Architectures for Accelerating Deep Neural Networks

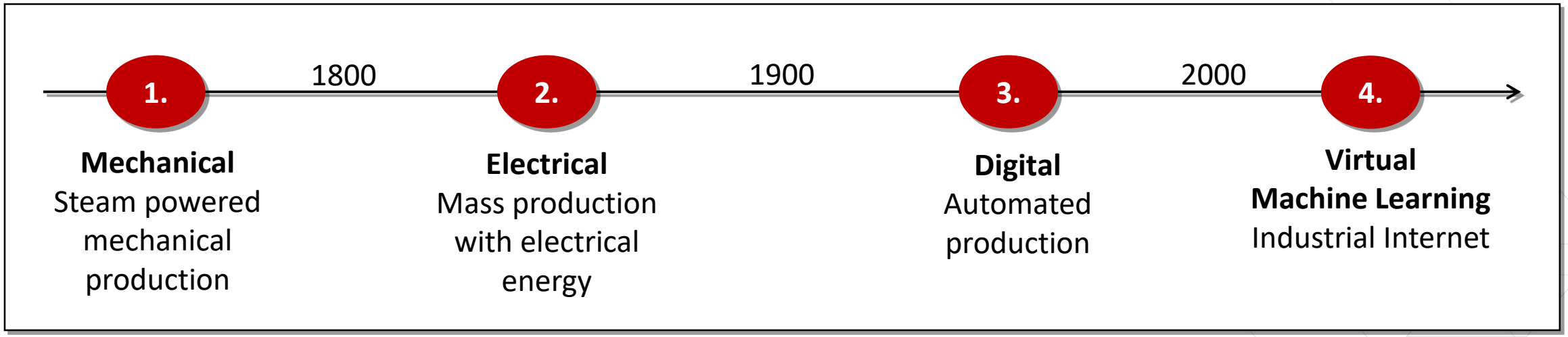
- **Part 1: Overview of Deep Learning and Computer Architectures for Accelerating DNNs**
  - Michaela Blott, Principal Engineer, Xilinx Research
- **Part 2: Accelerating Inference at the Edge**
  - Song Han, Assistant Professor, MIT
- **Part 3: Accelerating Training in the Cloud**
  - William L. Lynch, VP Engineering and Ardavan Pedram, MTS, Cerebras

Overview of Deep Learning and Computer Architectures for Accelerating DNNs

Michaela Blott
Principal Engineer
August 2018

XILINX.

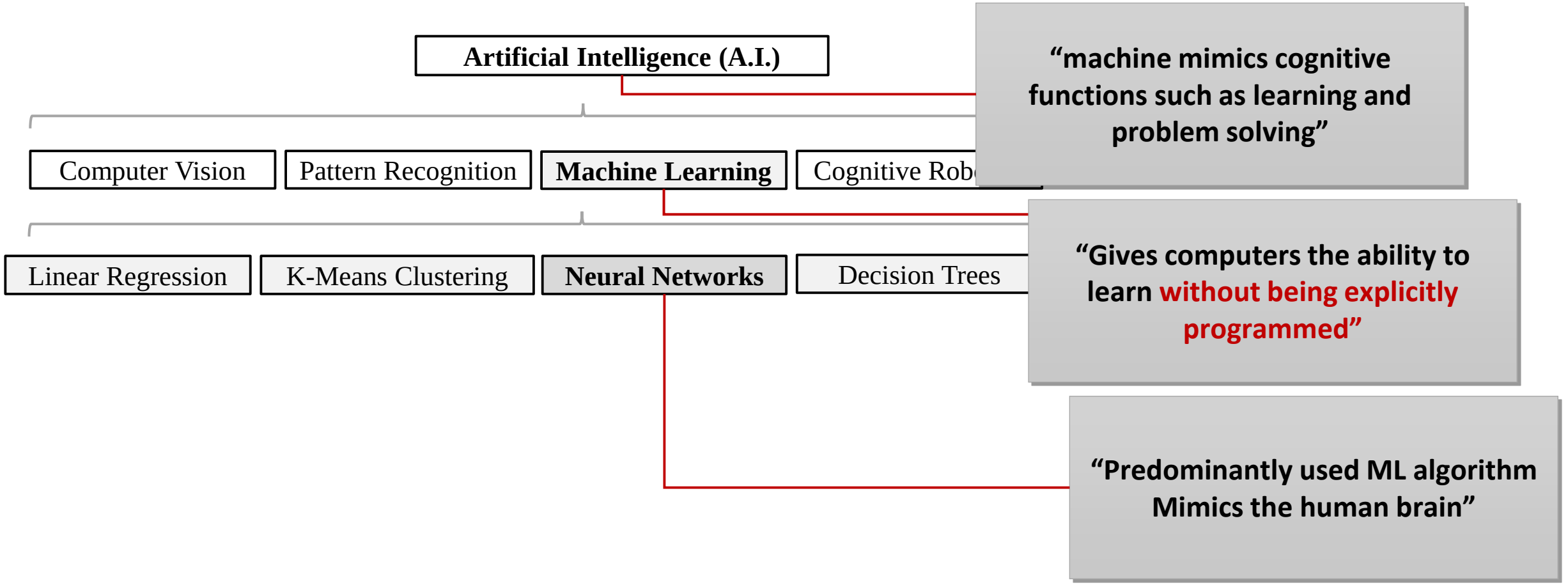
The Rise of The Machine (Learning Algorithms)



- **Potential to solve the unsolved problems**
  - Making solar energy economical, reverse engineering the brain (Jeff Dean, Google Brain 2017)
- **Many difficult ethical questions**
  - Will machines destroy jobs? AI apocalypse?
- **History has shown: We are going through cycles of inventions followed by society adjustments**
  - All of this has happened before and will happen again (Battlestar Galactica, 2014)
- **Let's look at what the technology can do, and how we computer architects can enable it further**

Neural Networks

A.I. – Machine Learning - Neural Networks



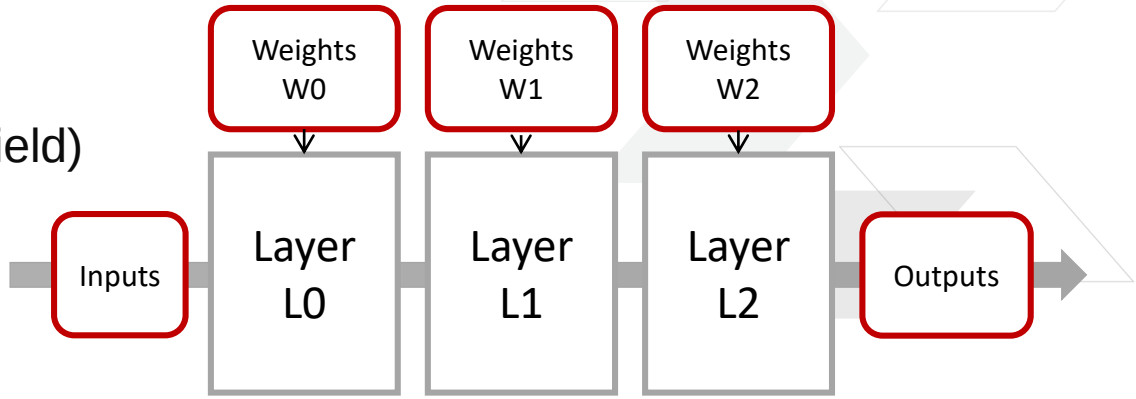
Convolutional Neural Networks (CNNs)
from a computational point of view

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

- **Each layer consists of neurons *ni* which are interconnected with synapses, associated with weights *wij***

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

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



* With exception of RNNs

Evolution: From Shallow to Deep Learning

“Shallow Learning”

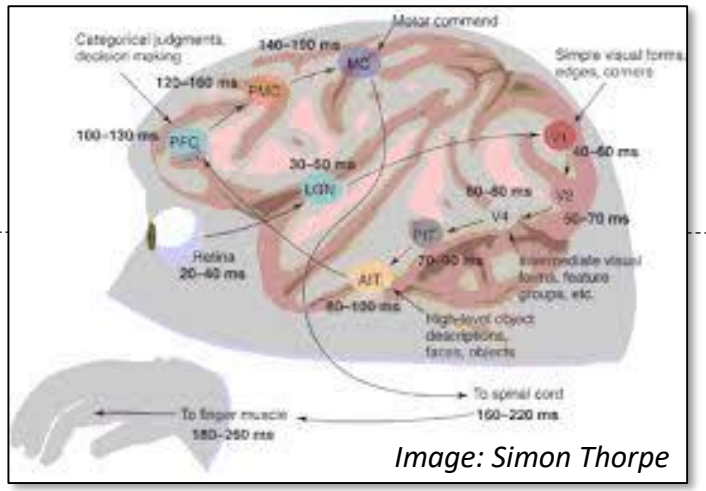


Image: Simon Thorpe

Inputs

extract features & encode in a vector

Handcrafted Feature Extraction

Weights Wn-1

Layer Ln-1

Outputs

“CAT”

“Hierarchical learning”

Feed into classifier (Support Vector Machine or similar)

“Deep Learning”

Learn low level to high level features (pixel, edges, textons, parts, objects)

Inputs

Weights W0

Weights W1

Weights W1

Layer L0

Layer L1

Layer L1

Weights Wn-1

Layer Ln-1

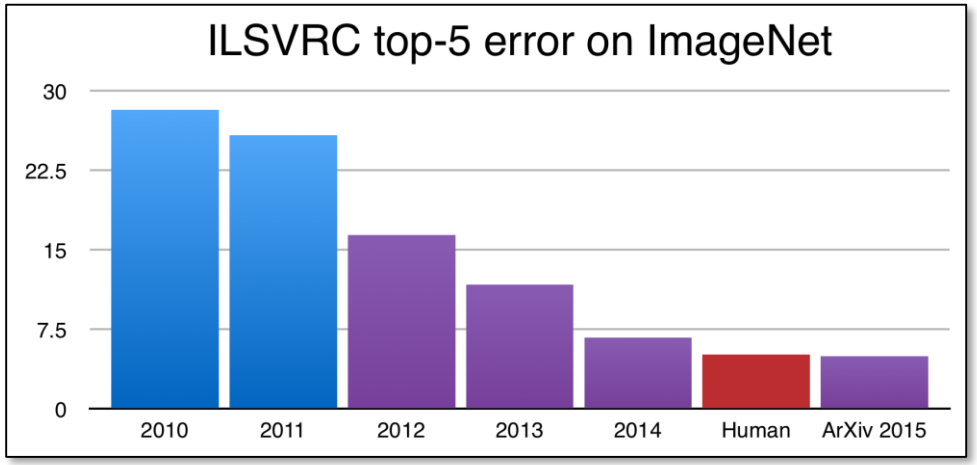
Outputs

Freely adapted from Yann LeCun and Kurt Keutzer

Convolutional Neural Networks (CNNs)

Why are they so popular?

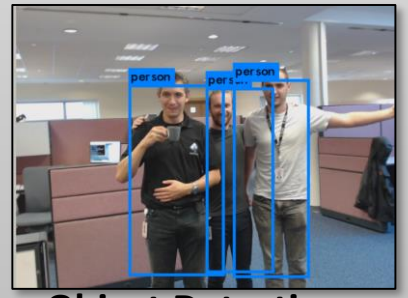
- **Requires little or no domain expertise**
- **NNs are a "universal approximation function"**
- **If you make it big enough and train it enough**
  - Can outperform humans on specific tasks



- **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**

Increasing Range of Applications

Image Classification



Object Detection

Semantic Segmentation

Computer Vision
CNNs

Speaker Diarization

Speech Recognition

Speech Recognition
RNNs, LSTMs

Translation

Sentiment Analysis

Natural Language Processing
Sequence to sequence

Recommender

GamePlay

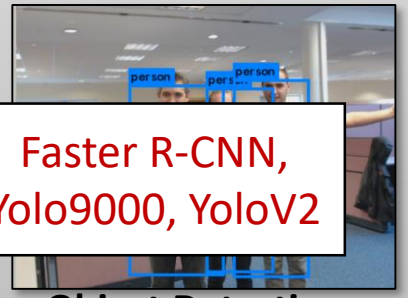
Many more emerging…

Others

Popular Neural Networks

ResNet50, VGG, AlexNet, InceptionV3

Image Classification



Faster R-CNN, Yolo9000, YoloV2

Object Detection

Mask-R-CNN, SSD

Semantic Segmentation

Computer Vision
CNNs

Speaker Diarization

DeepSpeech2

Speech Recognition

eech Recognition
RNNs, LSTMs

Seq2Seq, Transformer

Translation

Seq-CNN

Sentiment Analysis

Natural Language Processing
Sequence to sequence

NCF

Recommender

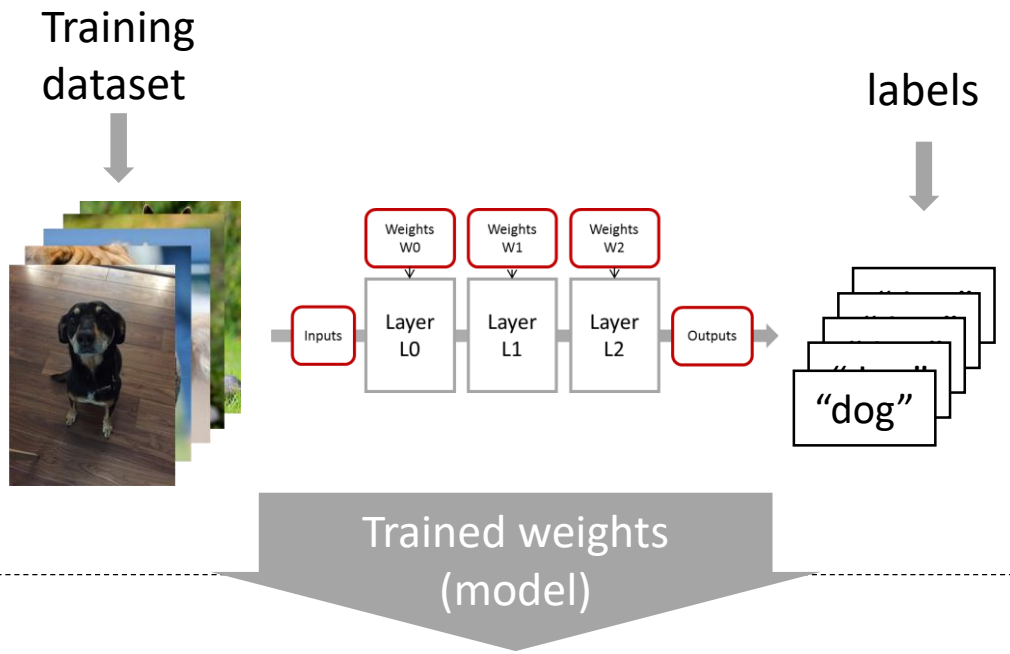
MiniGo, DeepQ, A3C

GamePlay

Others

Adopted from MLPerf, Fathom, TDP

From Training to Inference



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

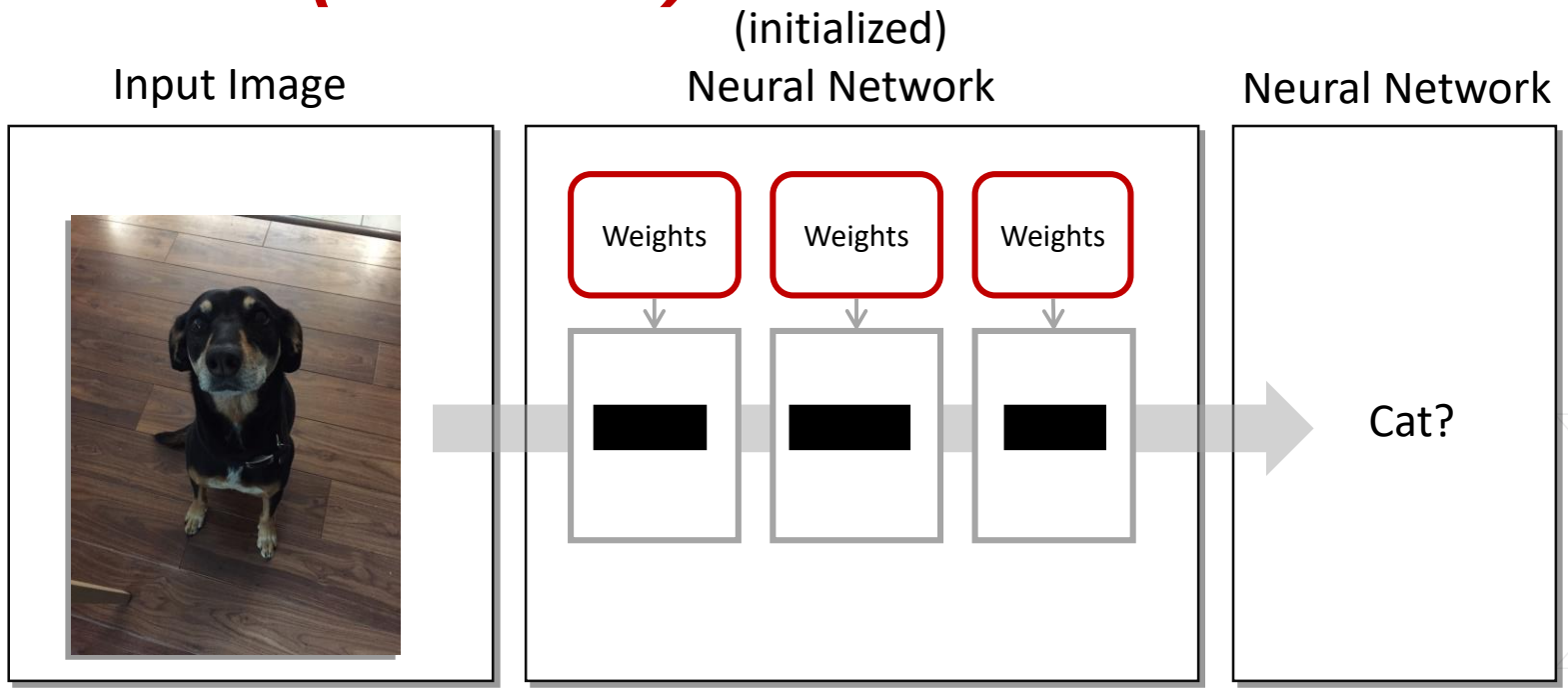
Typically computed in the cloud (part 3)

Inference
Using trained models to predict or estimate outcomes from new inputs.

Deployment at the edge (part 2)

Example: ResNet50

Forward Pass (Inference)



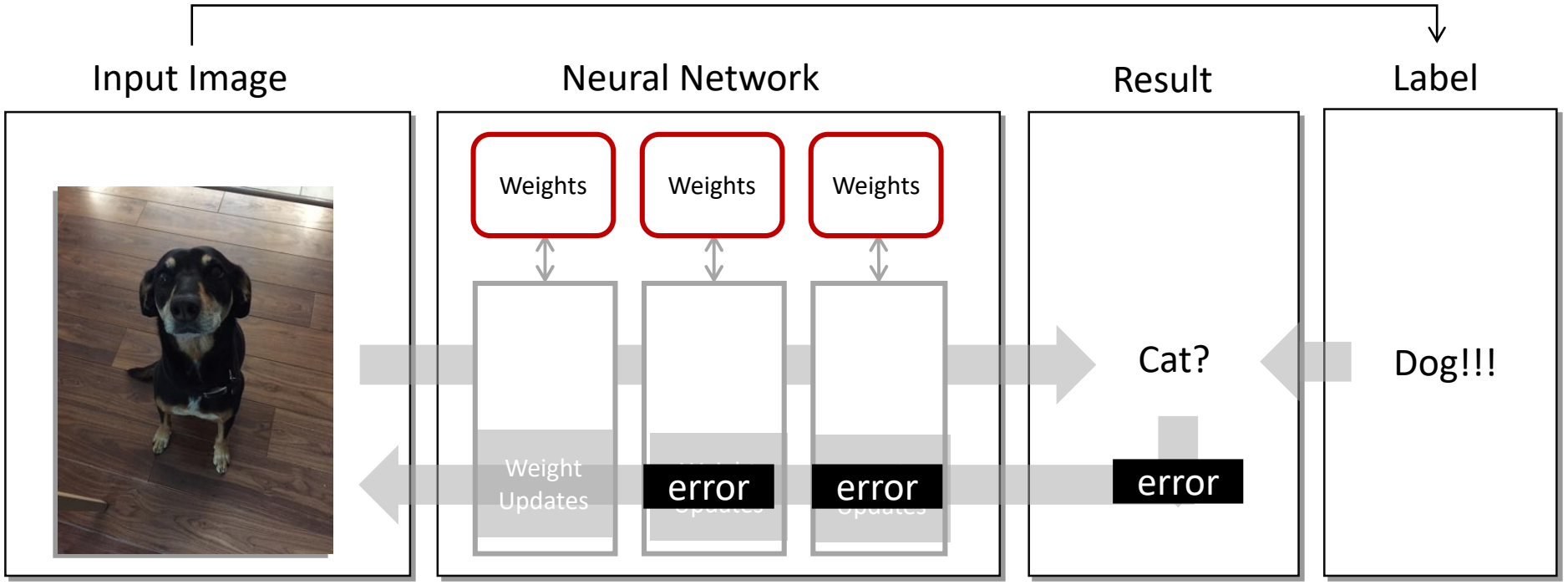
For ResNet50:

- 70 Layers
- 7.7 Billion operations
- 25.5 MBytes of weight storage*
- 10.1 MBytes for activations*

**Assuming int8*

Example: ResNet50

Backpropagation – 1 Image



For ResNet50:

23 Billion operations

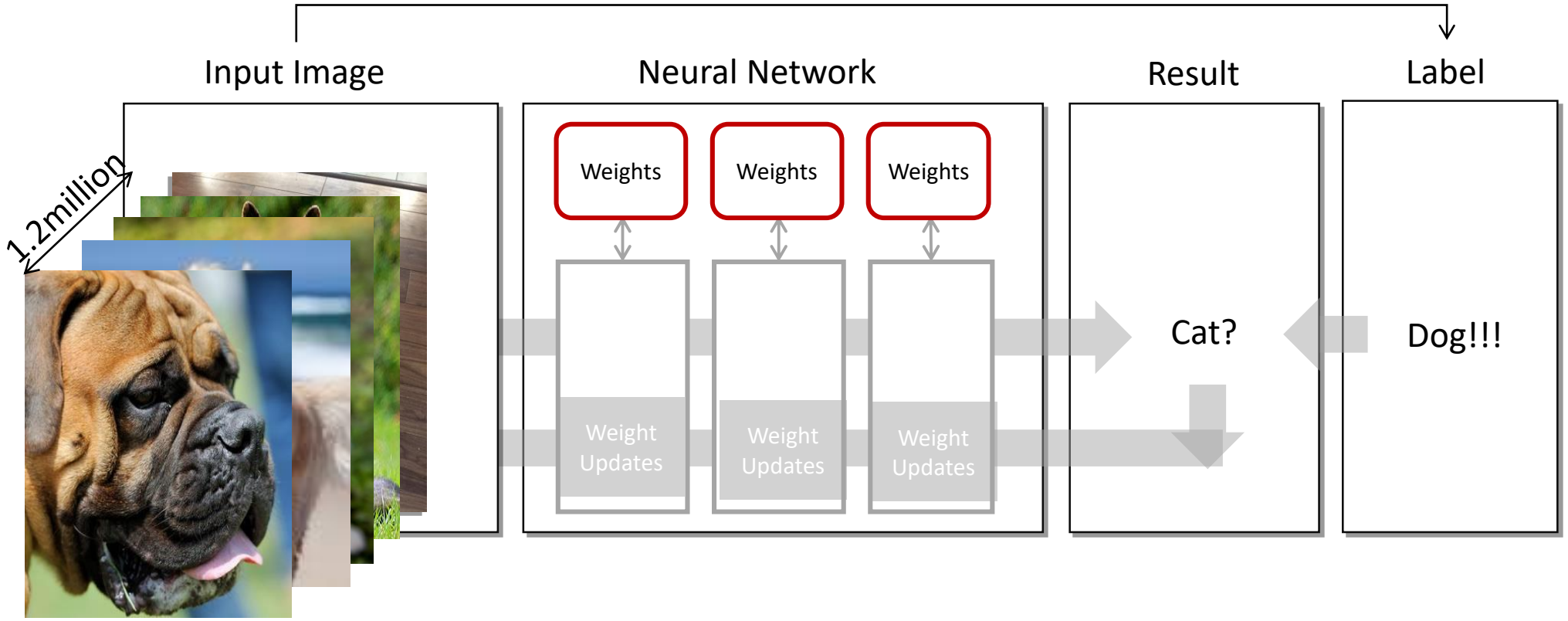
weights, weight gradients, updates: 303MBytes of storage (3-5x)

activations, gradients: 80 MBytes

**Assuming 32b SP*

Example: ResNet50

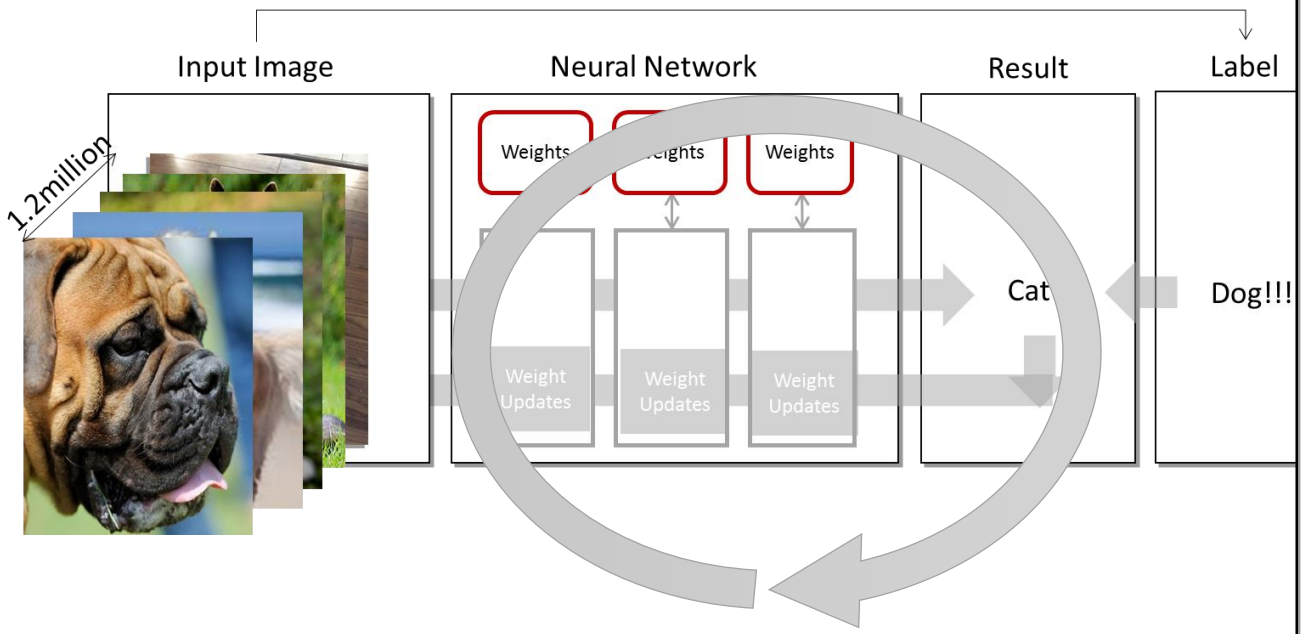
Training – 1.2 Million Images for 1 epoch

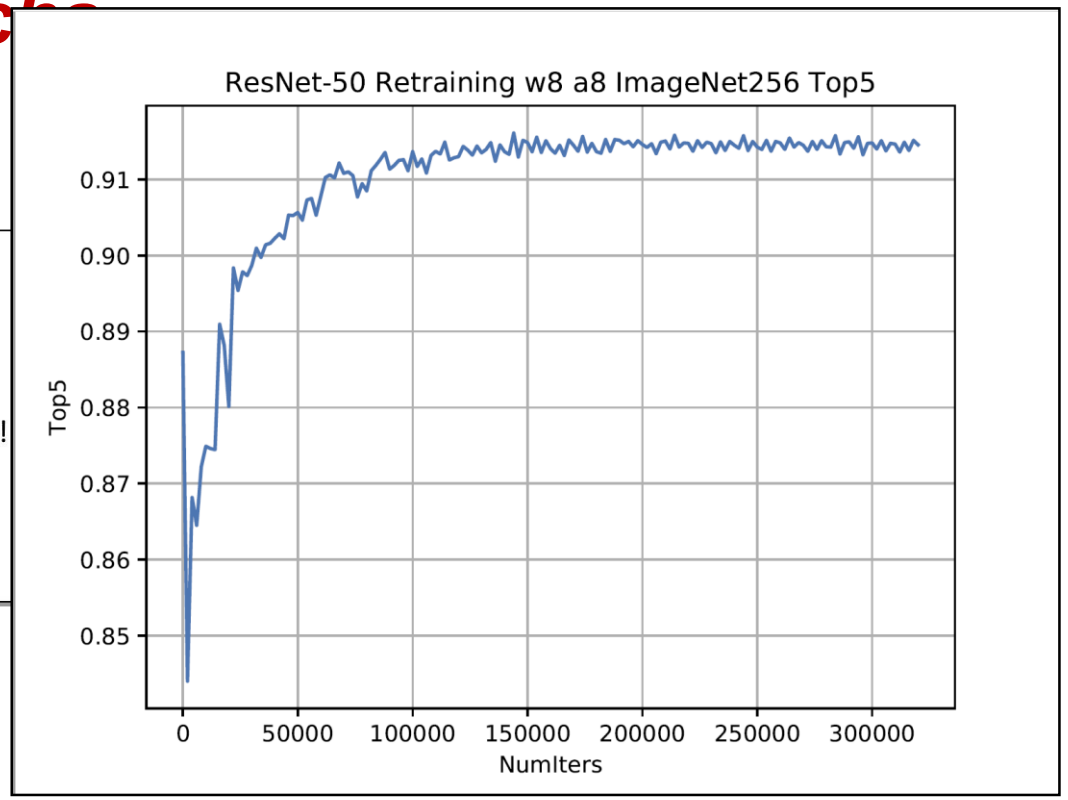


For ResNet50: 1 epoch takes 1.2M * 23 Billion operations = 23 * 10^{15} operations (peta)

Example: ResNet50

Training – Approximately 100 Epoch





For ResNet50: 100 * 23 10^{15} = 2.3 * 10^{18} (exa)

Single P40 GPU (12TFLOPS): 11days @ 100%, usually ~2 weeks

ResNet50:

- **For inference: Billions of operations, and 10s of MegaBytes**
- **For training: Quintillions/Exa of operations, and 100s of MegaBytes**

Inference and Training
Nested Loops

```
For all epochs do:
  For all images in data set do:
    # compute in batches of input images
        # inference
        For i = 1 to max_layers do:
            input(i+1) = act(input(i),weights);

        determine error;
        calculate gradients;

        # backpropagate with SGD
        For i = 1 to max_layers do:
            weight_grad = f(inputs(i), actgrad(i+1);
            weight_update += g(weight_grad);
            act_grad(i) = f(weights(i), actgrad(i+1);

    # at end of batch
    weights := f(weights,
```

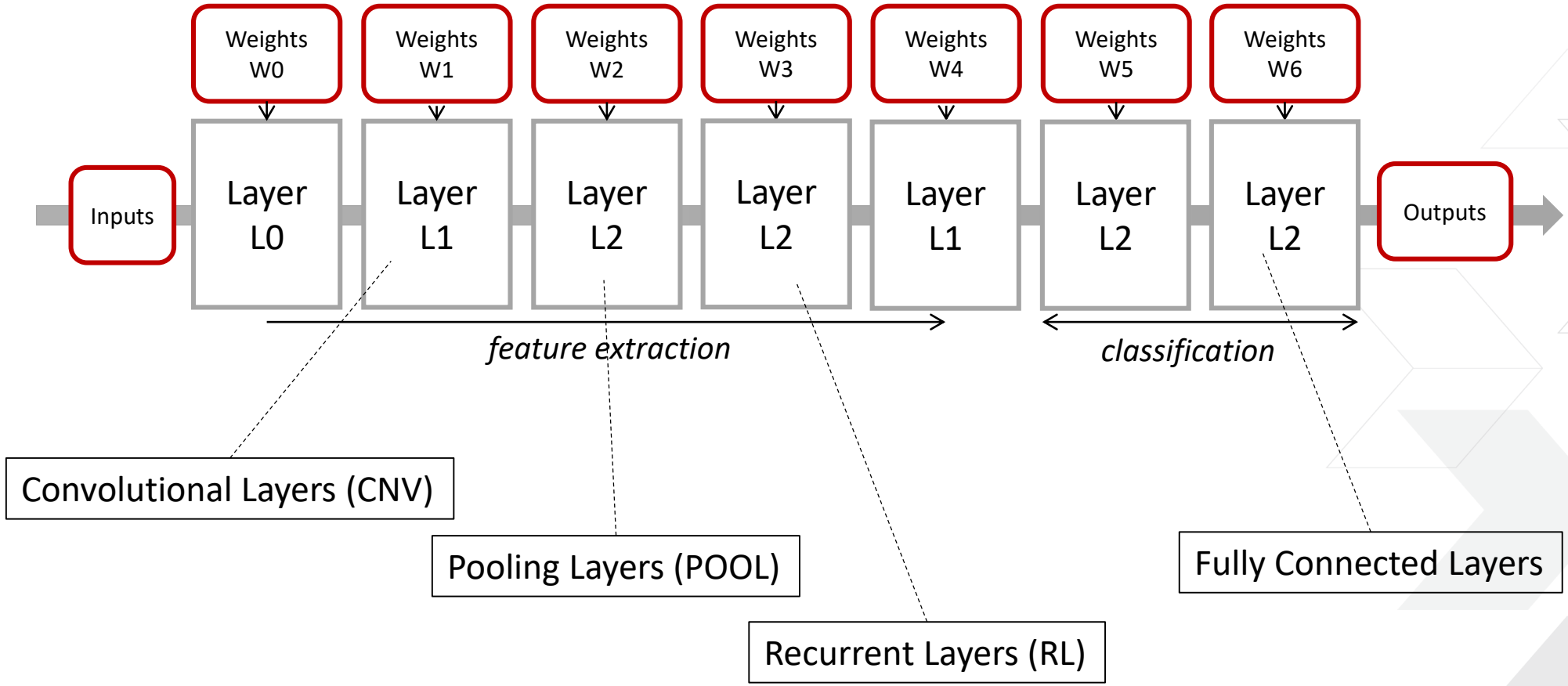
Group of inputs buffered to increase compute efficiency

Massively nested for loops in themselves

Dictates intervals between weight updates to reduce communication overhead, at potentially adverse effect on accuracy

How do we loop transform and unfold this best to maximize data reuse and compute efficiency?

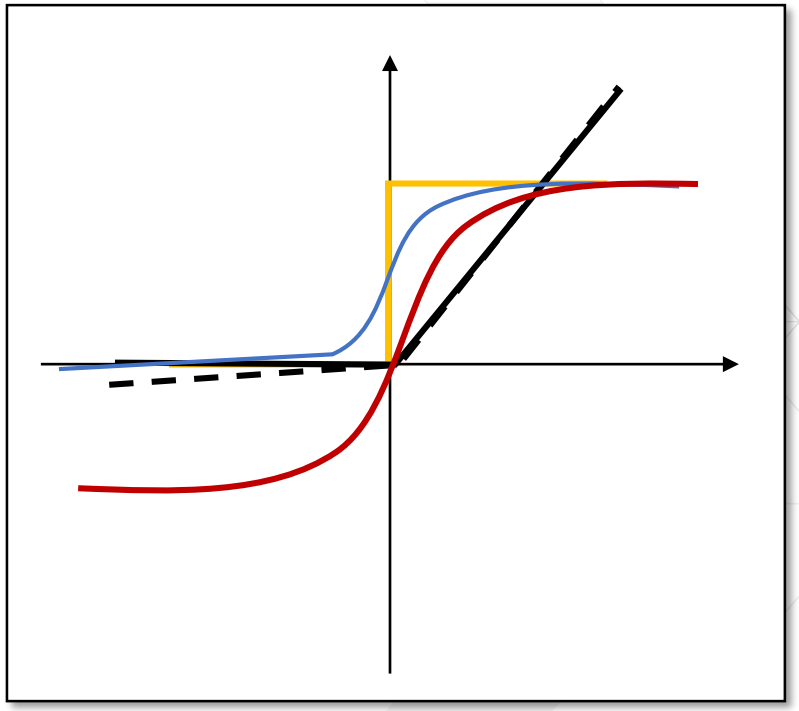
NNs in More Detail



Activation & Batch Normalization

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

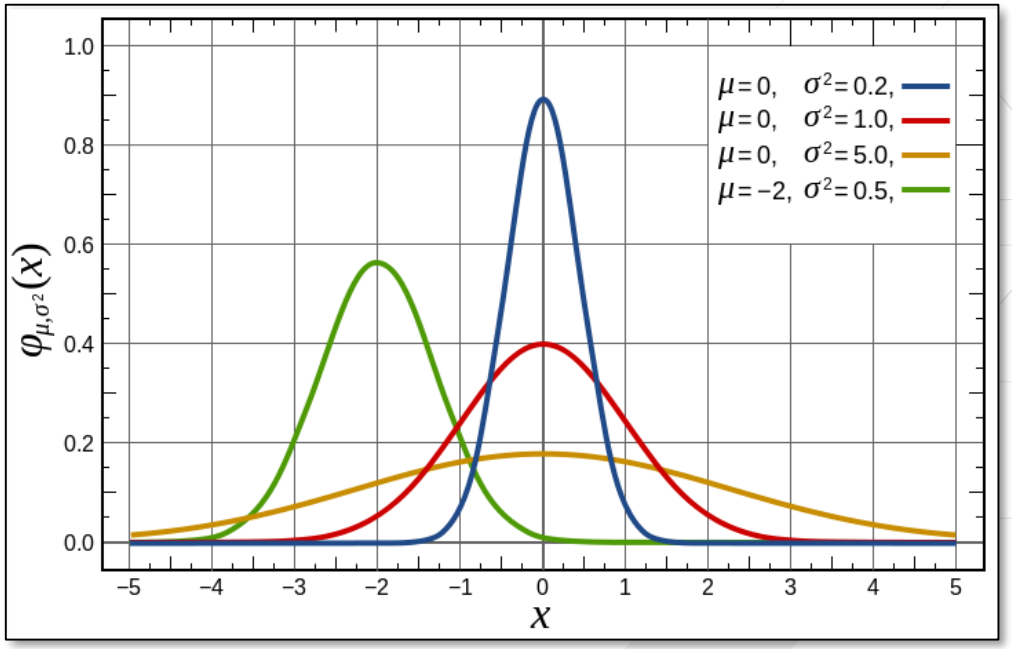


**Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S.S. and Pennington "Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks." arXiv preprint arXiv:1806.05393 (2018).*

***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).*

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

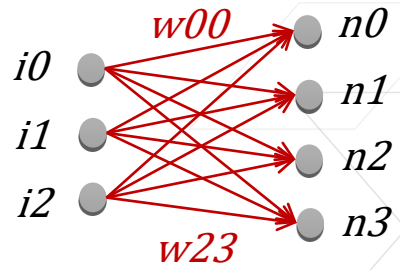


https://en.wikipedia.org/wiki/Normal_distribution

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

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



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

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

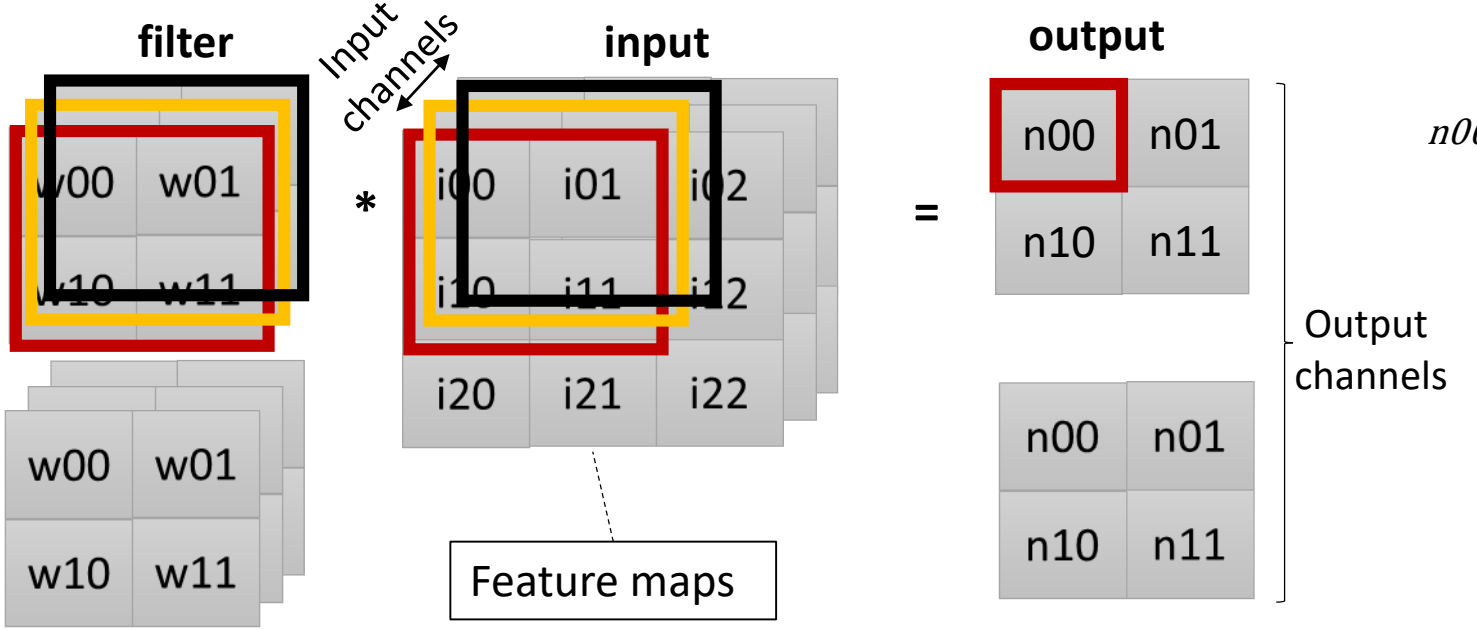
MODEL	CONV WEIGHTS (M)	FC WEIGHTS (M)
ResNet50	23.454912	2.048
AlexNet	2.332704	58.621952
VGG16	14.710464	123.633664

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

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**



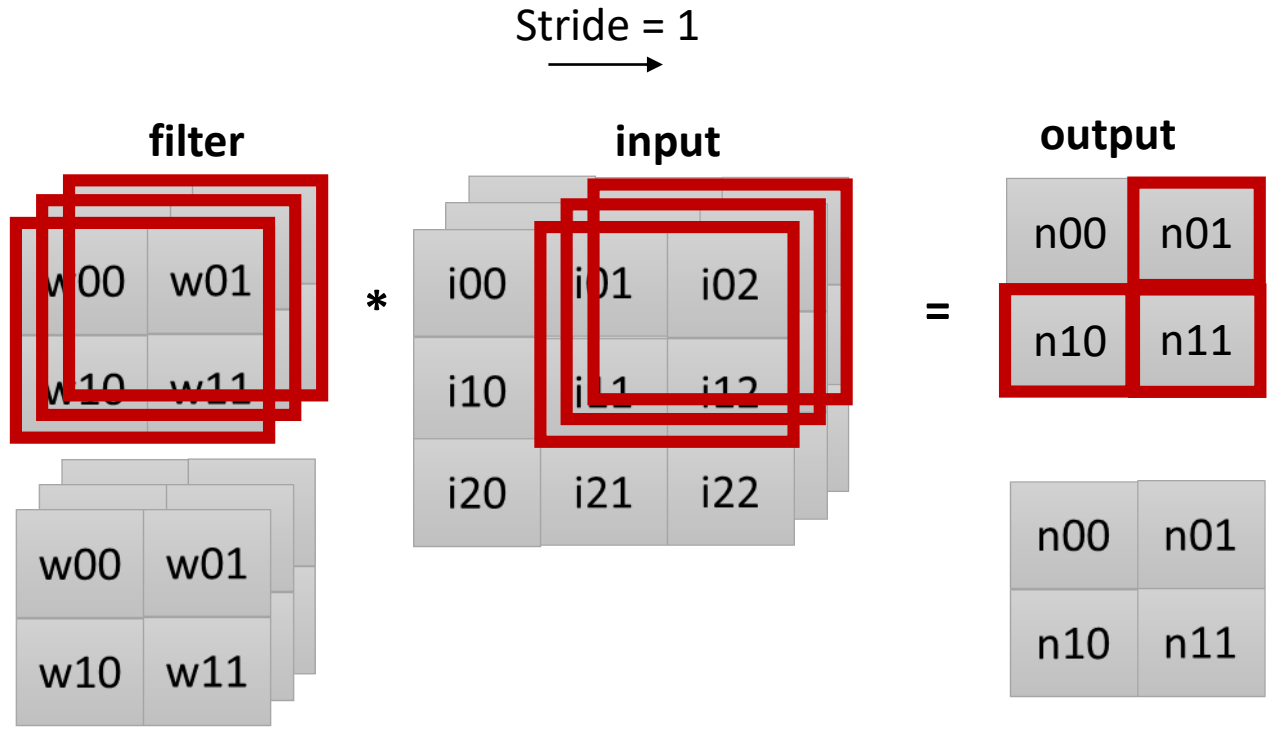
Input channel 0

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

XILINX

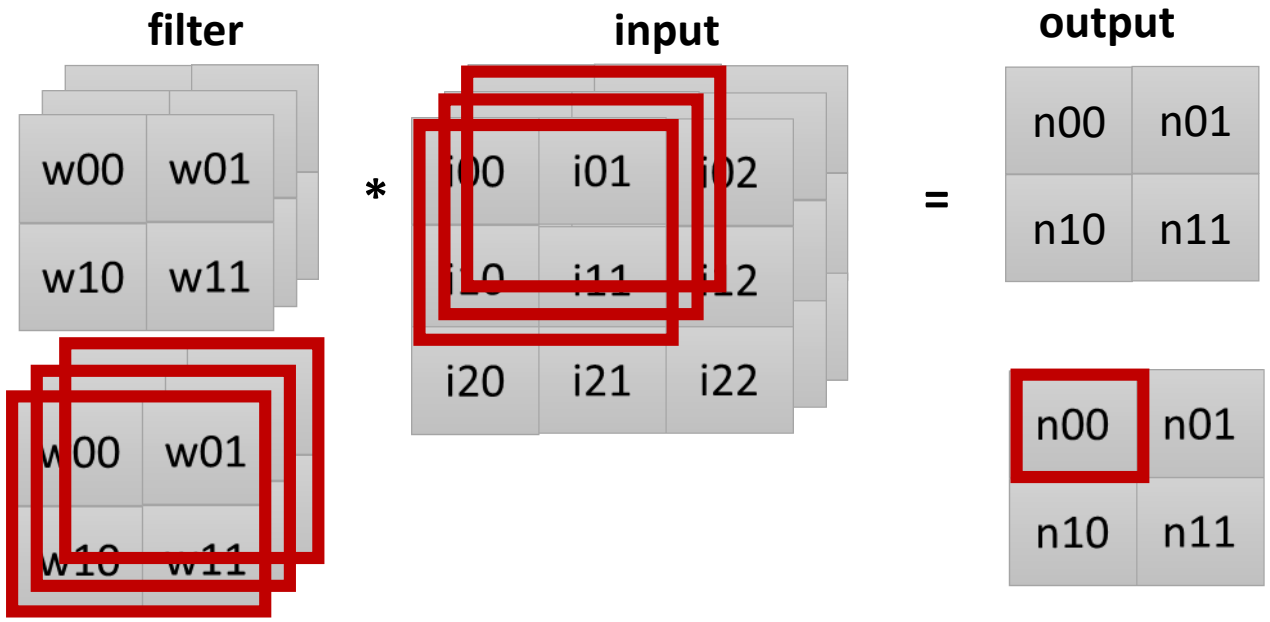
2D Convolutional Layers

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



2D Convolutional Layers

> **Compute next channel**



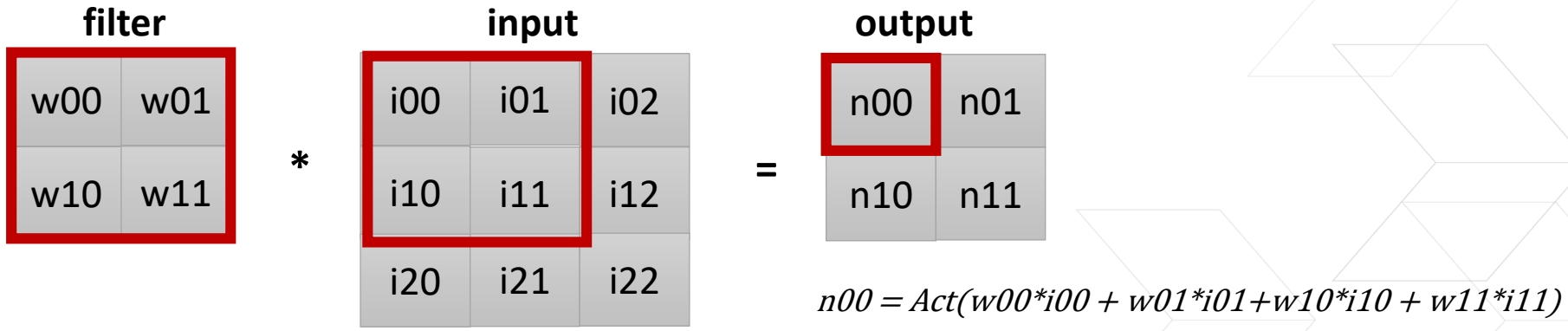
XILINX.

2D Convolutional Layers

1 input and 1 output channel

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

Convolution

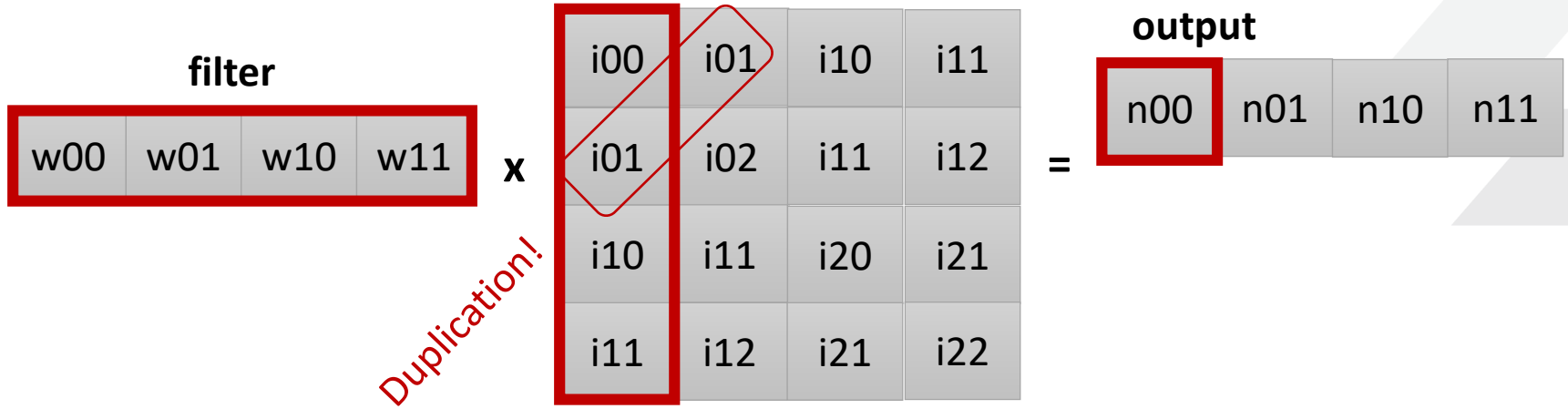


$$n00 = Act(w00*i00 + w01*i01 + w10*i10 + w11*i11)$$

<=>

Matrix Vector

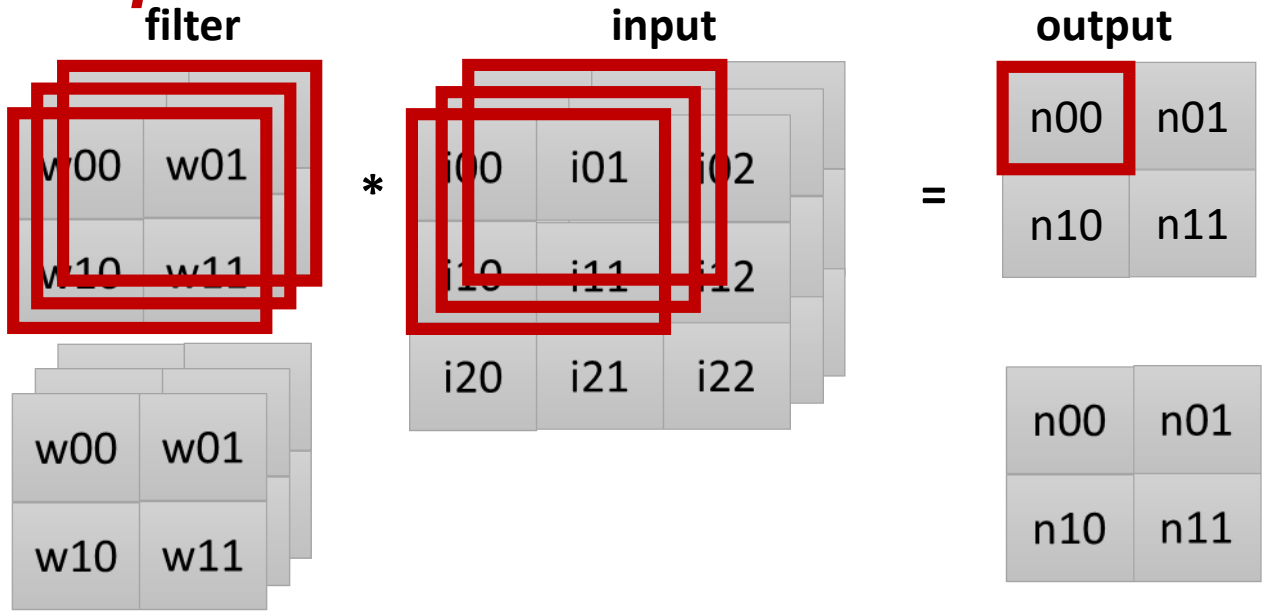
Toeplitz Matrix ("lowered image matrix")



2D Convolutional Layers

3 input and 2 output channels

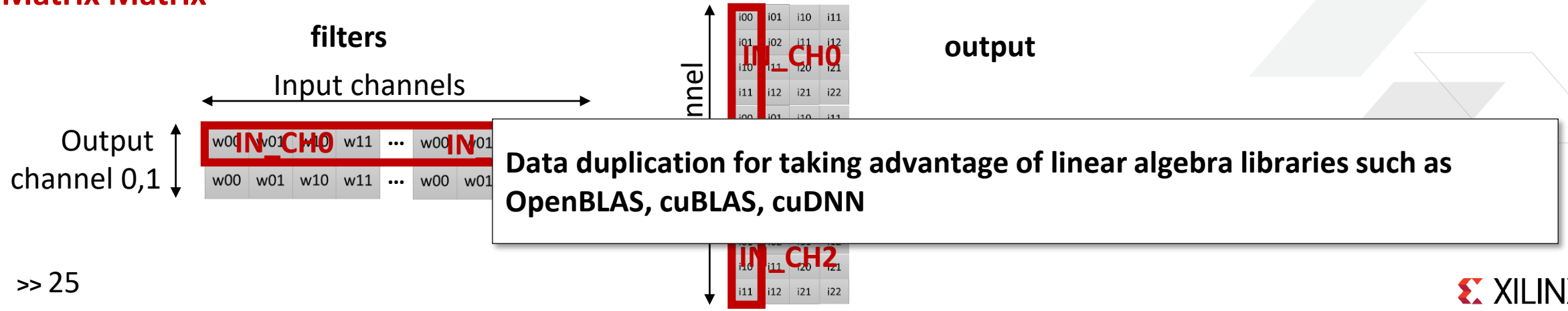
Convolution



<=>

Matrix Matrix

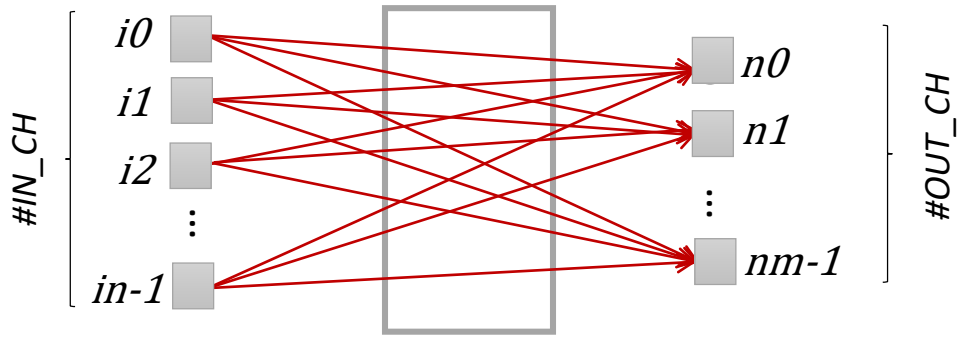
Toeplitz Matrix



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

Convolutions
Challenges

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



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

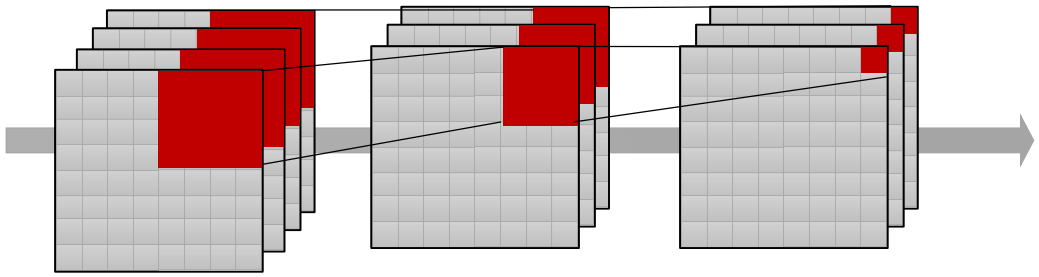
- **Novel (Non-Dense) Convolutions**
  - Less spatial convolutions (1x1) (SqueezeNet's FireModules)
  - Connectivity reduction between in and out channels (Shuffle, Shift layers)

=> Optimizations

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**

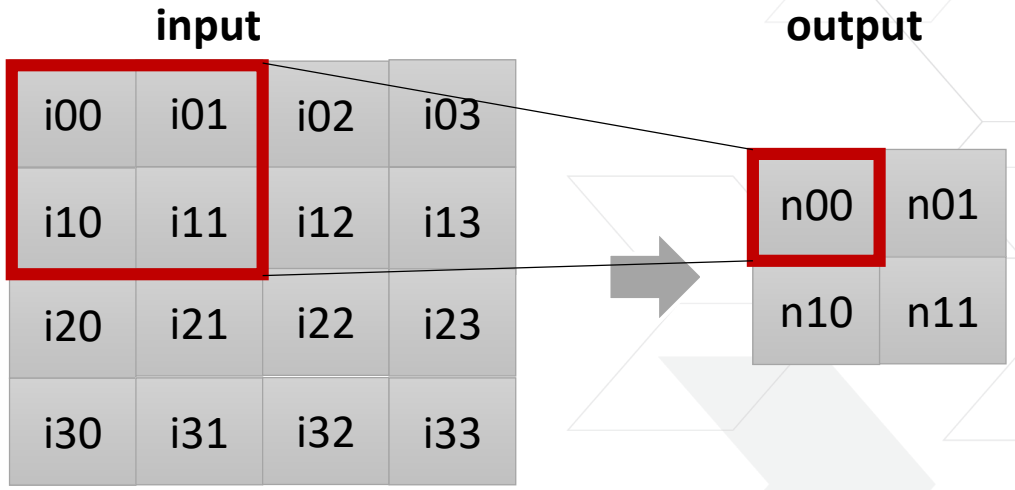


Alwani, M., Chen, H., Ferdman, M. and Milder, P., 2016, October. Fused-layer CNN accelerators. Micro 2016

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:

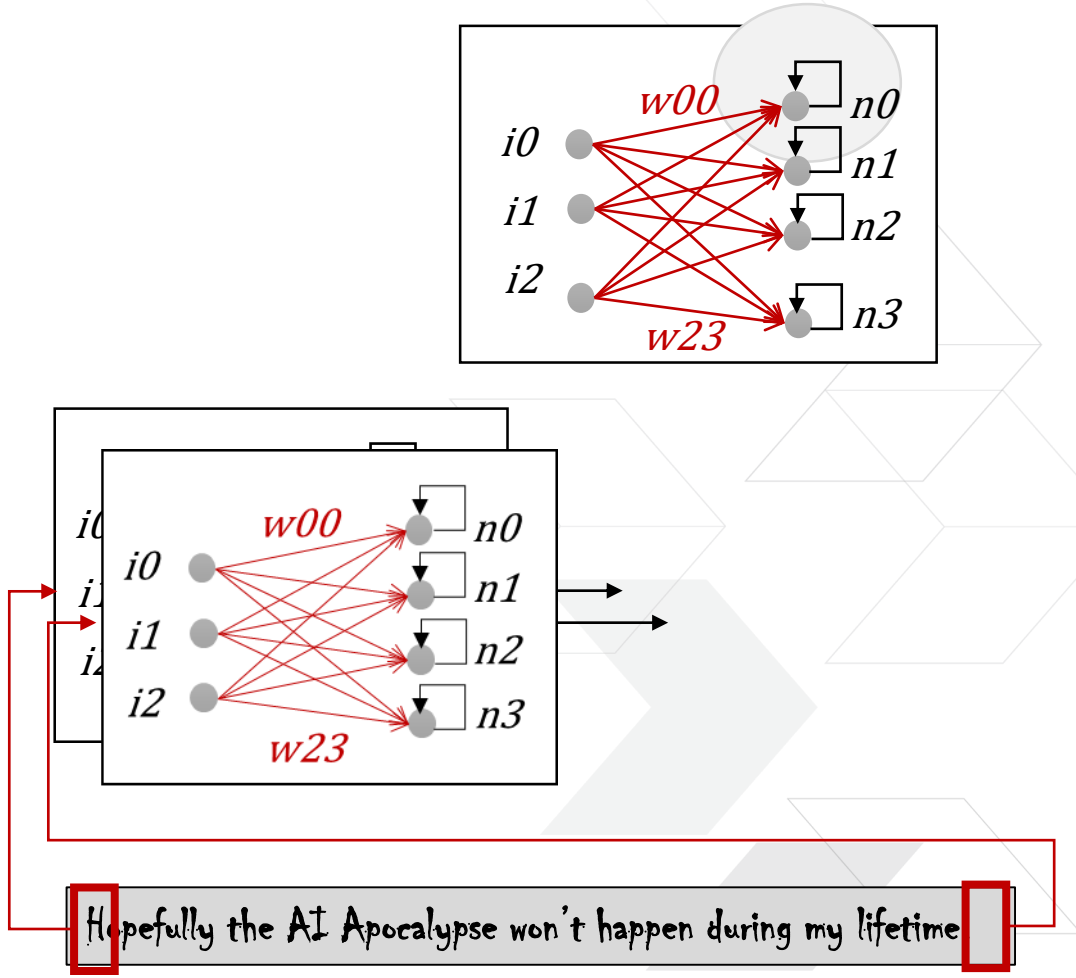


n00 = Max(i00, i01, i10, i11)

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

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)

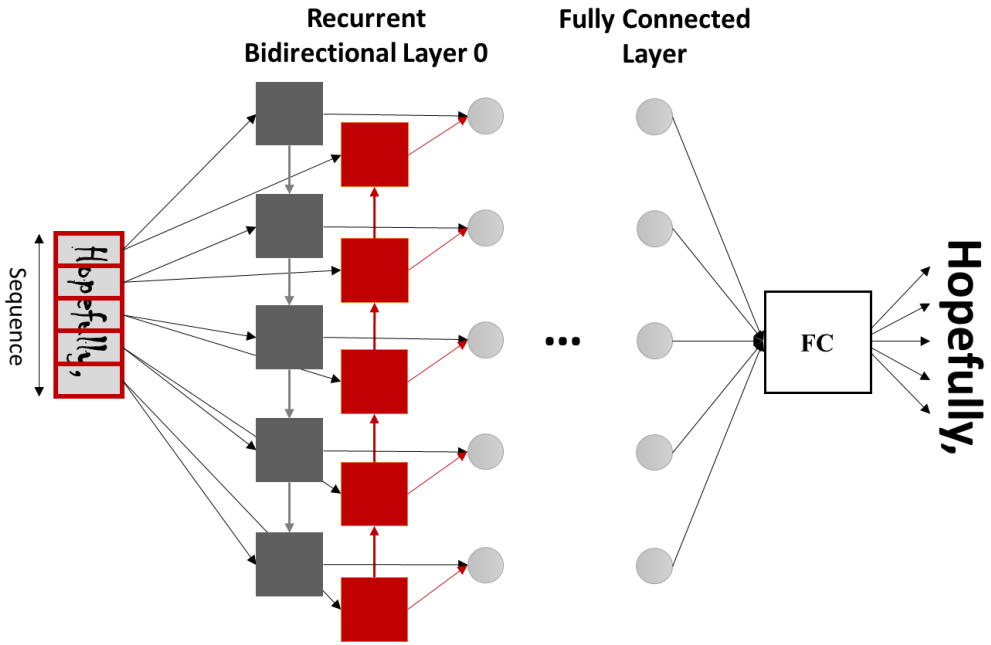


Recurrent Layers

Challenges in Additional Data Dependencies

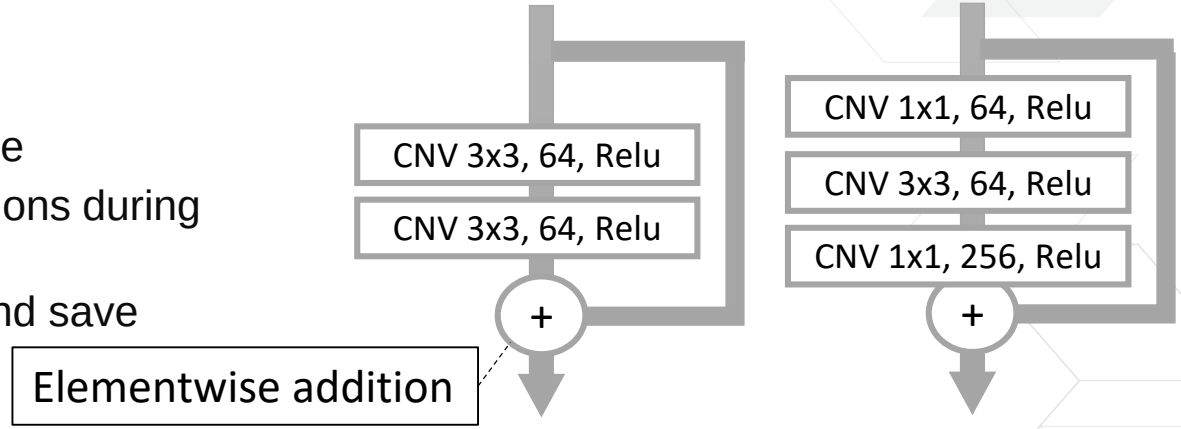
- **Input sequence**
  - Unlike batch, additional data dependencies between inputs of the same sequence and state

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

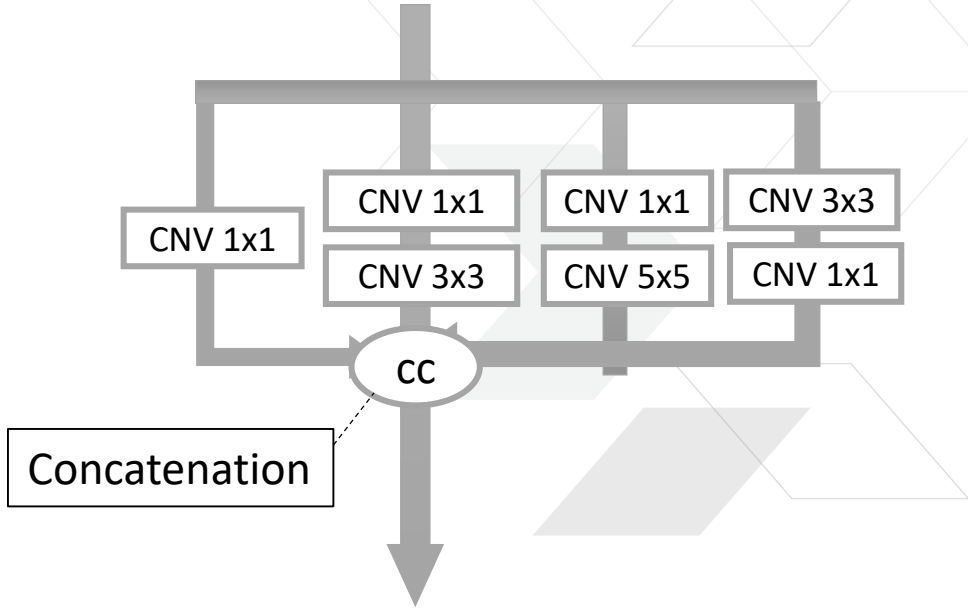


Meta-Layers

- **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



- **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



- **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.*

Computation & Memory Requirements

Compute and Memory Requirements
Architecture Neutral, Per Layer

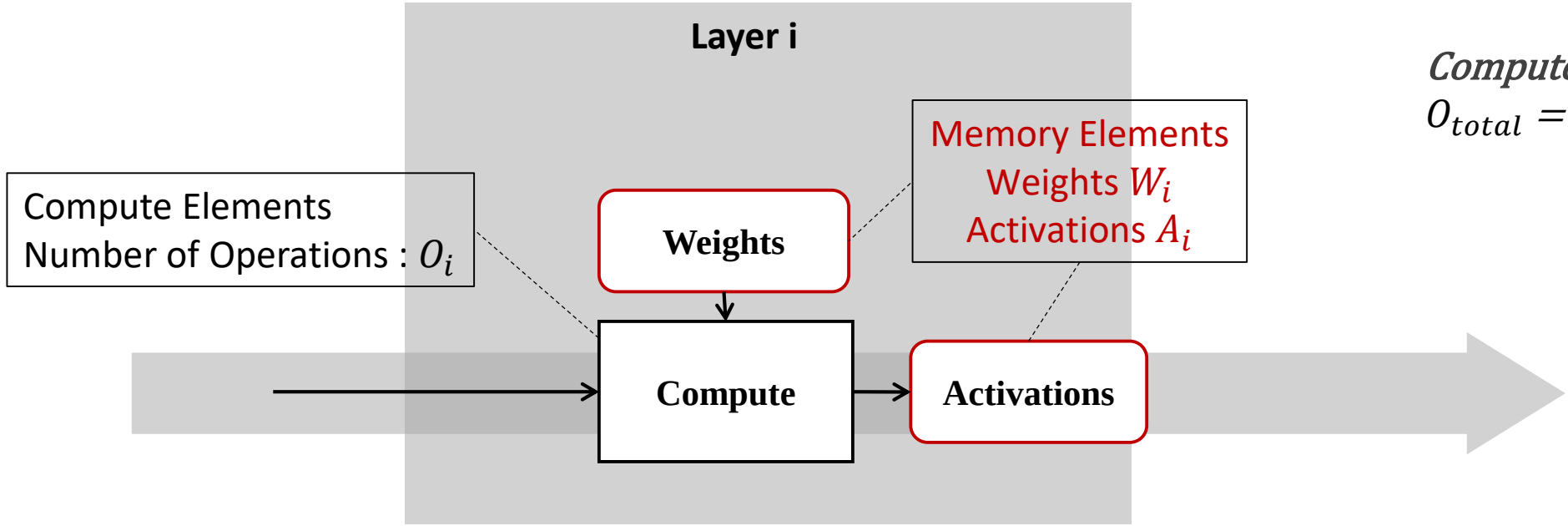
Memory Requirements:

$A_{total} = \sum A_i$

$W_{total} = \sum W_i$

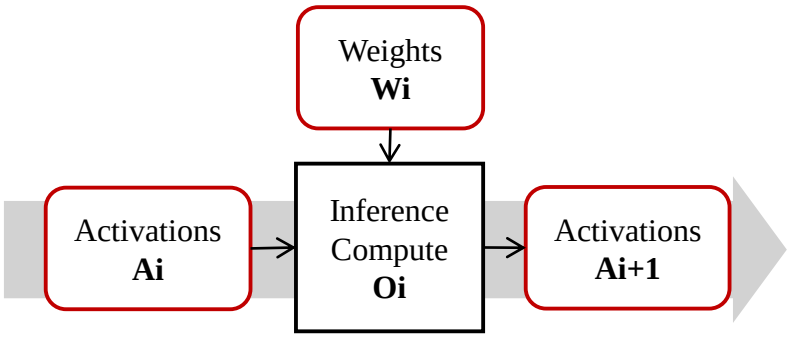
Compute Requirements:

$O_{total} = \sum O_i$



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*

Inference:
Compute & Memory Analysis



Fully Connected Layers:

$W_i \approx IN_i * OUT_i * BITS$

$A_i \approx BATCH * IN_i * BITS$

$O_i \approx 2 * BATCH * IN_i * OUT_i$

Convolutional Layers:

$W_i \approx IN\_CH_i * OUT\_CH_i * F\_DIM_i^2 * BITS$

$A_i \approx BATCH * FM\_DIM_i^2 * IN\_CH_i * BITS$

$O_i \approx 2 * BATCH * FM\_DIM_i^2 * IN\_CH_i * OUT\_CH_i * F\_DIM_i^2$

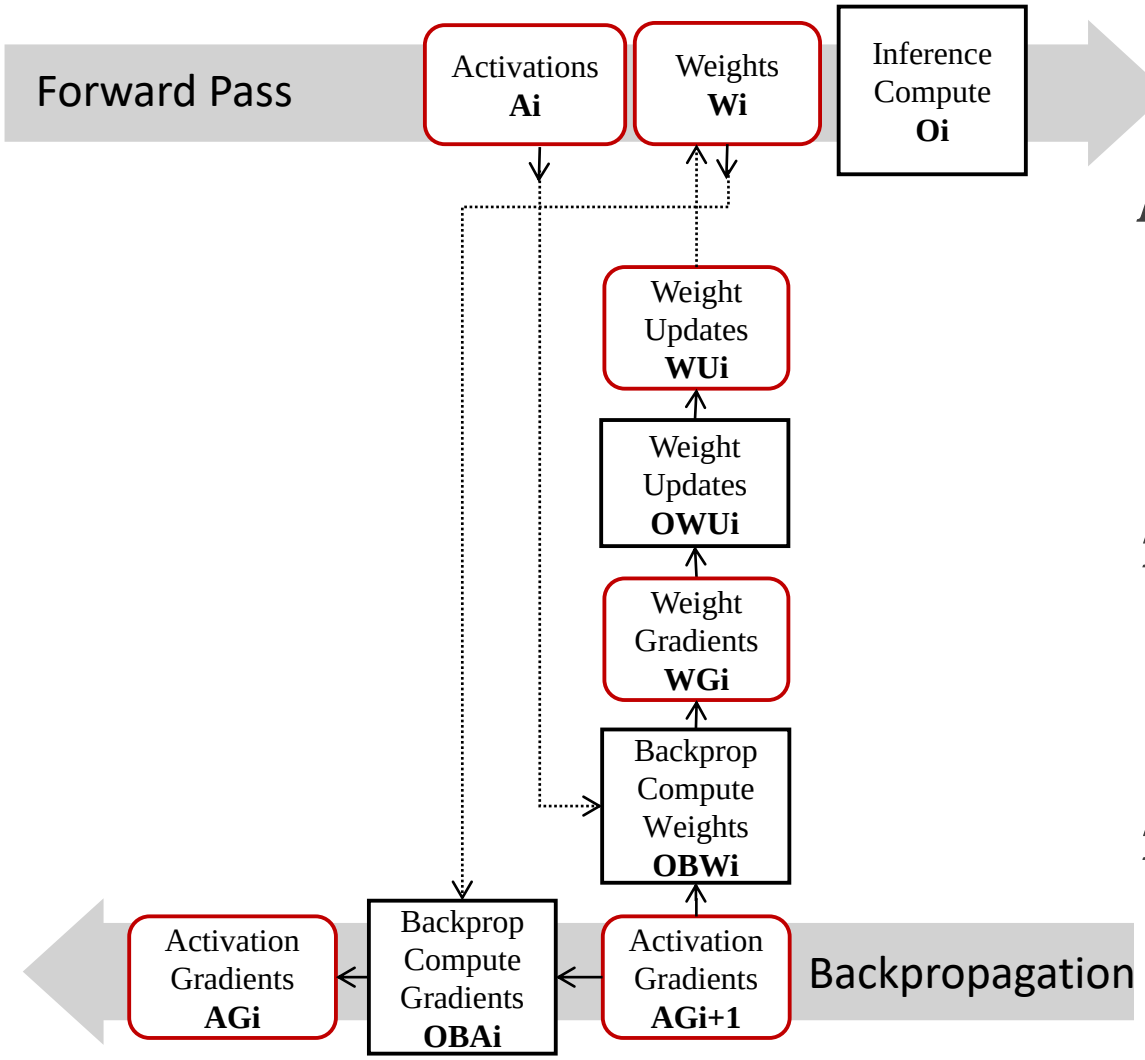
Recurrent Layers:

$W_i \approx (FM\_DIM_i + HID_i) * HID_i * GATES * DIRS * BITS$

$A_i \approx BATCH * FM\_DIM_i * DIRS * STATES * SEQ * BITS$

$O_i \approx 2 * (FM\_DIM_i + HID_i) * HID_i * GATES * DIRS * SEQ$

Backpropagation & Training:
Compute & Memory Analysis



Backpropagation for 1 image or 1 sequence and 1 layer

$WG_i \approx WU_i \approx W_i$ — *2x over inference*

$AG_i \approx A_i$ — *= inference*

$OBW_i \approx OBA_i \approx O_i$ — **2x over inference**

($OBWU_i$ negligible)

Training 1 batch – all layers

$A_{Batch} \approx 2 * A$

$W_{Batch} \approx 3 * BATCH * W$

$O_{Batch} \approx 3 * O_{inference}$

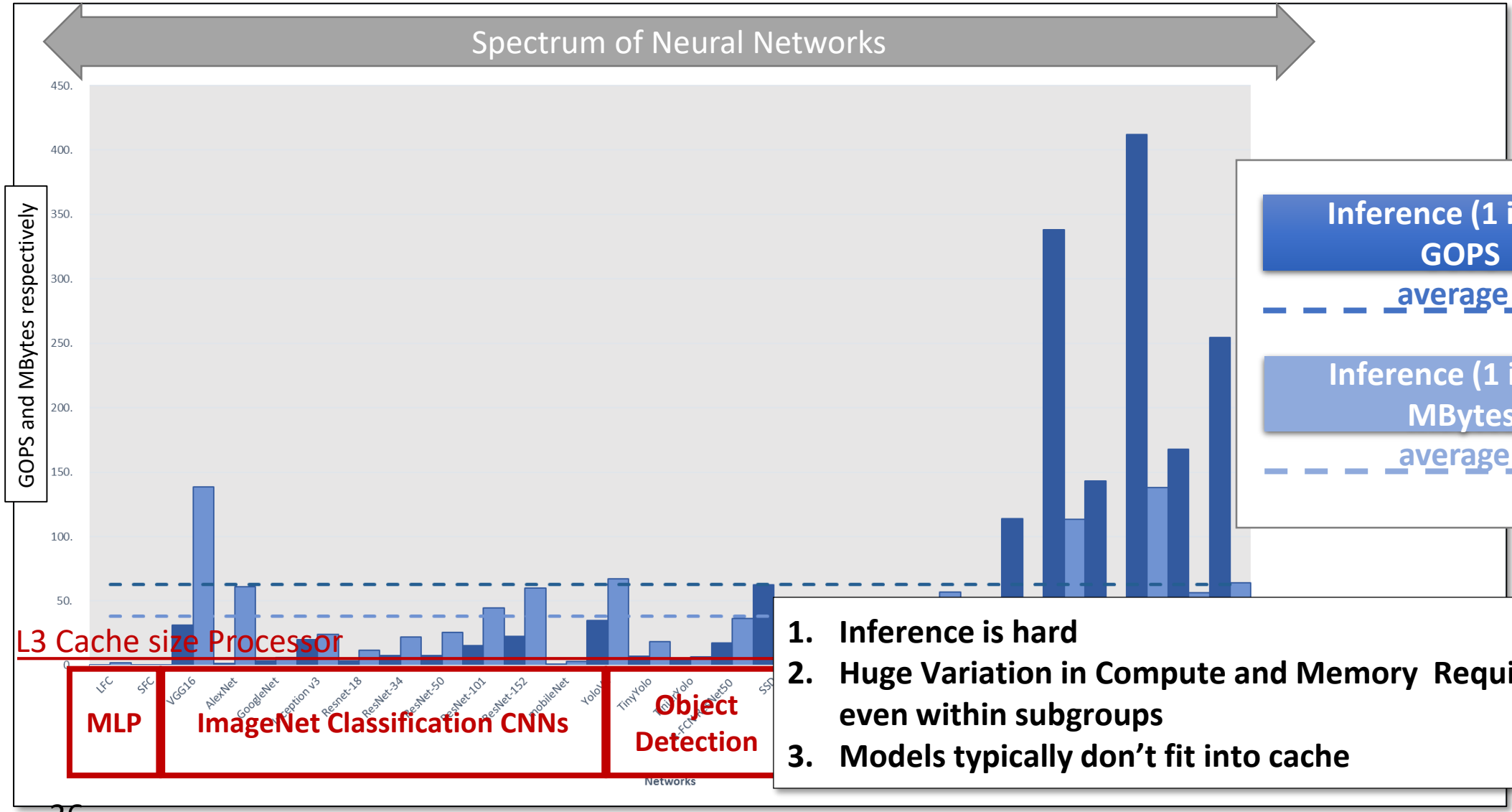
Training overall

$O_{Overall} \approx 3 * \#dataset * \#epochs * O_{inference}$

BATCH: *size of batch*
#epochs: *number of epochs*
#datasets: *number of inputs in dataset*

Inference Compute and Memory
Across a Spectrum of Neural Networks

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

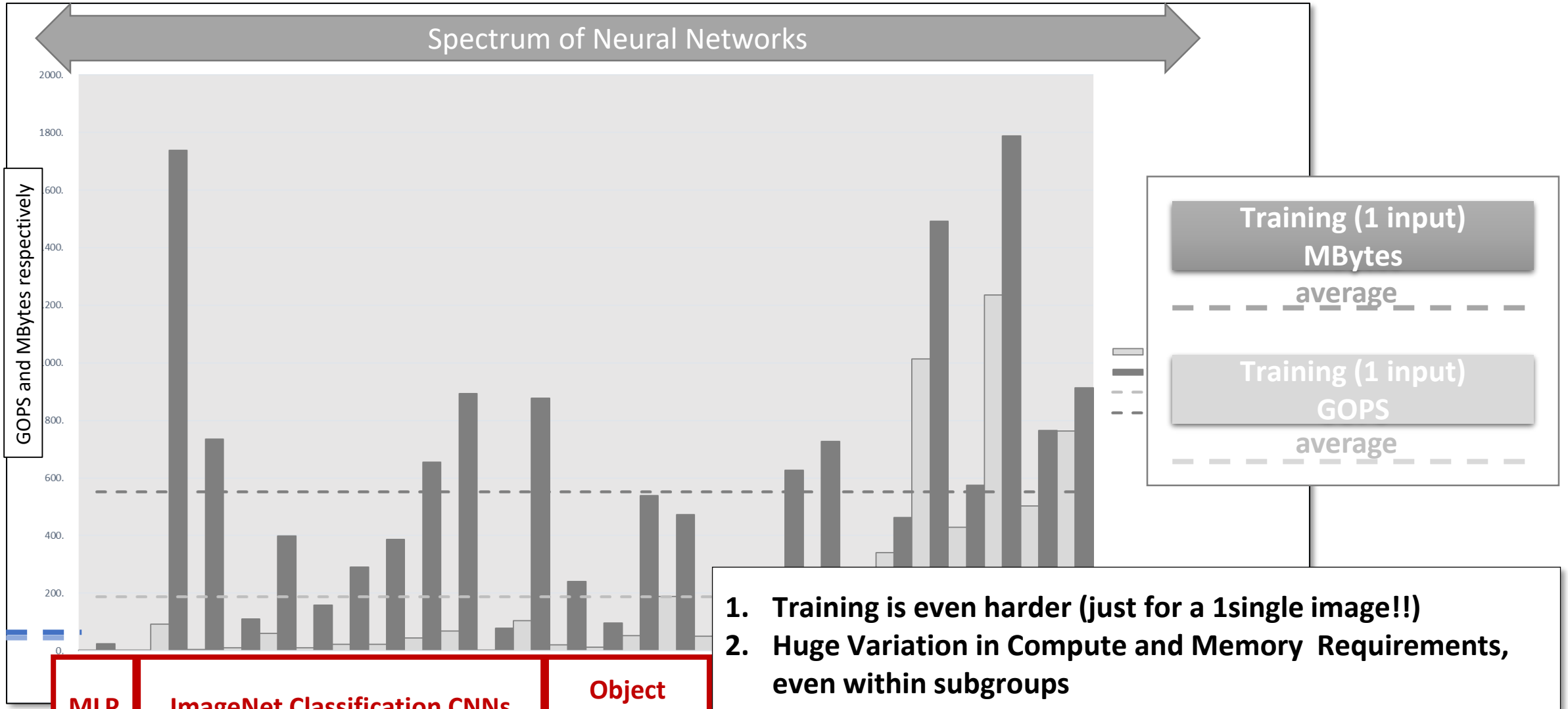


1. **Inference is hard**
2. **Huge Variation in Compute and Memory Requirements, even within subgroups**
3. **Models typically don't fit into cache**

XILINX

*architecture independent
**1 image forward and backprop
*** batch = 1

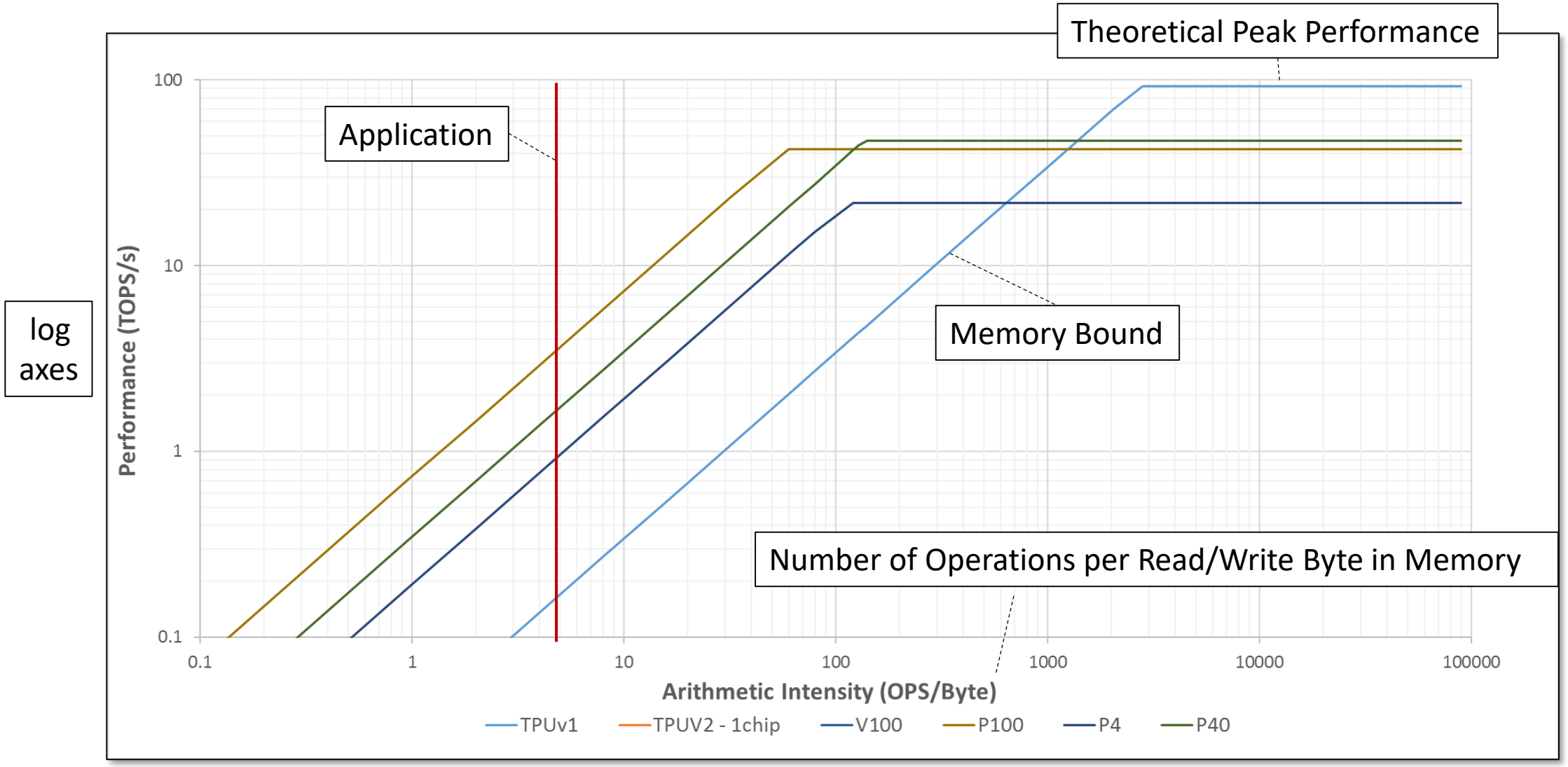
Training Compute and Memory
Across a Spectrum of Neural Networks



1. Training is even harder (just for a 1single image!!)
2. Huge Variation in Compute and Memory Requirements, even within subgroups

XILINX

Rooflines*

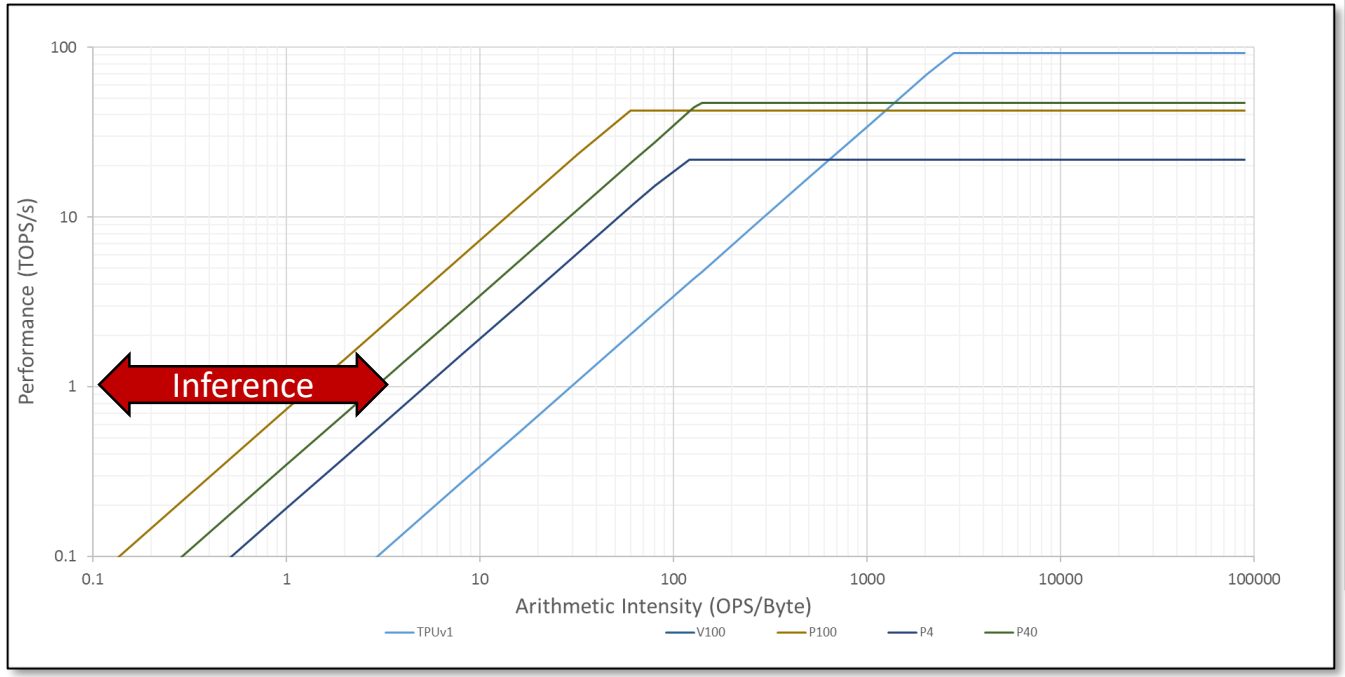


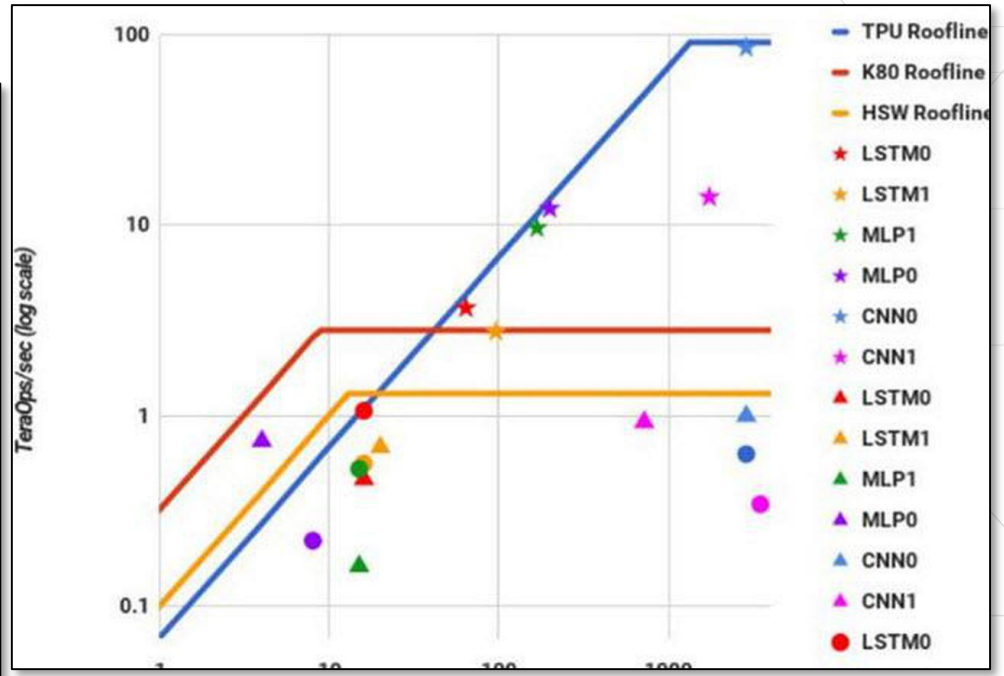
*Williams, S., Waterman, A. and Patterson, D., 2009.
Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM

Arithmetic Intensity
Across a Spectrum of Neural Networks

* batch = 1
** with respect to weights assuming weights are off-chip

- **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

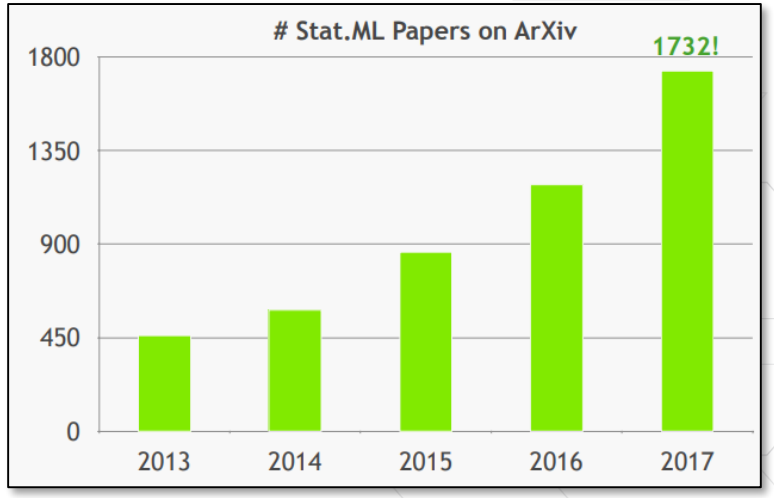




Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A. and Boyle, R., 2017, June. In-datacenter performance analysis of a tensor processing unit. ISCA'2017

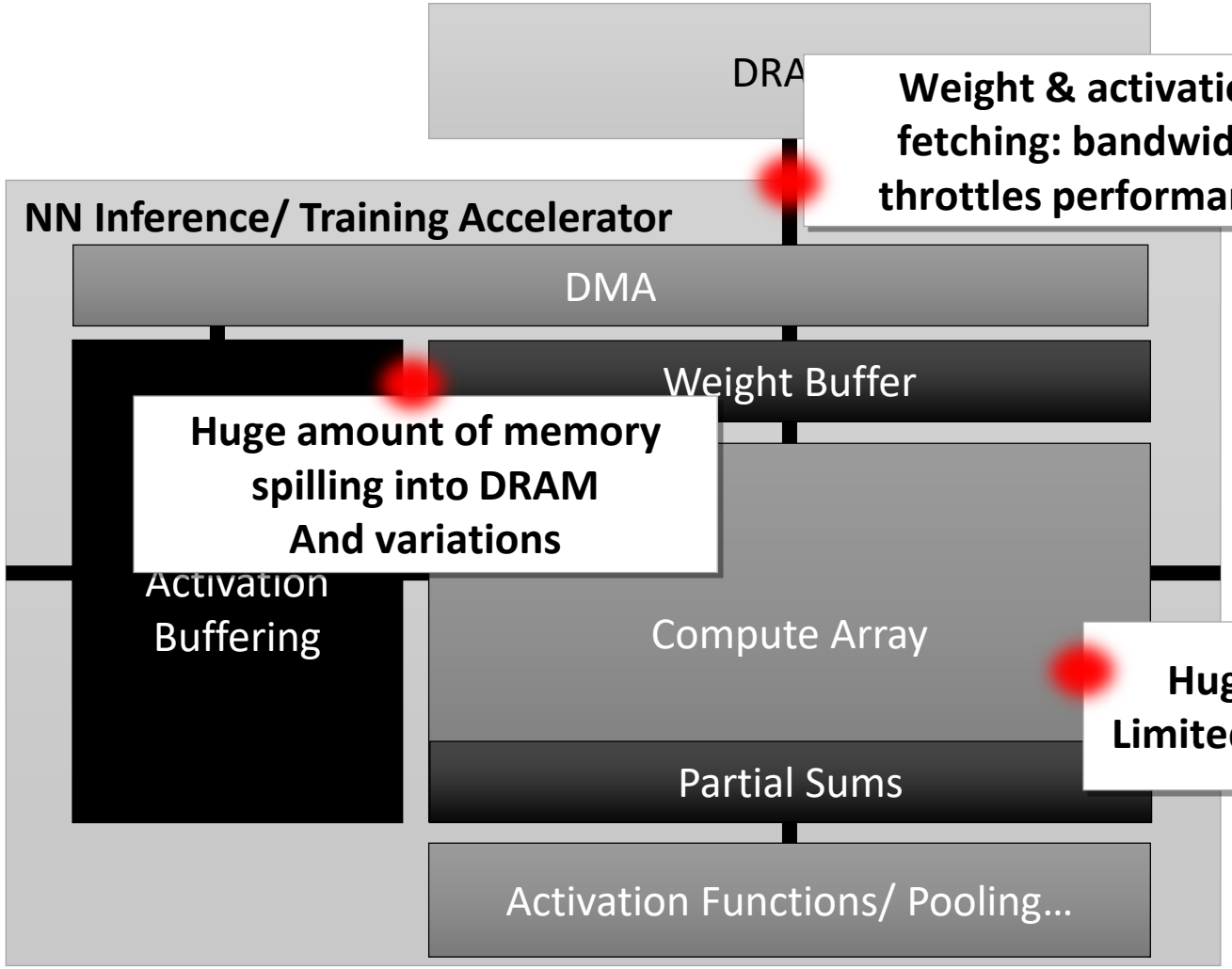
In Summary: CNNs are associated with…

- **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

Architectural Challenges/ Pain Points



Weight & activation fetching: bandwidth throttles performance

Huge amount of memory spilling into DRAM And variations

Power consumption for embedded

Latency in real-time processing

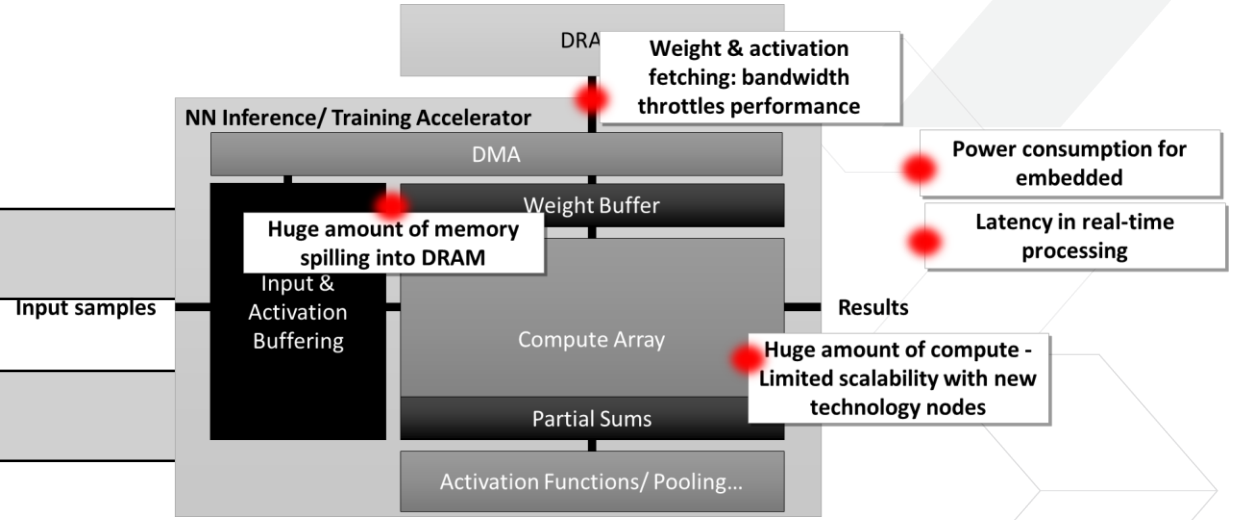
Huge amount of compute and variation- Limited scalability with new technology nodes

Requires algorithmic & architectural innovation

Algorithmic Optimization Techniques

Optimization Techniques

- Loop transformations to minimize memory access*
- Pruning
- Compression
- Winograd, Strassen and FFT
- Novel layer types (squeeze, shuffle, shift)
- Numerical Representations & Reducing Precision



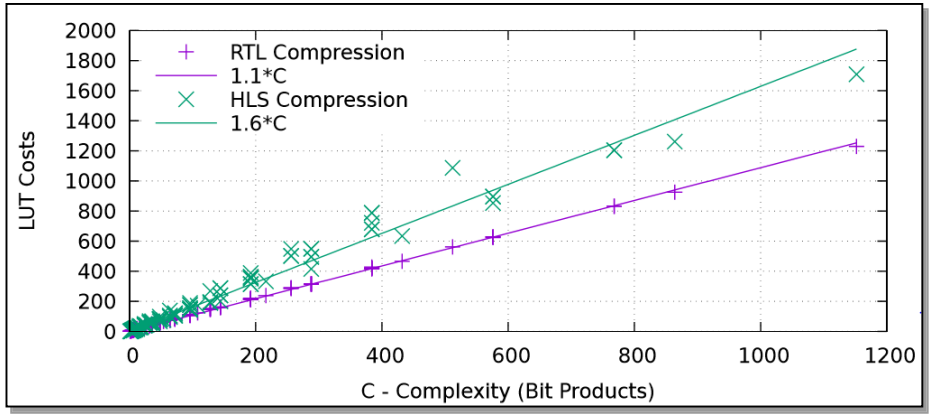
*Chen, Y.H., Krishna, T., Emer, J.S. and Sze, V., 2017. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits, 52(1), pp.127-13

Example: Reducing Bit-Precision

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

Precision	Modelsize [MB] (ResNet50)
1b	3.2
8b	25.5
32b	102.5

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

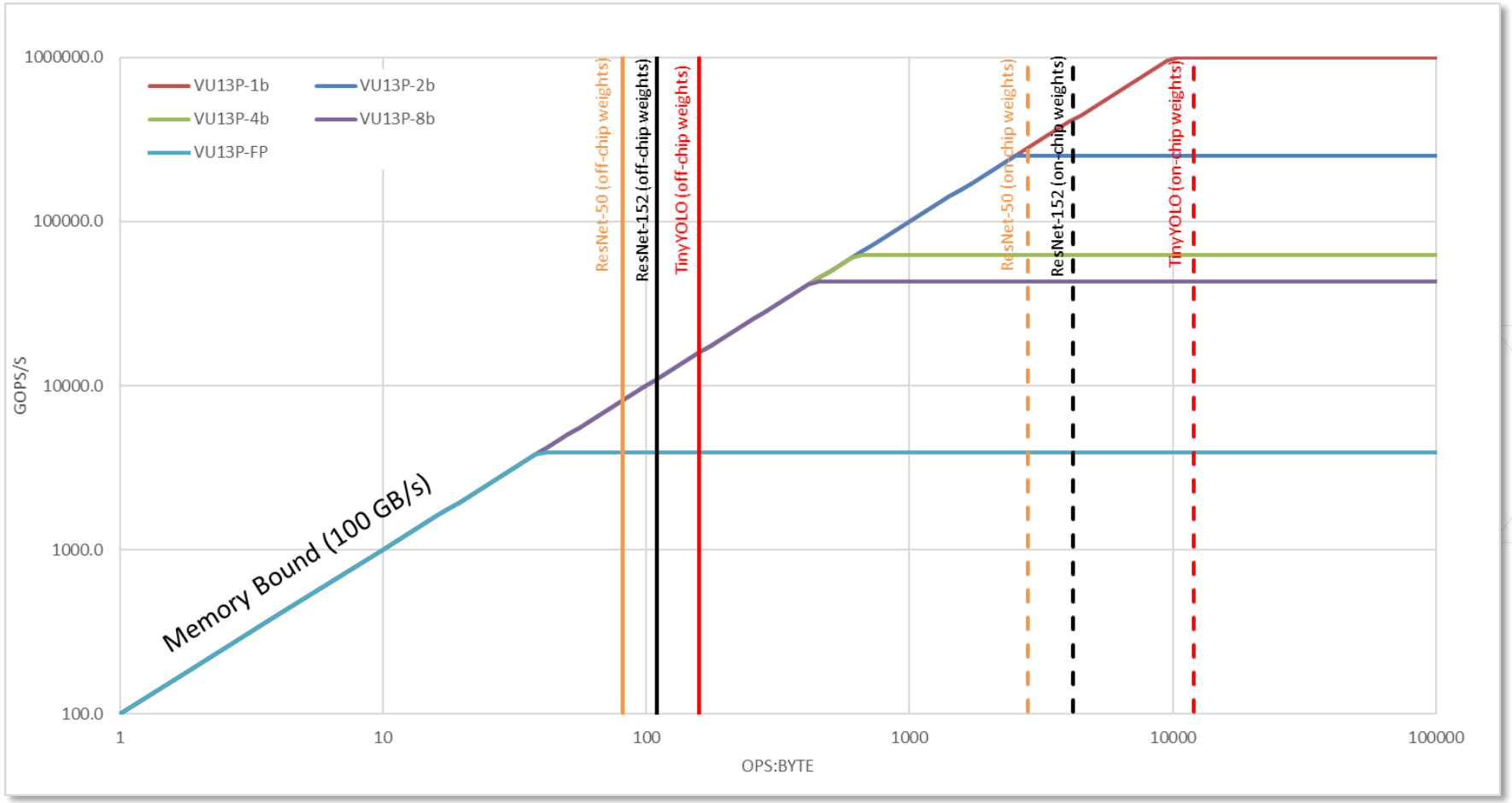


C= size of accumulator * size of weight * size of activation

(to appear in ACM TRETS SE on DL, FINN-R)

Reducing Precision provides Performance Scalability

Example: ResNet50, ResNet152 and TinyYolo



RP scales compute performance

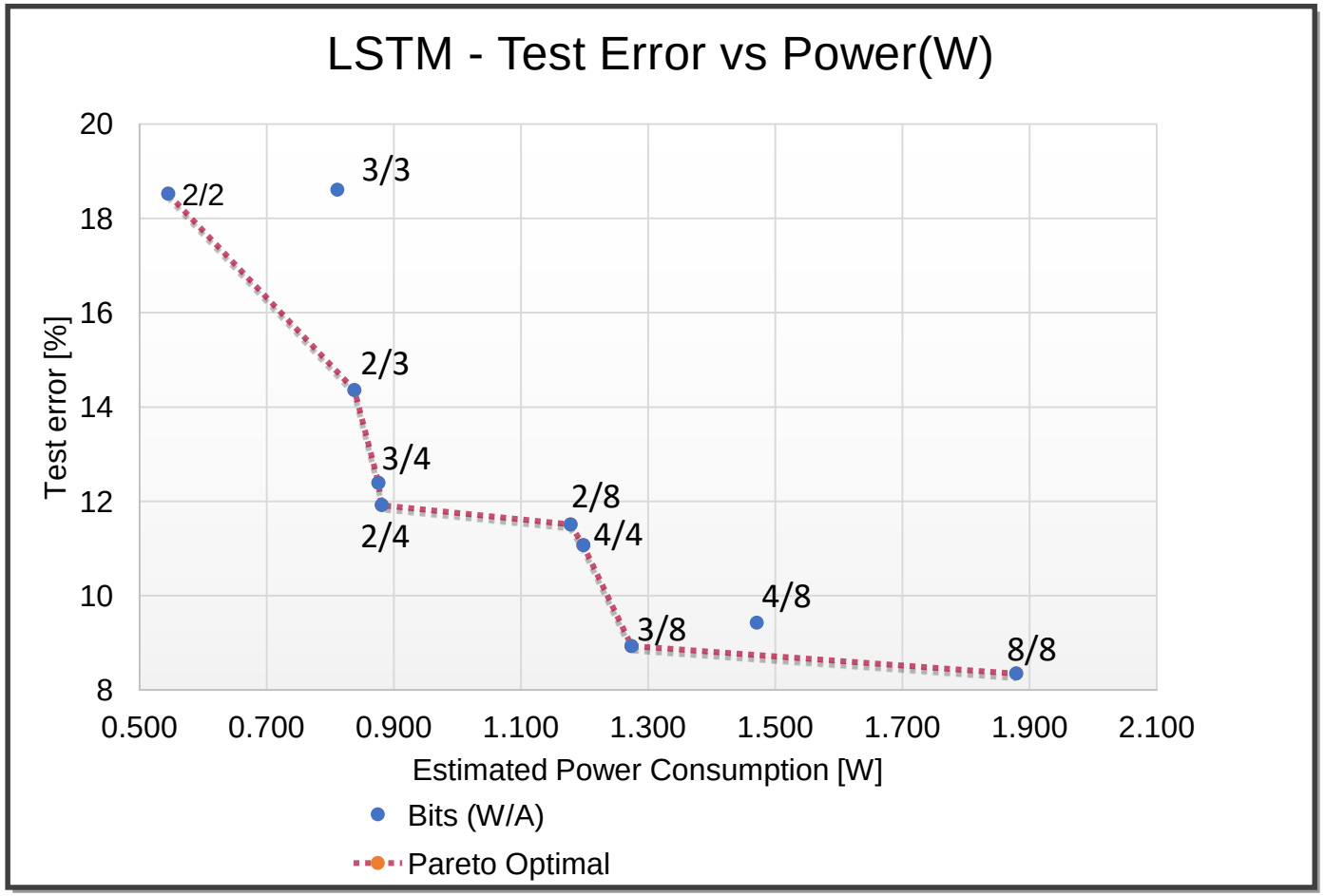
Theoretical Peak Performance for a VU13P with different Precision Operations

Assumptions: Application can fill device to 90% (fully parallelizable) 710MHz

RP reduces model size=> to stay on-chip

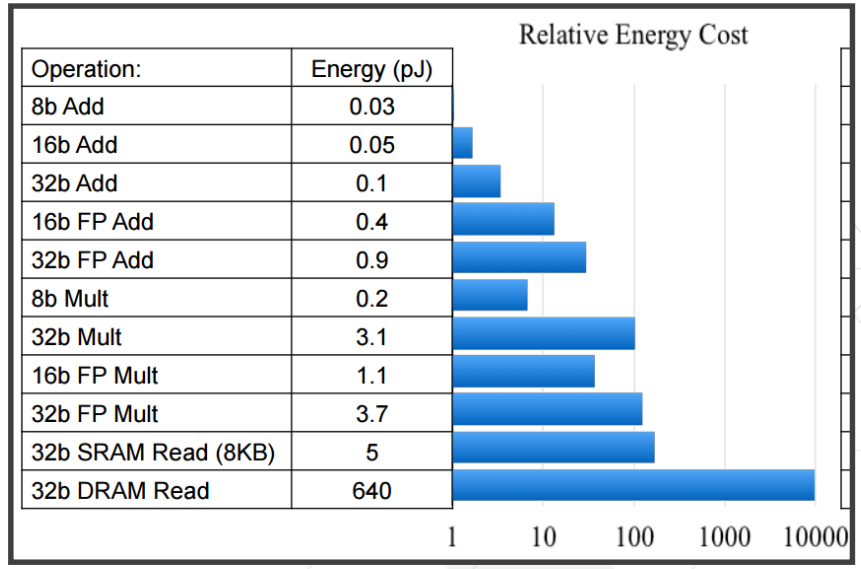
Reducing Precision Inherently Saves Power

FPGA:



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

ASIC:

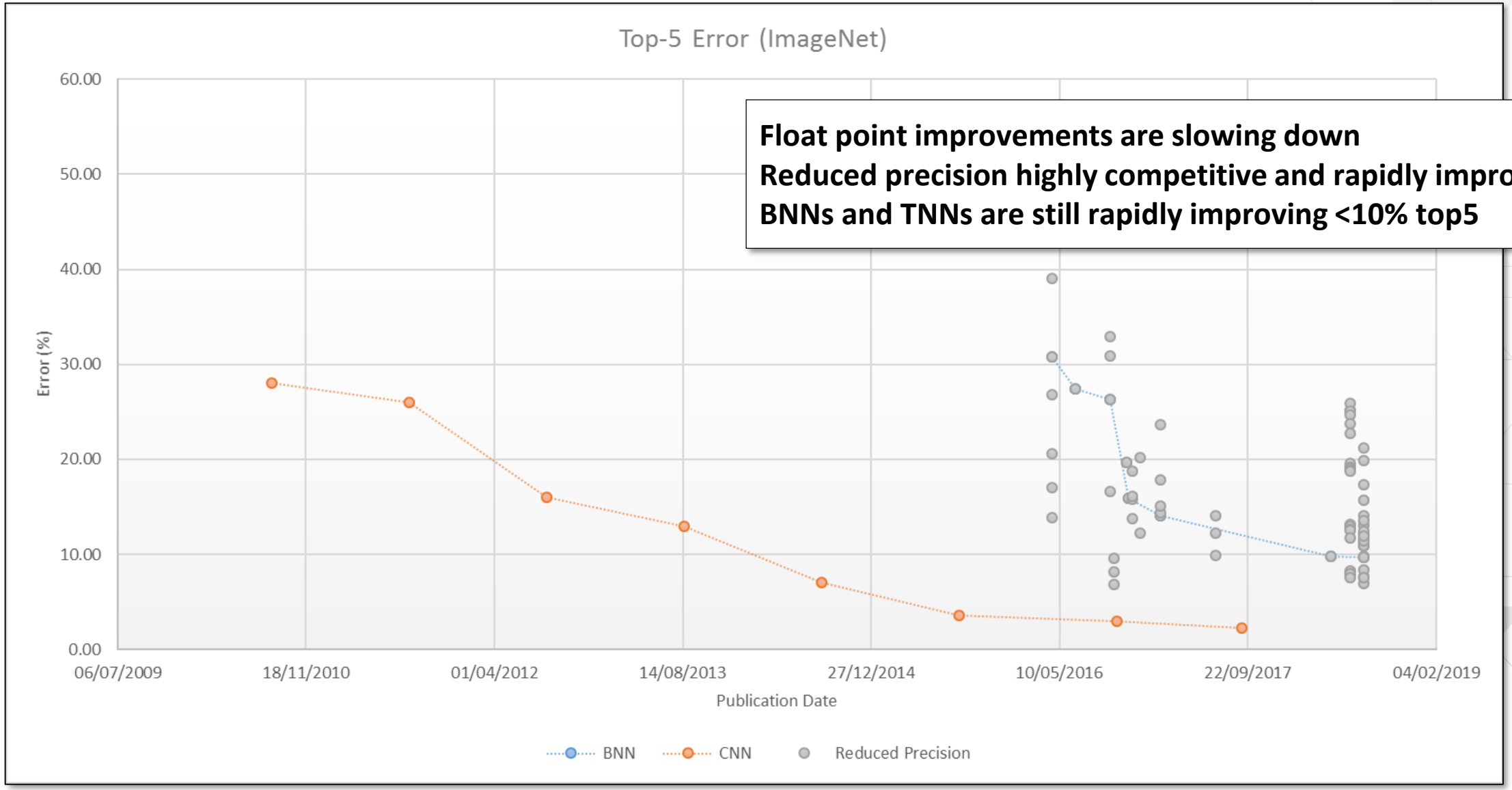


Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

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

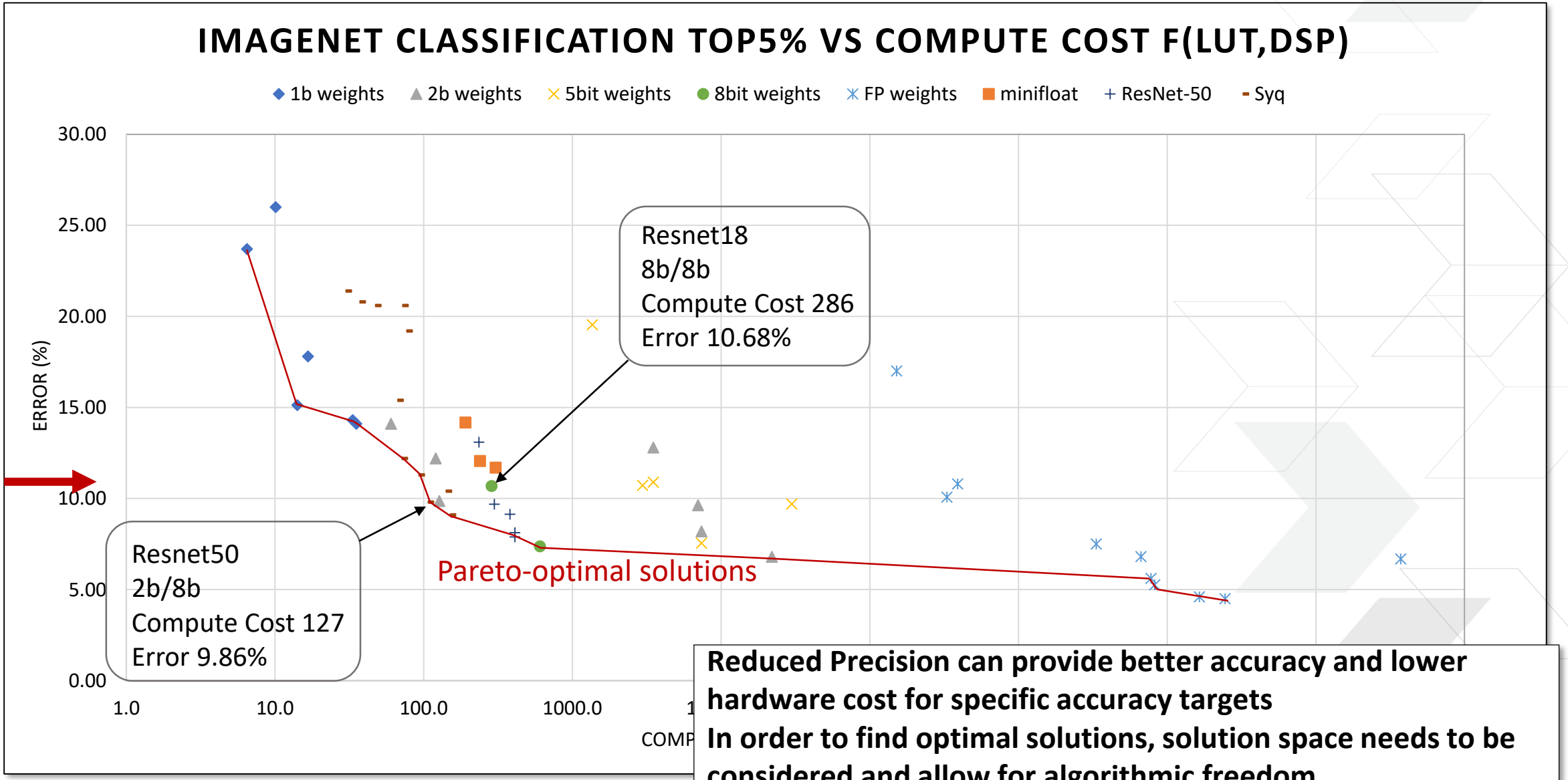
Rybalkin, V., Pappalardo, A., Ghaffar, M.M., Gambardella, G., Wehn, N. and Blott, M. "FINN-L: Library Extensions and Design Trade-off Analysis for Variable Precision LSTM Networks on FPGAs."

RPNNs: Closing the Accuracy Gap



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



Reduced Precision can provide better accuracy and lower hardware cost for specific accuracy targets

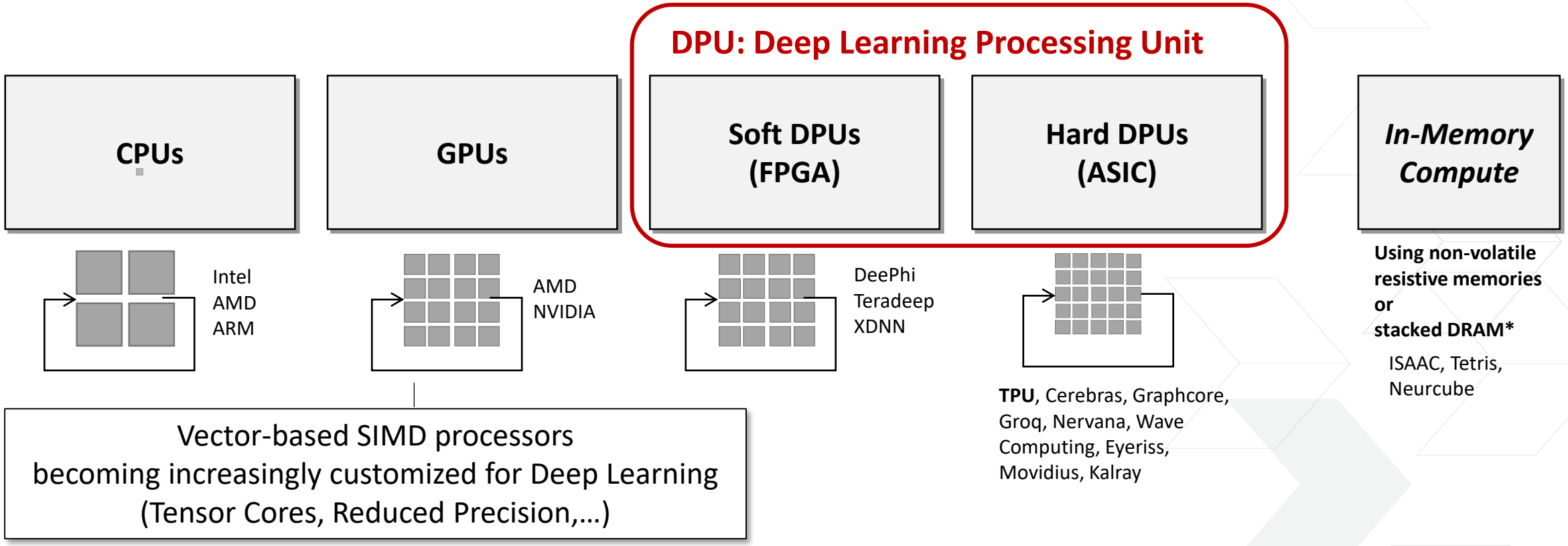
In order to find optimal solutions, solution space needs to be considered and allow for algorithmic freedom

Hardware Architectures and their Specialization Towards CNN Workloads

Exciting Times in Computer Architecture Research!

XILINX

Spectrum of New Architectures for Deep Learning



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

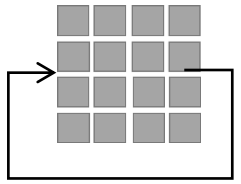
Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O., 2014, December. Dadiannao: A machine-learning supercomputer. In Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 609-622). IEEE Computer Society.

Architectural Choices – Macro-Architecture

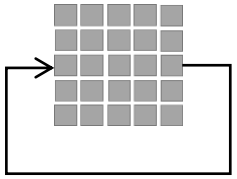
Soft DPUs (FPGA)

Hard DPUs (ASIC)

Matrix of PE

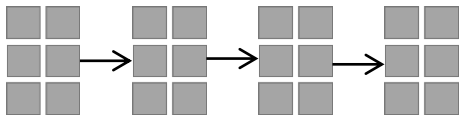


DeePhi
Teradeep
XDNN

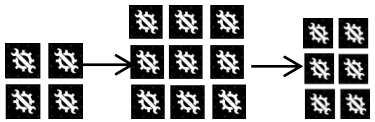


TPU, Cerebras, Graphcore, Groq, Nervana, Wave Computing, Eyeriss, Movidius, Kalray

Customized macro-architecture (Synchronous Dataflow)



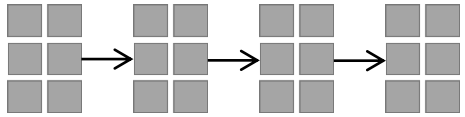
MSR Brainwave*



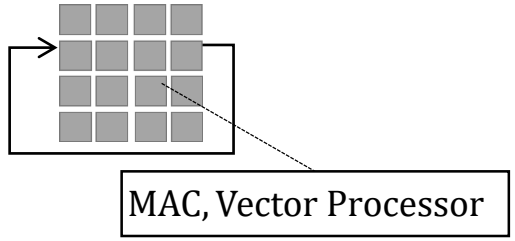
FINN**

**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*

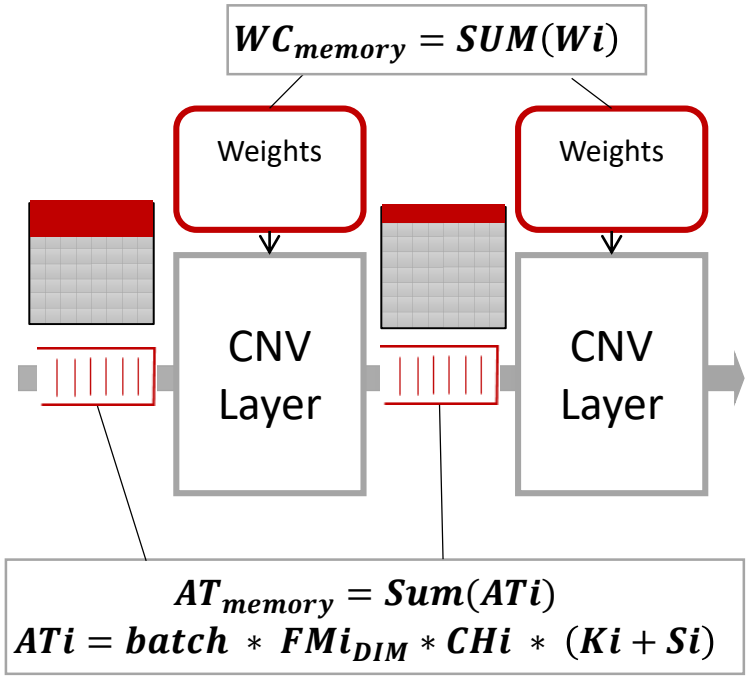
Synchronous Dataflow (SDF) vs Matrix of Processing Elements (MPE)

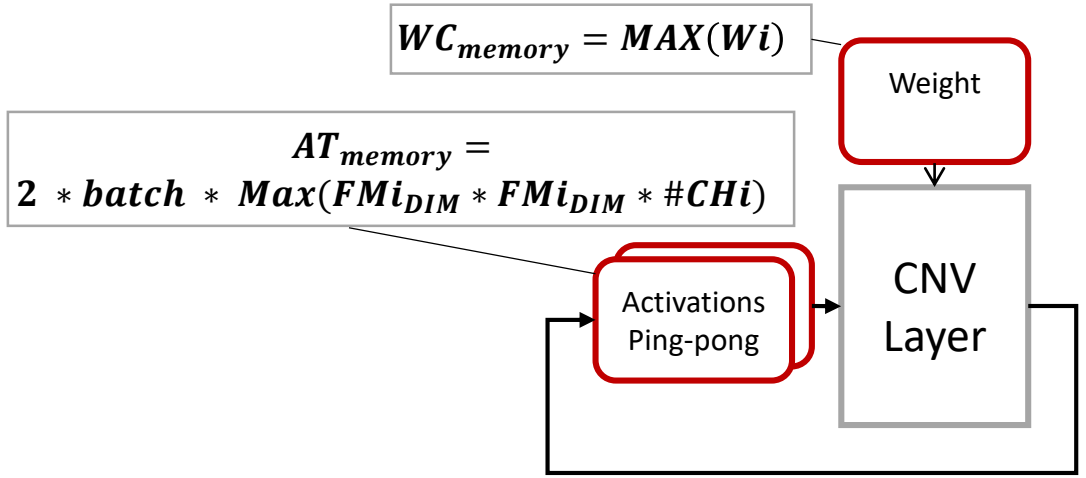


Spectrum of Options



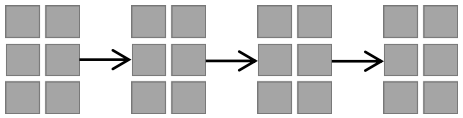
>> End points are pure layer-by-layer compute and feed-forward dataflow architecture

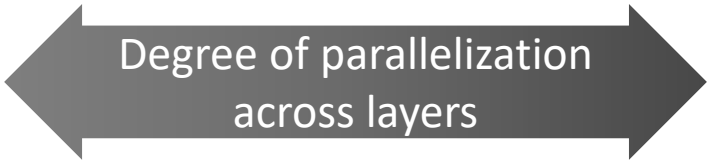


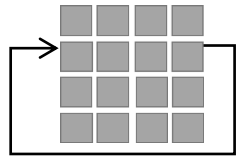


Lin, X., Yin, S., Tu, F., Liu, L., Li, X. and Wei, S. LCP: a layer clusters paralleling mapping method for accelerating inception and residual networks on FPGA. DAC'2016
Alwani, M., Chen, H., Ferdman, M. and Milder, P. Fused-layer CNN accelerators. MICRO 2016.

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



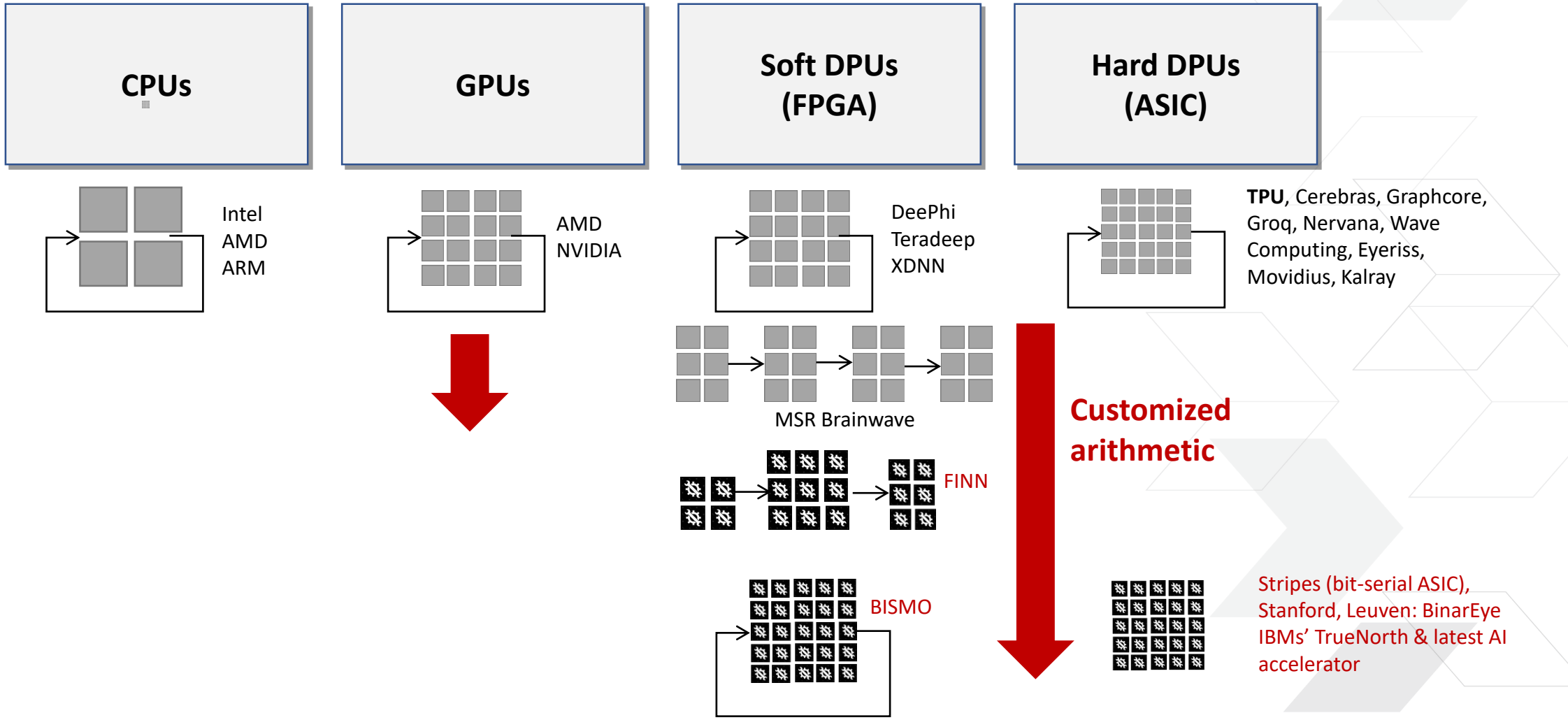




- Requires less activation buffering
- Higher compute and memory efficiency due to custom-tailored hardware design
- Less flexibility
- Less latency (reduced buffering)
- No control flow (static schedule)

- 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

Architectural Choices – Micro-Architecture

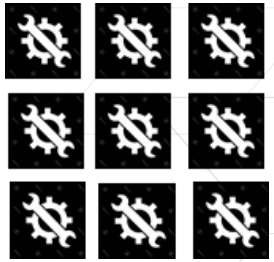


Judd, P., Albericio, J., Hetherington, T., Aamodt, T.M. and Moshovos, A., 2016, October. Stripes: Bit-serial deep neural network computing. MICRO'2016

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

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

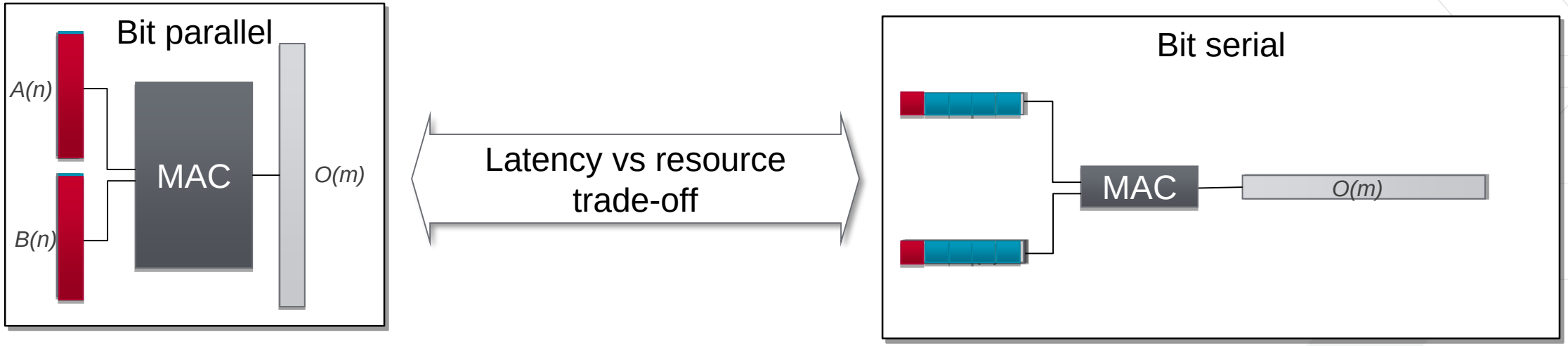
	DEC	INC	CONCAVE	CONVEX
Top-1 [%]	53.79	50.35	54.45	54.33
Top-5 [%]	77.59	74.89	76.43	78.20

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]*

Micro-Architecture:
Bit-Parallel vs Bit-Serial

- **Bit-serial can provide run-time programmable precision with a fixed architecture**
  - ASIC* or FPGA** overlay



- **FPGA: Flexibility comes at almost no cost and provides equivalent bit-level performance at chip-level for low precision***

*Judd, P., Albericio, J., Hetherington, T., Aamodt, T.M. and Moshovos, A., 2016, October. Stripes: Bit-serial deep neural network computing. MICRO'2016

**Umuroglu, Rasnayake, Sjalander"BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing." FPL'2018

https://arxiv.org/pdf/1806.08862.pdf

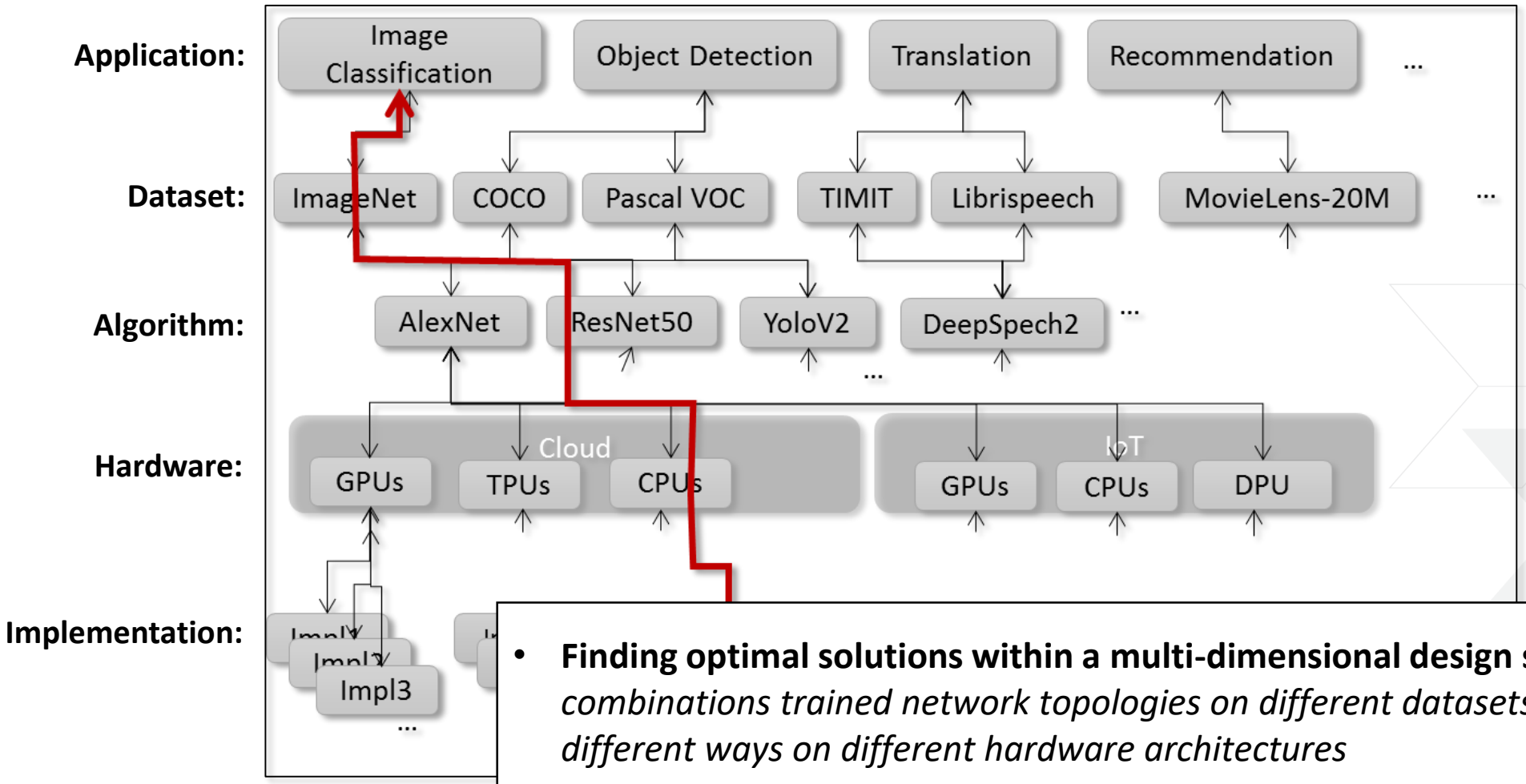
Summary

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

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

THANK YOU!

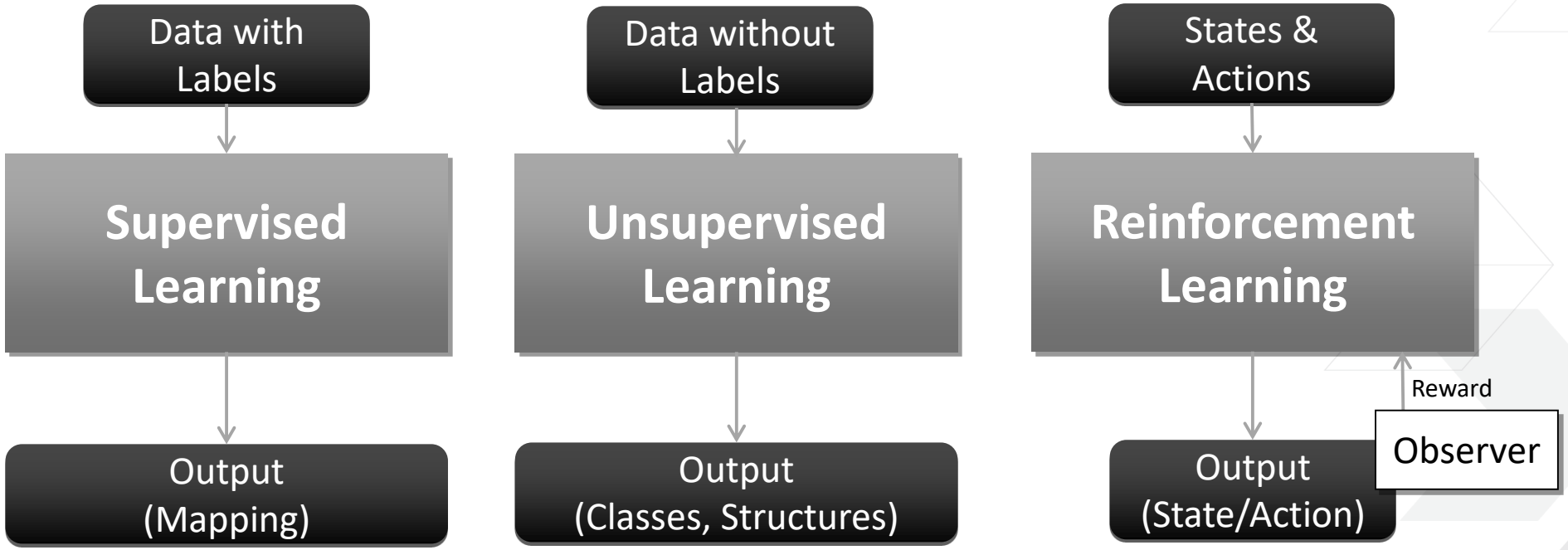
Adaptable.
Intelligent.

XILINX

Part 1 - Agenda

- Neural Networks
- Computation & Memory Requirements
- Algorithmic Optimization Techniques
- Hardware Architectures

Learning Paradigms



Batches*

*Caution: Overloaded Terms!

- **Batch:**
  - Collection of inputs buffered to capitalize on parallelism
- **Batches in Inference:**
  - Intention is to maximize the compute per loaded weights, helps increase compute efficiency when weight memory bound
    - Weight_bandwidth = weights*frame-rate/batch
  - Downside: adverse effects on latency:
    - Latency >= batch_size * 1/frame rate
- **Batches in Training:**
  - Batch size (mini-batch or iteration size) also dictates at what intervals weight updates happen)
  - Larger batch sizes require more memory, and can have potentially adverse effects on accuracy, and smaller batches might have adverse effects on training time