



# Can FPGAs beat GPUs in Accelerating Next-Generation Deep Neural Networks?

[Eriko Nurvitadhi](#), Ganesh Venkatesh, Jaewoong Sim, Debbie Marr (Accelerator Architecture Lab)

Randy Huang, Jason Gee Hock Ong, Yeong Tat Liew (Programmable Solutions Group)

Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra (FPGA Product Team)

Guy Boudoukh (Computer Vision Group)

*Disclaimer: the views expressed in this talk are those of the speaker and not his employer.*

# Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

The code names presented in this document are only for use by Intel to identify products, technologies, or services in development, that have not been made commercially available to the public, i.e., announced, launched or shipped. They are not "commercial" names for products or services and are not intended to function as trademarks.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com/design/literature.htm>.

Intel is a trademark of Intel Corporation in the US and other countries.

Copyright © 2017 Intel Corporation. All rights reserved.

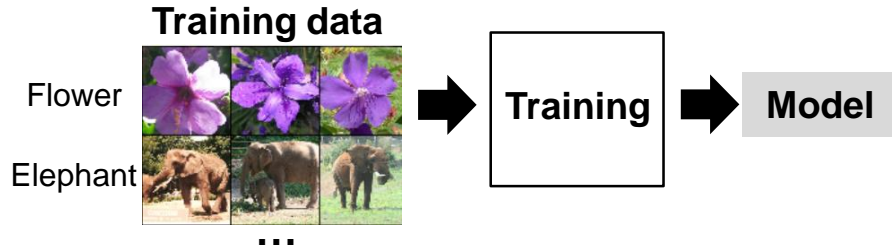
\* Other brands and names may be claimed as the property of others.

# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

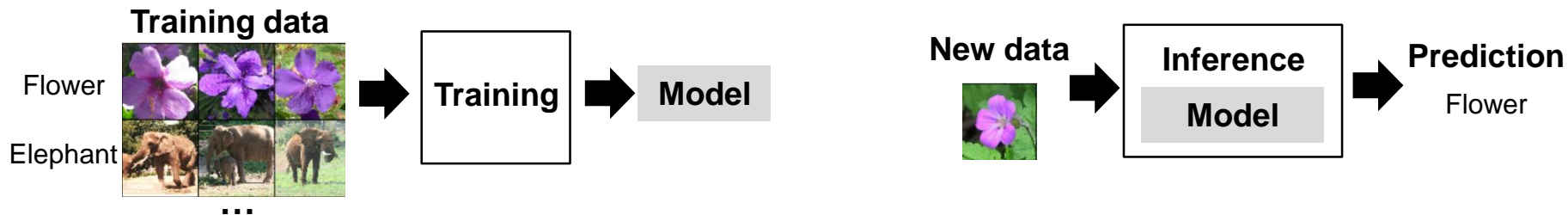
# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



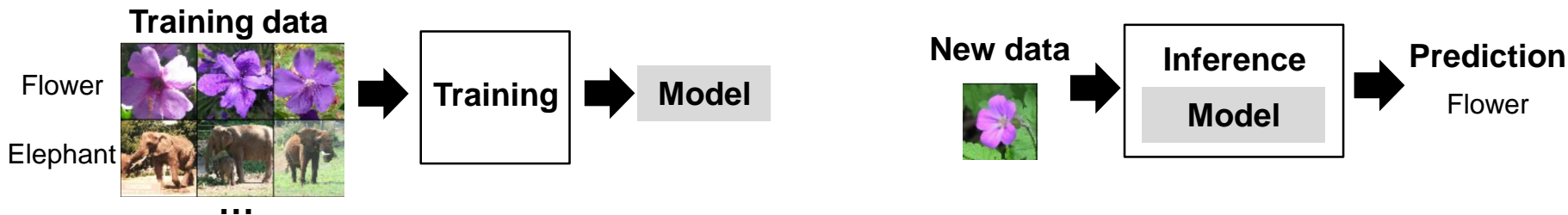
# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



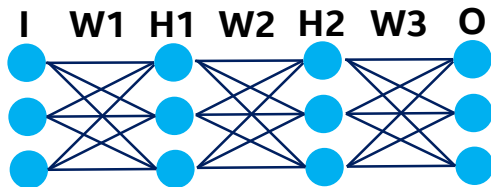
# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



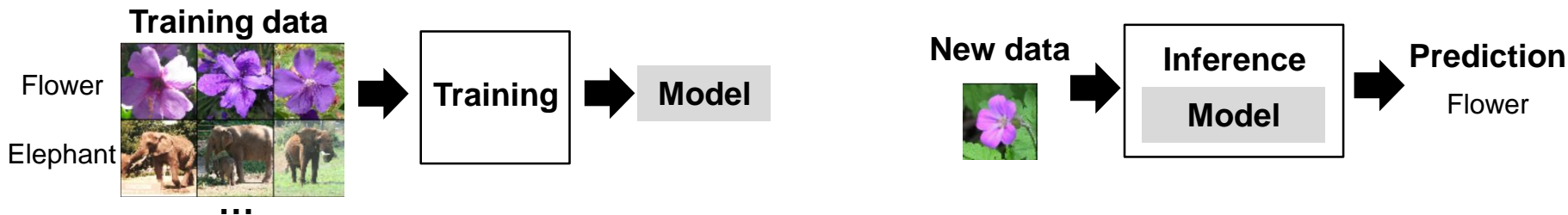
Consists of layers of neurons connected via weighted edges

E.g., 4-layer neural net



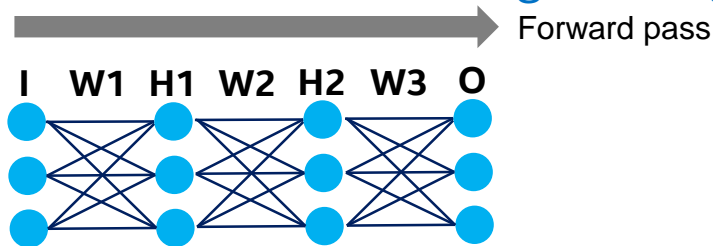
# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



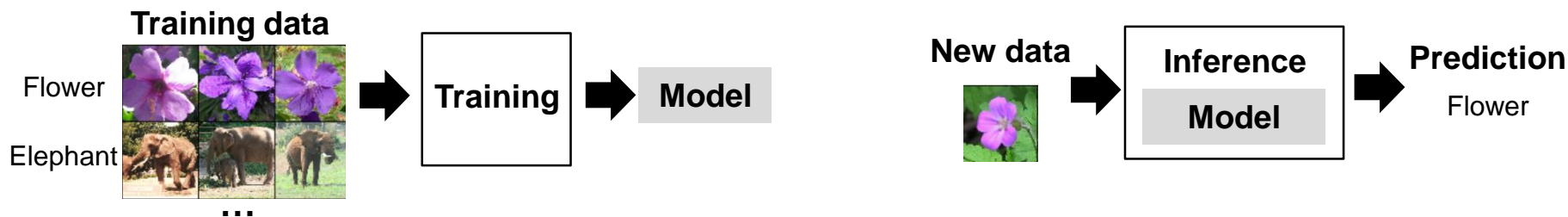
Consists of layers of neurons connected via weighted edges

E.g., 4-layer neural net

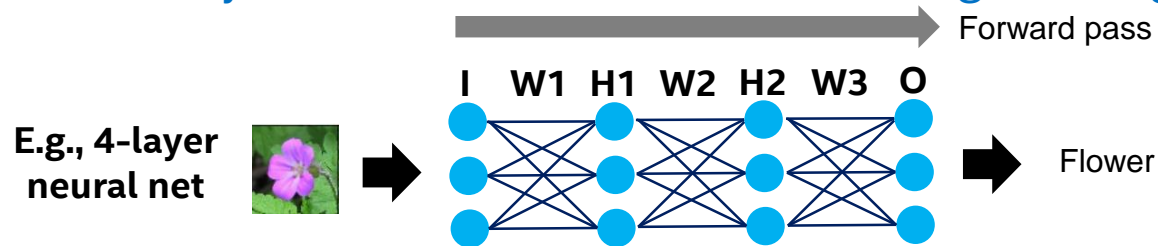


# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



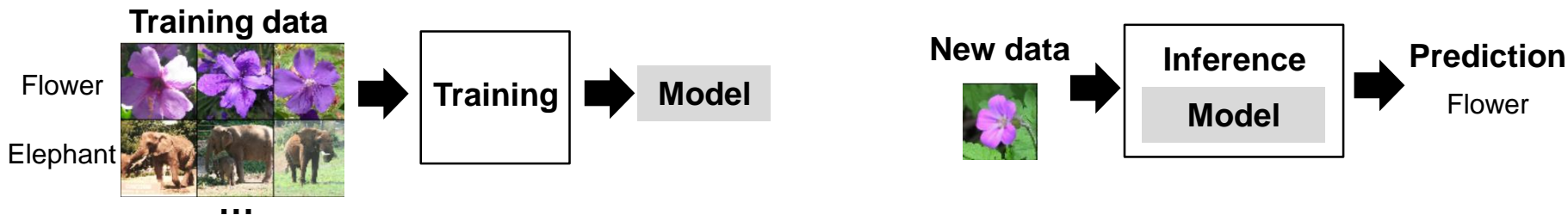
Consists of layers of neurons connected via weighted edges



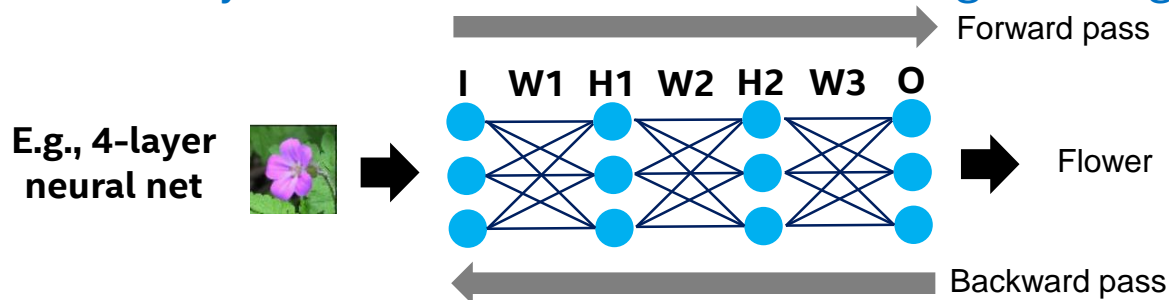


# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

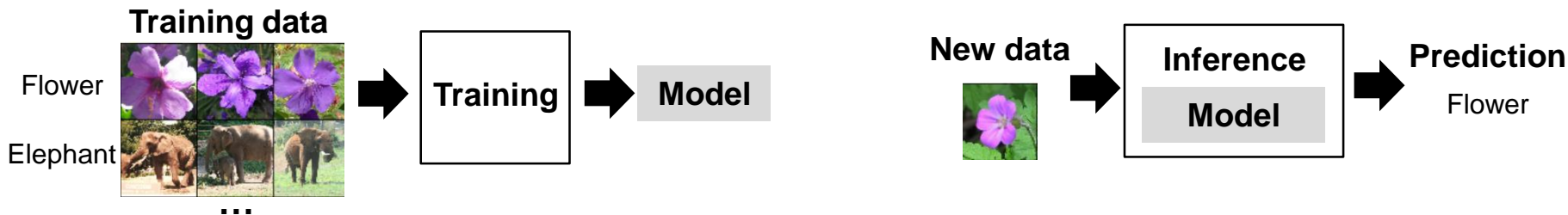


Consists of layers of neurons connected via weighted edges

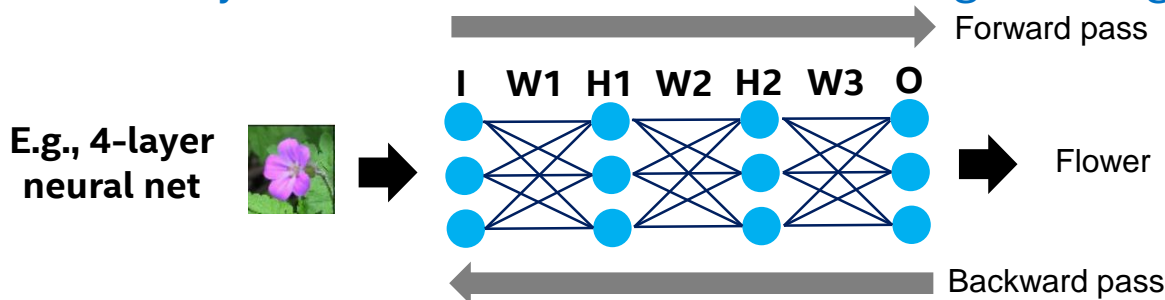


# Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics



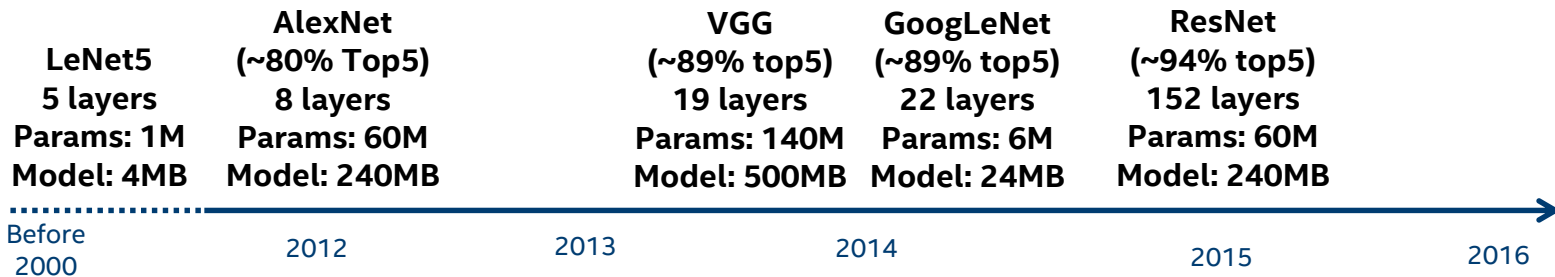
Consists of layers of neurons connected via weighted edges



State-of-the-art accuracies in multiple application domains

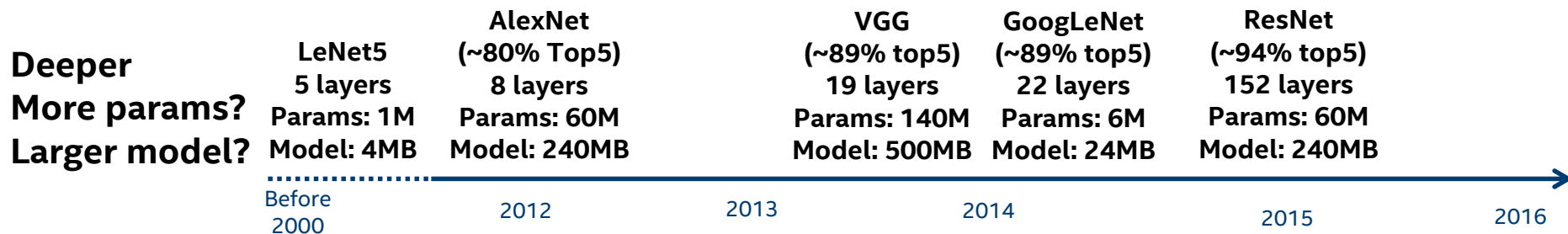
# DNNs Evolving Rapidly

**Deeper**  
**More params?**  
**Larger model?**



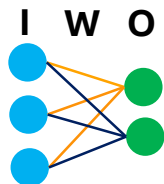
# DNNs Evolving Rapidly

Many efforts to improve efficiency

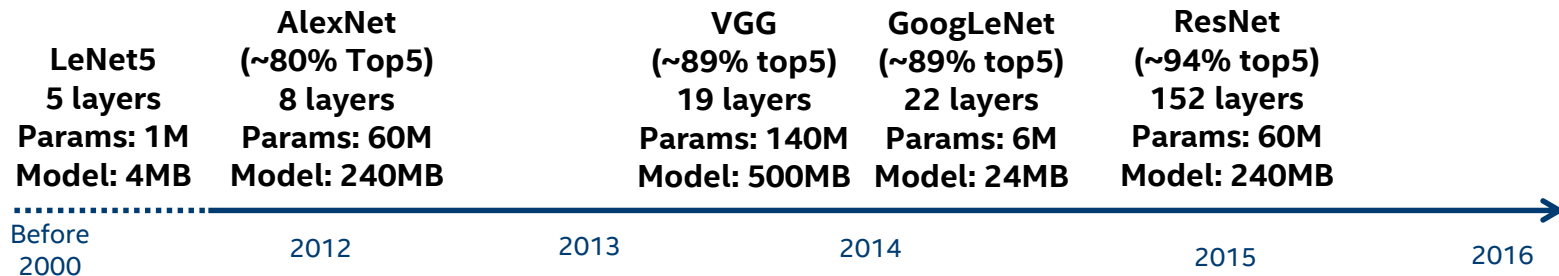


# DNNs Evolving Rapidly

Many efforts to improve efficiency

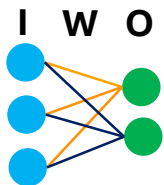


Deeper  
More params?  
Larger model?

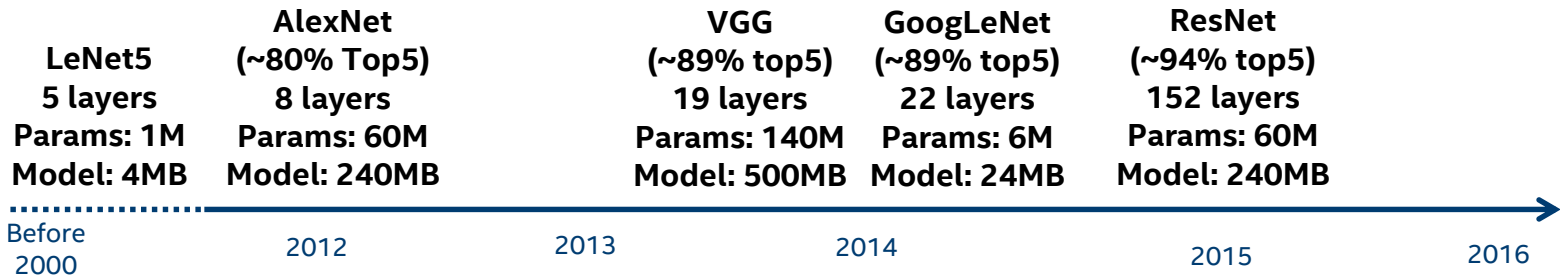


# DNNs Evolving Rapidly

Many efforts to improve efficiency



Deeper  
More params?  
Larger model?

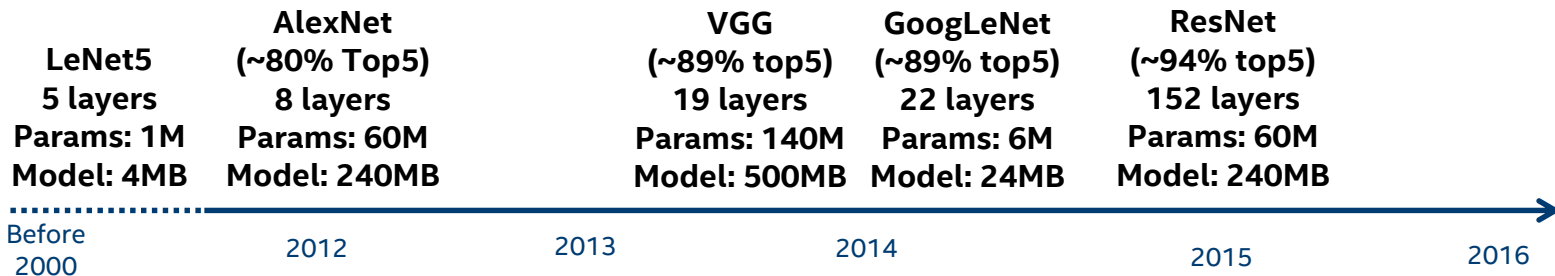


# DNNs Evolving Rapidly

Many efforts to improve efficiency

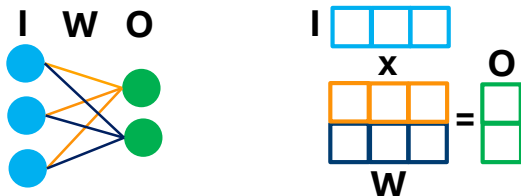


Deeper  
More params?  
Larger model?

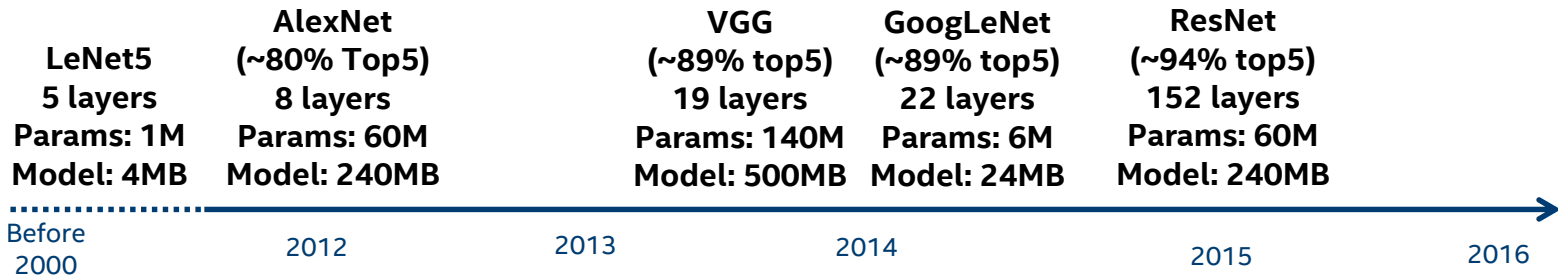


# DNNs Evolving Rapidly

Many efforts to improve efficiency



Deeper  
More params?  
Larger model?

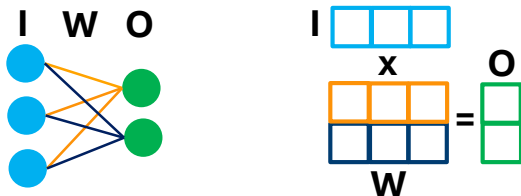




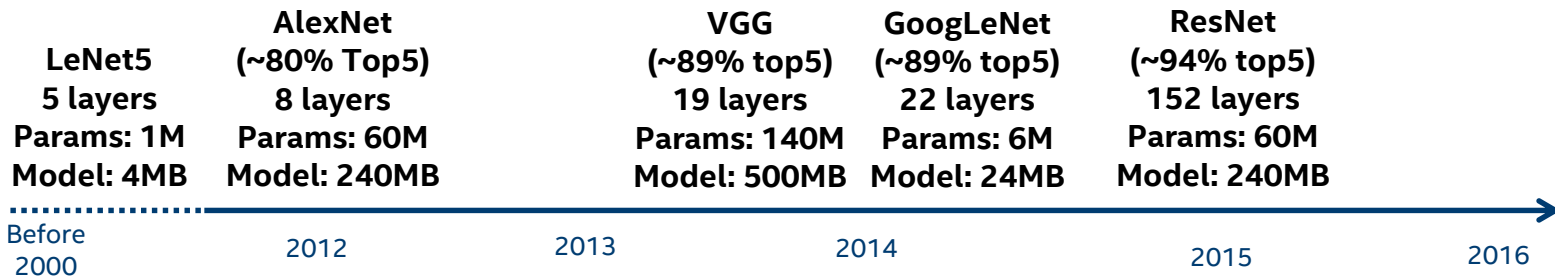
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching



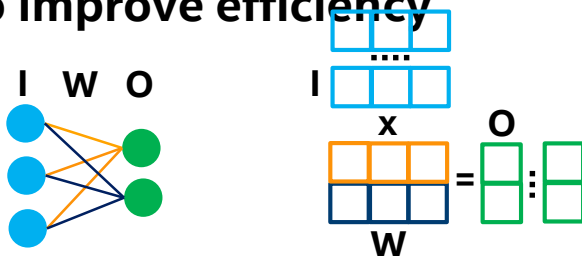
Deeper  
More params?  
Larger model?



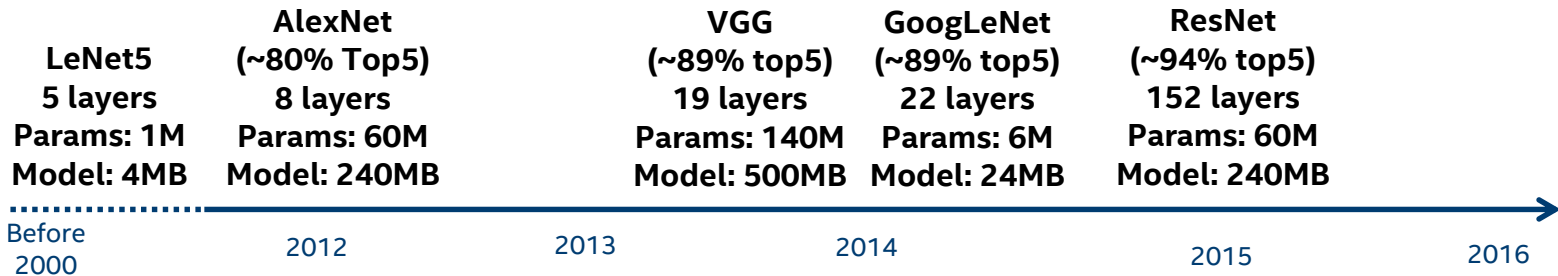
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching



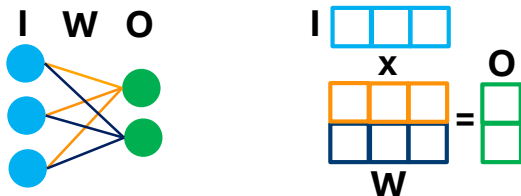
Deeper  
More params?  
Larger model?



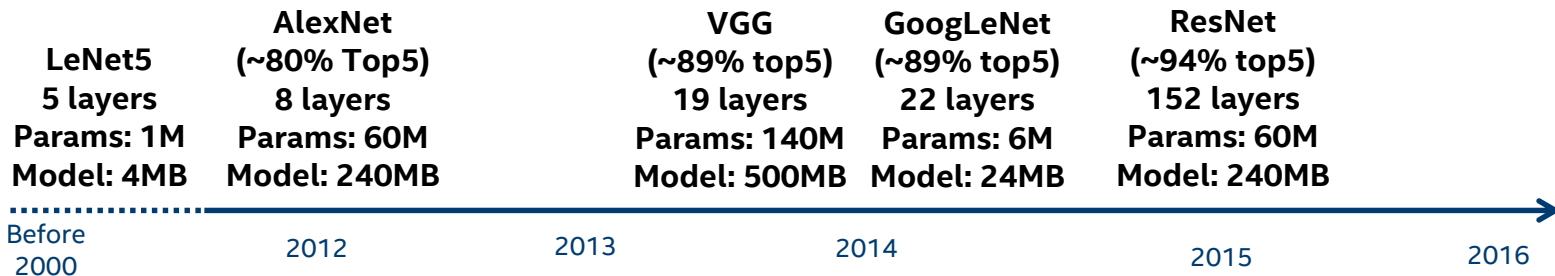
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching



Deeper  
More params?  
Larger model?

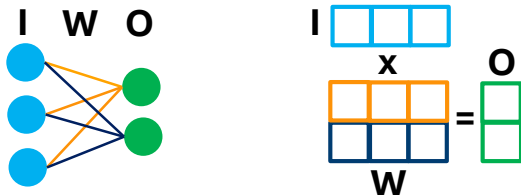


# DNNs Evolving Rapidly

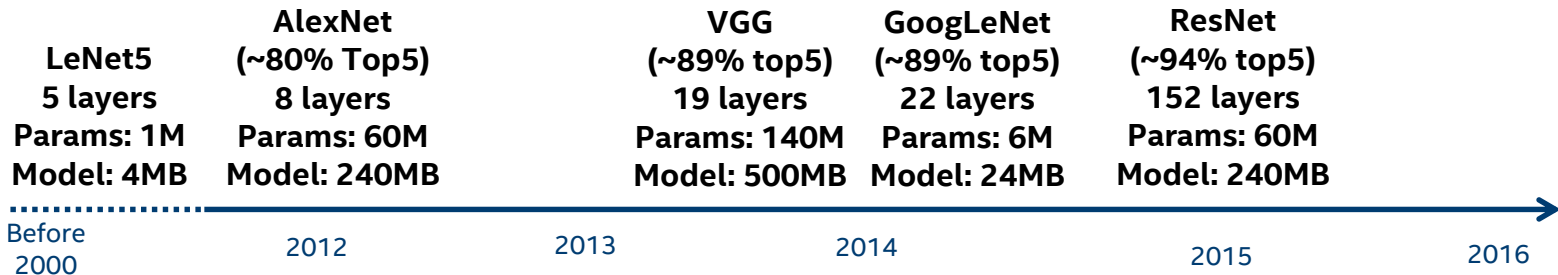
Many efforts to improve efficiency

Batching

Reduce bitwidth



Deeper  
More params?  
Larger model?

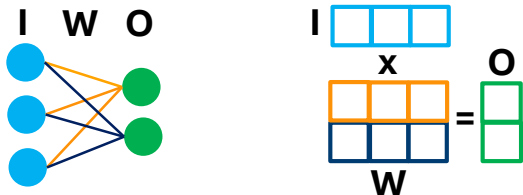


# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

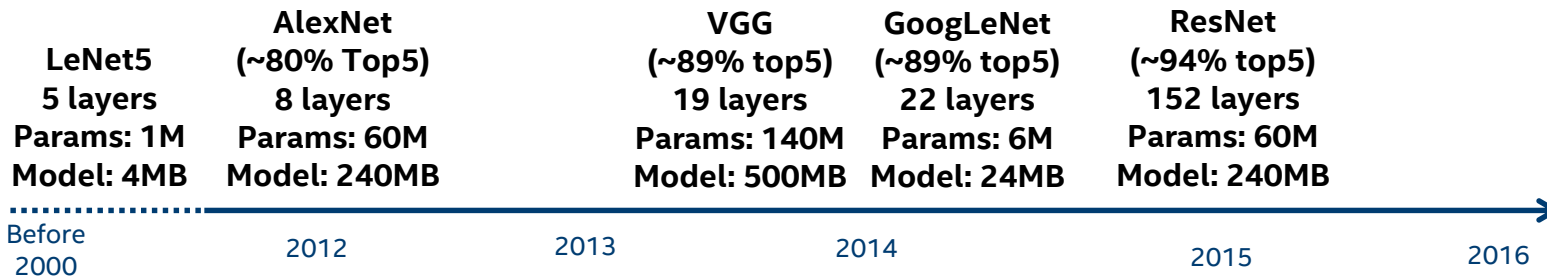
Reduce bitwidth



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



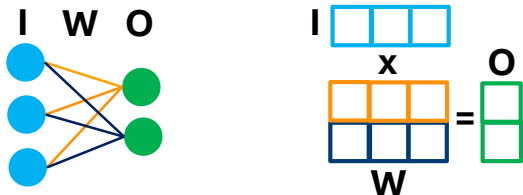
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

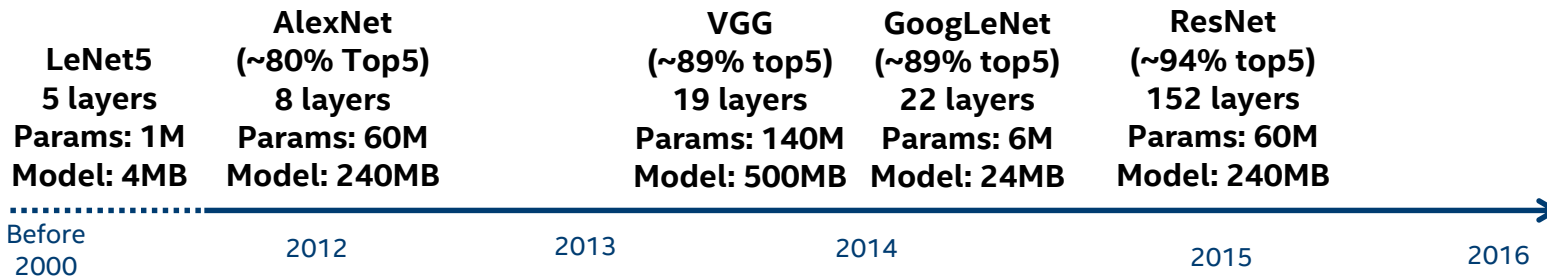
Sparse weights



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



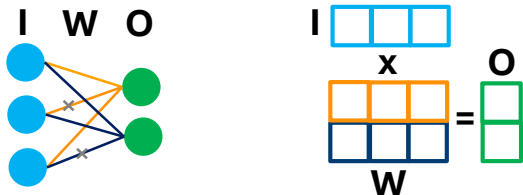
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

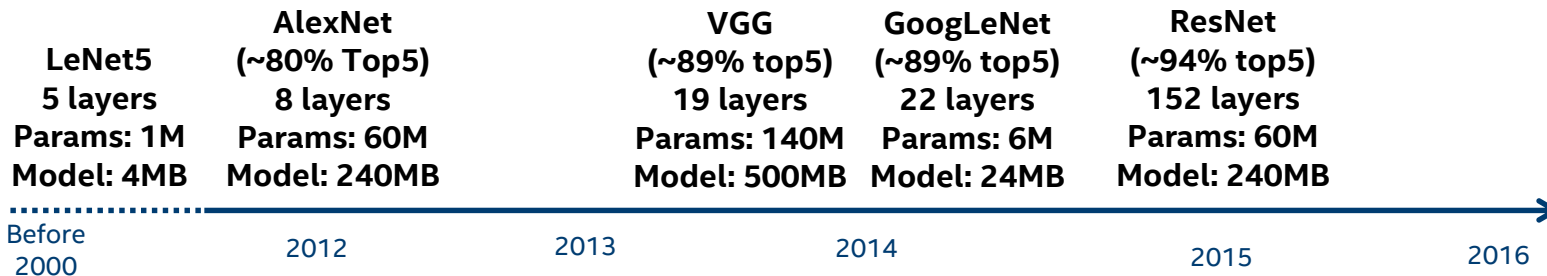
Sparse weights



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



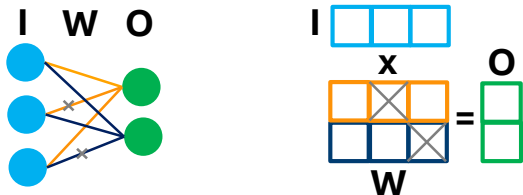
# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

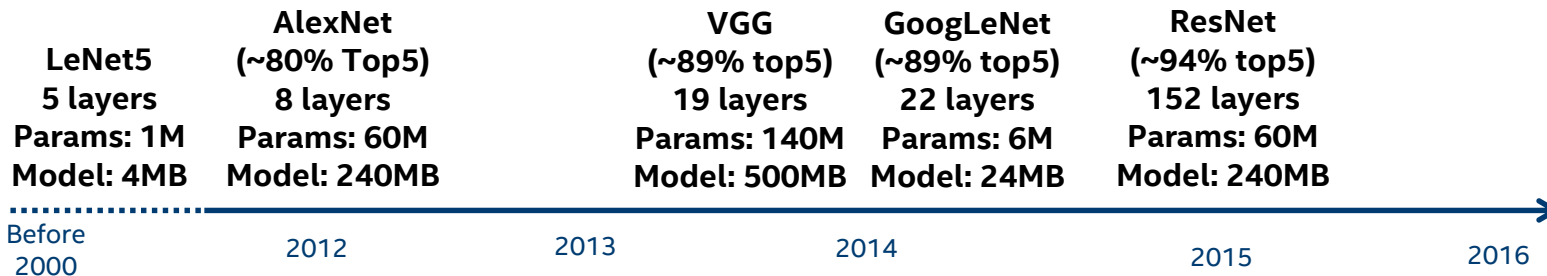
Sparse weights



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?





# DNNs Evolving Rapidly

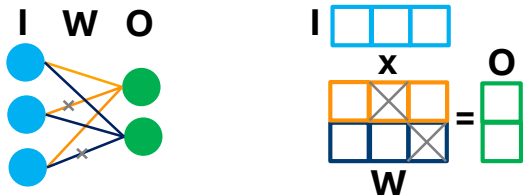
Many efforts to improve efficiency

Batching

Reduce bitwidth

Sparse weights

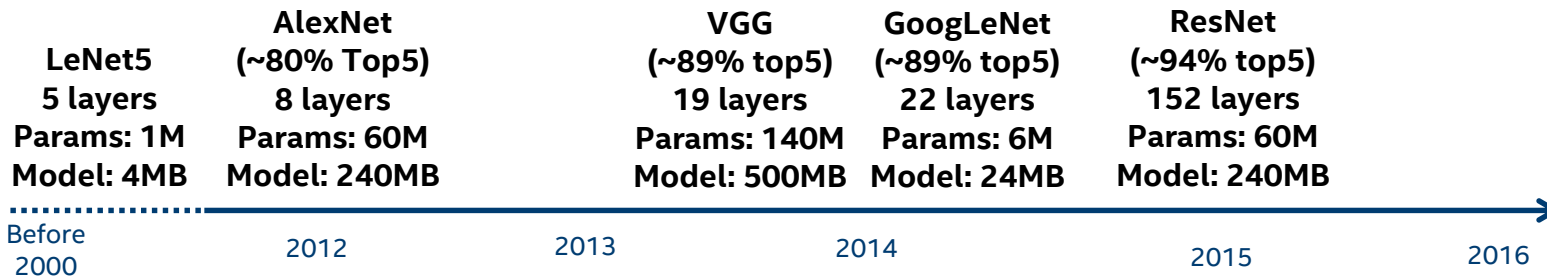
Sparse activations



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

Sparse weights

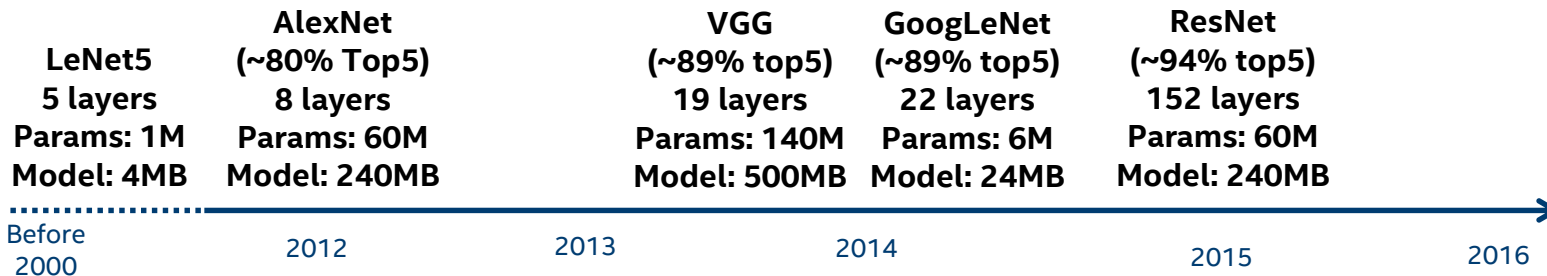
Sparse activations



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

Sparse weights

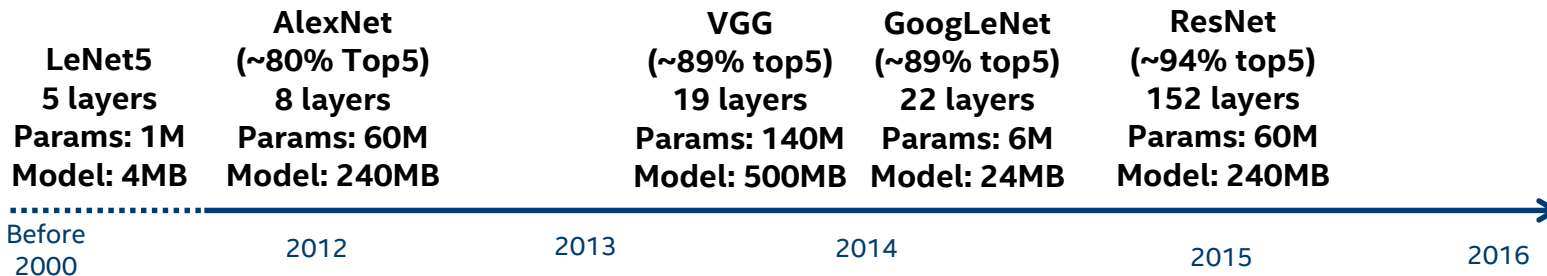
Sparse activations



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

Sparse weights

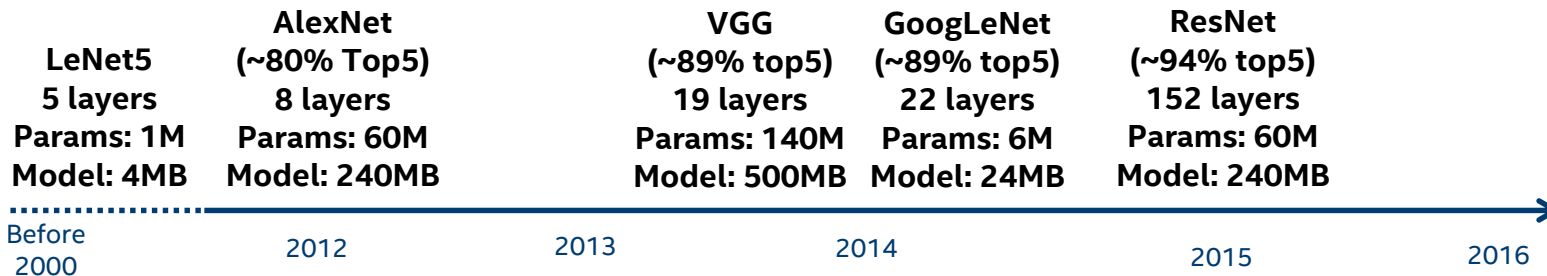
Sparse activations



BinaryConnect  
[NIPS'15]

XNORNet

Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

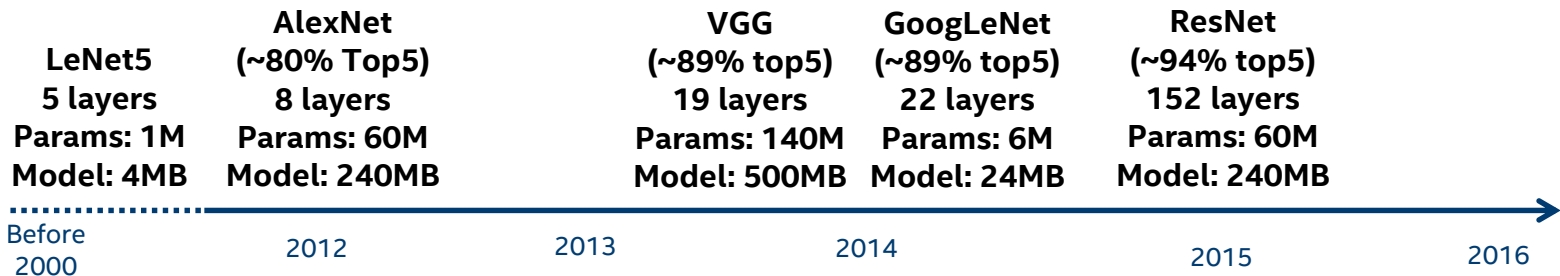
Reduce bitwidth

Sparse weights

Sparse activations



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

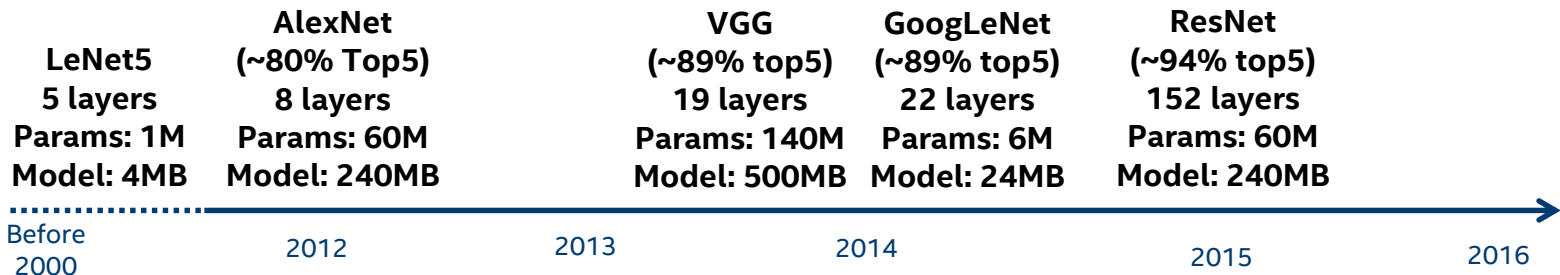
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

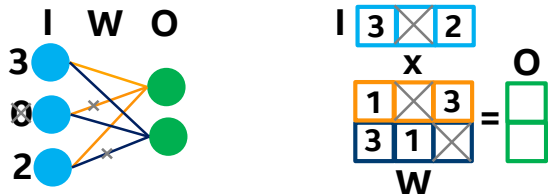
Batching

Reduce bitwidth

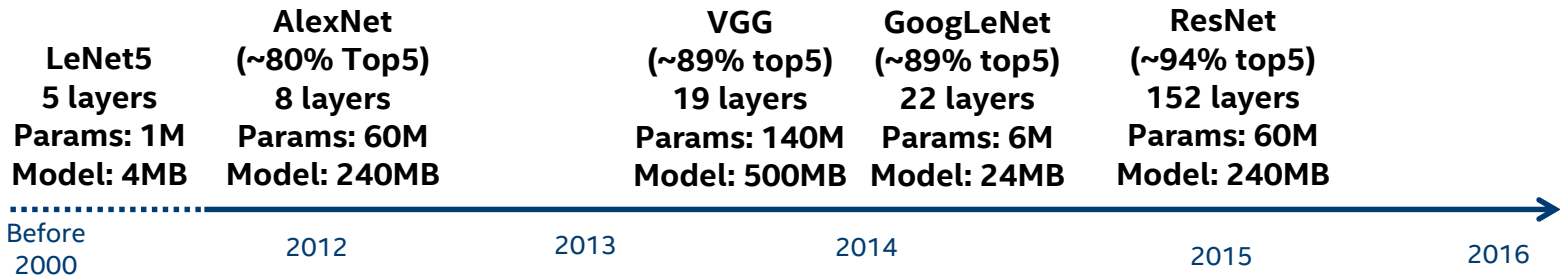
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

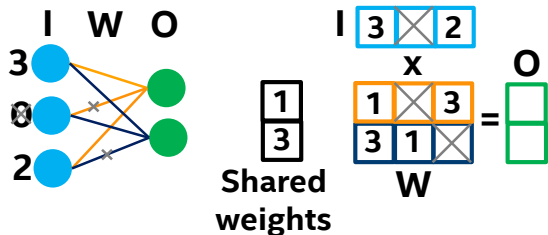
Batching

Reduce bitwidth

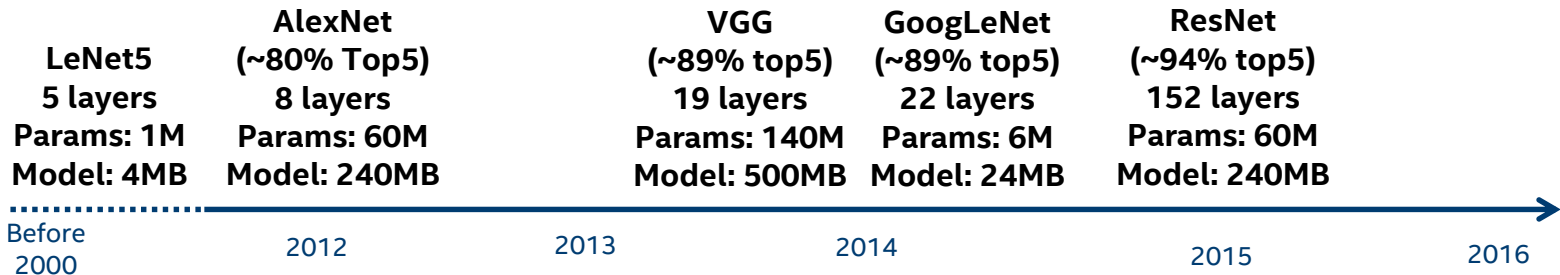
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?





# DNNs Evolving Rapidly

Many efforts to improve efficiency

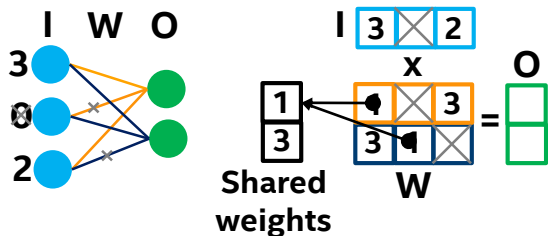
Batching

Reduce bitwidth

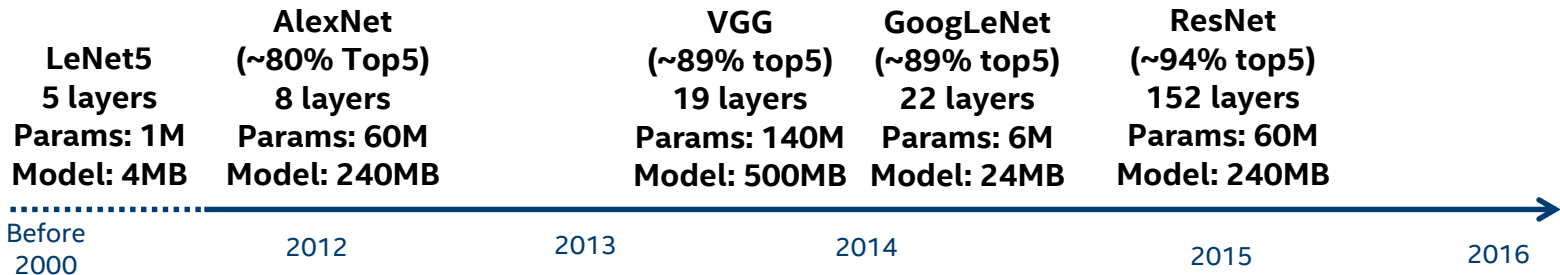
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

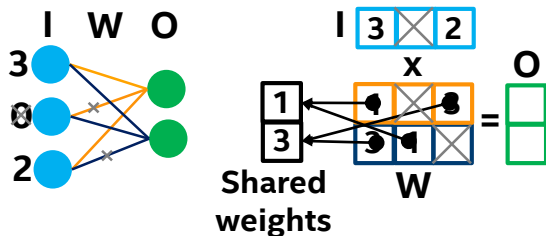
Batching

Reduce bitwidth

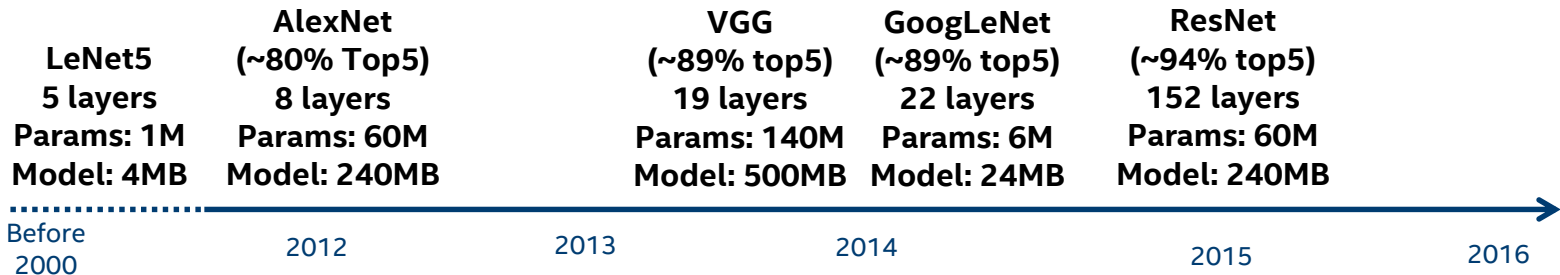
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

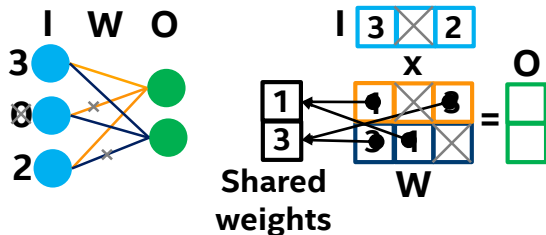
Batching

Reduce bitwidth

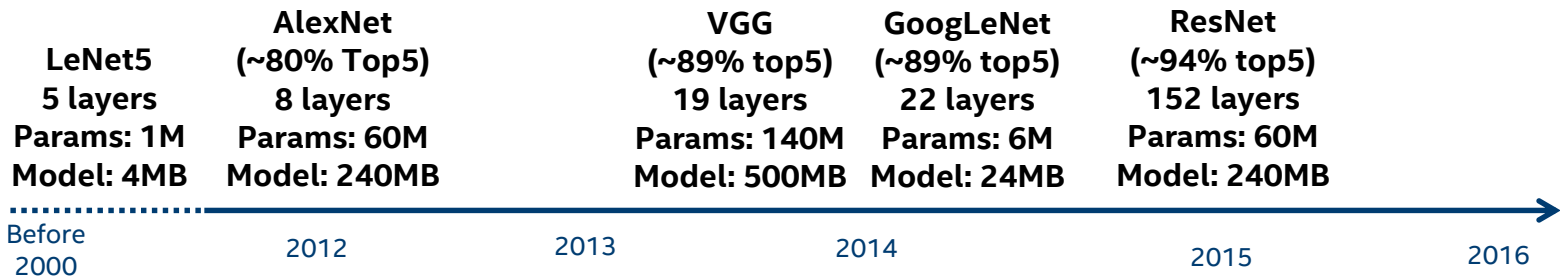
Sparse weights

Sparse activations

Compression



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

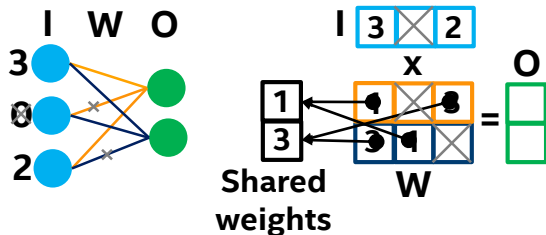
Reduce bitwidth

Sparse weights

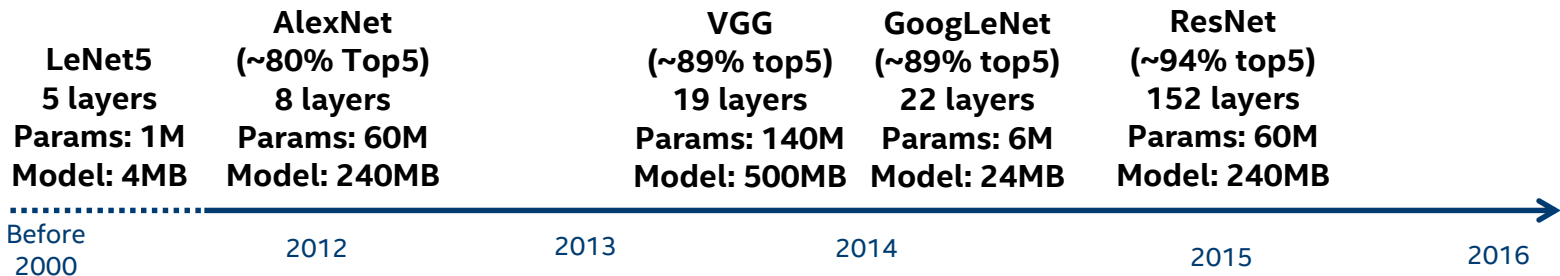
Sparse activations

Compression

Compact network



Deeper  
More params?  
Larger model?



Spatially SparseCNN [CIFAR-10 winner '14]

BinaryConnect [NIPS'15]  
SparseCNN [CVPR'15]  
Pruning [NIPS'15]  
HashedNets [ICML'15]

XNORNet  
TernaryConnect [ICLR'16]  
DeepComp [ICLR'16]

# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

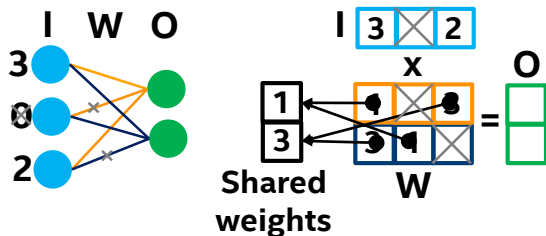
Reduce bitwidth

Sparse weights

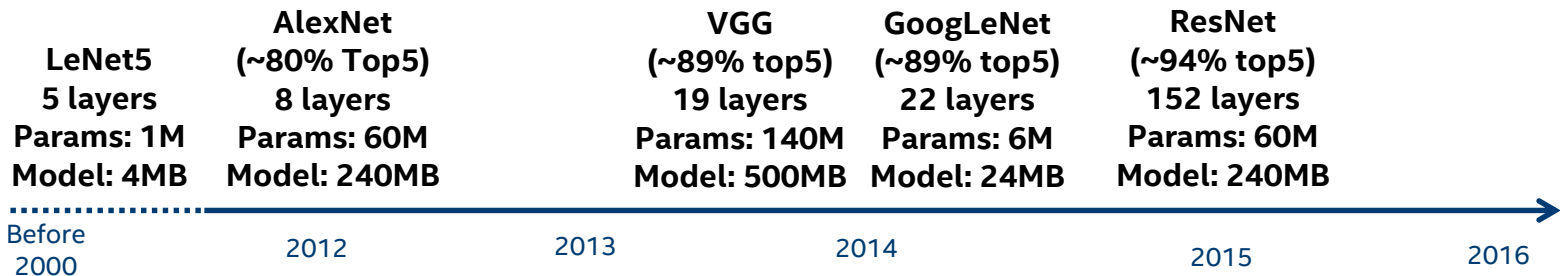
Sparse activations

Compression

Compact network



Deeper  
More params?  
Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

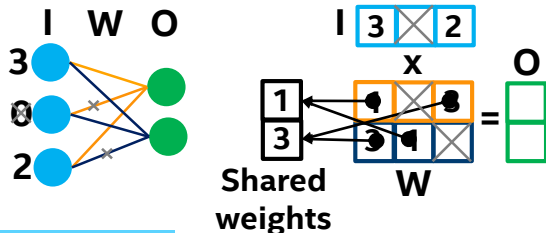
Reduce bitwidth

Sparse weights

Sparse activations

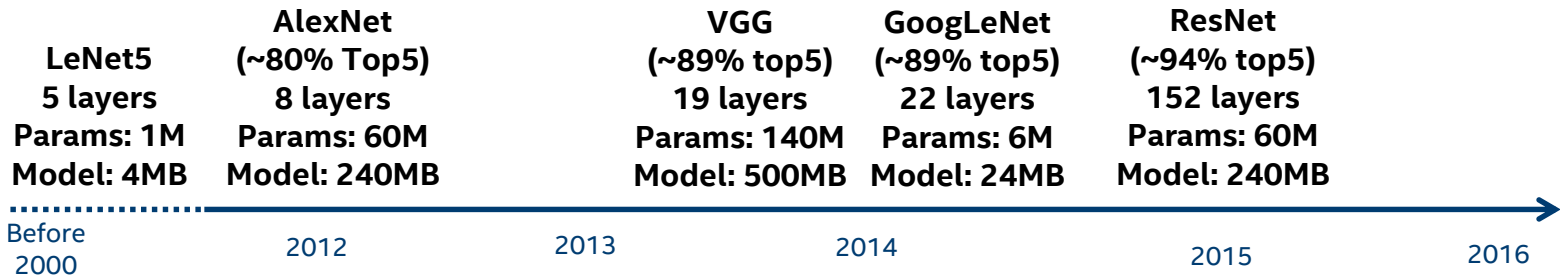
Compression

Compact network



All applicable for inference  
Some for training

Deeper  
More params?  
Larger model?



Spatially SparseCNN  
[CIFAR-10 winner '14]

BinaryConnect [NIPS'15]  
SparseCNN [CVPR'15]  
Pruning [NIPS'15]  
HashedNets [ICML'15]

XNORNet  
TernaryConnect [ICLR'16]  
DeepComp [ICLR'16]  
SqueezeNet

# DNNs Evolving Rapidly

**SqueezeNet + DeepCompression:**  
 6-bit, 20-50% sparse  
 AlexNet accuracy, ~500x smaller (0.5MB)

Many efforts to improve efficiency

Batching

Reduce bitwidth

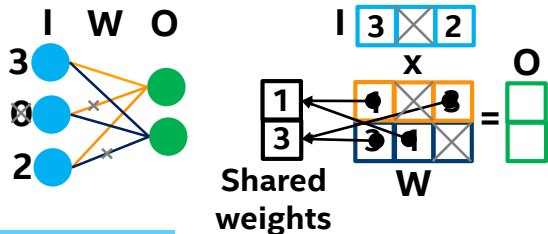
Sparse weights

Sparse activations

Compression

Compact network

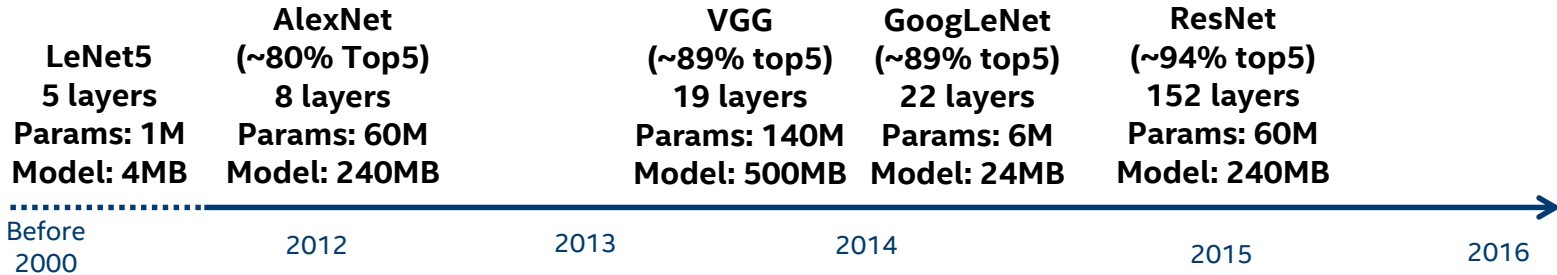
All applicable for inference  
 Some for training



- BinaryConnect [NIPS'15]
- SparseCNN [CVPR'15]
- Pruning [NIPS'15]
- HashedNets [ICML'15]
- XNORnet
- TernaryConnect [ICLR'16]
- DeepComp [ICLR'16]
- SqueezeNet

Spatially SparseCNN [CIFAR-10 winner '14]

Deeper  
 More params?  
 Larger model?



# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

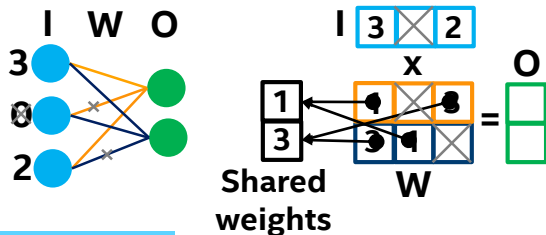
Reduce bitwidth

Sparse weights

Sparse activations

Compression

Compact network



All applicable for inference  
Some for training

**SqueezeNet + DeepCompression:**  
6-bit, 20-50% sparse  
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

BinaryConnect  
[NIPS'15]

SparseCNN  
[CVPR'15]

Pruning  
[NIPS'15]

HashedNets  
[ICML'15]

XNORnet

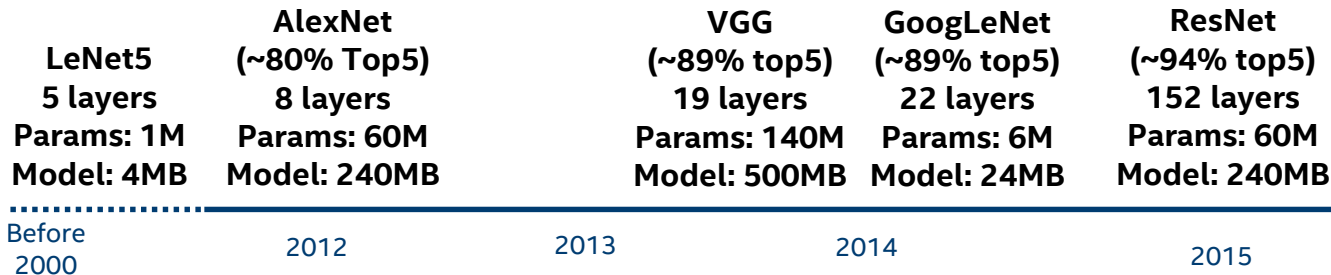
TernaryConnect  
[ICLR'16]

DeepComp  
[ICLR'16]

SqueezeNet

Spatially  
SparseCNN  
[CIFAR-10  
winner '14]

Deeper  
More params?  
Larger model?





# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

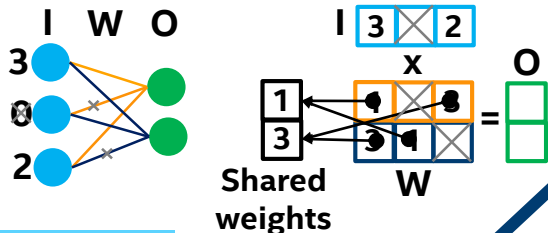
Sparse weights

Sparse activations

Compression

Compact network

All applicable for inference  
Some for training



**SqueezeNet + DeepCompression:**  
6-bit, 20-50% sparse  
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

BinaryConnect  
[NIPS'15]

SparseCNN  
[CVPR'15]

Pruning  
[NIPS'15]

HashedNets  
[ICML'15]

XNORnet

TernaryConnect  
[ICLR'16]

DeepComp  
[ICLR'16]

SqueezeNet

Spatially  
SparseCNN  
[CIFAR-10  
winner '14]

~3.5  
years

Deeper  
More params?  
Larger model?

LeNet5  
5 layers  
Params: 1M  
Model: 4MB

AlexNet  
(~80% Top5)  
8 layers  
Params: 60M  
Model: 240MB

VGG  
(~89% top5)  
19 layers  
Params: 140M  
Model: 500MB

GoogLeNet  
(~89% top5)  
22 layers  
Params: 6M  
Model: 24MB

ResNet  
(~94% top5)  
152 layers  
Params: 60M  
Model: 240MB

Before  
2000

2012

2013

2014

2015

2016

# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

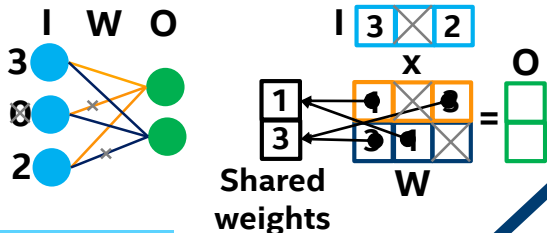
Sparse weights

Sparse activations

Compression

Compact network

All applicable for inference  
Some for training



Shared weights

~3.5 years

**SqueezeNet + DeepCompression:**  
6-bit, 20-50% sparse  
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

TernaryNet (2-bit, 50% sparse) → ~1% ResNet

BinaryConnect  
[NIPS'15]

XNORnet

SparseCNN  
[CVPR'15]

TernaryConnect  
[ICLR'16]

Pruning  
[NIPS'15]

DeepComp  
[ICLR'16]

HashedNets  
[ICML'15]

SqueezeNet

Spatially  
SparseCNN  
[CIFAR-10  
winner '14]

Deeper  
More params?  
Larger model?

LeNet5  
5 layers  
Params: 1M  
Model: 4MB  
.....  
Before  
2000

AlexNet  
(~80% Top5)  
8 layers  
Params: 60M  
Model: 240MB

2012

2013

VGG  
(~89% top5)  
19 layers  
Params: 140M  
Model: 500MB

2014

GoogLeNet  
(~89% top5)  
22 layers  
Params: 6M  
Model: 24MB

2015

ResNet  
(~94% top5)  
152 layers  
Params: 60M  
Model: 240MB

2016

# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

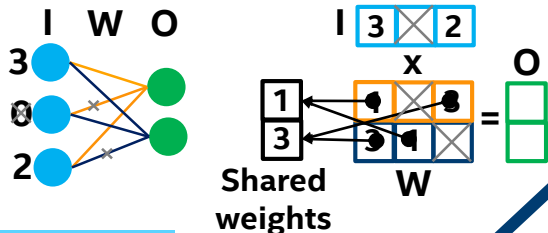
Sparse weights

Sparse activations

Compression

Compact network

All applicable for inference  
Some for training



Deeper  
More params?  
Larger model?

LeNet5  
5 layers  
Params: 1M  
Model: 4MB  
Before 2000

AlexNet  
(~80% Top5)  
8 layers  
Params: 60M  
Model: 240MB  
2012

2013

VGG  
(~89% top5)  
19 layers  
Params: 140M  
Model: 500MB  
2014

GoogLeNet  
(~89% top5)  
22 layers  
Params: 6M  
Model: 24MB  
2014

ResNet  
(~94% top5)  
152 layers  
Params: 60M  
Model: 240MB  
2015

2016

~3.5 years

Spatially SparseCNN  
[CIFAR-10 winner '14]

BinaryConnect  
[NIPS'15]  
SparseCNN  
[CVPR'15]  
Pruning  
[NIPS'15]  
HashedNet  
[ICML'15]

~1 year

XNORnet  
TernaryConnect  
[ICLR'16]  
DeepComp  
[ICLR'16]  
SqueezeNet

SqueezeNet + DeepCompression:  
6-bit, 20-50% sparse  
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

TernaryNet (2-bit, 50% sparse) → ~1% ResNet

# DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth

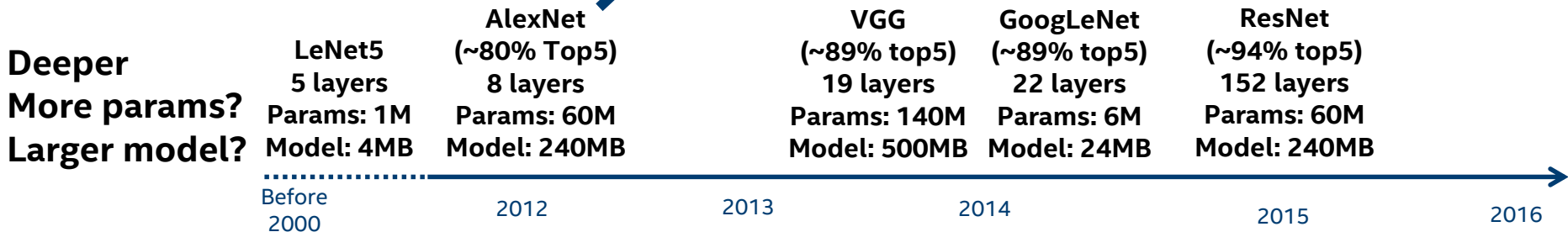
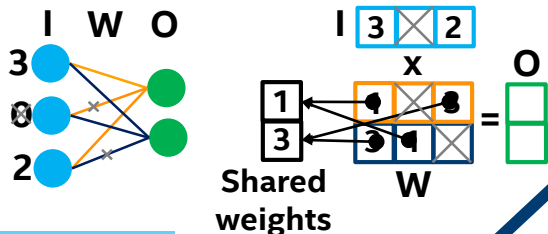
Sparse weights

Sparse activations

Compression

Compact network

All applicable for inference  
Some for training



**SqueezeNet + DeepCompression:**  
6-bit, 20-50% sparse  
AlexNet accuracy, ~500x smaller (0.5MB)

**XNORnet (1-bit) → ~2% AlexNet**

**TernaryNet (2-bit, 50% sparse) → ~1% ResNet**

Spatially SparseCNN [CIFAR-10 winner '14]

BinaryConnect [NIPS'15]  
SparseCNN [CVPR'15]  
Pruning [NIPS'15]  
HashedNet [ICML'15]

XNORnet [ICLR'16]  
TernaryConnect [ICLR'16]  
DeepComp [ICLR'16]  
SqueezeNet [ICML'15]

~1 year

~3.5 years

**Next-gen DNNs: more irregular with custom data types**

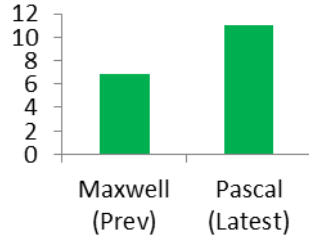


So far, high-end GPUs are gaining traction in DNNs

# So far, high-end GPUs are gaining traction in DNNs

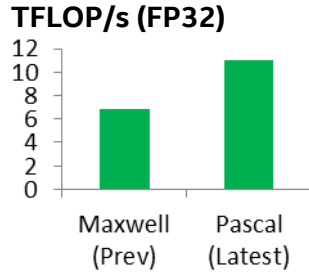
Lots of FLOP/s

TFLOP/s (FP32)

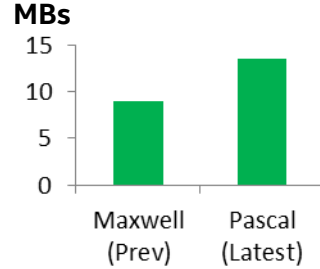


# So far, high-end GPUs are gaining traction in DNNs

## Lots of FLOP/s

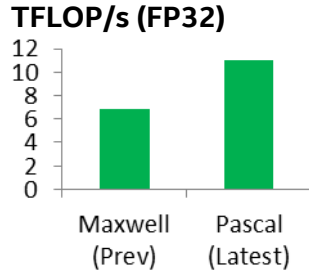


## On-chip RAMs

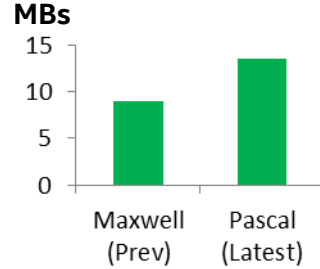


# So far, high-end GPUs are gaining traction in DNNs

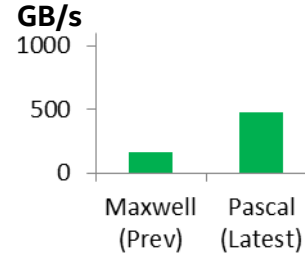
## Lots of FLOP/s



## On-chip RAMs



## Memory BW

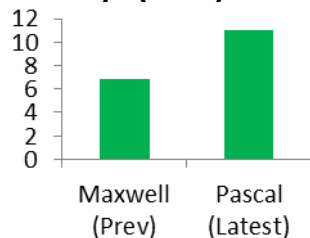




# So far, high-end GPUs are gaining traction in DNNs

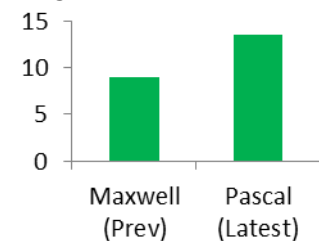
## Lots of FLOP/s

TFLOP/s (FP32)



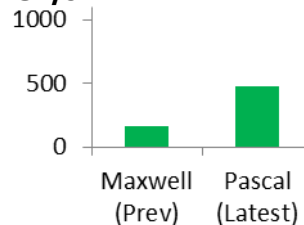
## On-chip RAMs

MBs



## Memory BW

GB/s



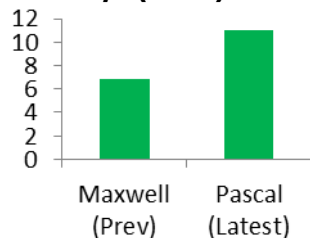
## But, Power Hungry

**~200+ W**

# So far, high-end GPUs are gaining traction in DNNs

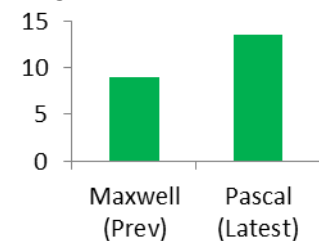
## Lots of FLOP/s

TFLOP/s (FP32)



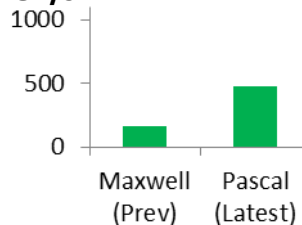
## On-chip RAMs

MBs



## Memory BW

GB/s



## But, Power Hungry

**~200+ W**

## Programmable



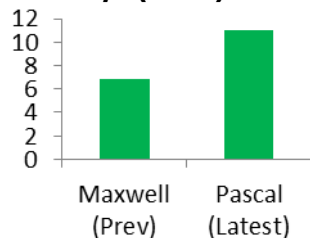
cuBLAS

cuDNN

# So far, high-end GPUs are gaining traction in DNNs

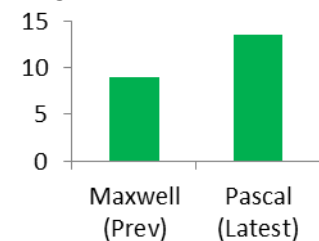
## Lots of FLOP/s

TFLOP/s (FP32)



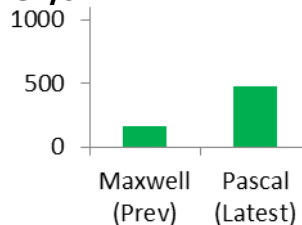
## On-chip RAMs

MBs



## Memory BW

GB/s



## But, Power Hungry

**~200+ W**

## Programmable



cuBLAS

cuDNN

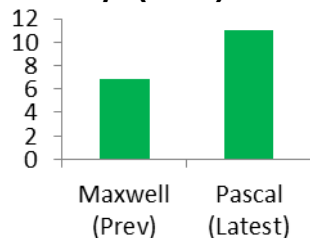
## Integrated



# So far, high-end GPUs are gaining traction in DNNs

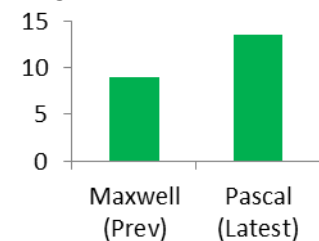
## Lots of FLOP/s

TFLOP/s (FP32)



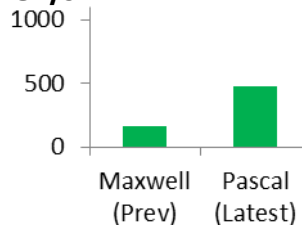
## On-chip RAMs

MBs



## Memory BW

GB/s



## But, Power Hungry

**~200+ W**

## Programmable



cuBLAS

cuDNN

## Integrated

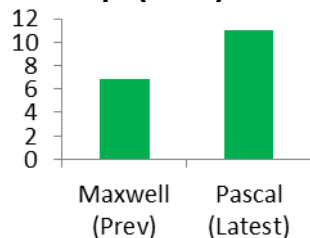


**High achievable FLOP/s on GEMM**  
(for big enough regular dense matrix, using native data types – FP32, FP16, INT8)

# So far, high-end GPUs are gaining traction in DNNs

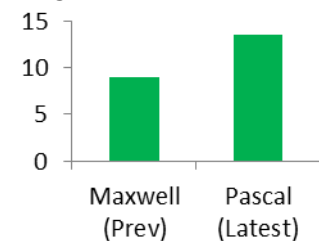
## Lots of FLOP/s

TFLOP/s (FP32)



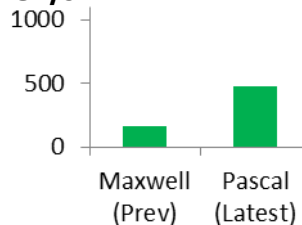
## On-chip RAMs

MBs



## Memory BW

GB/s



## But, Power Hungry

~200+ W

## Programmable



cuBLAS

cuDNN

## Integrated

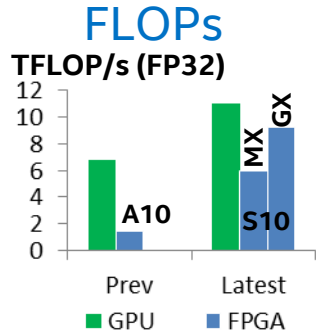


**High achievable FLOP/s on GEMM**  
(for big enough regular dense matrix, using native data types – FP32, FP16, INT8)

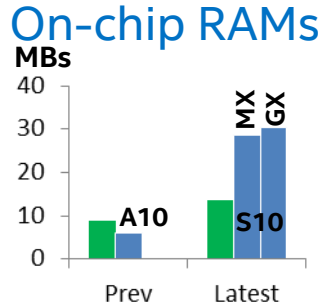
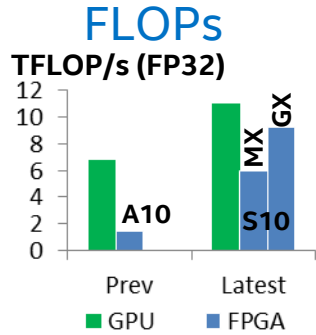
How much GPU's FLOPs can be utilized in next-gen DNNs?

Meanwhile, FPGAs becoming much more capable

# Meanwhile, FPGAs becoming much more capable

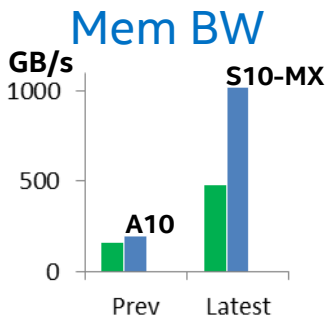
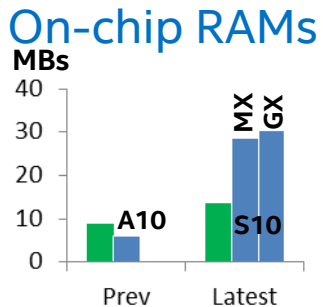
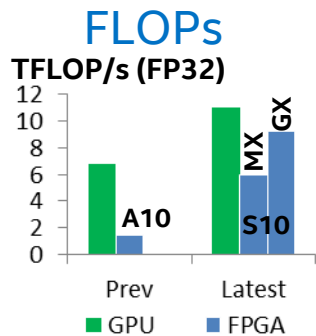


# Meanwhile, FPGAs becoming much more capable

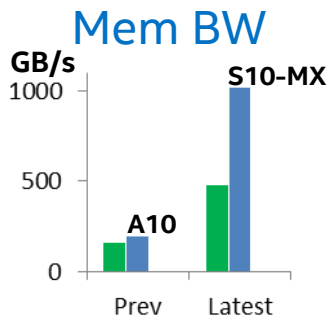
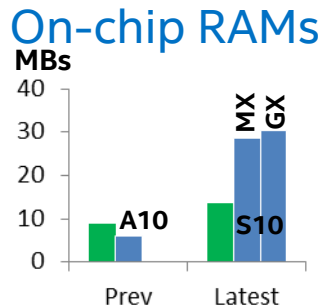
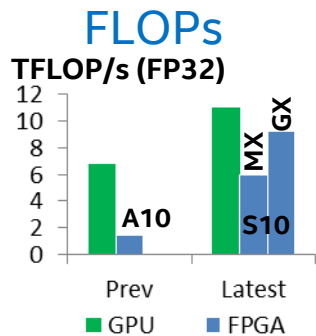




# Meanwhile, FPGAs becoming much more capable



# Meanwhile, FPGAs becoming much more capable

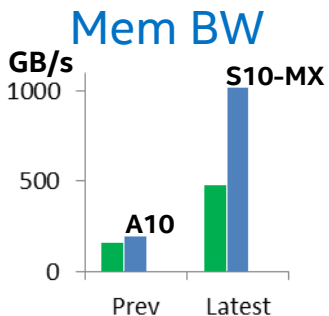
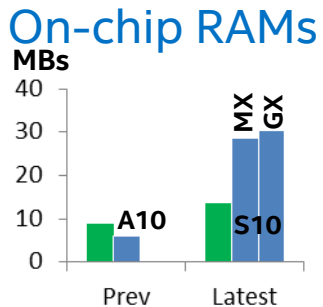
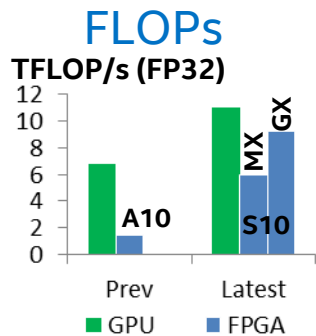


Higher Freq

**HyperFlex™**  
ARCHITECTURE

2x core frequency

# Meanwhile, FPGAs becoming much more capable



Higher Freq

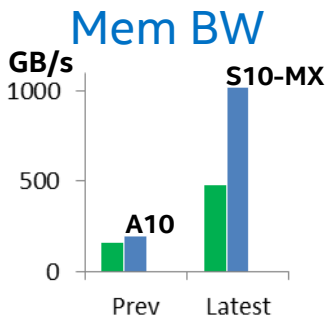
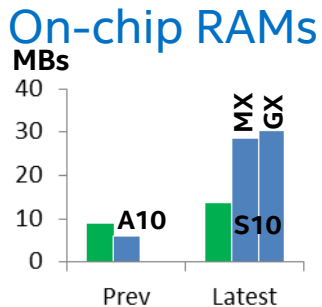
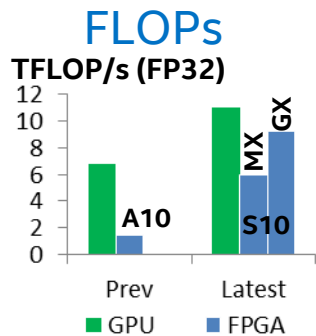
**HyperFlex™**  
ARCHITECTURE

2x core frequency

Power Efficient

10s-100s W

# Meanwhile, FPGAs becoming much more capable



Higher Freq

**HyperFlex™**  
ARCHITECTURE

2x core frequency

Power Efficient

10s-100s W

## High-Level Programming

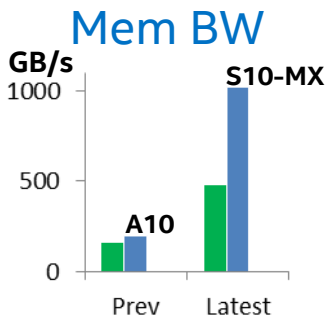
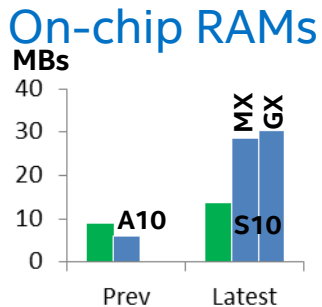
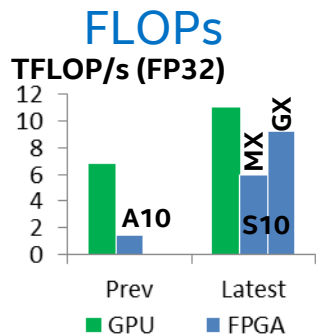


OpenCL



A++

# Meanwhile, FPGAs becoming much more capable



Higher Freq

Power Efficient

**HyperFlex™**  
ARCHITECTURE

**10s-100s W**

**2x core frequency**

## High-Level Programming

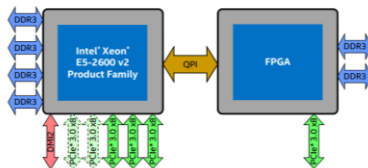


OpenCL



**A++**

## More Integrated



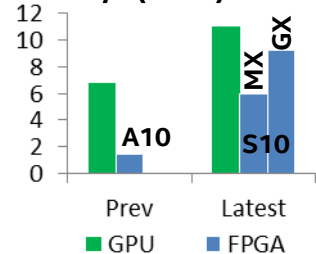
**Xeon+FPGA**



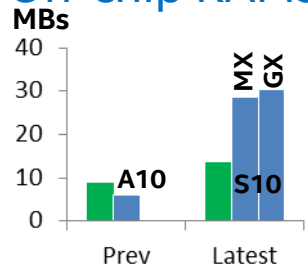
**Discrete cards**

# Meanwhile, FPGAs becoming much more capable

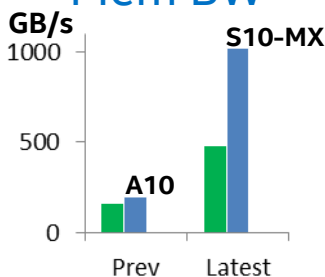
**FLOPs**  
TFLOP/s (FP32)



**On-chip RAMs**  
MBs



**Mem BW**  
GB/s



**Higher Freq**

**HyperFlex™**  
ARCHITECTURE

**2x core frequency**

**Power Efficient**

**10s-100s W**

## High-Level Programming

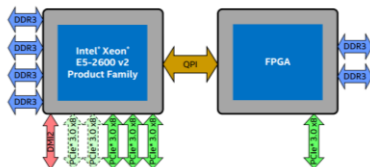


OpenCL



**A++**

## More Integrated



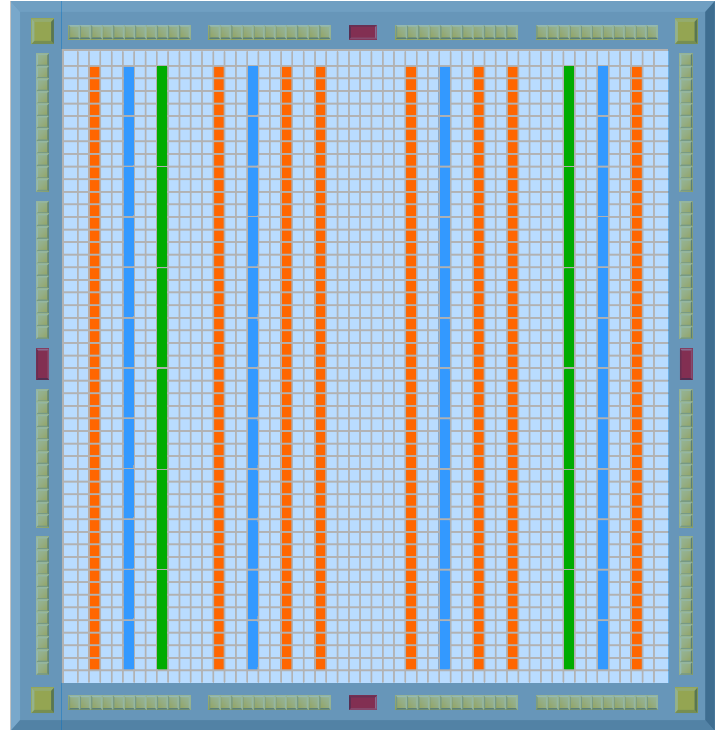
**Xeon+FPGA**



**Discrete cards**

**Upcoming Stratix 10 will be more competitive to GPUs**

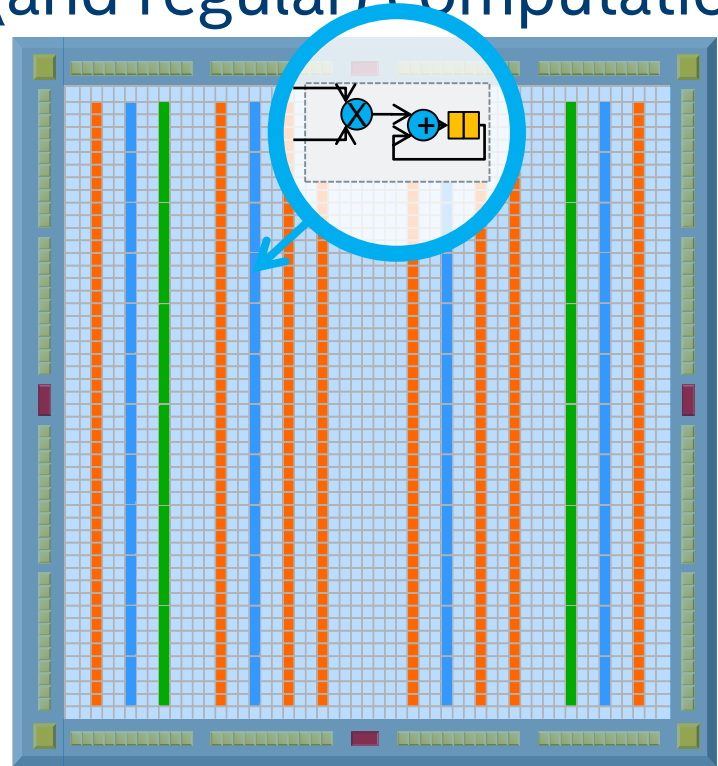
# FPGA fabric is great for irregular (and regular) computation



Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)



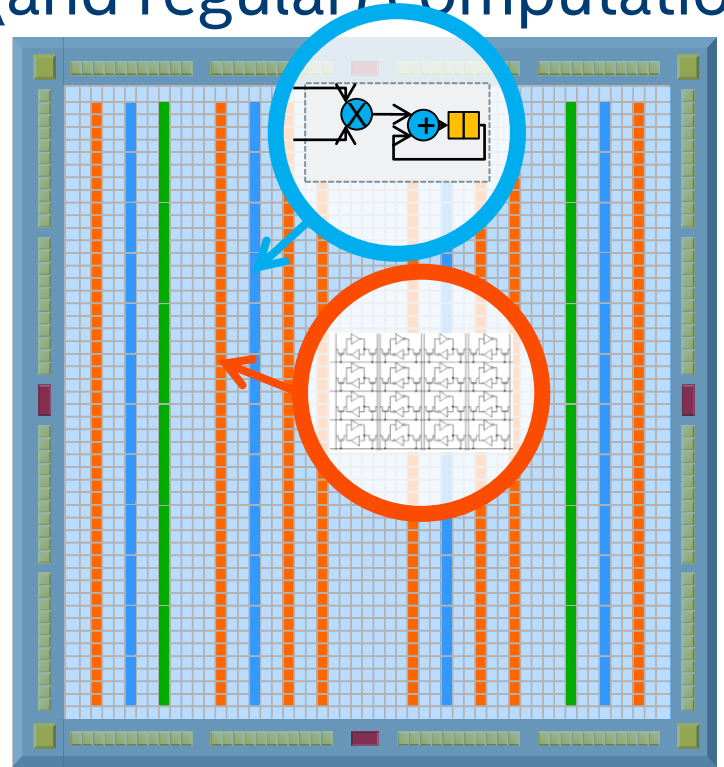
Figures courtesy of Gordon Chiu



# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)



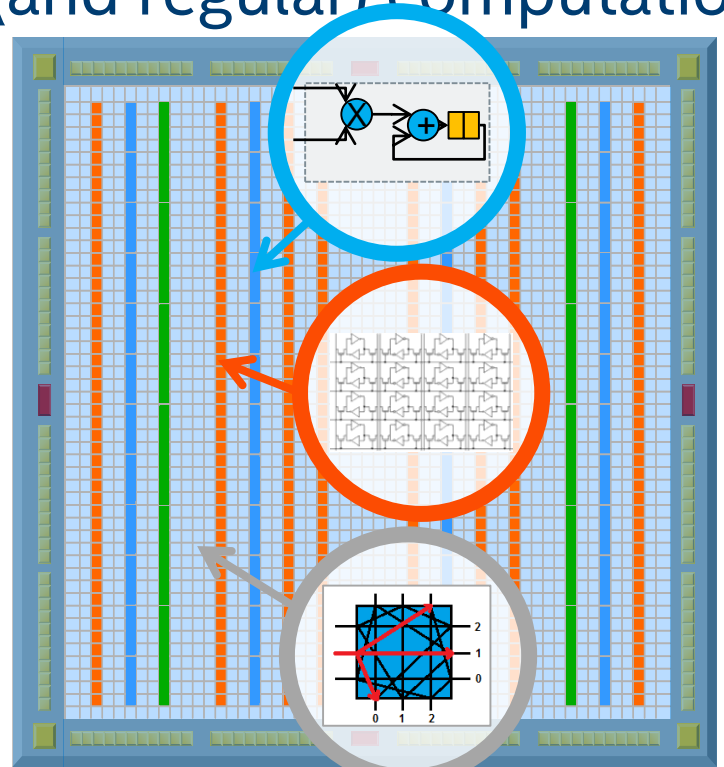
Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing



Figures courtesy of Gordon Chiu

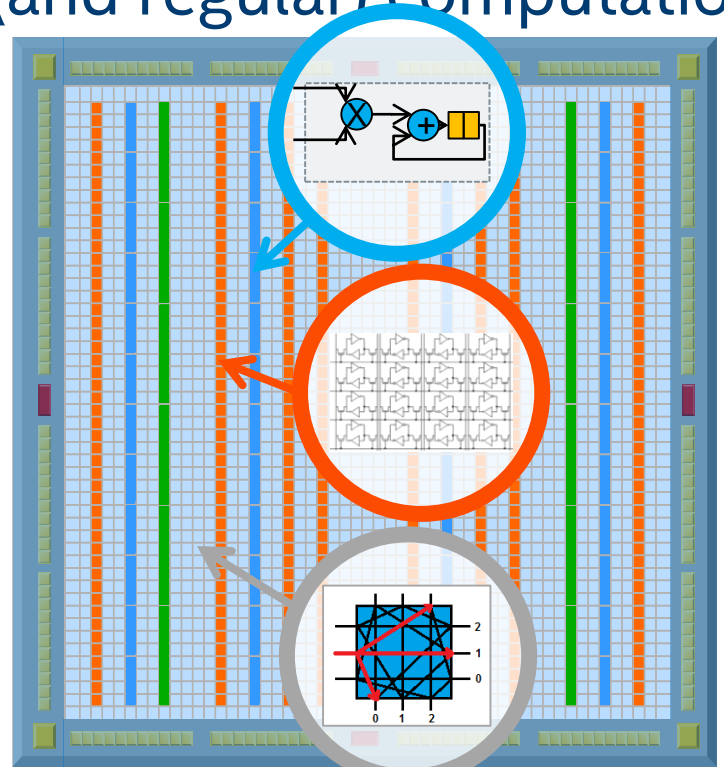
# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing

Extreme degree of customizations



Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

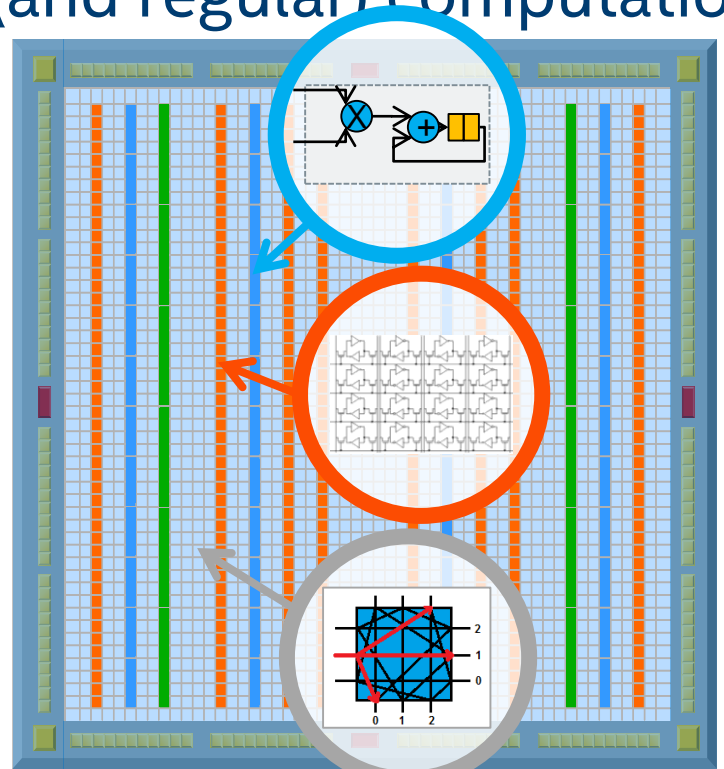
1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing

**Extreme degree of customizations**

Arbitrary bitwidth, mix bitwidths, etc



Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

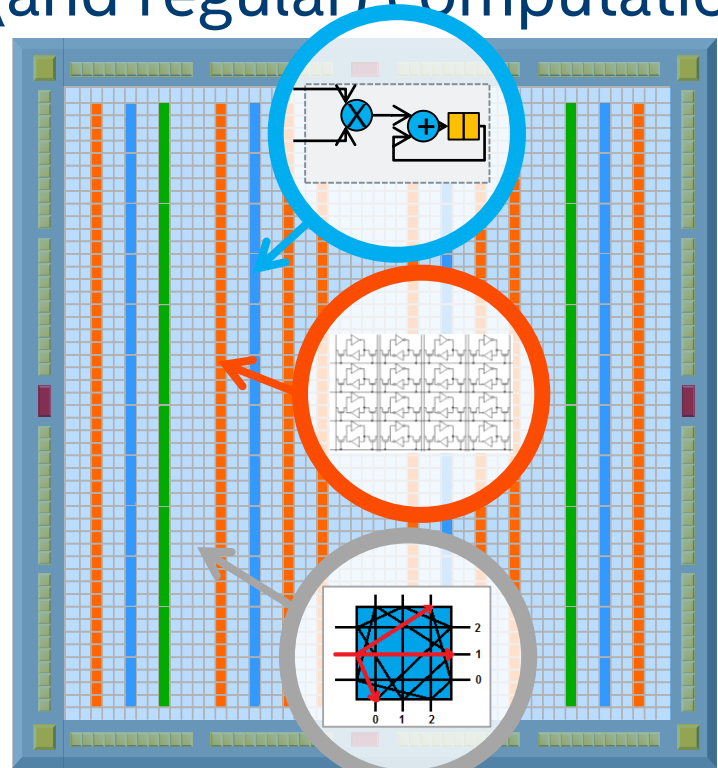
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing

**Extreme degree of customizations**

Arbitrary bitwidth, mix bitwidths, etc

Arbitrary SRAMs compositions (spad, \$, fifo, ..)



Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

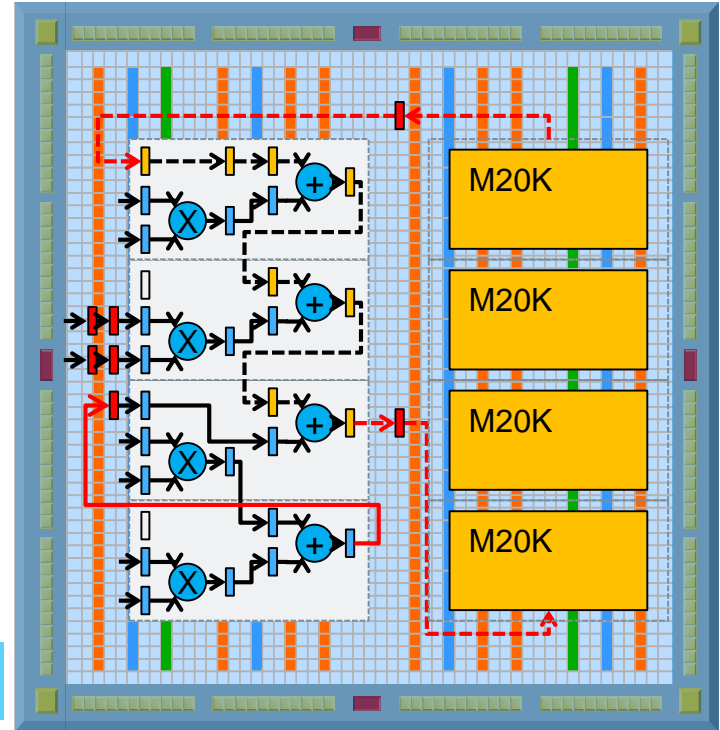
Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc

Arbitrary SRAMs compositions (spad, \$, fifo, ..)

Arbitrary DNN architectures



Figures courtesy of Gordon Chiu

# FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

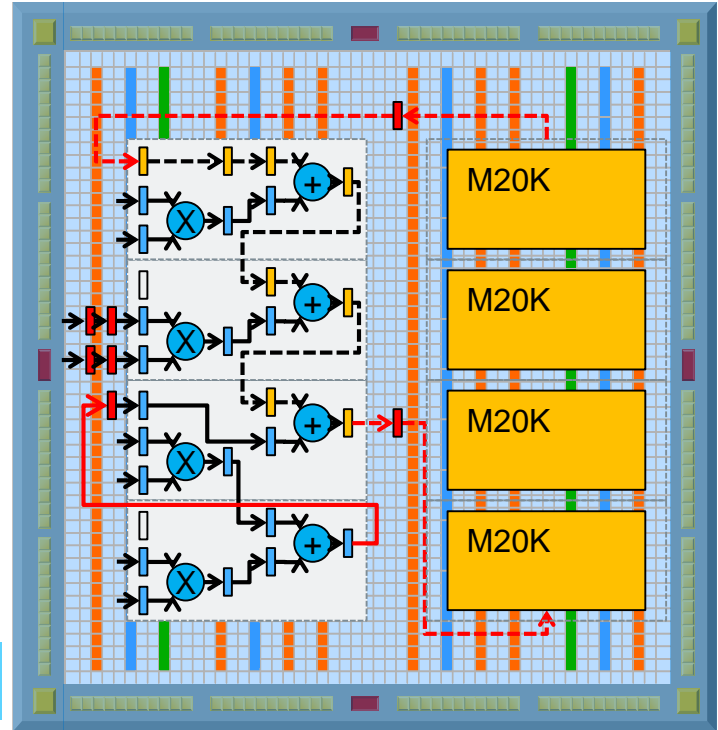
Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc

Arbitrary SRAMs compositions (spad, \$, fifo, ..)

Arbitrary DNN architectures



Figures courtesy of Gordon Chiu

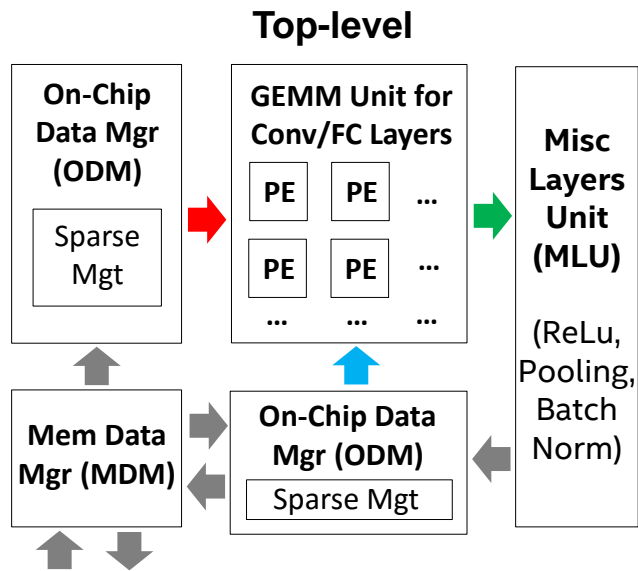
FPGAs well positioned for deep learning

# This work: compare high-end GPU vs. FPGA for DNNs

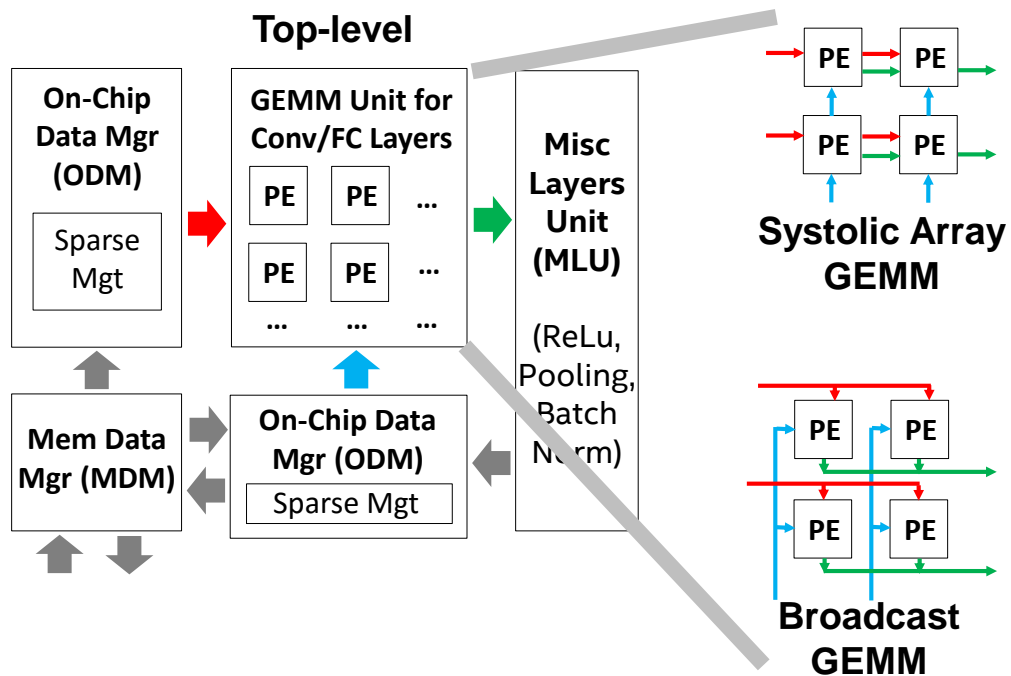
- Collected measurements on latest high-end GPU (Titan X Pascal)
- Compared against projections for upcoming Stratix 10 FPGA
- Study 1: compare various GEMMs used by next-gen DNNs
- Study 2: case study on ternarized ResNet



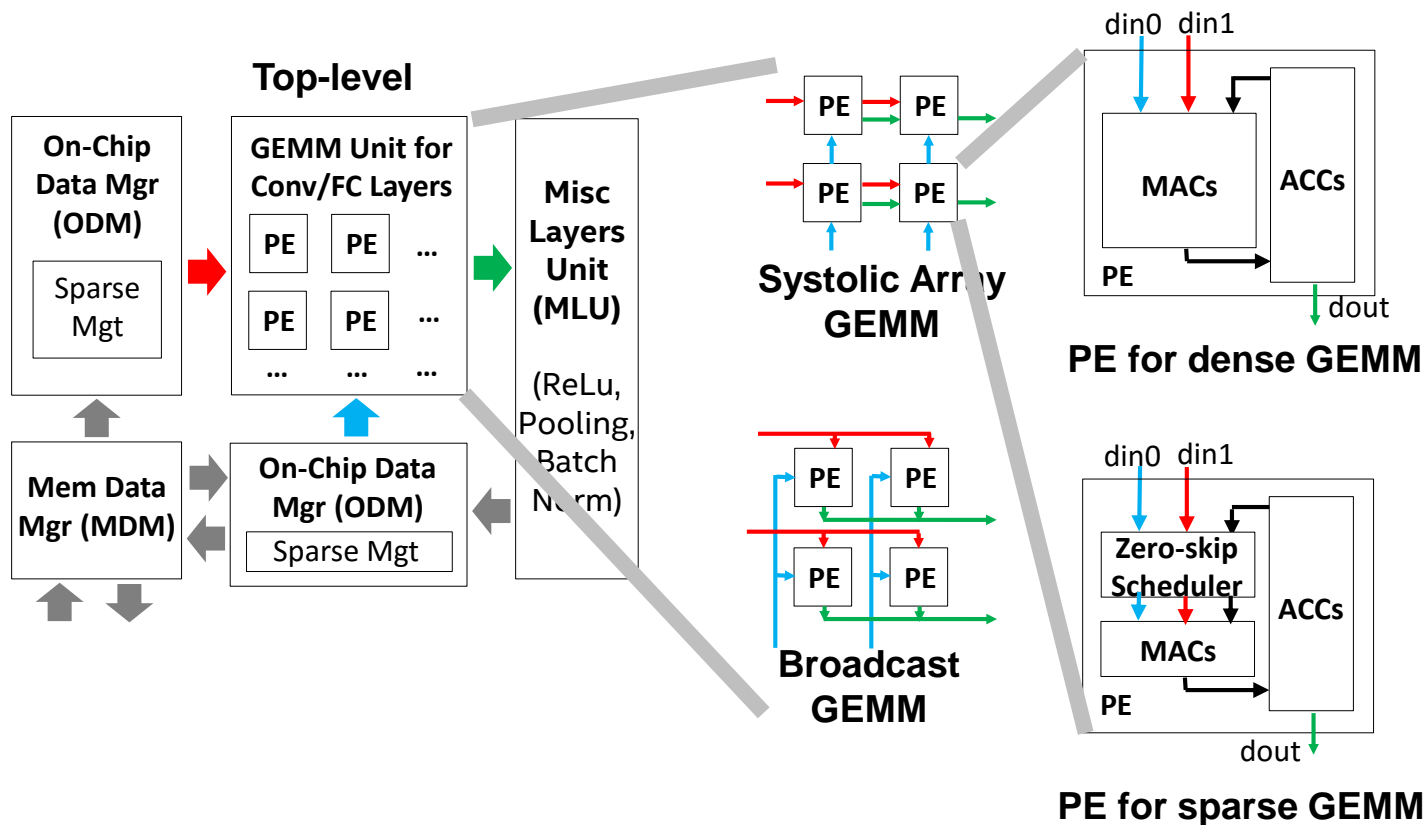
# DNN accelerator template for FPGA used in our studies



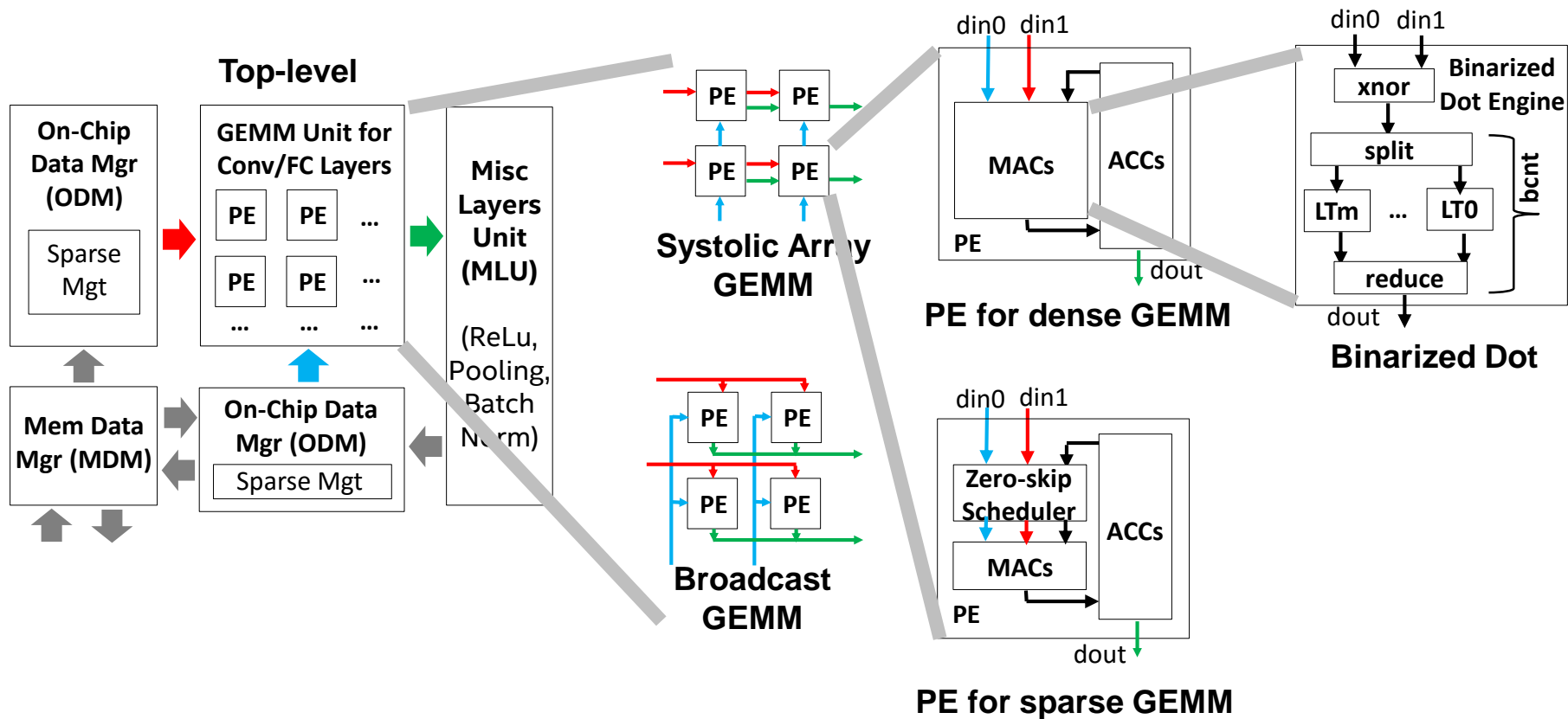
# DNN accelerator template for FPGA used in our studies



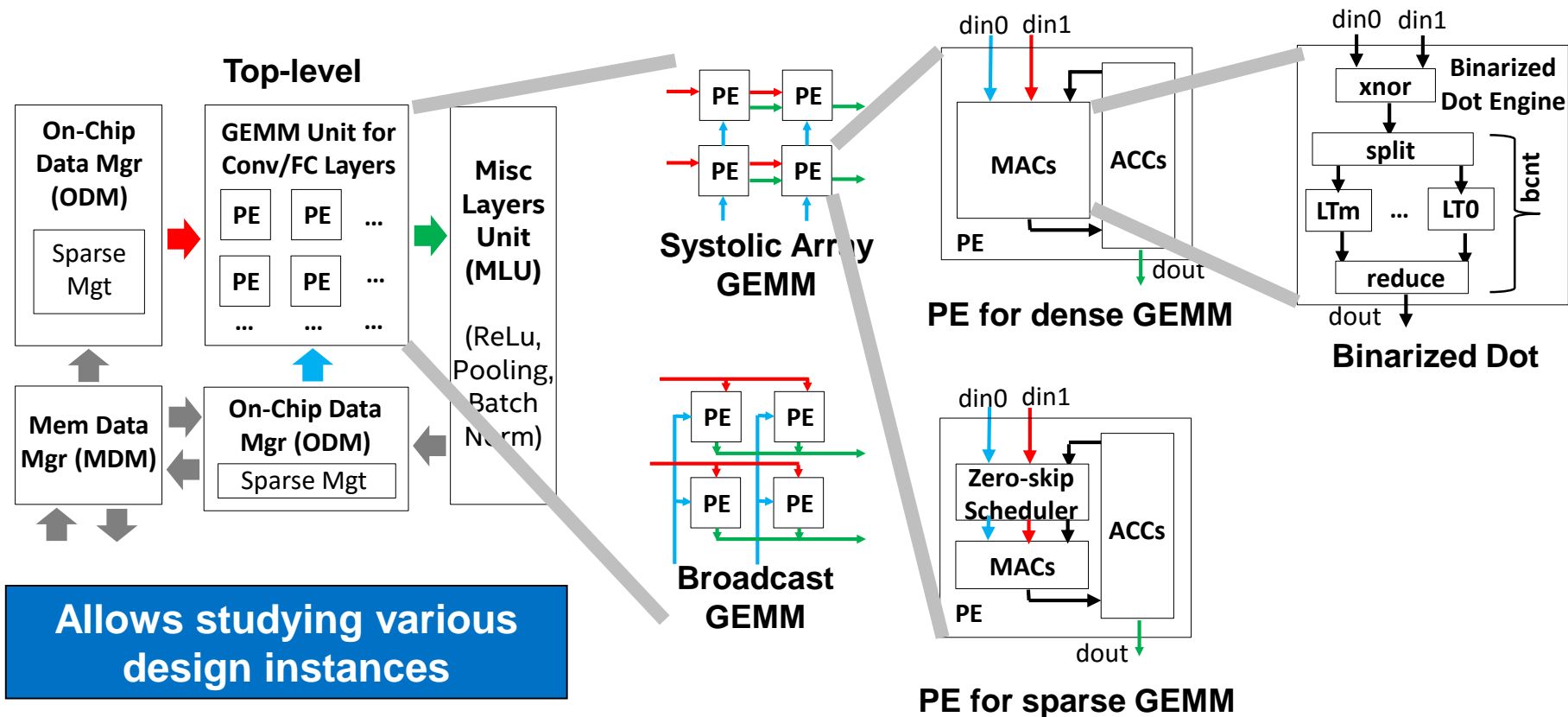
# DNN accelerator template for FPGA used in our studies



# DNN accelerator template for FPGA used in our studies



# DNN accelerator template for FPGA used in our studies



# Methodology

## FPGAs and GPU under study

	Arria 10 1150 FPGA	Stratix 10 2800 FPGA	TitanX Pascal GPU
Peak FP32 TFLOPs	1.36	9.2	11
On-chip RAMs	6.6 MB (M20Ks)	28.6 MB (M20Ks)	13.5 MB (RF, SM, L2)
Memory BW	Assume same as Titan X	Assume same as Titan X	480 GB/s

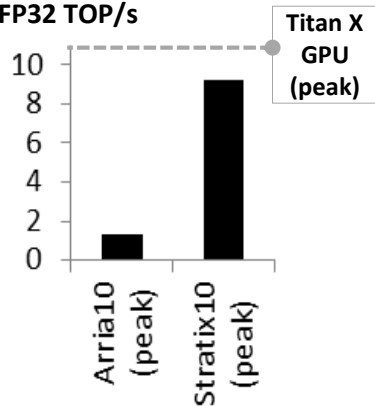
## Evaluation

- GPU: used known library (cuBLAS) or framework (Torch with cuDNN)
- FPGA: estimated using Quartus and PowerPlay
  - For Stratix 10, we use Quartus Early Beta release. Note that its quality is not necessarily reflective of future more mature releases of Quartus for Stratix 10

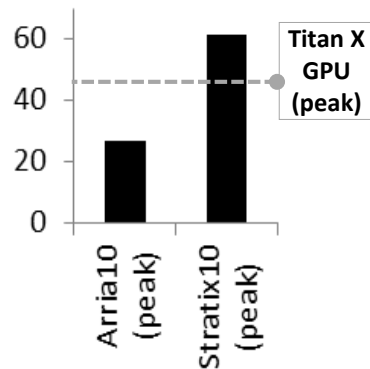
# Study 1 results: Dense GEMM

## “Classic” FP32 GEMM

FP32 TOP/s

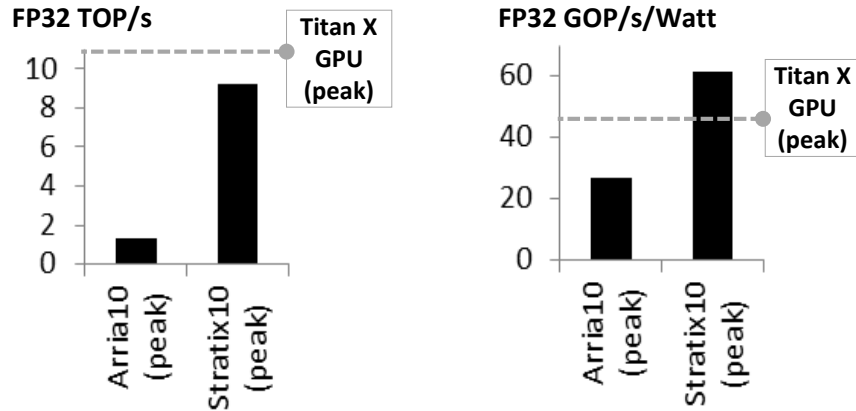


FP32 GOP/s/Watt



# Study 1 results: Dense GEMM

## “Classic” FP32 GEMM

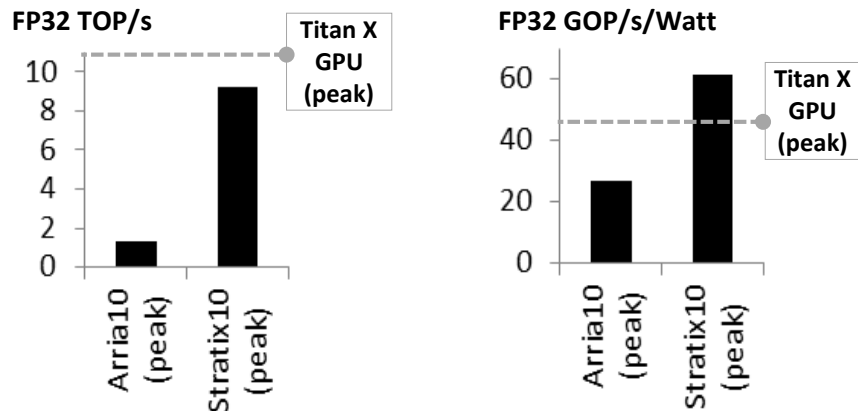


**For “classic” FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt**

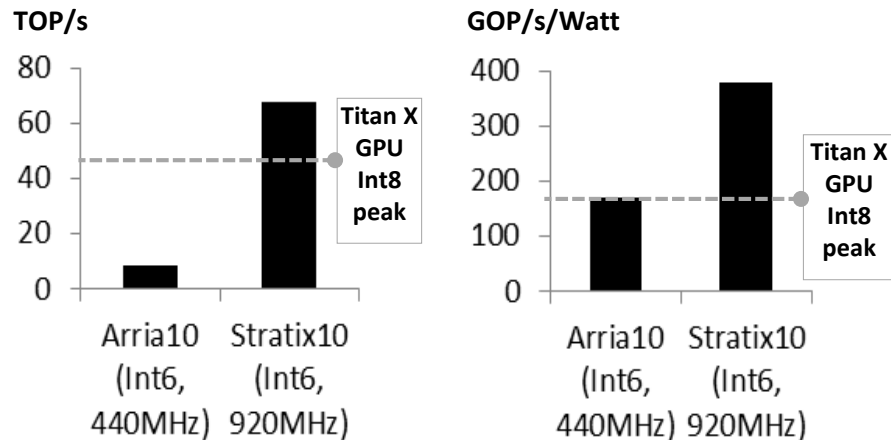


# Study 1 results: Dense GEMM

## “Classic” FP32 GEMM



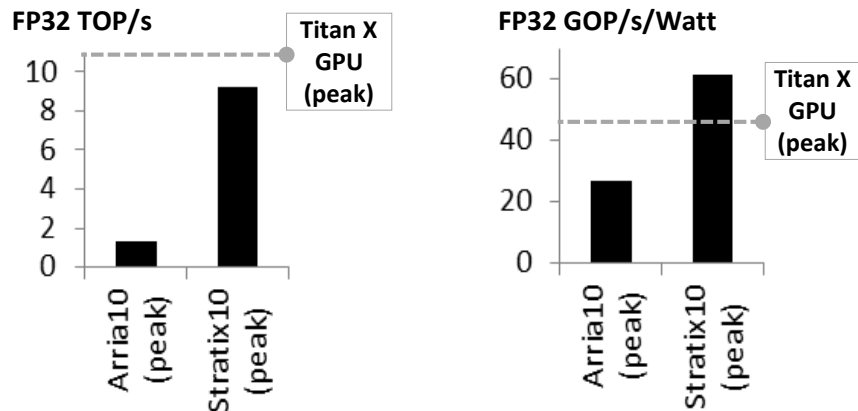
## Low precision (INT6, INT8) GEMM



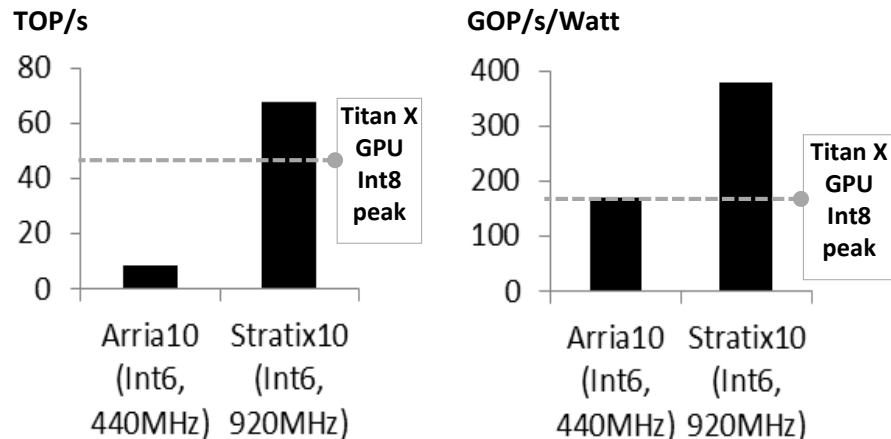
For “classic” FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt

# Study 1 results: Dense GEMM

## “Classic” FP32 GEMM



## Low precision (INT6, INT8) GEMM



For “classic” FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt

For low precision 6bit, S10 FPGA can offer better performance, and even better perf/watt.

# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

Matrix x Vector, with +1 or -1

$$\begin{array}{c|c|c|c} \mathbf{I} & & & \\ \hline -1 & & & \\ \hline +1 & & & \\ \hline +1 & & & \\ \hline \end{array} \times \begin{array}{c|c|c|c} \mathbf{W} & & & \\ \hline -1 & +1 & +1 & \\ \hline -1 & -1 & -1 & \\ \hline +1 & +1 & -1 & \\ \hline \end{array} = \begin{array}{c} (-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1) \\ (-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1) \\ (-1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1) \end{array} \begin{array}{c} \mathbf{O} \\ \hline 3 \\ \hline -1 \\ \hline -1 \\ \hline \end{array}$$

# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

Matrix x Vector, with +1 or -1

I	W		O
-1	-1 +1 +1	$(-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1)$	3
+1	-1 -1 -1	$(-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1)$	-1
+1	+1 +1 -1	$(-1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1)$	-1

Binarized Matrix x Vector

I	W		O
0	0 1 1	$\text{bcnt}(\text{xnor}(011, 011))$	3
1	0 0 0	$= \text{bcnt}(\text{xnor}(011, 000))$	-1
1	1 1 0	$\text{bcnt}(\text{xnor}(011, 110))$	-1

# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

Matrix x Vector, with +1 or -1

$$\begin{array}{c|ccc} \mathbf{I} & & & \\ \hline -1 & & & \\ \hline +1 & & & \\ \hline +1 & & & \\ \hline \end{array} \times \begin{array}{c|ccc} \mathbf{W} & & & \\ \hline -1 & +1 & +1 & \\ \hline -1 & -1 & -1 & \\ \hline +1 & +1 & -1 & \\ \hline \end{array} = \begin{array}{c|ccc} \mathbf{O} & & & \\ \hline (-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1) & & & \\ \hline (-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1) & = & -1 & \\ \hline (-1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1) & & & \\ \hline \end{array}$$

Binarized Matrix x Vector

$$\begin{array}{c|ccc} \mathbf{I} & & & \\ \hline 0 & & & \\ \hline 1 & & & \\ \hline 1 & & & \\ \hline \end{array} \times \begin{array}{c|ccc} \mathbf{W} & & & \\ \hline 0 & 1 & 1 & \\ \hline 0 & 0 & 0 & \\ \hline 1 & 1 & 0 & \\ \hline \end{array} = \begin{array}{c|ccc} \mathbf{O} & & & \\ \hline \text{bcnt}(\text{xnor}(011,011)) & & & \\ \hline \text{bcnt}(\text{xnor}(011,000)) & = & -1 & \\ \hline \text{bcnt}(\text{xnor}(011,110)) & & & \\ \hline \end{array}$$

Used optimized GPU code and  
FPGA design from [FPT'16]

# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

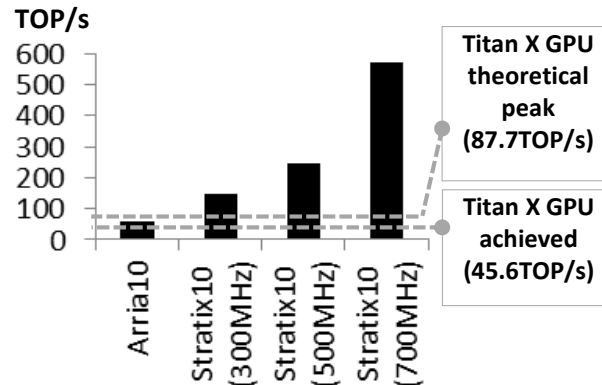
Matrix x Vector, with +1 or -1

I	W		O
-1	-1 +1 +1	$(-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1)$	3
+1	-1 -1 -1	$(-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1)$	-1
+1	+1 +1 -1	$(-1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1)$	-1

Binarized Matrix x Vector

I	W		O
0	0 1 1	$\text{bcnt}(\text{xnor}(011, 011))$	3
1	0 0 0	$= \text{bcnt}(\text{xnor}(011, 000))$	-1
1	1 1 0	$\text{bcnt}(\text{xnor}(011, 110))$	-1

Used optimized GPU code and FPGA design from [FPT'16]



# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

Matrix x Vector, with +1 or -1

I	W	O
-1	-1 +1 +1	3
+1	-1 -1 -1	-1
+1	+1 +1 -1	-1

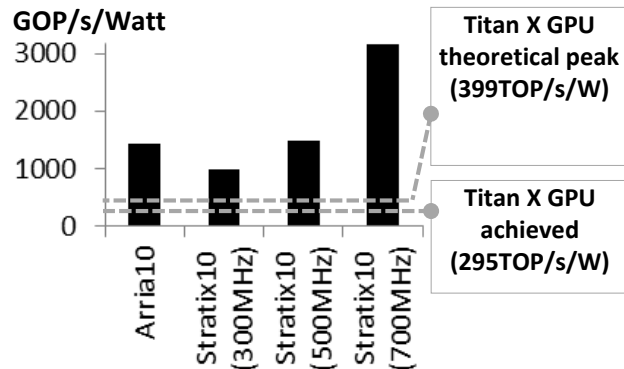
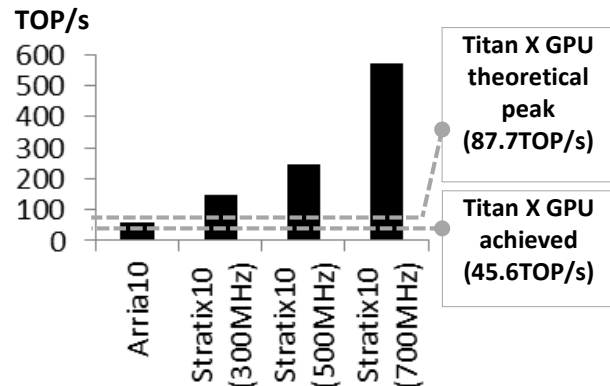
$$\begin{aligned}
 &(-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1) \\
 &= (-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1) \\
 &= (-1.1) + (1.1) + (1.1) \\
 &= (-1.1) + (1.1) + (1.1)
 \end{aligned}$$

Binarized Matrix x Vector

I	W	O
0	0 1 1	3
1	0 0 0	-1
1	1 1 0	-1

$$\begin{aligned}
 &\text{bcnt}(\text{xnor}(011,011)) \\
 &= \text{bcnt}(\text{xnor}(011,000)) \\
 &= \text{bcnt}(\text{xnor}(011,110))
 \end{aligned}$$

Used optimized GPU code and FPGA design from [FPT'16]



# Study 1 results: Very low precision 1bit (Binarized) GEMM

## Neural networks with parameters of +1 or -1

Matrix x Vector, with +1 or -1

I	W	O
-1	-1 +1 +1	3
+1	-1 -1 -1	-1
+1	+1 +1 -1	-1

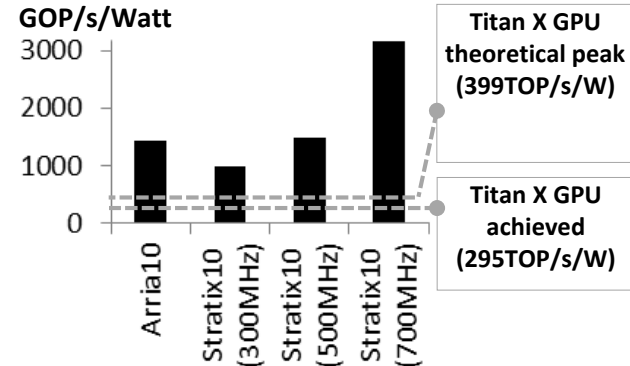
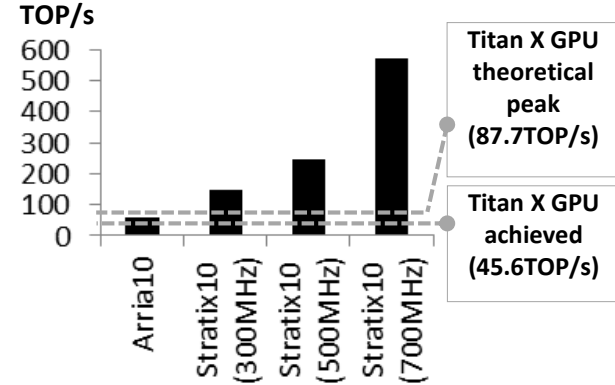
$$\begin{aligned}
 &(-1 \cdot -1) + (1 \cdot 1) + (1 \cdot 1) \\
 &= (-1 \cdot -1) + (1 \cdot -1) + (1 \cdot -1) \\
 &= (-1 \cdot 1) + (1 \cdot 1) + (1 \cdot -1)
 \end{aligned}$$

Binarized Matrix x Vector

I	W	O
0	0 1 1	3
1	0 0 0	-1
1	1 1 0	-1

$$\begin{aligned}
 &\text{bcnt}(\text{xnor}(011,011)) \\
 &= \text{bcnt}(\text{xnor}(011,000)) \\
 &= \text{bcnt}(\text{xnor}(011,110))
 \end{aligned}$$

Used optimized GPU code and FPGA design from [FPT'16]



**S10 FPGA can offer significantly better performance than Titan X GPU**



# Study 1 results: Sparse GEMM

## Sparse NN

I		W				O
0.1		0.2	0	0.1	$(0.1 \times 0.2) + (0 \times 0) + (0.2 \times 0.1)$	<input type="text"/>
0	x	0	0.3	0	$= (0.1 \times 0) + (0 \times 0.3) + (0.2 \times 0)$	<input type="text"/>
0.2		0.1	0	0	$(0.1 \times 0.1) + (0 \times 0) + (0.2 \times 0)$	<input type="text"/>

# Study 1 results: Sparse GEMM

## Sparse NN

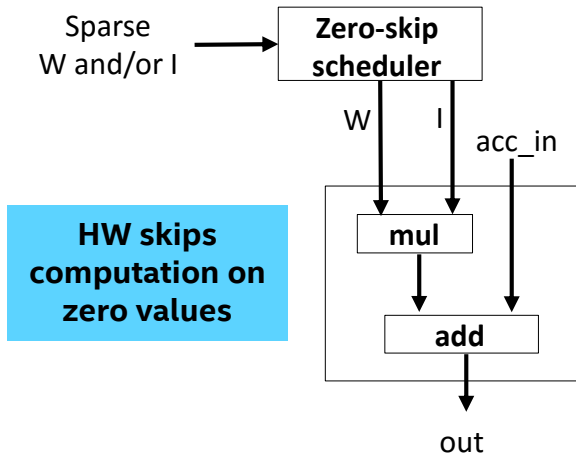
I	W						O
0.1	0.2	0	0.1	(0.1x0.2)+(0.2x0)	+ (0.2x0.1)		
0	0	0.3	0	(0x0)	+ (0x3)	+ (0x0)	
0.2	0.1	0	0	(0.1x0.1)	+ (0x0)	+ (0x0)	

# Study 1 results: Sparse GEMM

## Sparse NN

$$\begin{array}{c} \mathbf{I} \\ \begin{array}{|c|c|c|c|} \hline 0.1 & 0.2 & 0 & 0.1 \\ \hline 0 & 0.3 & 0 & \\ \hline 0.2 & 0.1 & 0 & 0 \\ \hline \end{array} \end{array} \times \begin{array}{c} \mathbf{W} \\ \begin{array}{|c|c|c|} \hline 0.2 & 0 & 0.1 \\ \hline 0 & 0.3 & 0 \\ \hline 0.1 & 0 & 0 \\ \hline \end{array} \end{array} = \begin{array}{c} \mathbf{O} \\ \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \end{array} \end{array}$$

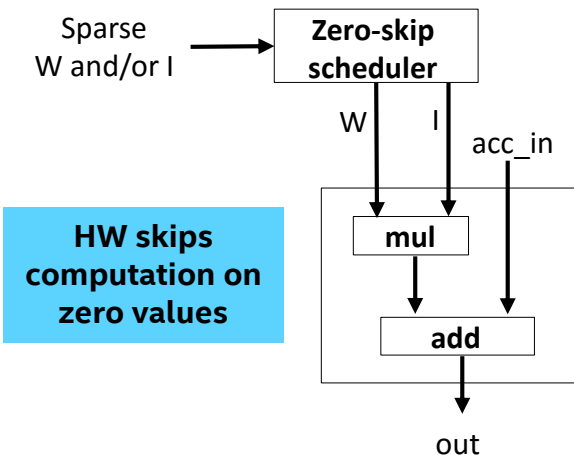
$(0.1 \times 0.2) + (0.2 \times 0.1) + (0.2 \times 0.1)$   
 $(0 \times 0) + (0 \times 3) + (0 \times 0)$   
 $(0.1 \times 0.1) + (0 \times 0) + (0 \times 0)$



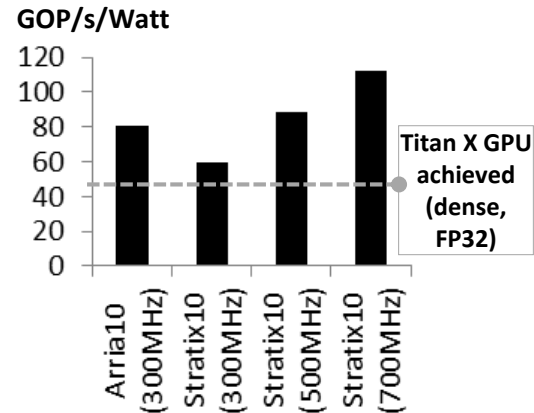
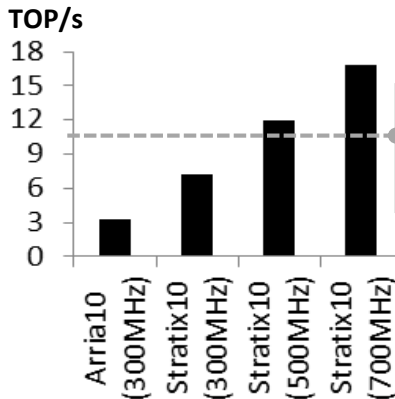
# Study 1 results: Sparse GEMM

## Sparse NN

$$\begin{array}{c} \mathbf{I} \\ \begin{array}{|c|} \hline 0.1 \\ \hline 0 \\ \hline 0.2 \\ \hline \end{array} \end{array} \times \begin{array}{c} \mathbf{W} \\ \begin{array}{|c|c|c|} \hline 0.2 & 0 & 0.1 \\ \hline 0 & 0.3 & 0 \\ \hline 0.1 & 0 & 0 \\ \hline \end{array} \end{array} = \begin{array}{c} (0.1 \times 0.2) + (0 \times 0.1) + (0.2 \times 0.1) \\ (0 \times 0) + (0 \times 0.3) + (0 \times 0) \\ (0.1 \times 0.1) + (0 \times 0) + (0 \times 0) \end{array} = \begin{array}{|c|} \hline \\ \hline \\ \hline \end{array} \mathbf{O}$$



## Sparse FP32 GEMM (at 85% sparsity based on AlexNet)

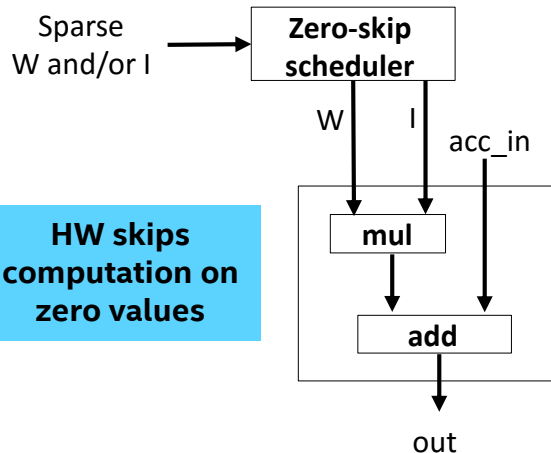


# Study 1 results: Sparse GEMM

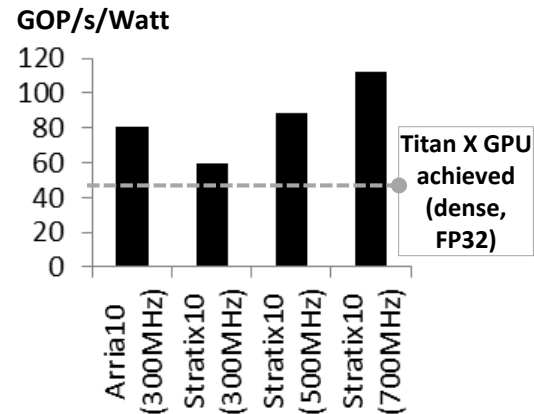
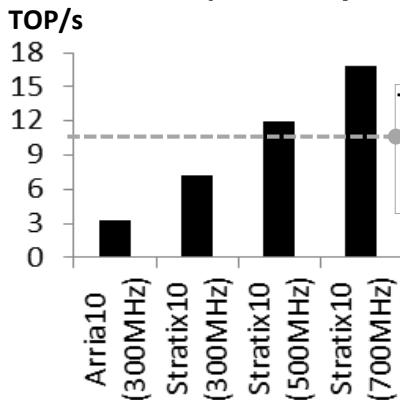
## Sparse NN

$$\begin{array}{c} \mathbf{I} \\ \begin{array}{|c|} \hline 0.1 \\ \hline 0 \\ \hline 0.2 \\ \hline \end{array} \end{array} \times \begin{array}{c} \mathbf{W} \\ \begin{array}{|c|c|c|} \hline 0.2 & 0 & 0.1 \\ \hline 0 & 0.3 & 0 \\ \hline 0.1 & 0 & 0 \\ \hline \end{array} \end{array} = \begin{array}{c} \mathbf{O} \\ \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \end{array} \end{array}$$

$(0.1 \times 0.2) + (0 \times 0.1) + (0.2 \times 0.1)$   
 $(0 \times 0.2) + (0 \times 0.3) + (0 \times 0)$   
 $(0.1 \times 0.1) + (0 \times 0) + (0 \times 0)$



## Sparse FP32 GEMM (at 85% sparsity based on AlexNet)



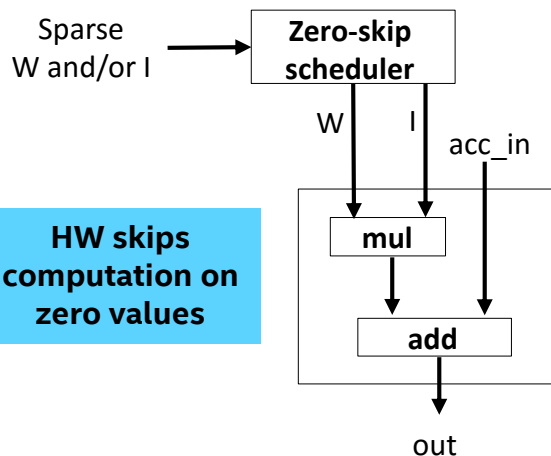
Also tried zero-skipping in GPU, which performed worse than dense. See paper.

# Study 1 results: Sparse GEMM

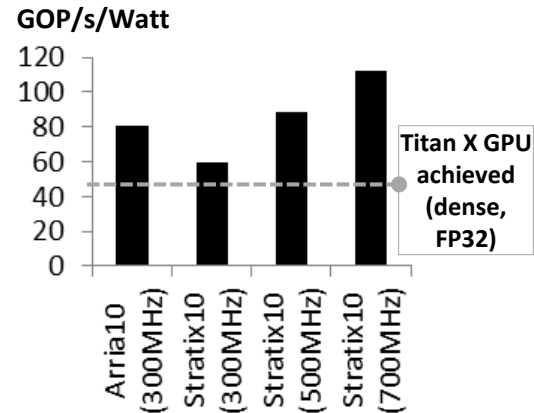
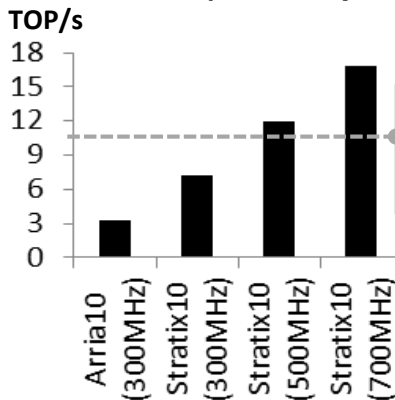
## Sparse NN

$$\begin{array}{c} \mathbf{I} \\ \begin{array}{|c|c|c|} \hline 0.1 & 0 & 0 \\ \hline 0 & 0.3 & 0 \\ \hline 0.2 & 0.1 & 0 \\ \hline \end{array} \end{array} \times \begin{array}{c} \mathbf{W} \\ \begin{array}{|c|c|c|} \hline 0.2 & 0 & 0.1 \\ \hline 0 & 0.3 & 0 \\ \hline 0.1 & 0 & 0 \\ \hline \end{array} \end{array} = \begin{array}{c} \mathbf{O} \\ \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \end{array}$$

$(0.1 \times 0.2) + (0 \times 0) + (0.2 \times 0.1)$   
 $(0 \times 0) + (0 \times 0.3) + (0 \times 0)$   
 $(0.1 \times 0.1) + (0 \times 0) + (0 \times 0)$



## Sparse FP32 GEMM (at 85% sparsity based on AlexNet)



Also tried zero-skipping in GPU, which performed worse than dense. See paper.

FPGA can offer better performance and efficiency than GPU, depending on sparsity and frequency achieved.

# Study 2: Ternary ResNet Case Study

# Study 2: Ternary ResNet Case Study

**Ternary NN: neural net with weights of +1,-1,0**

I		W		O
0.1		-1 0 +1	$(0.1 \times -1) + (0 \times 0) + (0.2 \times 1)$	<input type="text"/>
0	x	0 -1 -1	$(0.1 \times 0) + (0 \times -1) + (0.2 \times -1)$	<input type="text"/>
0.2		+1 0 0	$(0.1 \times 1) + (0 \times 0) + (0.2 \times 0)$	<input type="text"/>



# Study 2: Ternary ResNet Case Study

**Ternary NN: neural net with weights of +1,-1,0**

I		W		O
0.1		-1 0 +1	$(0.1x-1)+(0 \times 0)+(0.2x1)$	<input type="text"/>
0	x	0 -1 -1	$(0 \times 0)+(0 \times -1)+(0.2x-1)$	<input type="text"/>
0.2		+1 0 0	$(0.1x1)+(0 \times 0)+(0. \times 0)$	<input type="text"/>

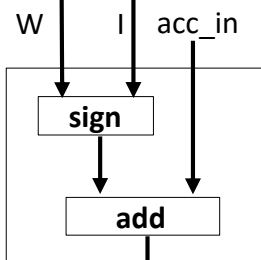
# Study 2: Ternary ResNet Case Study

## Ternary NN: neural net with weights of +1,-1,0

I	W		O
0.1	-1 0 +1	$(0.1x-1)+(0 \times 0)+(0.2x1)$	<input type="text"/>
0	0 -1 -1	$(0 \times 0)+(0 \times -1)+(0.2x-1)$	<input type="text"/>
0.2	+1 0 0	$(0.1x1)+(0 \times 0)+(0. \times 0)$	<input type="text"/>

Sparse I and/or ternary W

Zero-skip scheduler



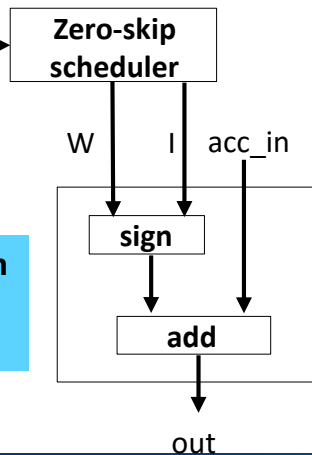
Skip computation on zero values, and no multiply

# Study 2: Ternary ResNet Case Study

## Ternary NN: neural net with weights of +1,-1,0

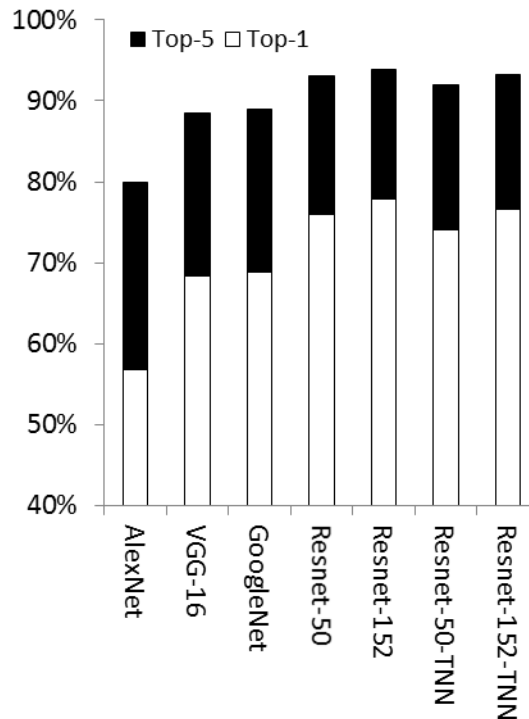
$$\begin{array}{|c|} \hline \mathbf{I} \\ \hline 0.1 \\ \hline 0 \\ \hline 0.2 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline \mathbf{W} \\ \hline -1 & 0 & +1 \\ \hline 0 & -1 & -1 \\ \hline +1 & 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline (0.1 \times -1) + (0 \times 0) + (0.2 \times 1) \\ \hline (0 \times 0) + (0 \times -1) + (0.2 \times -1) \\ \hline (0.1 \times 1) + (0 \times 0) + (0 \times 0) \\ \hline \end{array} = \begin{array}{|c|} \hline \mathbf{O} \\ \hline \\ \hline \\ \hline \end{array}$$

Sparse I and/or ternary W



**Skip computation on zero values, and no multiply**

## ImageNet Accuracy



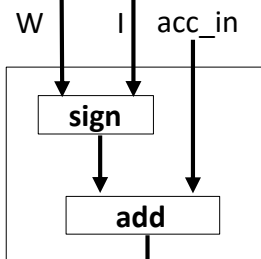
# Study 2: Ternary ResNet Case Study

**Ternary NN: neural net with weights of +1,-1,0**

I	W		O			
0.1	-1 0 +1	$(0.1x-1)+(0x0)+(0.2x1)$	<table border="1"> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> </table>			
0	0 -1 -1	$(0x0)+(0x-1)+(0.2x-1)$				
0.2	+1 0 0	$(0.1x1)+(0x0)+(0x0)$				

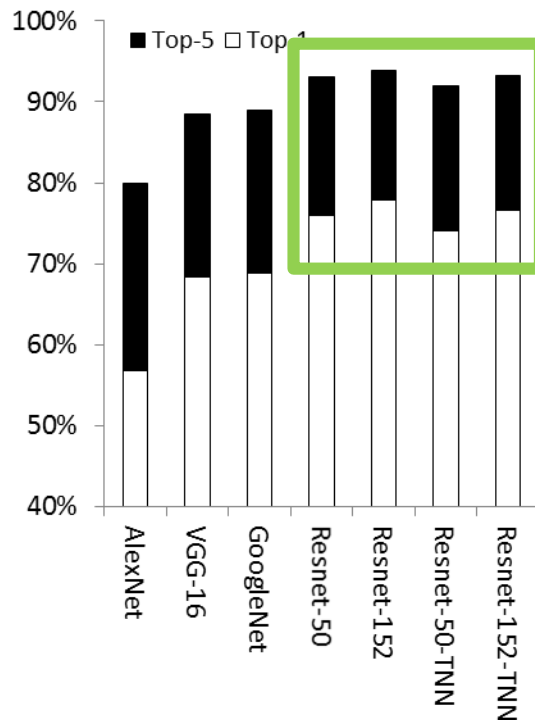
Sparse I and/or ternary W

Zero-skip scheduler



**Skip computation on zero values, and no multiply**

ImageNet Accuracy

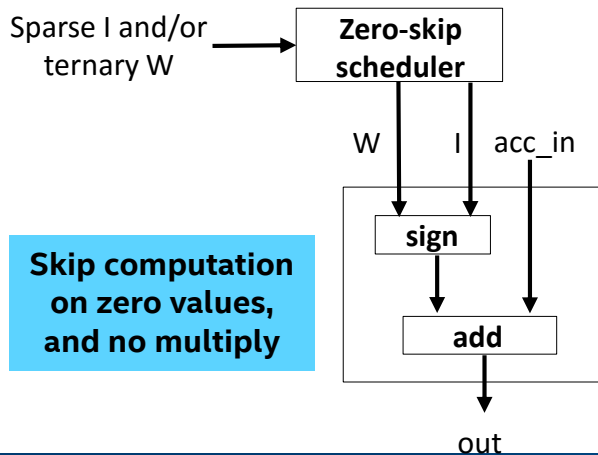


**Ternary ResNet offers state-of-the-art accuracy [ICASSP'17]**

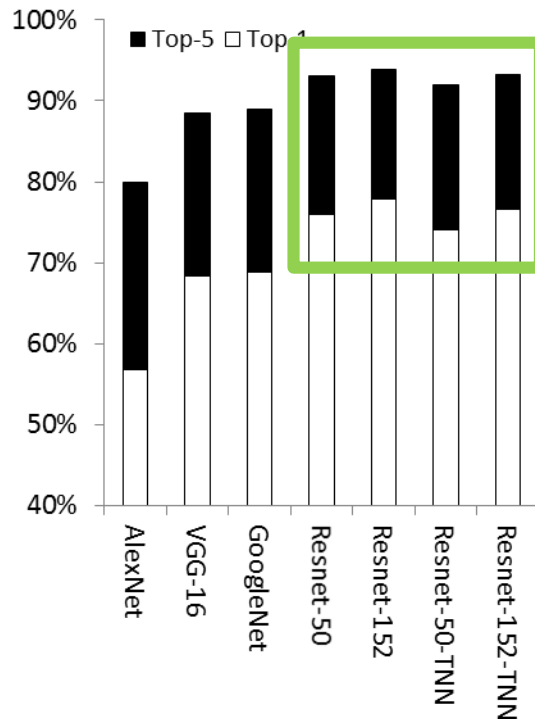
# Study 2: Ternary ResNet Case Study

**Ternary NN: neural net with weights of +1,-1,0**

I	W		O			
0.1	-1 0 +1	$(0.1x-1)+(0x0)+(0.2x1)$	<table border="1"> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> </table>			
0	0 -1 -1	$(0x0)+(0x-1)+(0.2x-1)$				
0.2	+1 0 0	$(0.1x1)+(0x0)+(0x0)$				



**ImageNet Accuracy**



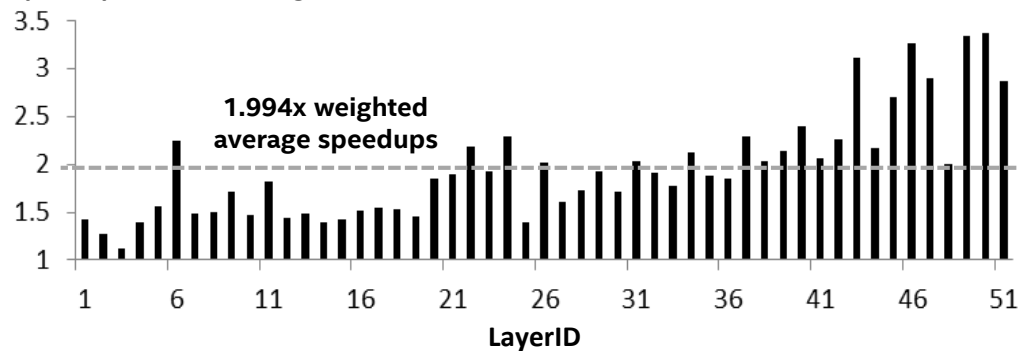
**Ternary ResNet offers state-of-the-art accuracy [ICASSP'17]**

**We target Resnet-50-TNN in this study**

# Results

Per layer speedup varies depends on sparsity

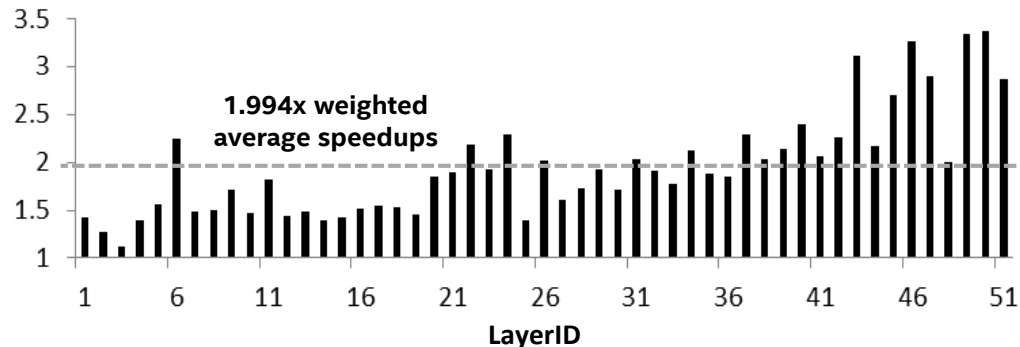
Speedup vs. dense design



# Results

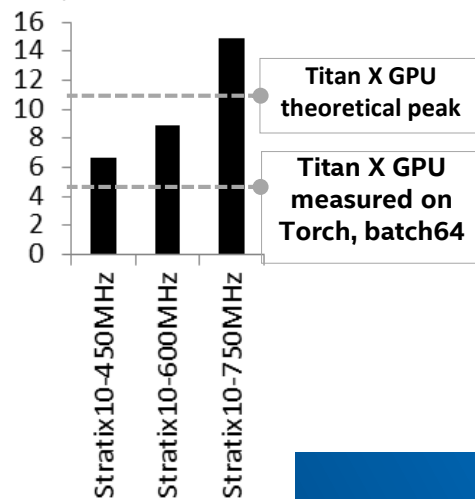
Per layer speedup varies depends on sparsity

Speedup vs. dense design

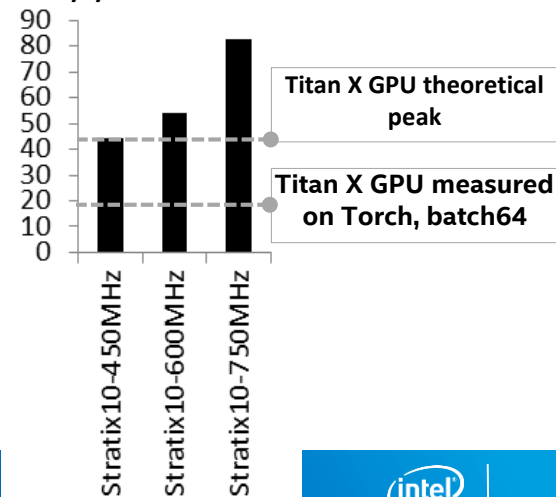


S10 FPGA performs better, across all frequency targets

TOP/s



GOP/s/Watt



# More Opportunities for FPGAs

## Various further optimizations & customizations for deep learning

- e.g., math transforms (FFT, Winograd), further quantizations, compression schemes

## Other irregular applications

- e.g., other classes of ML, apps outside of ML

## Latency sensitive applications

- e.g., ADAS, industrial usages



# So, can FPGAs beat GPUs for next-gen DNNs?

For classic DNNs using 32b dense GEMMs, FPGAs catching up to GPUs

- But next-gen DNNs may not rely purely on big dense 32b GEMM anymore

# So, can FPGAs beat GPUs for next-gen DNNs?

For classic DNNs using 32b dense GEMMs, FPGAs catching up to GPUs

- But next-gen DNNs may not rely purely on big dense 32b GEMM anymore

For upcoming more irregular DNNs, FPGAs show great promise

- Arbitrary data types (2b, 1b, ..), sparsity

# So, can FPGAs beat GPUs for next-gen DNNs?

For classic DNNs using 32b dense GEMMs, FPGAs catching up to GPUs

- But next-gen DNNs may not rely purely on big dense 32b GEMM anymore

For upcoming more irregular DNNs, FPGAs show great promise

- Arbitrary data types (2b, 1b, ..), sparsity

**Current trends favor FPGAs. Can we do even better by purposely formulating deep learning to take full advantage of FPGA strengths?**

