutions

Reduced Precision can provide better accuracy and lower hardware cost for specific accuracy targets
In order to find optimal solutions, solution space needs to be considered and allow for algorithmic freedom

✗ XILINX.

Hardware Architectures and their Specialization Towards CNN Workloads
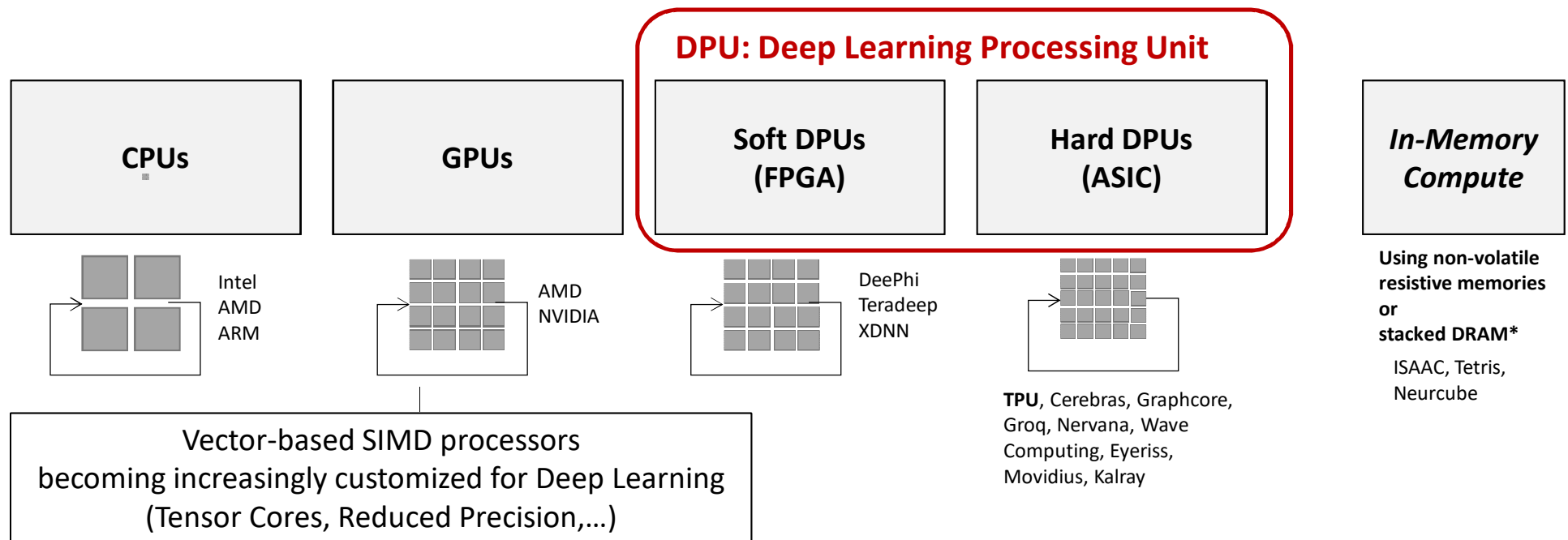## *Exciting Times in Computer Architecture Research!*

XILINX.

Spectrum of New Architectures for Deep Learning

DPU: Deep Learning Processing Unit

CPUs	GPUs	Soft DPUs (FPGA)	Hard DPUs (ASIC)	In-Memory Compute

Intel
AMD
ARM

AMD
NVIDIA

DeePhi
Teradeep
XDNN

TPU, Cerebras, Graphcore, Groq, Nervana, Wave Computing, Eyeriss, Movidius, Kalray

Using non-volatile resistive memories or stacked DRAM*

ISAAC, Tetris, Neurcube

Vector-based SIMD processors becoming increasingly customized for Deep Learning (Tensor Cores, Reduced Precision,...)
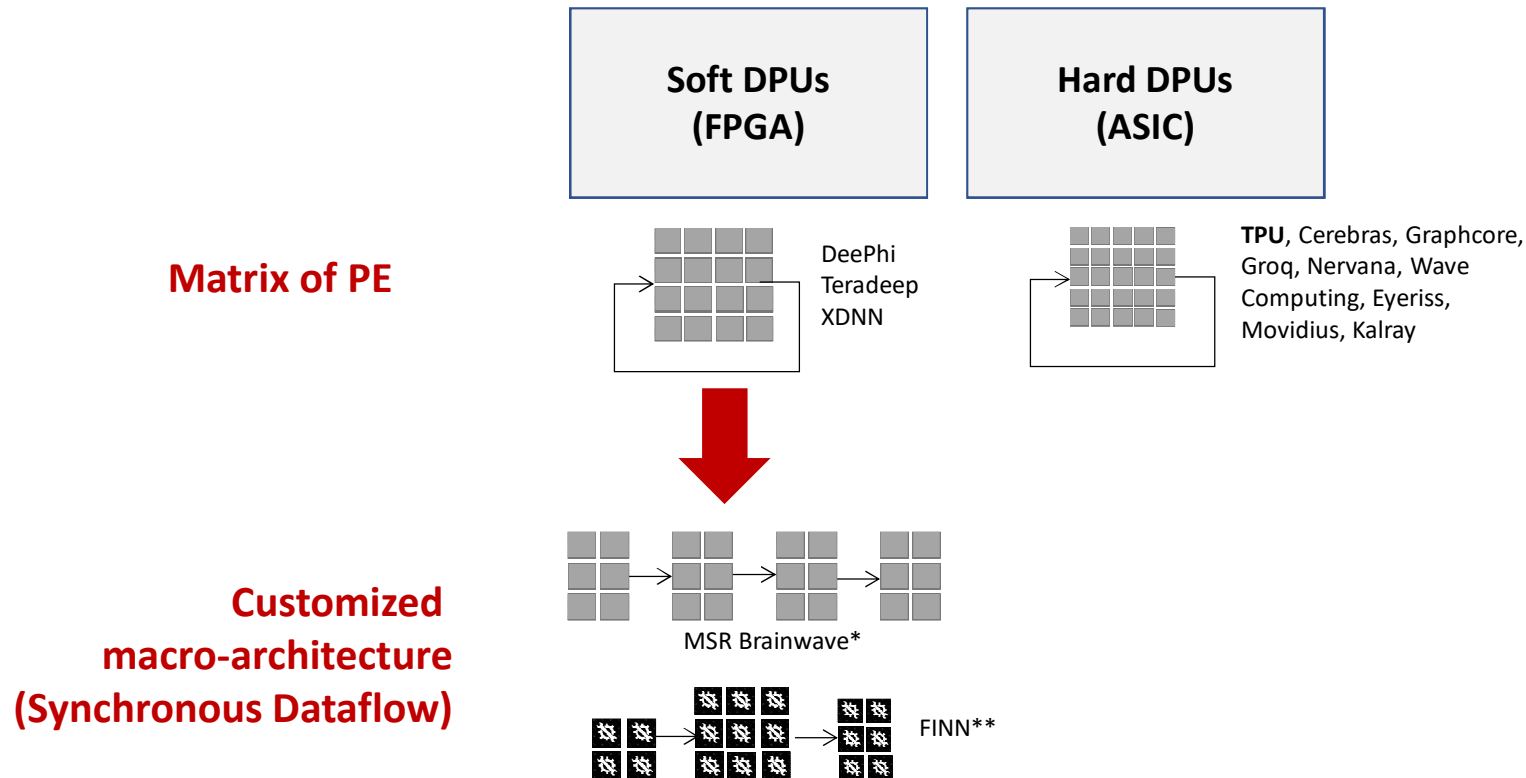
*Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J.P., Hu, M., Williams, R.S. and Srikumar, V., 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. ACM SIGARCH
Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y. and Xie, Y., 2016, June. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In ACM SIGARCH
Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O., 2014, December. Dadiannao: A machine-learning supercomputer. In Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 609-622). IEEE Computer Society.

XILINX.

Architectural Choices – Macro-Architecture

Soft DPUs (FPGA)

Hard DPUs (ASIC)

Matrix of PE

DeePhi
Teradeep
XDNN

TPU, Cerebras, Graphcore, Groq, Nervana, Wave Computing, Eyeriss, Movidius, Kalray

**Customized macro-architecture (Synchronous Dataflow)**

MSR Brainwave*

FINN**

*Chung, E., Fowers, J., Ovtcharov, K., Papamichael, M., Caulfield, A., Massengill, T., Liu, M., Lo, D., Alkalay, S., Haselman, M. and Abeydeera, M.Serving DNNs in Real Time at Datacenter Scale with Project Brainwave. IEEE Micro, 38(2)
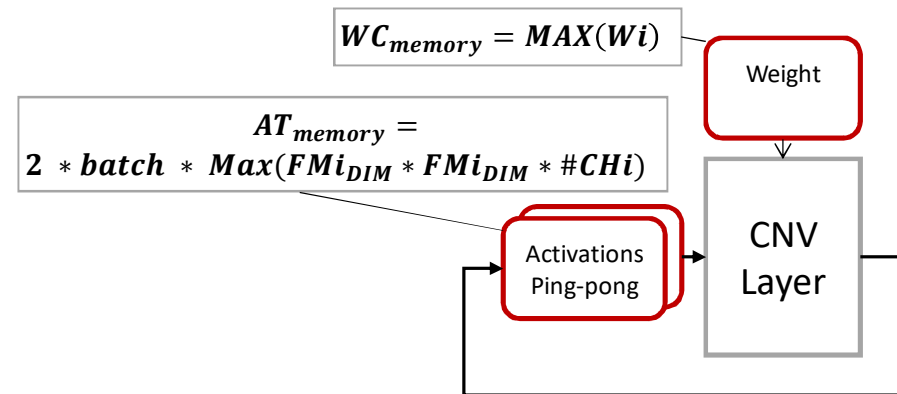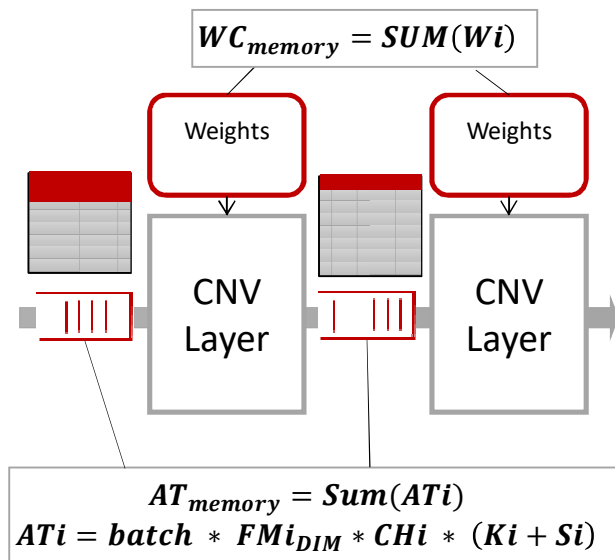https://www.microsoft.com/en-us/research/uploads/prod/2018/06/ISCA18-Brainwave-CameraReady.pdf
**Umuroglu, Yaman, Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P., Jahre, M. and Vissers, K. "FINN: A framework for fast, scalable binarized neural network inference." ISFPGA'2017

XILINX.

Synchronous Dataflow (SDF) vs Matrix of Processing Elements (MPE)

Spectrum of Options
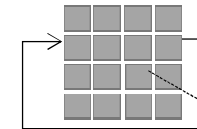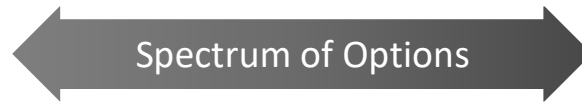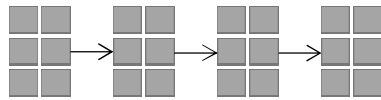
MAC, Vector Processor

>> End points are pure layer-by-layer compute and feed-forward dataflow architecture

$$WC_{memory} = SUM(Wi)$$

Weights

Weights

CNV Layer

CNV Layer

$$AT_{memory} = Sum(ATi)$$
$$ATi = batch * FMi_{DIM} * CHi * (Ki + Si)$$

$$WC_{memory} = MAX(Wi)$$

Weight

$$AT_{memory} = 2 * batch * Max(FMi_{DIM} * FMi_{DIM} * \#CHi)$$

Activations Ping-pong

CNV Layer

Lin, X., Yin, S., Tu, F., Liu, L., Li, X. and Wei, S. LCP: a layer clusters paralleling mapping method for accelerating inception and residual networks on FPGA. DAC'2016.
Alwani, M., Chen, H., Ferdman, M. and Milder, P. Fused-layer CNN accelerators. MICRO 2016.

XILINX

Synchronous Dataflow (SDF) vs Matrix of Processing Elements (MPE)

Degree of parallelization across layers

- Requires less activation buffering

- Higher compute and memory efficiency due to custom-tailored hardware design

- Less flexibility

- Less latency (reduced buffering)

- No control flow (static schedule)

- Requires less on-chip weight memory, but more activation buffers

- Efficiency of memory for weights and activations depends on how well balanced the topology is

- Flexible hardware, which can scale to arbitrary large networks

- Compute efficiency is a scheduling problem => generating sophisticated scheduling algorithms

XILINX.

Architectural Choices – Micro-Architecture

CPUs	GPUs	Soft DPUs (FPGA)	Hard DPUs (ASIC)

Intel
AMD
ARM

AMD
NVIDIA

DeePhi
Teradeep
XDNN

TPU, Cerebras, Graphcore, Groq, Nervana, Wave Computing, Eyeriss, Movidius, Kalray

MSR Brainwave

FINN

Customized arithmetic

BISMO

Stripes (bit-serial ASIC), Stanford, Leuven: BinarEye IBMs' TrueNorth & latest AI accelerator
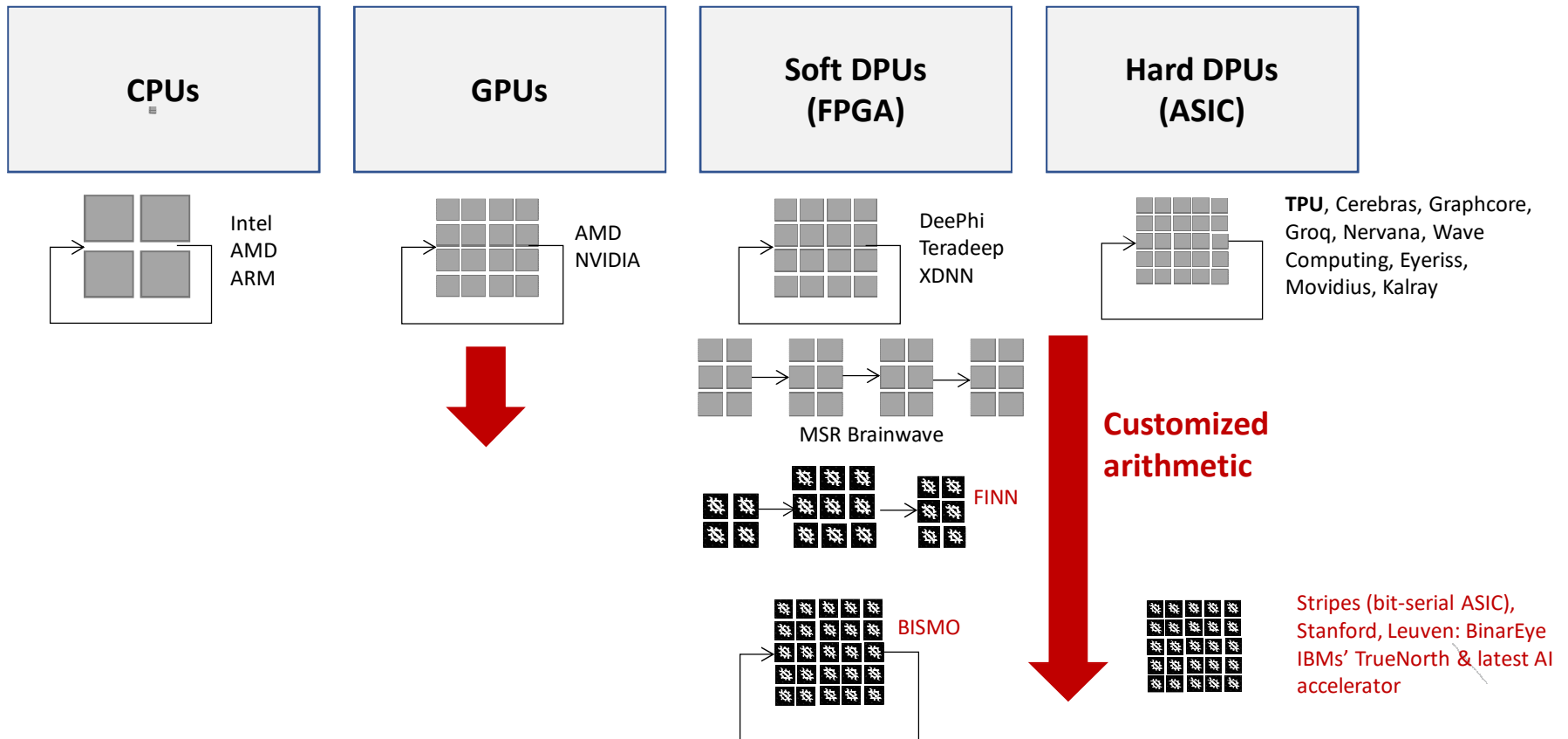
Judd, P., Albericio, J., Hetherington, T., Aamodt, T.M. and Moshovos, A., 2016, October. Stripes: Bit-serial deep neural network computing. MICRO'2016

Moons, B., Bankman, D., Yang, L., Murmann, B. and Verhelst, M. BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS, ICC'2018

Lin, X., Yin, S., Tu, F., Liu, L., Li, X. and Wei, S. LCP: a layer clusters paralleling mapping method for accelerating inception and residual networks on FPGA. DAC'2016

XILINX.

Micro-Architecture:
Customized Arithmetic for Specific Numerical Representations

> **Customizing arithmetic compute allows to maximize performance at minimal accuracy loss**
>> Flexpoint, Microsoft Floating Point formats, Binary & Ternary, Bfloat16
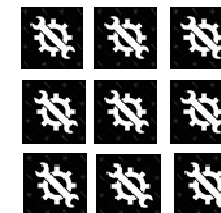
> **Which do we support?**
>> Perhaps too risky to support numerous, and too risky to fix on one?

> **What's more, non-uniform arithmetic can yield more efficient hardware implementations for a fixed accuracy***
>> Run-time programmable precision: Bit-Serial

	DEC	INC	CONCAVE	CONVEX
Top-1 [%]	53.79	50.35	54.45	54.33
Top-5 [%]	77.59	74.89	76.43	78.20

Table 2. Accuracy comparison of our approach under different styles of layer-wise quantization.

*Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo, "Weighted-Entropy-based Quantization for Deep Neural Networks" CVPR'2017]

XILINX

Summary

Summary

> **CNNs are increasingly being adopted for new workloads and key to the current industrial revolution and perhaps the next**

> **Associated with significant challenges**

> **Requires algorithmic and architectural innovation (co-designed)**

> **Emerging: Huge spectrum of algorithms and increasingly diverse & heterogenous hardware architectures**

> **Clear metrics for comparison needed**
>> Hardware performance always tying back to application performance (accuracy) to allow for algorithmic optimizations
>> Ideally in form of pareto curves: Accuracy - performance (TOPS/sec) - response time (1 input) - power consumption

>> 48

XILINX

Exciting Times for our Community:
Many New Architectures Evolving - Programmable and Hardened

- **Finding optimal solutions within a multi-dimensional design space**
  combinations trained network topologies on different datasets implemented in different ways on different hardware architectures

XILINX.