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

Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

Training data

Flower

Elephant

...

Training → **Model**

Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

Training data

Flower

Elephant

...

Training → **Model**

New data

→ **Inference** **Model** → **Prediction**

Flower

Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

Training data

Flower

Elephant

...

→ **Training** → **Model**

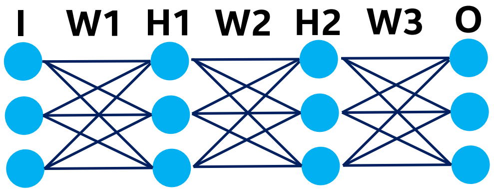
New data

→ **Inference** **Model** → **Prediction**

Flower

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

Training data

Flower

Elephant

...

→ **Training** → **Model**

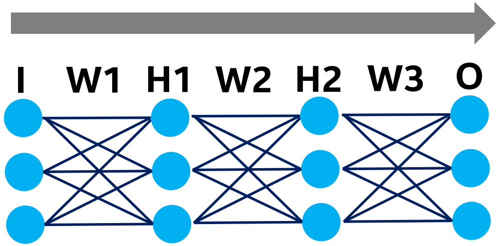
New data

→ **Inference** **Model** → **Prediction**

Flower

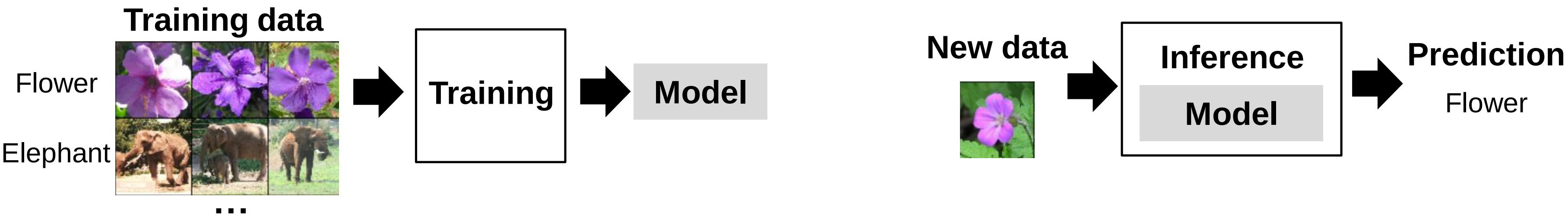
Consists of layers of neurons connected via weighted edges

E.g., 4-layer neural net

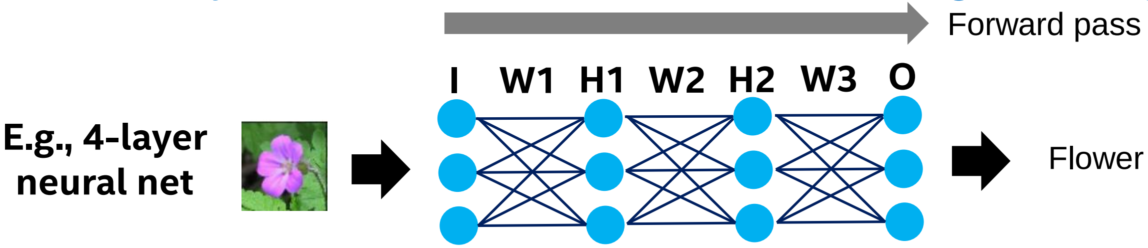


Deep Neural Networks (DNNs)

Popular machine learning (ML) approach for data analytics

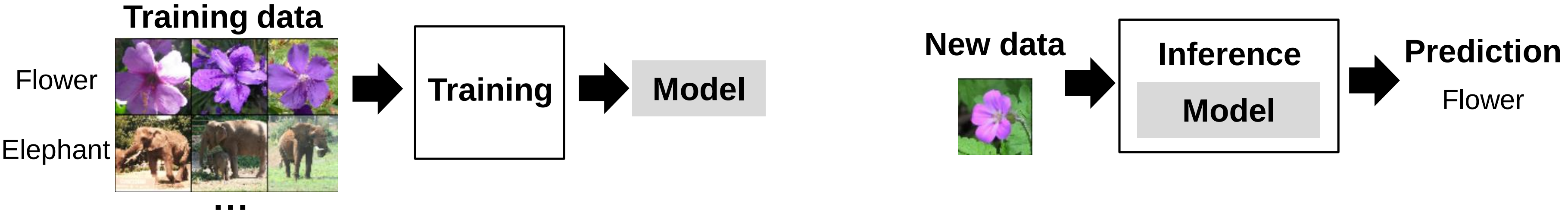


Consists of layers of neurons connected via weighted edges

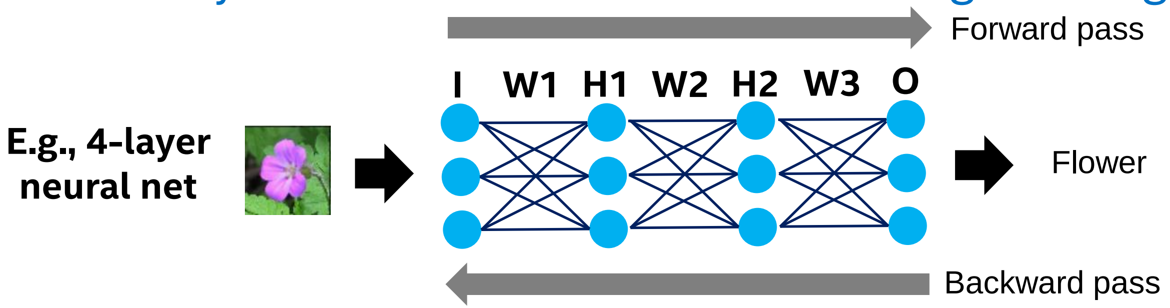


Deep Neural Networks (DNNs)

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

Training data

Flower

Elephant

...

Training → **Model**

New data

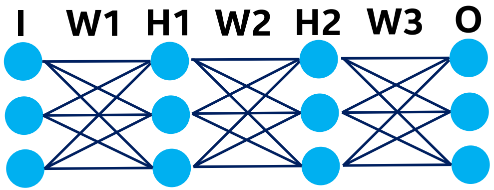
→ **Inference** **Model** → **Prediction**

Flower

Consists of layers of neurons connected via weighted edges

Forward pass

E.g., 4-layer neural net



Flower

Backward pass

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

DNNs Evolving Rapidly

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

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

2014

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

2015

2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

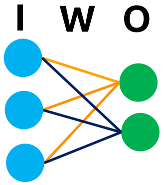
Deeper
More params?
Larger model?

LeNet5	AlexNet	VGG	GoogLeNet	ResNet
	(~80% Top5)	(~89% top5)	(~89% top5)	(~94% top5)
5 layers	8 layers	19 layers	22 layers	152 layers
Params: 1M	Params: 60M	Params: 140M	Params: 6M	Params: 60M
Model: 4MB	Model: 240MB	Model: 500MB	Model: 24MB	Model: 240MB

Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency



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

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

2014

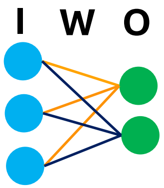
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

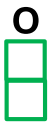
2015

2016

DNNs Evolving Rapidly

Many efforts to improve efficiency





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

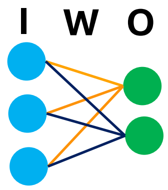
GoogLeNet
(~89% top5)
22 layers
Params: 6M
Model: 24MB

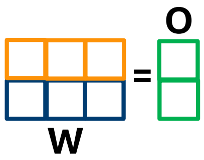
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency





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

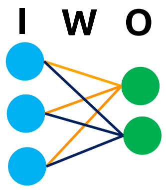
GoogLeNet
(~89% top5)
22 layers
Params: 6M
Model: 24MB
2014

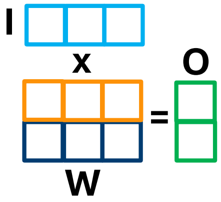
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB
2015

2016

DNNs Evolving Rapidly

Many efforts to improve efficiency





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

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

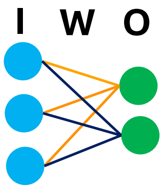
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB
2015

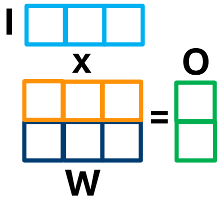
2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching





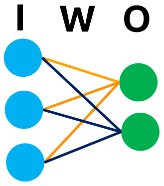
Deeper
More params?
Larger model?

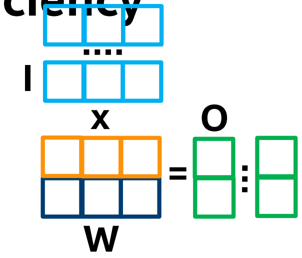
Before 2000	2012	2013	2014	2014	2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching





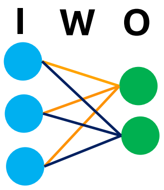
Deeper
More params?
Larger model?

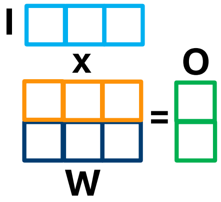
Before 2000	2012	2013	2014		2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching





Deeper
More params?
Larger model?

Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

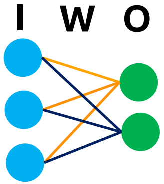
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

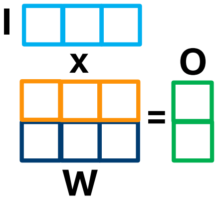
DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth





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

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

2015

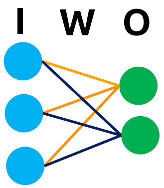
2016

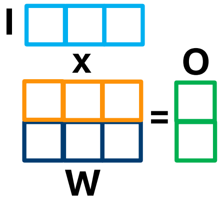
DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching

Reduce bitwidth





BinaryConnect
[NIPS'15]

XNORNet

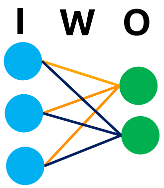
Deeper
More params?
Larger model?

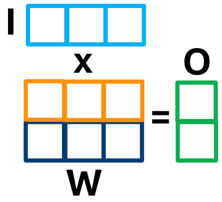
Before 2000	2012	2013	2014		2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights





BinaryConnect [NIPS'15]

XNORNet

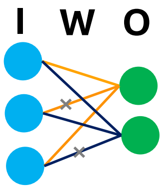
Deeper
More params?
Larger model?

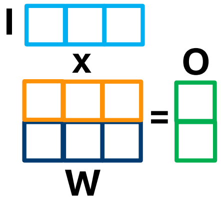
Before 2000	2012	2013	2014	2014	2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights





BinaryConnect
[NIPS'15]

XNORNet

Deeper
More params?
Larger model?

LeNet5	AlexNet	VGG	GoogLeNet	ResNet
	(~80% Top5)	(~89% top5)	(~89% top5)	(~94% top5)
5 layers	8 layers	19 layers	22 layers	152 layers
Params: 1M	Params: 60M	Params: 140M	Params: 6M	Params: 60M
Model: 4MB	Model: 240MB	Model: 500MB	Model: 24MB	Model: 240MB

Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

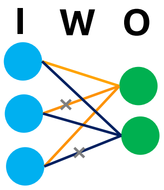
DNNs Evolving Rapidly

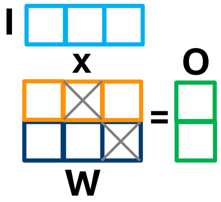
Many efforts to improve efficiency

Batching

Reduce bitwidth

Sparse weights





BinaryConnect
[NIPS'15]

XNORNet

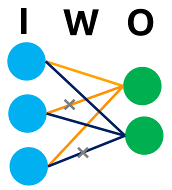
Deeper
More params?
Larger model?

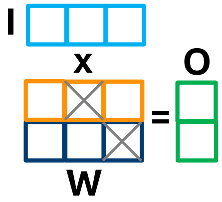
Before 2000	2012	2013	2014		2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations





BinaryConnect
[NIPS'15]

XNORNet

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

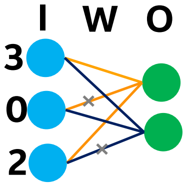
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

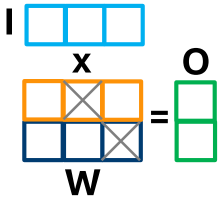
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations





BinaryConnect
[NIPS'15]

XNORNet

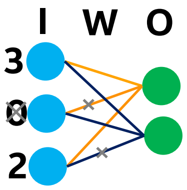
Deeper
More params?
Larger model?

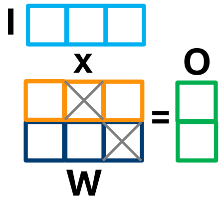
Before 2000	2012	2013	2014	2014	2015	2016
LeNet5 5 layers Params: 1M Model: 4MB	**AlexNet** (~80% Top5) 8 layers Params: 60M Model: 240MB		**VGG** (~89% top5) 19 layers Params: 140M Model: 500MB	**GoogLeNet** (~89% top5) 22 layers Params: 6M Model: 24MB	**ResNet** (~94% top5) 152 layers Params: 60M Model: 240MB	

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations





BinaryConnect [NIPS'15]

XNORNet

Deeper
More params?
Larger model?

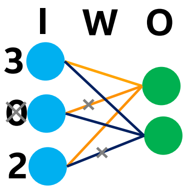
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

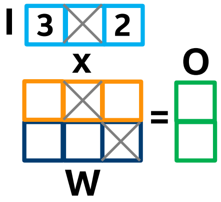
Before 2000 | 2012 | 2013 | 2014 | 2015 | 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations





BinaryConnect
[NIPS'15]

XNORNet

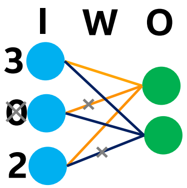
Deeper
More params?
Larger model?

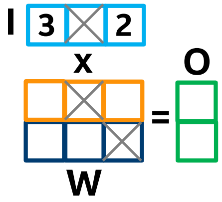
Before 2000	2012	2013	2014	2014	2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations





BinaryConnect [NIPS'15]

SparseCNN [CVPR'15]

Pruning [NIPS'15]

XNORNet

TernaryConnect [ICLR'16]

Spatially SparseCNN [CIFAR-10 winner '14]

Deeper
More params?
Larger model?

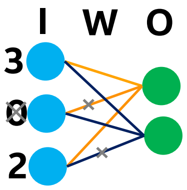
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

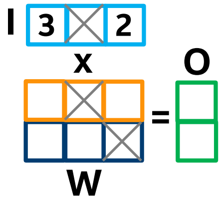
Before 2000 · 2012 · 2013 · 2014 · 2015 · 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression





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

XNORNet
TernaryConnect [ICLR'16]

Spatially SparseCNN [CIFAR-10 winner '14]

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

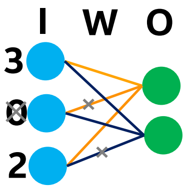
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

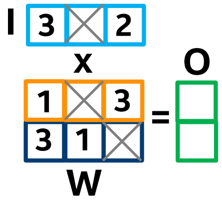
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression





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

XNORNet
TernaryConnect [ICLR'16]

Spatially SparseCNN [CIFAR-10 winner '14]

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

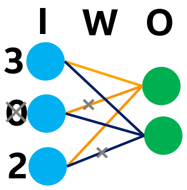
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

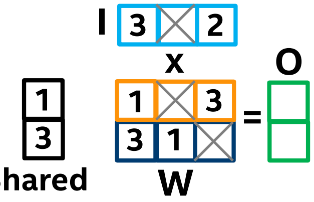
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression





Shared weights

Spatially SparseCNN [CIFAR-10 winner '14]

BinaryConnect [NIPS'15]

SparseCNN [CVPR'15]

Pruning [NIPS'15]

XNORNet

TernaryConnect [ICLR'16]

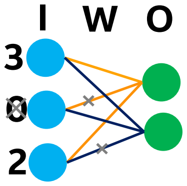
Deeper
More params?
Larger model?

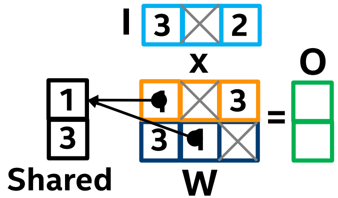
Before 2000	2012	2013	2014	2014	2015	2016
LeNet5 **5 layers** **Params: 1M** **Model: 4MB**	**AlexNet** **(~80% Top5)** **8 layers** **Params: 60M** **Model: 240MB**		**VGG** **(~89% top5)** **19 layers** **Params: 140M** **Model: 500MB**	**GoogLeNet** **(~89% top5)** **22 layers** **Params: 6M** **Model: 24MB**	**ResNet** **(~94% top5)** **152 layers** **Params: 60M** **Model: 240MB**	

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression





Spatially SparseCNN [CIFAR-10 winner '14]

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

XNORNet

TernaryConnect [ICLR'16]

Deeper
More params?
Larger model?

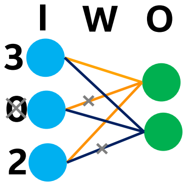
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

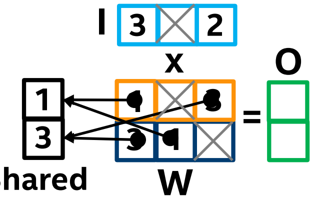
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression





Spatially SparseCNN [CIFAR-10 winner '14]

BinaryConnect [NIPS'15]

SparseCNN [CVPR'15]

Pruning [NIPS'15]

XNORNet

TernaryConnect [ICLR'16]

Deeper
More params?
Larger model?

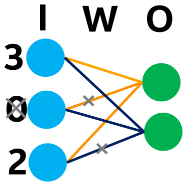
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

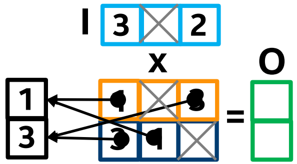
Before 2000 · 2012 · 2013 · 2014 · 2015 · 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression





Shared weights

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]

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

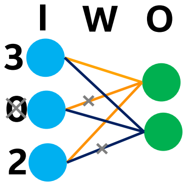
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

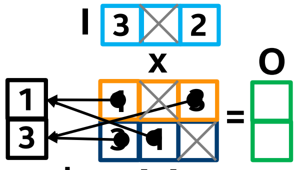
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network





Shared weights

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]

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

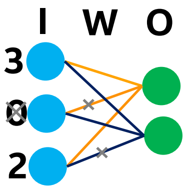
ResNet
(~94% top5)
152 layers
Params: 60M
Model: 240MB

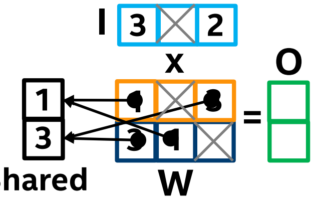
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network





Shared weights

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

Deeper
More params?
Larger model?

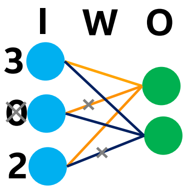
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

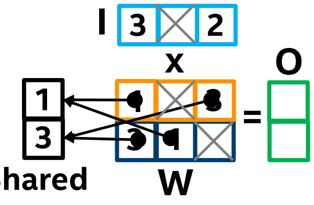
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

DNNs Evolving Rapidly

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network





Shared weights

All applicable for inference
Some for training

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

Deeper
More params?
Larger model?

Before 2000	2012	2014	2014	2015
LeNet5	AlexNet	VGG	GoogLeNet	ResNet
	(~80% Top5)	(~89% top5)	(~89% top5)	(~94% top5)
5 layers	8 layers	19 layers	22 layers	152 layers
Params: 1M	Params: 60M	Params: 140M	Params: 6M	Params: 60M
Model: 4MB	Model: 240MB	Model: 500MB	Model: 24MB	Model: 240MB

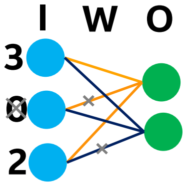
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

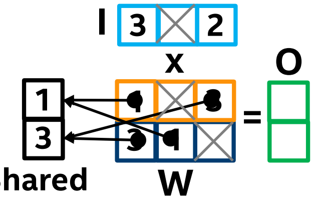
DNNs Evolving Rapidly

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





Shared weights

All applicable for inference
Some for training

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

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

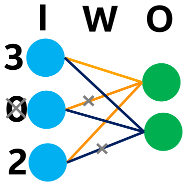
DNNs Evolving Rapidly

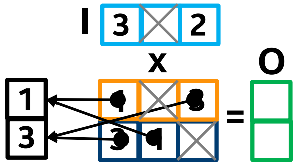
SqueezeNet + DeepCompression:
6-bit, 20-50% sparse
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

Many efforts to improve efficiency

Batching
Reduce bitwidth
Sparse weights
Sparse activations
Compression
Compact network





Shared weights

All applicable for inference
Some for training

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

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

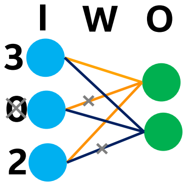
DNNs Evolving Rapidly

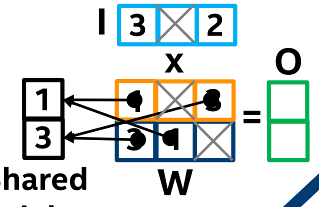
SqueezeNet + DeepCompression:
6-bit, 20-50% sparse
AlexNet accuracy, ~500x smaller (0.5MB)

XNORnet (1-bit) → ~2% AlexNet

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network





Shared weights

All applicable for inference
Some for training

~3.5 years

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

Deeper
More params?
Larger model?

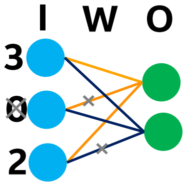
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

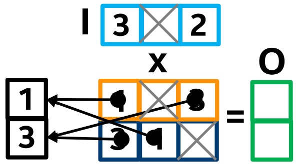
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

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

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

~3.5 years

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

Deeper
More params?
Larger model?

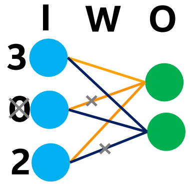
Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

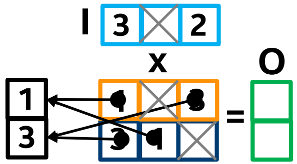
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

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

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

~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

Deeper
More params?
Larger model?

Before 2000	2012	2013	2014	2014	2015	2016
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**	

DNNs Evolving Rapidly

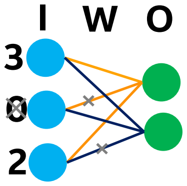
SqueezeNet + DeepCompression:
6-bit, 20-50% sparse
AlexNet accuracy, ~500x smaller (0.5MB)

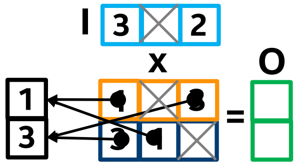
XNORnet (1-bit) → ~2% AlexNet

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

Many efforts to improve efficiency

- Batching
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network





Shared weights

All applicable for inference
Some for training

~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

Deeper
More params?
Larger model?

Before 2000	2012	2014	2014	2015
LeNet5	**AlexNet**	**VGG**	**GoogLeNet**	**ResNet**
	(~80% Top5)	**(~89% top5)**	**(~89% top5)**	**(~94% top5)**
5 layers	**8 layers**	**19 layers**	**22 layers**	**152 layers**
Params: 1M	**Params: 60M**	**Params: 140M**	**Params: 6M**	**Params: 60M**
Model: 4MB	**Model: 240MB**	**Model: 500MB**	**Model: 24MB**	**Model: 240MB**

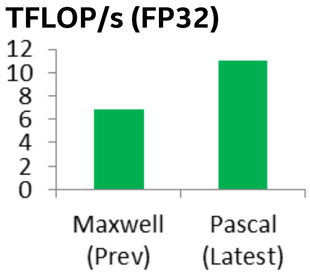
Before 2000 — 2012 — 2013 — 2014 — 2015 — 2016

Next-gen DNNs: more irregular with custom data types

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

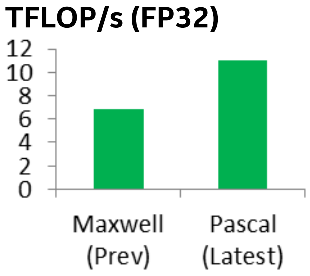
So far, high-end GPUs are gaining traction in DNNs

Lots of FLOP/s

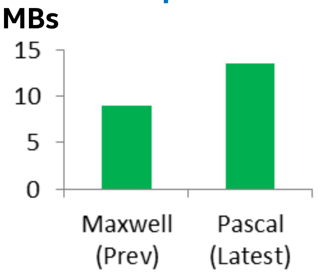


So far, high-end GPUs are gaining traction in DNNs

Lots of FLOP/s

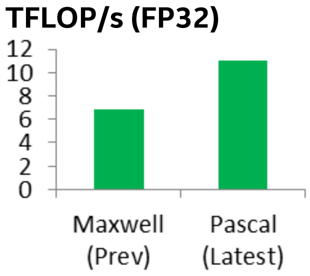


On-chip RAMs

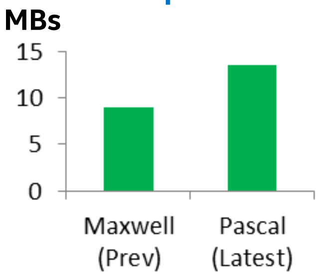


So far, high-end GPUs are gaining traction in DNNs

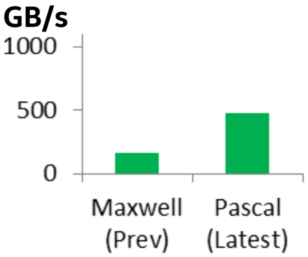
Lots of FLOP/s



On-chip RAMs

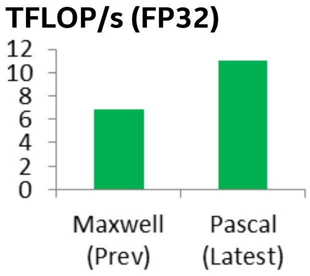


Memory BW

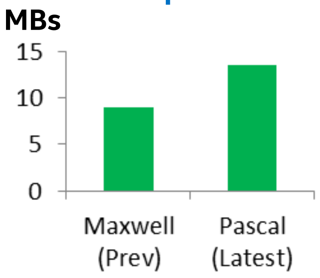


So far, high-end GPUs are gaining traction in DNNs

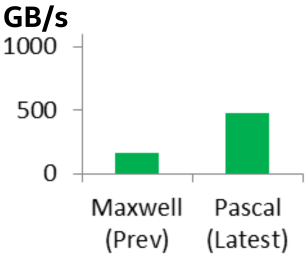
Lots of FLOP/s



On-chip RAMs



Memory BW

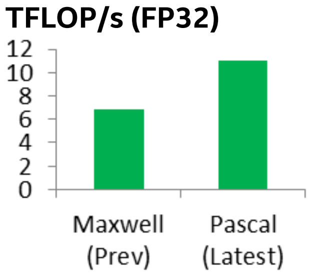


But, Power Hungry

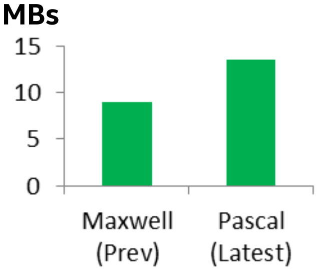
~200+ W

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

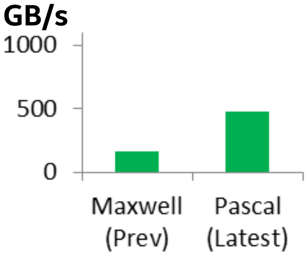
Lots of FLOP/s



On-chip RAMs



Memory BW



But, Power Hungry

~200+ W

Programmable

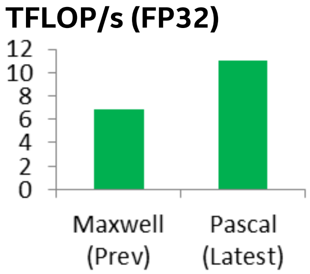
DESIGNED FOR NVIDIA CUDA

cuBLAS

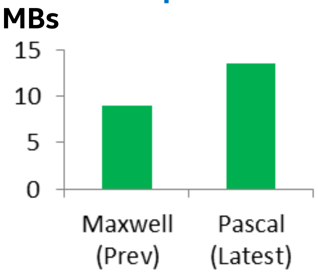
cuDNN

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

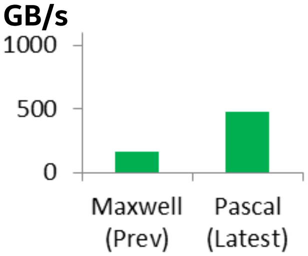
Lots of FLOP/s



On-chip RAMs



Memory BW



But, Power Hungry

~200+ W

Programmable

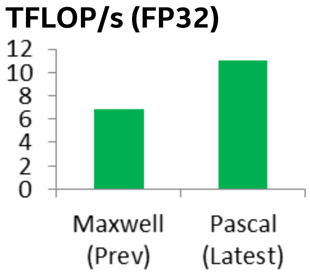
cuBLAS

cuDNN

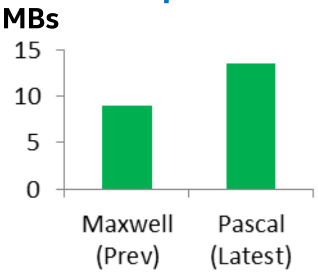
Integrated

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

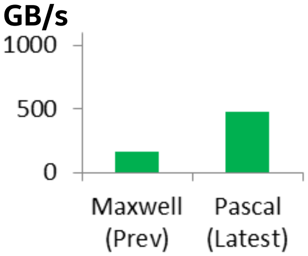
Lots of FLOP/s



On-chip RAMs



Memory BW



But, Power Hungry

~200+ W

Programmable

cuBLAS

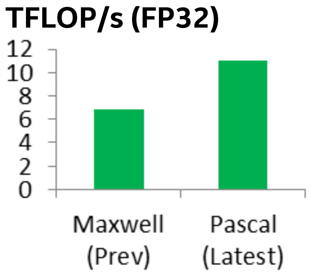
cuDNN

Integrated

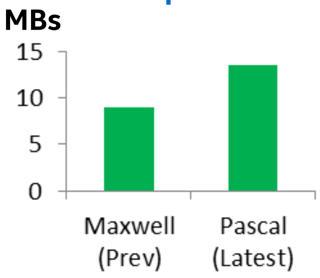
High achievable FLOP/s on GEMM
(for big enough regular dense matrix, using native data types – FP32, FP16, INT8)

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

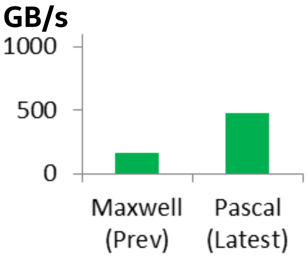
Lots of FLOP/s



On-chip RAMs



Memory BW



But, Power Hungry

~200+ W

Programmable

DESIGNED FOR NVIDIA CUDA

cuBLAS

cuDNN

Integrated

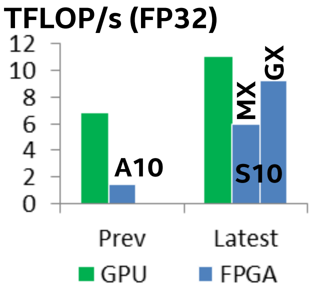
High achievable FLOP/s on GEMM
(for big enough regular dense matrix, using native data types – FP32, FP16, INT8)

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

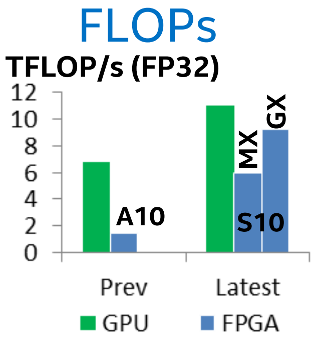
Meanwhile, FPGAs becoming much more capable

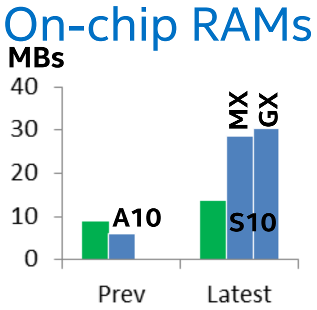
Meanwhile, FPGAs becoming much more capable

FLOPs

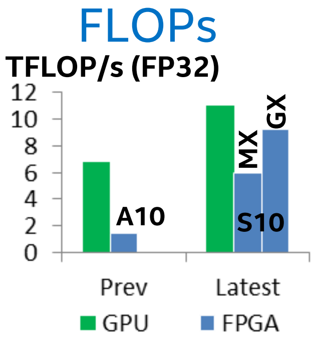


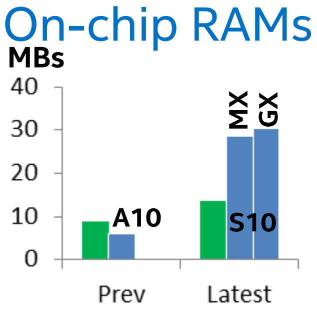
Meanwhile, FPGAs becoming much more capable

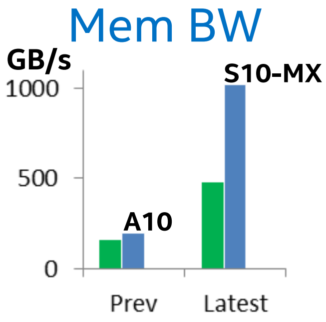




Meanwhile, FPGAs becoming much more capable

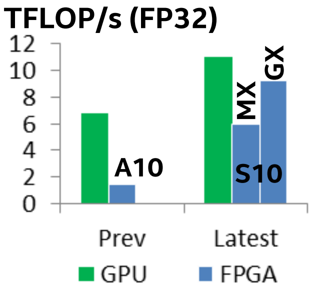




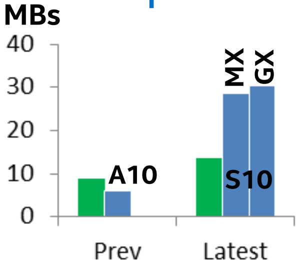


Meanwhile, FPGAs becoming much more capable

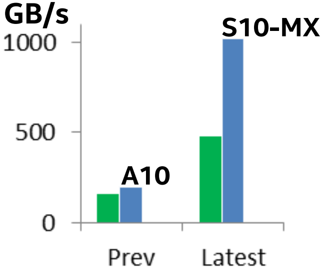
FLOPs



On-chip RAMs



Mem BW



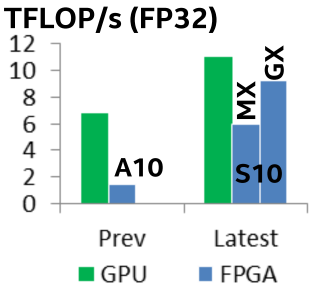
Higher Freq

HyperFlex™ ARCHITECTURE

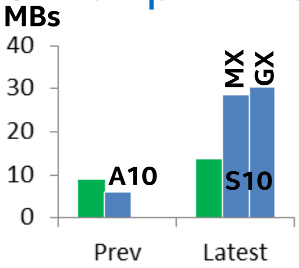
2x core frequency

Meanwhile, FPGAs becoming much more capable

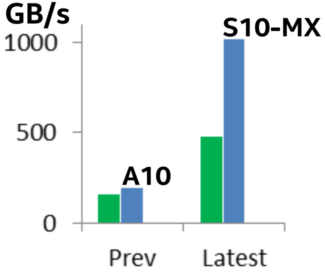
FLOPs



On-chip RAMs



Mem BW



Higher Freq

HyperFlex™ ARCHITECTURE

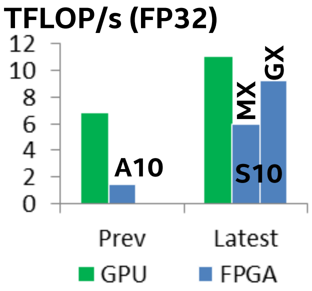
2x core frequency

Power Efficient

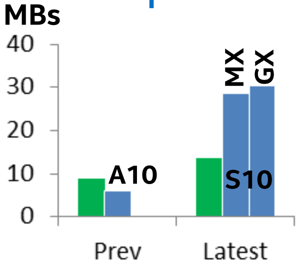
10s-100s W

Meanwhile, FPGAs becoming much more capable

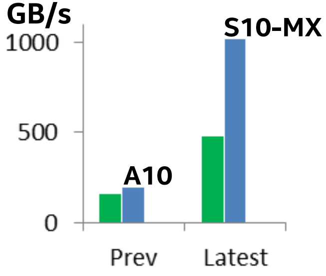
FLOPs



On-chip RAMs



Mem BW



Higher Freq

HyperFlex™ ARCHITECTURE

2x core frequency

Power Efficient

10s-100s W

High-Level Programming

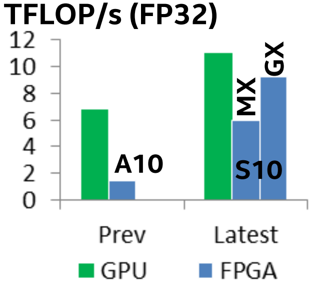
OpenCL

Altera® for OpenCL SDK

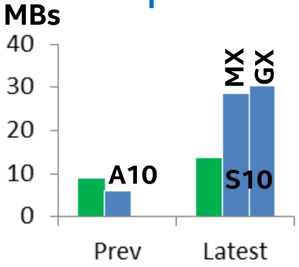
A++

Meanwhile, FPGAs becoming much more capable

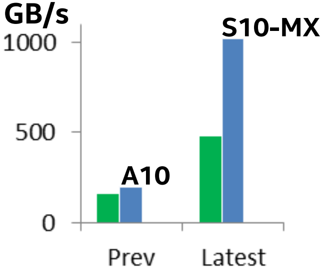
FLOPs



On-chip RAMs



Mem BW



Higher Freq

HyperFlex™ ARCHITECTURE

2x core frequency

Power Efficient

10s-100s W

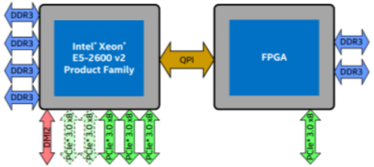
High-Level Programming

OpenCL

Altera® for OpenCL SDK

A++

More Integrated

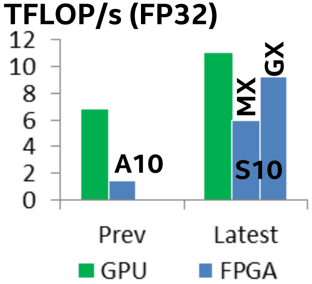


Xeon+FPGA

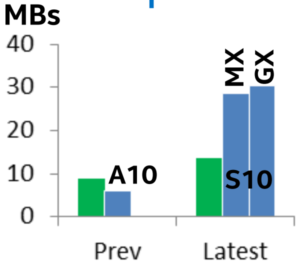
Discrete cards

Meanwhile, FPGAs becoming much more capable

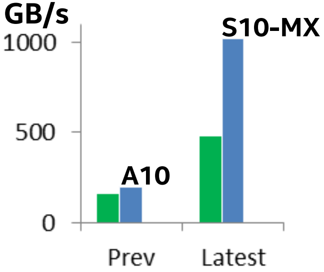
FLOPs



On-chip RAMs



Mem BW



Higher Freq

HyperFlex™ ARCHITECTURE

2x core frequency

Power Efficient

10s-100s W

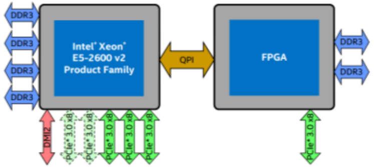
High-Level Programming

OpenCL

Altera® for OpenCL SDK

A++

More Integrated

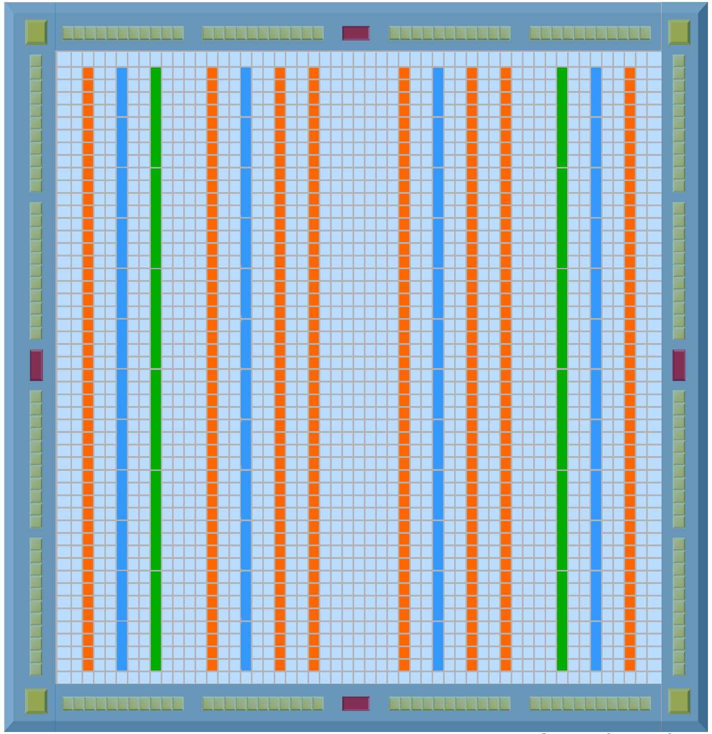


Xeon+FPGA

Discrete cards

Upcoming Stratix 10 will be more competitive to GPUs

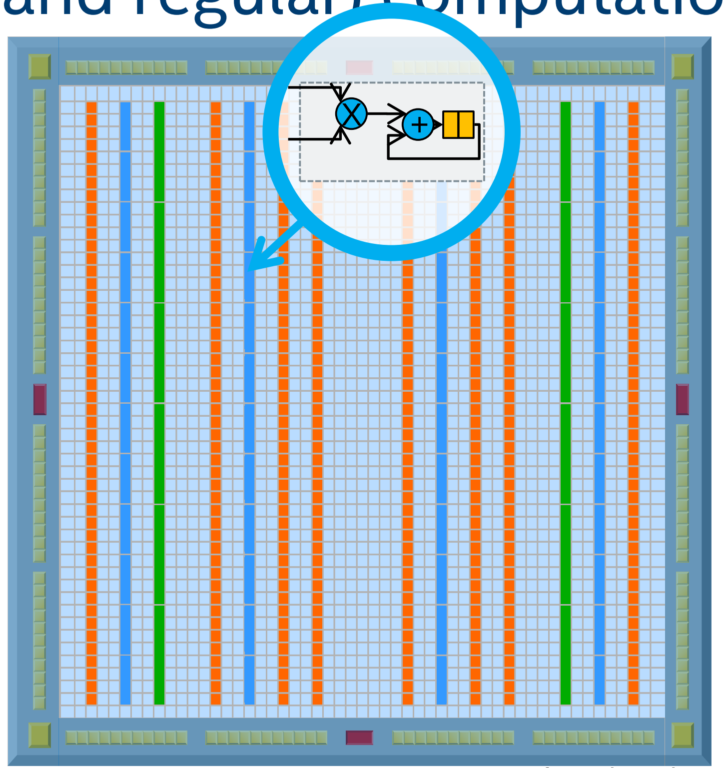
FPGA fabric is great for irregular (and regular) computation



Figures courtesy of Gordon Chiu

FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

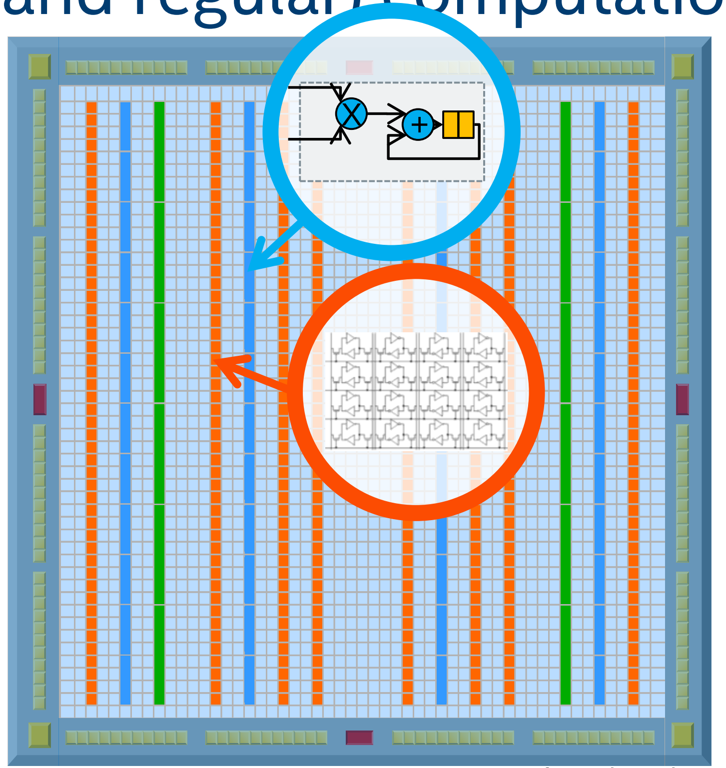


Figures courtesy of Gordon Chiu

FPGA fabric is great for irregular (and regular) computation

1000s of hard DSPs (floating-point units)

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



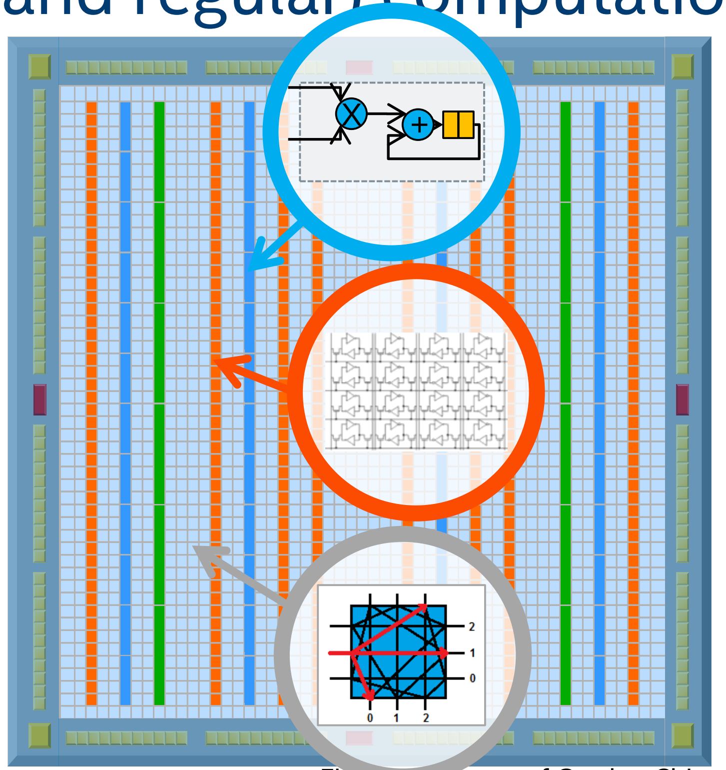
Figures courtesy of Gordon Chiu

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

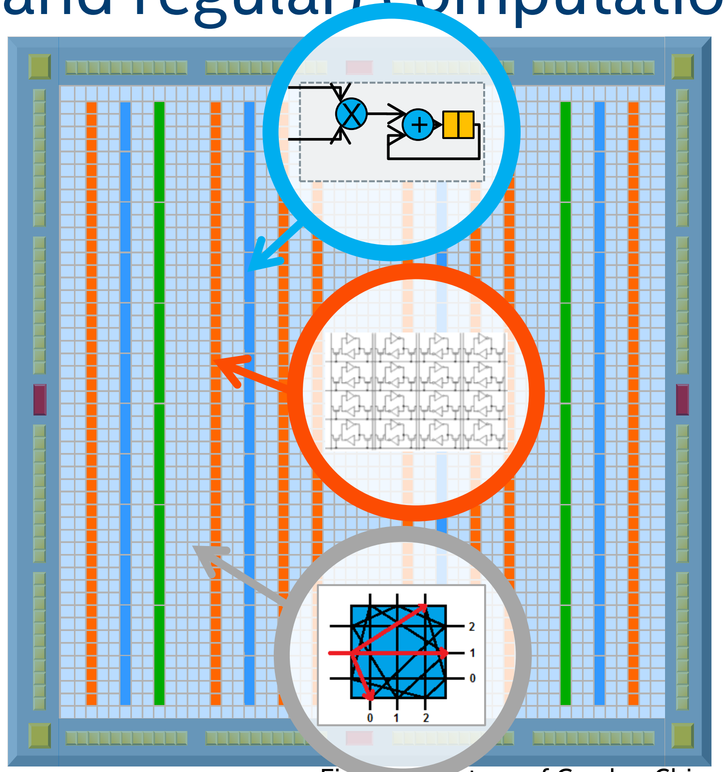
FPGA fabric is great for irregular (and regular) computation

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

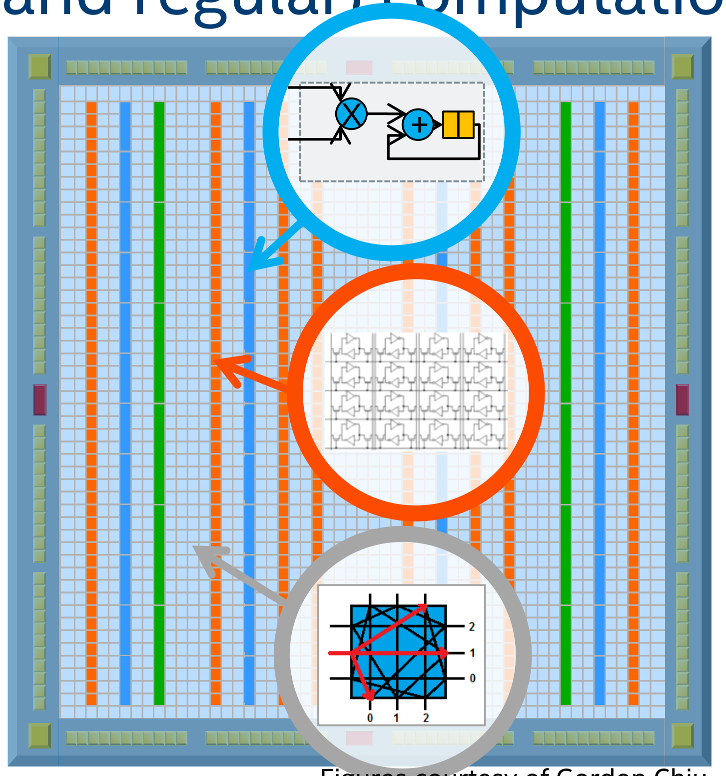
1000s of hard DSPs (floating-point units)

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)

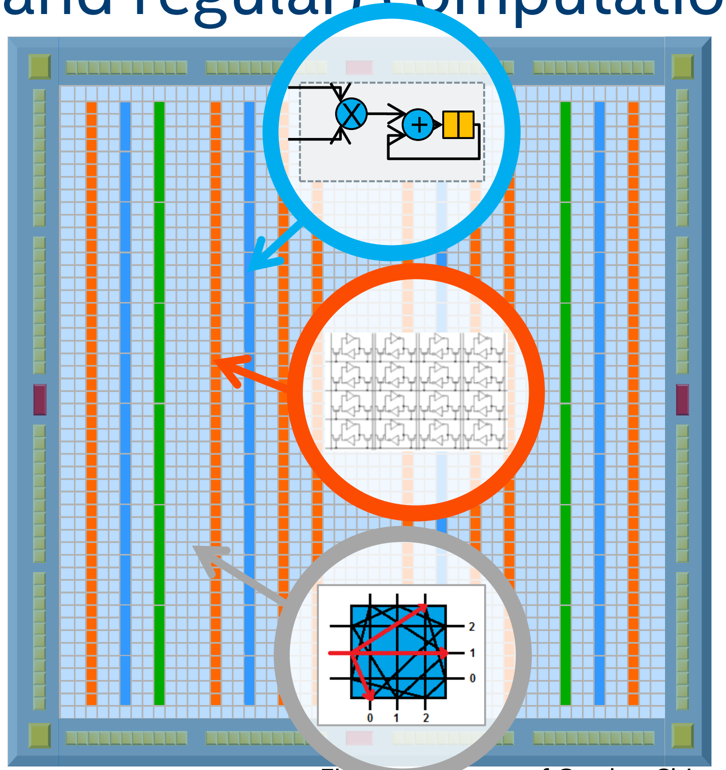
1000s of Hard "M20K" SRAMs (2.5KB/SRAM)

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)

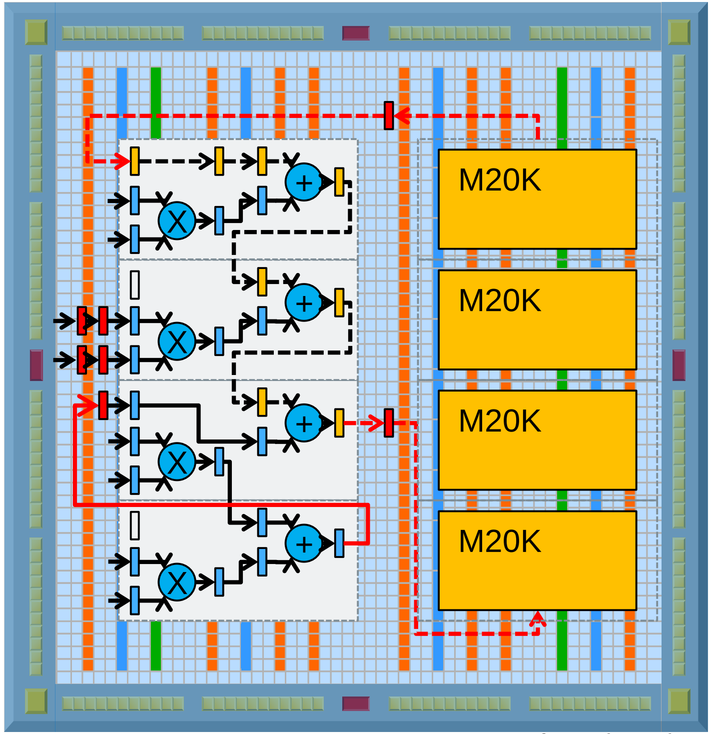
Sea of Programmable Logic and Routing

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)

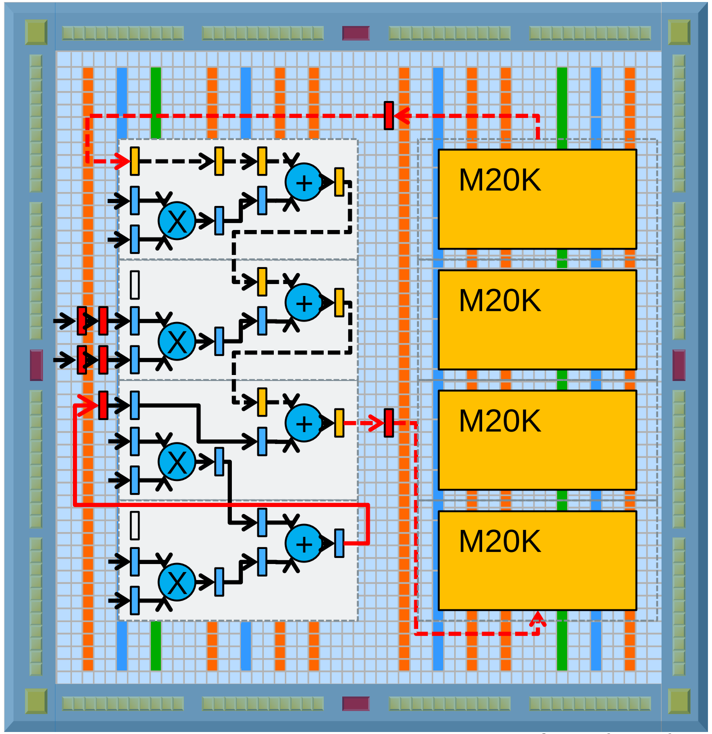
Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc

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

Arbitrary DNN architectures



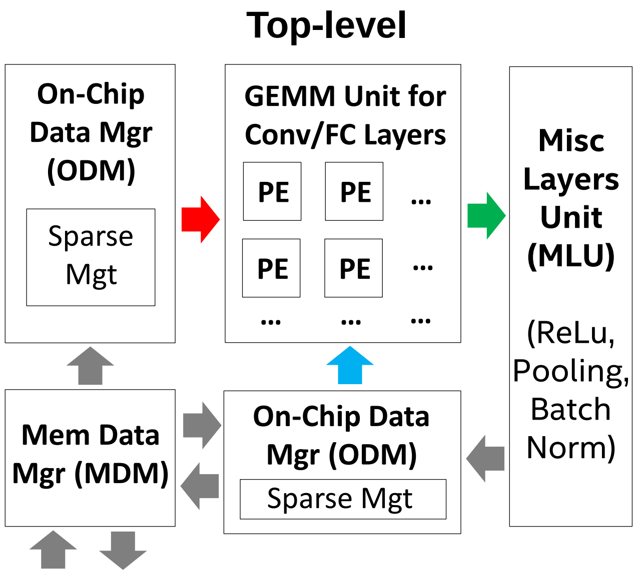
Figures courtesy of Gordon Chiu

FPGAs well positioned for deep learning

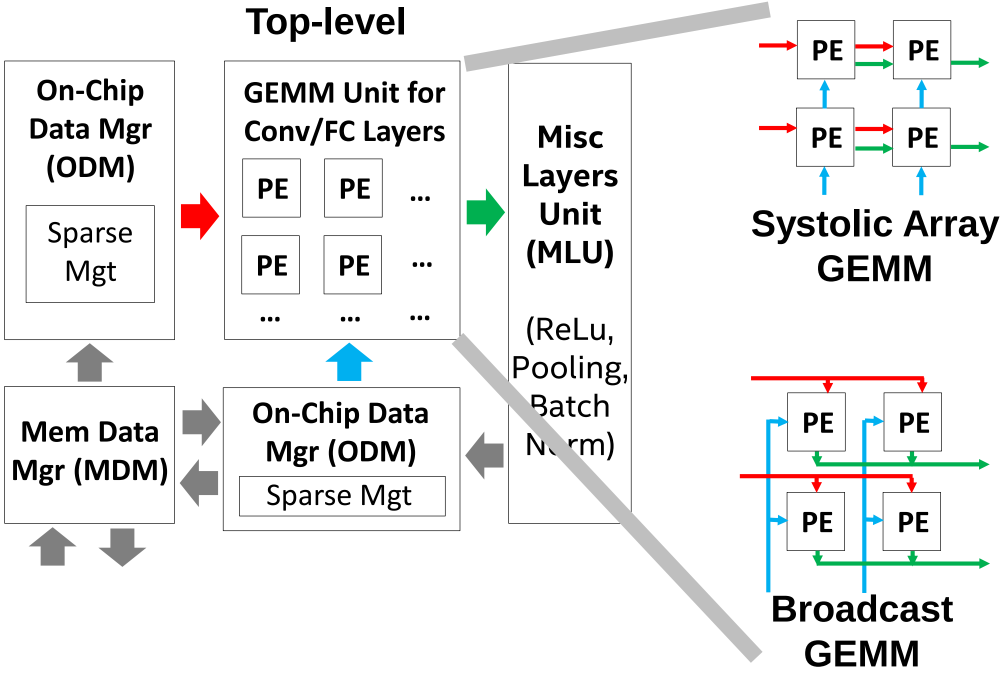
This work: compare high-end GPU vs. FPGA for DNNs

- 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

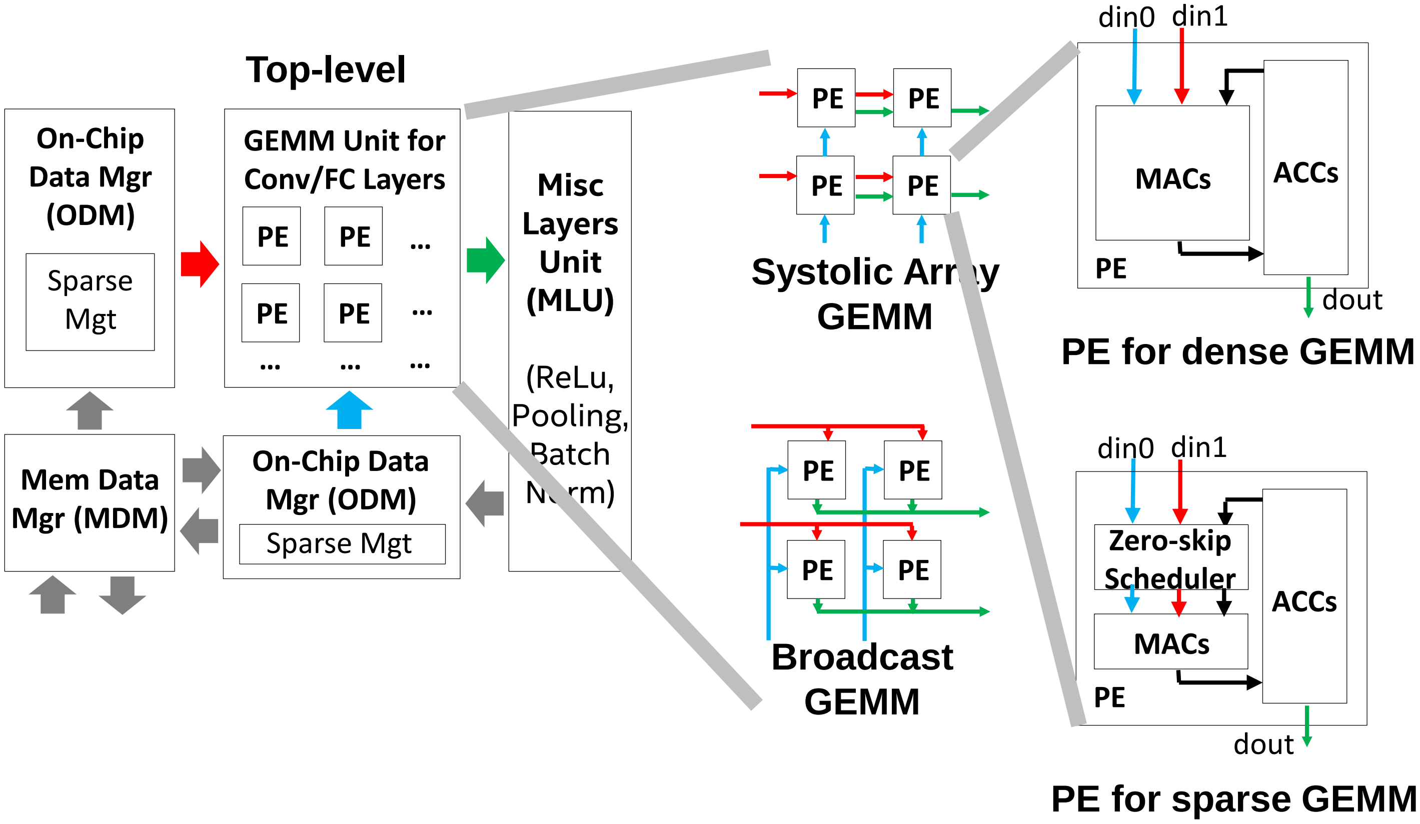
DNN accelerator template for FPGA used in our studies



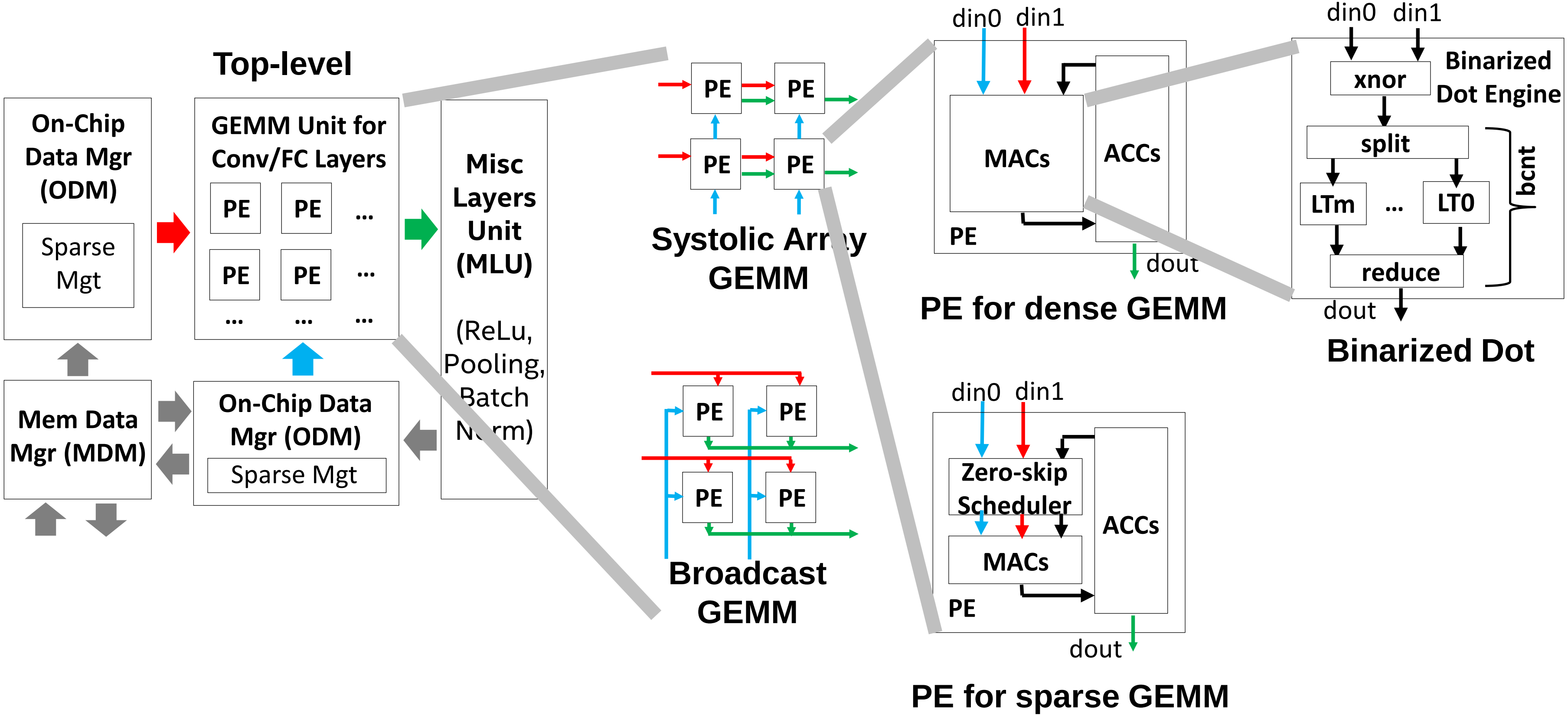
DNN accelerator template for FPGA used in our studies



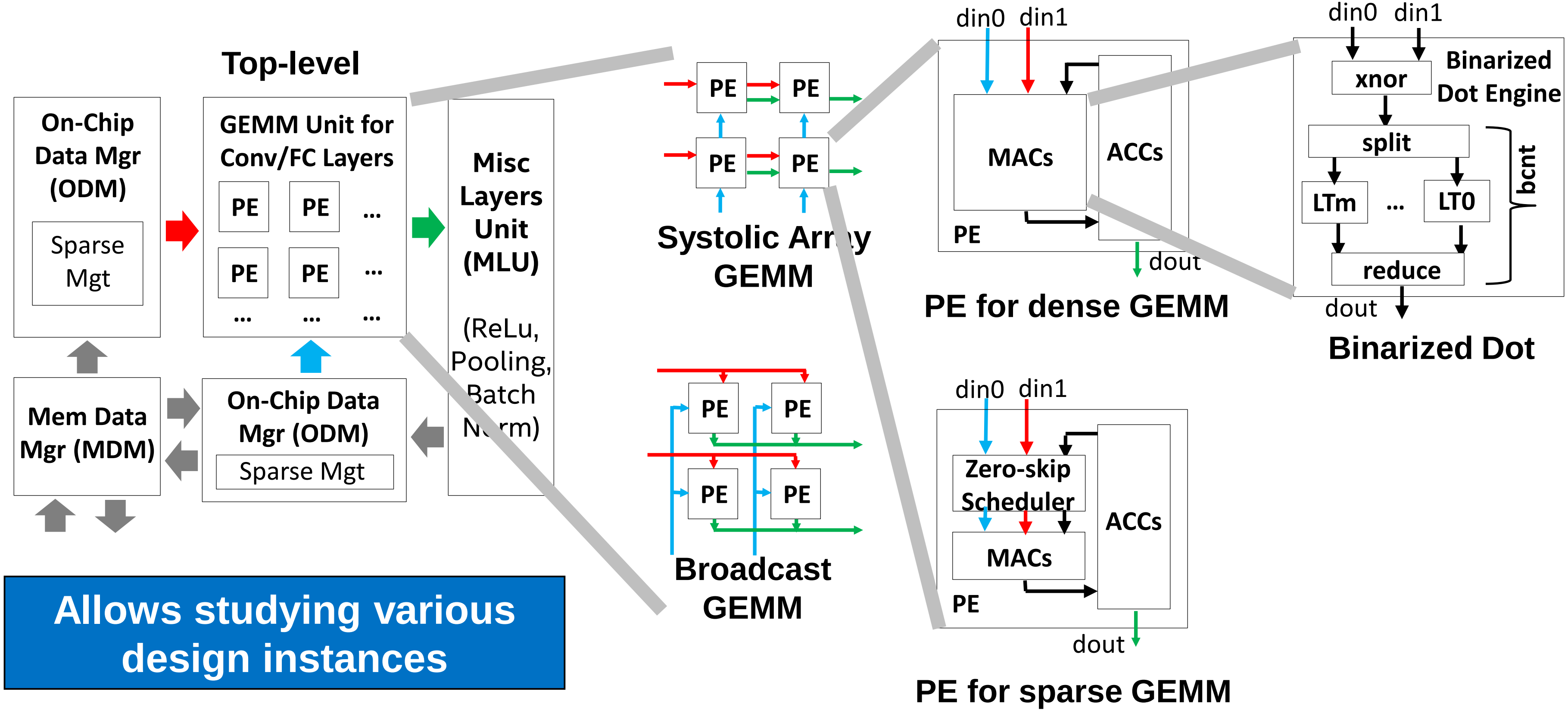
DNN accelerator template for FPGA used in our studies



DNN accelerator template for FPGA used in our studies



DNN accelerator template for FPGA used in our studies



Allows studying various design instances

Methodology

FPGAs and GPU under study

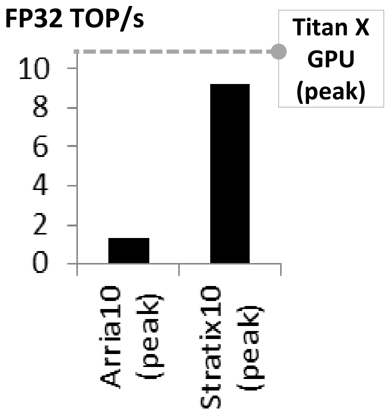
	Arria 10 1150 FPGA	Stratix 10 2800 FPGA	TitanX Pascal GPU
Peak FP32 TFLOPs	1.36	9.2	11
On-chip RAMs	6.6 MB (M20Ks)	28.6 MB (M20Ks)	13.5 MB (RF, SM, L2)
Memory BW	Assume same as Titan X	Assume same as Titan X	480 GB/s

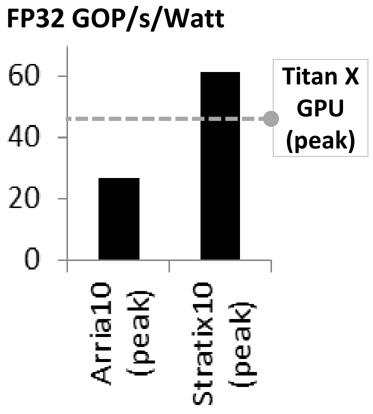
Evaluation

- 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

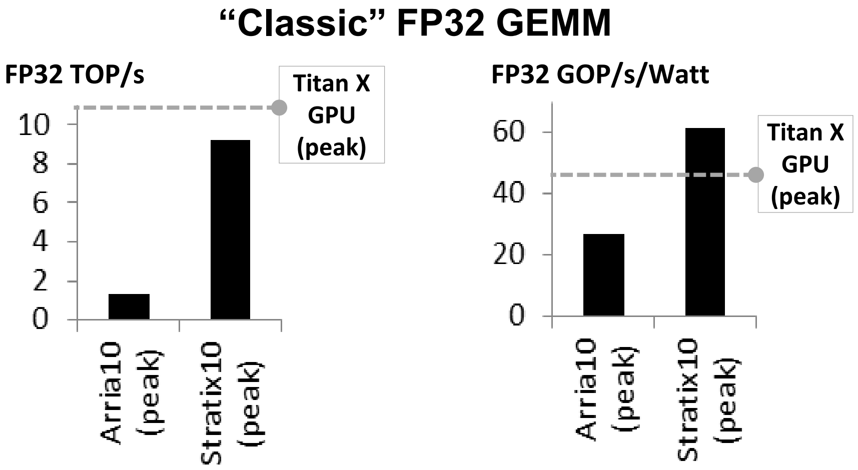
Study 1 results: Dense GEMM

"Classic" FP32 GEMM





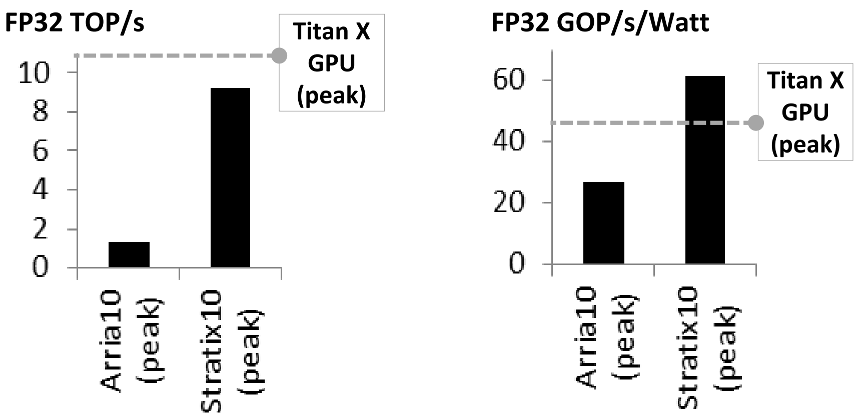
Study 1 results: Dense GEMM



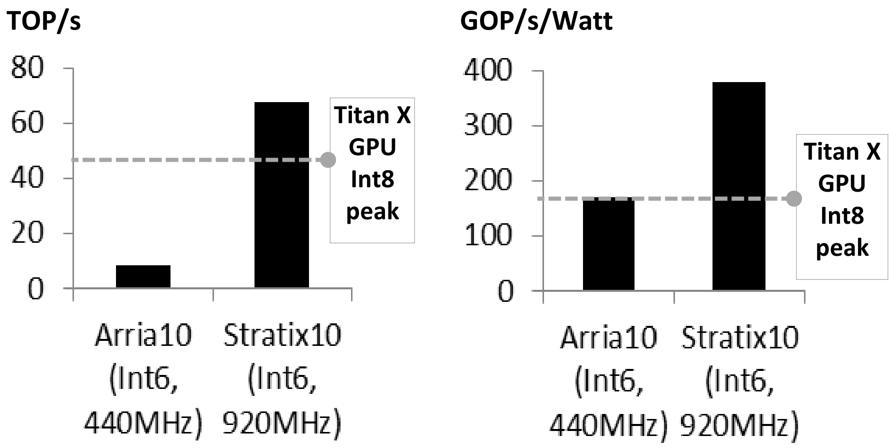
For "classic" FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt

Study 1 results: Dense GEMM

"Classic" FP32 GEMM



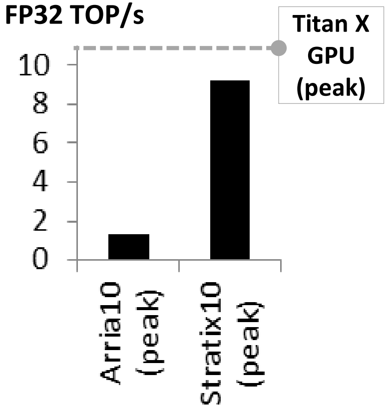
Low precision (INT6, INT8) GEMM

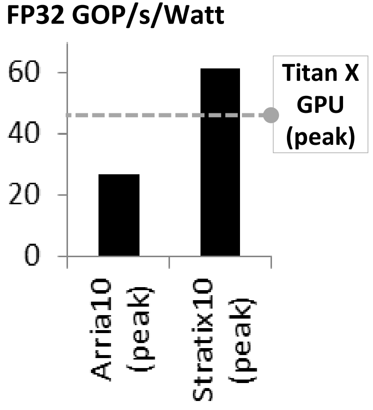


For "classic" FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt

Study 1 results: Dense GEMM

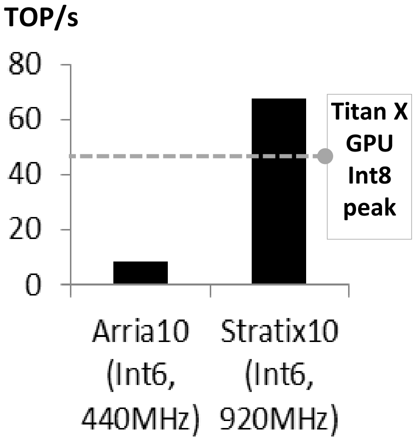
"Classic" FP32 GEMM

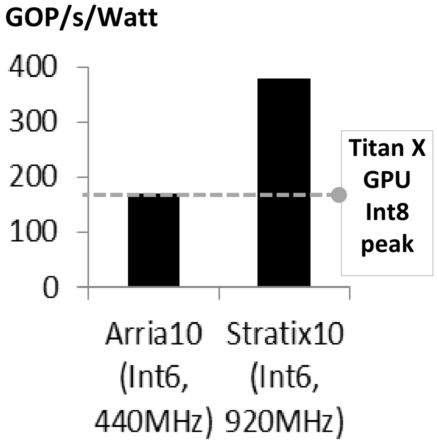




For "classic" FP32 Dense GEMM, S10 FPGA is catching up to GPU in performance, and better in perf/Watt

Low precision (INT6, INT8) GEMM





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

$$\overset{\textbf{I}}{\begin{bmatrix}-1\\+1\\+1\end{bmatrix}} \text{ x } \overset{\textbf{W}}{\begin{bmatrix}-1 & +1 & +1\\-1 & -1 & -1\\+1 & +1 & -1\end{bmatrix}} = \begin{matrix}(-1.-1)+(1.1)+(1.1)\\(-1.-1)+(1.-1)+(1.-1)\\(-1.1)+(1.1)+(1.-1)\end{matrix} = \overset{\textbf{O}}{\begin{bmatrix}3\\-1\\-1\end{bmatrix}}$$

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

Neural networks with parameters of +1 or –1

Matrix x Vector, with +1 or -1

$$\underset{I}{\begin{bmatrix} -1 \\ +1 \\ +1 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} -1 & +1 & +1 \\ -1 & -1 & -1 \\ +1 & +1 & -1 \end{bmatrix}} = \begin{matrix} (-1.-1)+(1.1)+(1.1) \\ (-1.-1)+(1.-1)+(1.-1) \\ (-1.1)+(1.1)+(1.-1) \end{matrix} = \underset{O}{\begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}}$$

Binarized Matrix x Vector

$$\underset{I}{\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}} = \begin{matrix} \mathrm{bcnt}(\mathrm{xnor}(011,011)) \\ \mathrm{bcnt}(\mathrm{xnor}(011,000)) \\ \mathrm{bcnt}(\mathrm{xnor}(011,110)) \end{matrix} = \underset{O}{\begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}}$$

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

Neural networks with parameters of +1 or –1

Matrix x Vector, with +1 or -1

$$
\begin{array}{c}\textbf{I}\\ \begin{bmatrix}-1\\+1\\+1\end{bmatrix}\end{array}
\times
\begin{array}{c}\textbf{W}\\ \begin{bmatrix}-1 & +1 & +1\\ -1 & -1 & -1\\ +1 & +1 & -1\end{bmatrix}\end{array}
=
\begin{array}{c}(-1.-1)+(1.1)+(1.1)\\ (-1.-1)+(1.-1)+(1.-1)\\ (-1.1)+(1.1)+(1.-1)\end{array}
=
\begin{array}{c}\textbf{O}\\ \begin{bmatrix}3\\-1\\-1\end{bmatrix}\end{array}
$$

Binarized Matrix x Vector

$$
\begin{array}{c}\textbf{I}\\ \begin{bmatrix}0\\1\\1\end{bmatrix}\end{array}
\times
\begin{array}{c}\textbf{W}\\ \begin{bmatrix}0 & 1 & 1\\ 0 & 0 & 0\\ 1 & 1 & 0\end{bmatrix}\end{array}
=
\begin{array}{c}\text{bcnt(xnor(011,011))}\\ \text{bcnt(xnor(011,000))}\\ \text{bcnt(xnor(011,110))}\end{array}
=
\begin{array}{c}\textbf{O}\\ \begin{bmatrix}3\\-1\\-1\end{bmatrix}\end{array}
$$

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

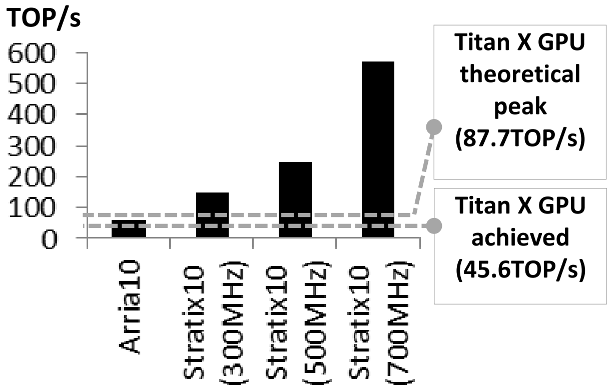
Matrix x Vector, with +1 or -1

I		W					O
-1		-1	+1	+1		(-1.-1)+(1.1)+(1.1)	3
+1	x	-1	-1	-1	=	(-1.-1)+(1.-1)+(1.-1) =	-1
+1		+1	+1	-1		(-1.1)+(1.1)+(1.-1)	-1

Binarized Matrix x Vector

I		W					O
0		0	1	1		bcnt(xnor(011,011))	3
1	x	0	0	0	=	bcnt(xnor(011,000))=	-1
1		1	1	0		bcnt(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

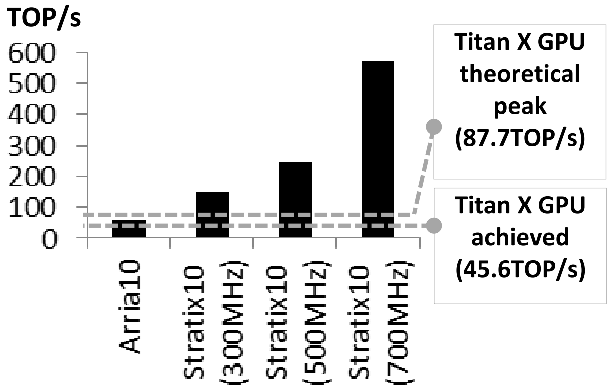
Matrix x Vector, with +1 or -1

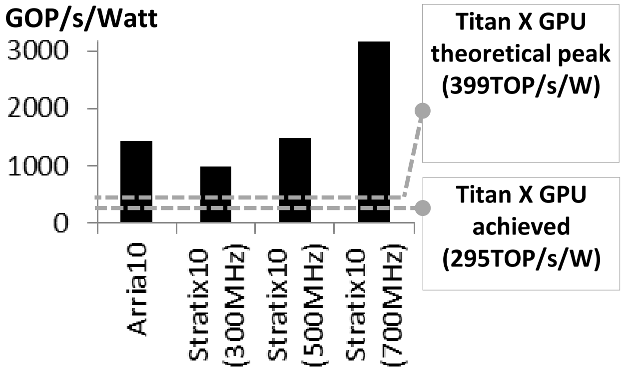
$$\underset{I}{\begin{bmatrix} -1 \\ +1 \\ +1 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} -1 & +1 & +1 \\ -1 & -1 & -1 \\ +1 & +1 & -1 \end{bmatrix}} = \begin{matrix} (-1.-1)+(1.1)+(1.1) \\ (-1.-1)+(1.-1)+(1.-1) \\ (-1.1)+(1.1)+(1.-1) \end{matrix} = \underset{O}{\begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}}$$

Binarized Matrix x Vector

$$\underset{I}{\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}} = \begin{matrix} \mathrm{bcnt}(\mathrm{xnor}(011,011)) \\ \mathrm{bcnt}(\mathrm{xnor}(011,000)) \\ \mathrm{bcnt}(\mathrm{xnor}(011,110)) \end{matrix} = \underset{O}{\begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}}$$

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

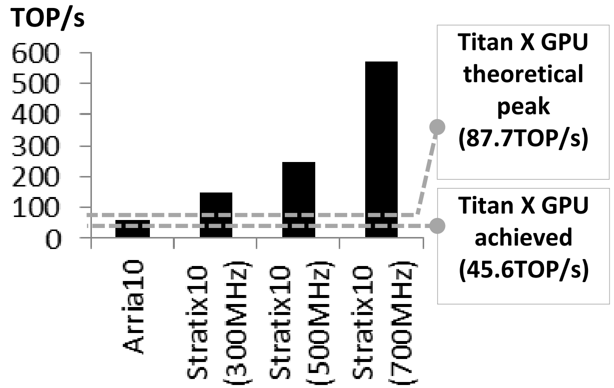
Matrix x Vector, with +1 or -1

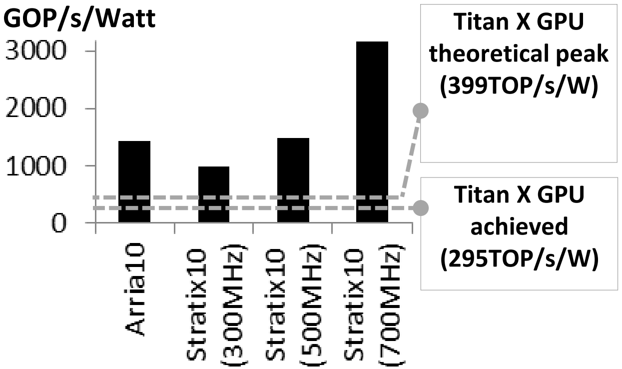
$$\underset{I}{\begin{bmatrix}-1\\+1\\+1\end{bmatrix}} \times \underset{W}{\begin{bmatrix}-1 & +1 & +1\\-1 & -1 & -1\\+1 & +1 & -1\end{bmatrix}} = \begin{matrix}(-1.-1)+(1.1)+(1.1)\\(-1.-1)+(1.-1)+(1.-1)\\(-1.1)+(1.1)+(1.-1)\end{matrix} = \underset{O}{\begin{bmatrix}3\\-1\\-1\end{bmatrix}}$$

Binarized Matrix x Vector

$$\underset{I}{\begin{bmatrix}0\\1\\1\end{bmatrix}} \times \underset{W}{\begin{bmatrix}0 & 1 & 1\\0 & 0 & 0\\1 & 1 & 0\end{bmatrix}} = \begin{matrix}\mathrm{bcnt}(\mathrm{xnor}(011,011))\\\mathrm{bcnt}(\mathrm{xnor}(011,000))\\\mathrm{bcnt}(\mathrm{xnor}(011,110))\end{matrix} = \underset{O}{\begin{bmatrix}3\\-1\\-1\end{bmatrix}}$$

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

$$\underset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} 0.2 & 0 & 0.1 \\ 0 & 0.3 & 0 \\ 0.1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1\text{x}0.2)+(0\text{x}0)+(0.2\text{x}0.1) \\ (0.1\text{x}0)+(0\text{x}0.3)+(0.2\text{x}0) \\ (0.1\text{x}0.1)+(0\text{x}0)+(0.2\text{x}0) \end{matrix} = \underset{O}{\begin{bmatrix} \ \\ \ \\ \ \end{bmatrix}}$$

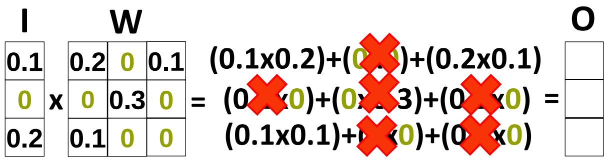
Study 1 results: Sparse GEMM

Sparse NN

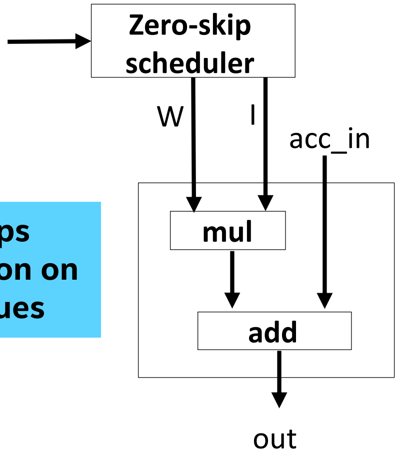
$$
\overset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \overset{W}{\begin{bmatrix} 0.2 & 0 & 0.1 \\ 0 & 0.3 & 0 \\ 0.1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1\mathrm{x}0.2)+\cancel{(0\mathrm{x}0)}+(0.2\mathrm{x}0.1) \\ \cancel{(0.1\mathrm{x}0)}+\cancel{(0\mathrm{x}0.3)}+\cancel{(0.2\mathrm{x}0)} \\ (0.1\mathrm{x}0.1)+\cancel{(0\mathrm{x}0)}+\cancel{(0.2\mathrm{x}0)} \end{matrix} = \overset{O}{\begin{bmatrix} \ \\ \ \\ \ \end{bmatrix}}
$$

Study 1 results: Sparse GEMM

Sparse NN



Sparse
W and/or I

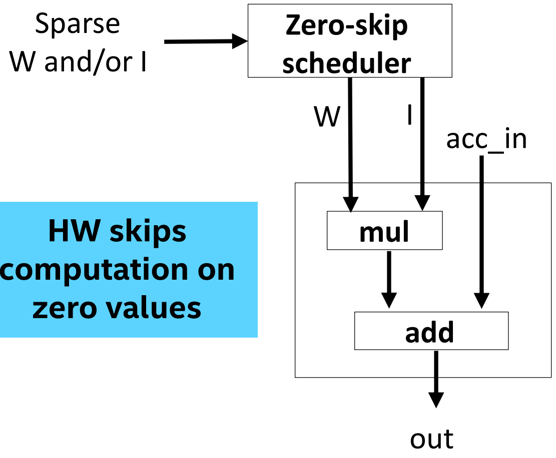


HW skips computation on zero values

Study 1 results: Sparse GEMM

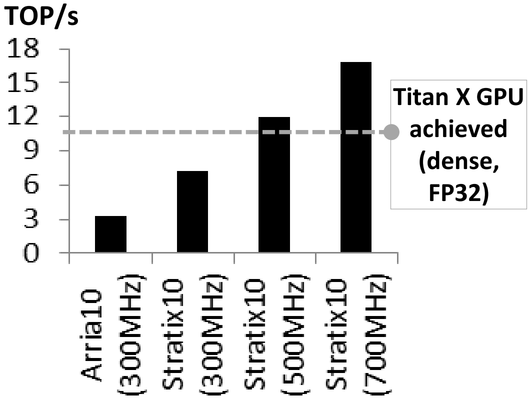
Sparse NN

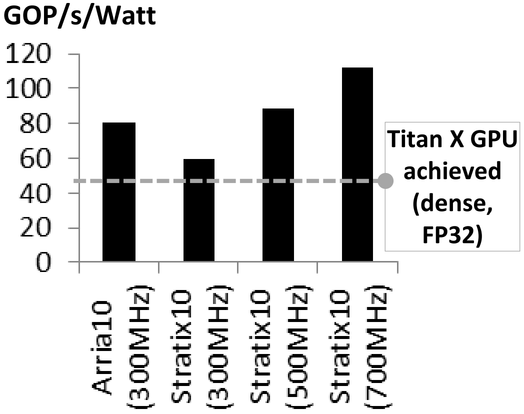
$$
\underset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} 0.2 & 0 & 0.1 \\ 0 & 0.3 & 0 \\ 0.1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1x0.2)+\cancel{(0x0)}+(0.2x0.1) \\ \cancel{(0.1x0)}+\cancel{(0x0.3)}+\cancel{(0.2x0)} \\ (0.1x0.1)+\cancel{(0x0)}+\cancel{(0.2x0)} \end{matrix} = \underset{O}{\begin{bmatrix} \; \\ \; \\ \; \end{bmatrix}}
$$



HW skips computation on zero values

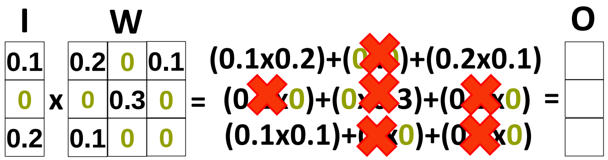
Sparse FP32 GEMM
(at 85% sparsity based on AlexNet)

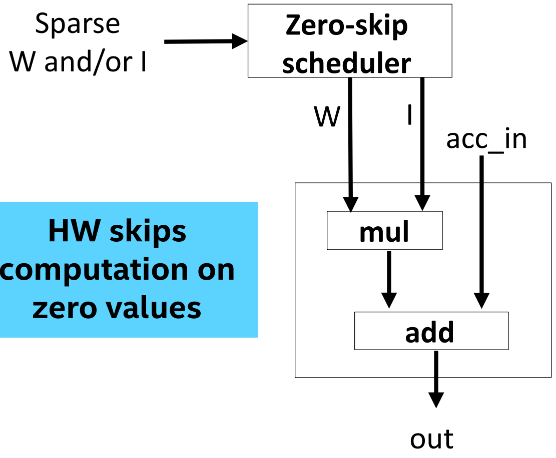




Study 1 results: Sparse GEMM

Sparse NN

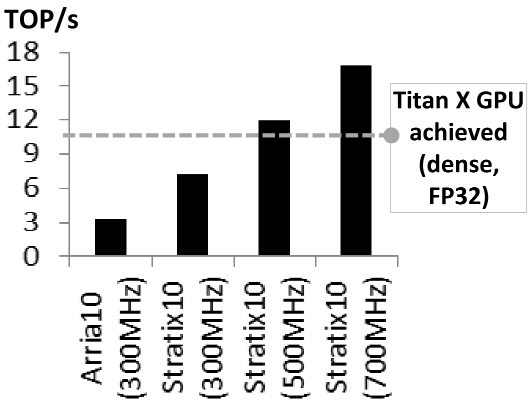


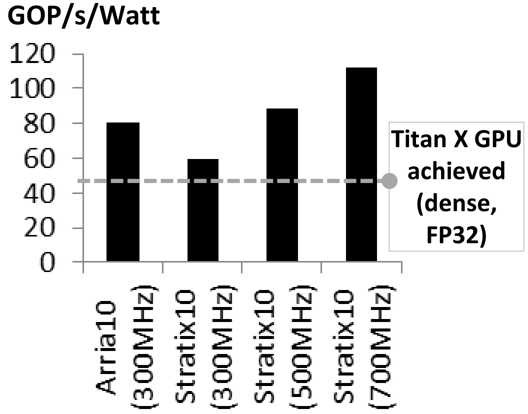


HW skips computation on zero values

Sparse FP32 GEMM

(at 85% sparsity based on AlexNet)

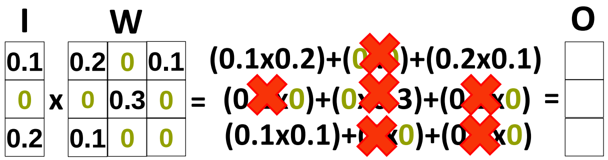


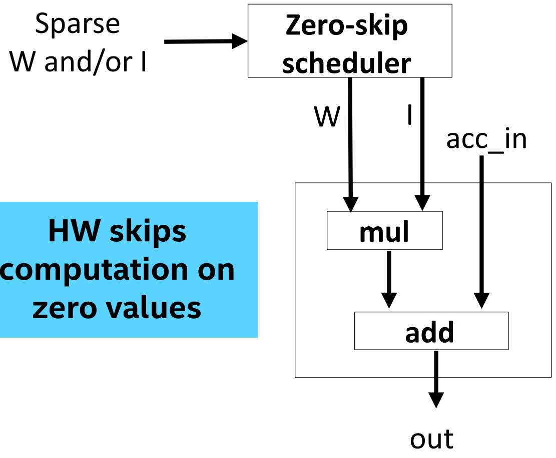


Also tried zero-skipping in GPU, which performed worse than dense. See paper.

Study 1 results: Sparse GEMM

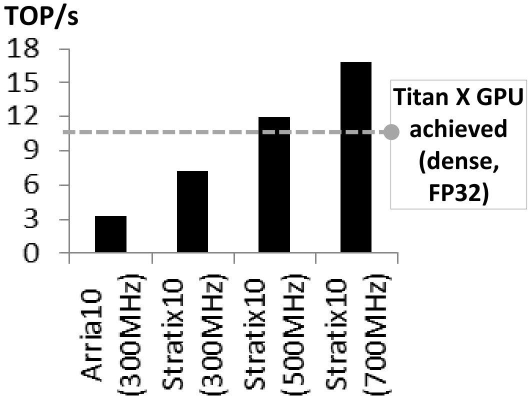
Sparse NN

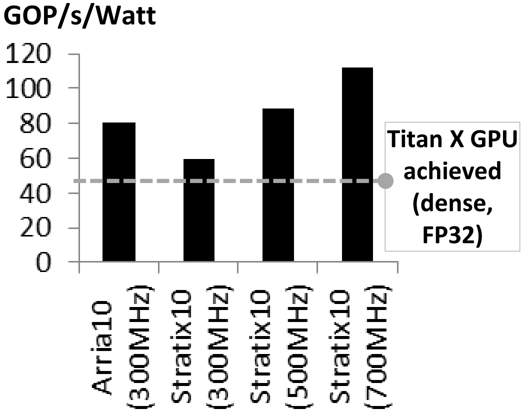




HW skips computation on zero values

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

$$\overset{I}{\begin{bmatrix}0.1\\0\\0.2\end{bmatrix}} \times \overset{W}{\begin{bmatrix}-1 & 0 & +1\\0 & -1 & -1\\+1 & 0 & 0\end{bmatrix}} = \begin{matrix}(0.1\text{x}-1)+(0\text{x}0)+(0.2\text{x}1)\\(0.1\text{x}0)+(0\text{x}-1)+(0.2\text{x}-1)\\(0.1\text{x}1)+(0\text{x}0)+(0.2\text{x}0)\end{matrix} = \overset{O}{\begin{bmatrix}\ \\ \ \\ \ \end{bmatrix}}$$

Study 2: Ternary ResNet Case Study

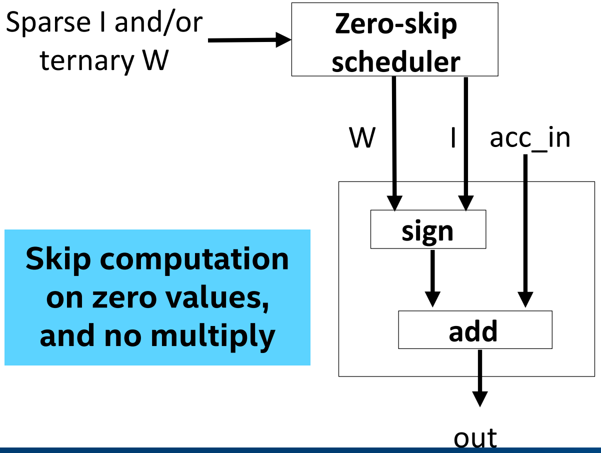
Ternary NN: neural net with weights of +1,-1,0

$$
\overset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \overset{W}{\begin{bmatrix} -1 & 0 & +1 \\ 0 & -1 & -1 \\ +1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1x\text{-}1)+\cancel{(0x0)}+(0.2x1) \\ \cancel{(0.1x0)}+\cancel{(0x\text{-}1)}+(0.2x\text{-}1) \\ (0.1x1)+\cancel{(0x0)}+\cancel{(0.2x0)} \end{matrix} = \overset{O}{\begin{bmatrix} \\ \\ \end{bmatrix}}
$$

Study 2: Ternary ResNet Case Study

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

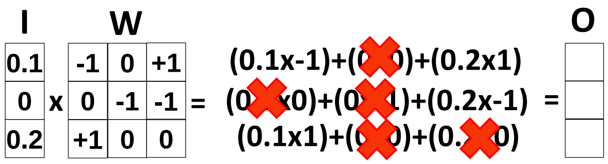
$$
\underset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} -1 & 0 & +1 \\ 0 & -1 & -1 \\ +1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1\text{x}-1)+\cancel{(0\text{x}0)}+(0.2\text{x}1) \\ \cancel{(0\text{x}0)}+\cancel{(0\text{x}-1)}+(0.2\text{x}-1) \\ (0.1\text{x}1)+\cancel{(0\text{x}0)}+\cancel{(0.2\text{x}0)} \end{matrix} = \underset{O}{\begin{bmatrix} \; \\ \; \\ \; \end{bmatrix}}
$$

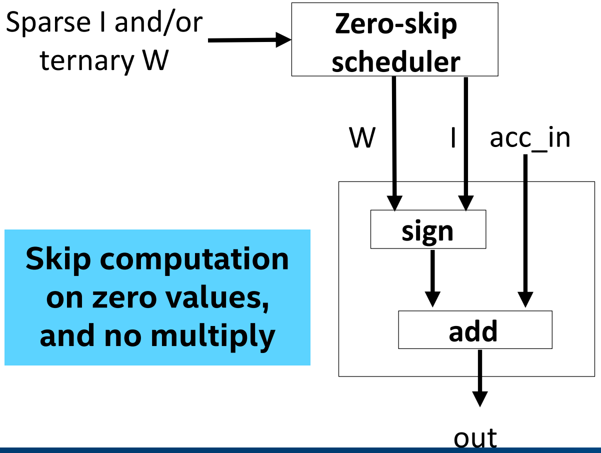


Skip computation on zero values, and no multiply

Study 2: Ternary ResNet Case Study

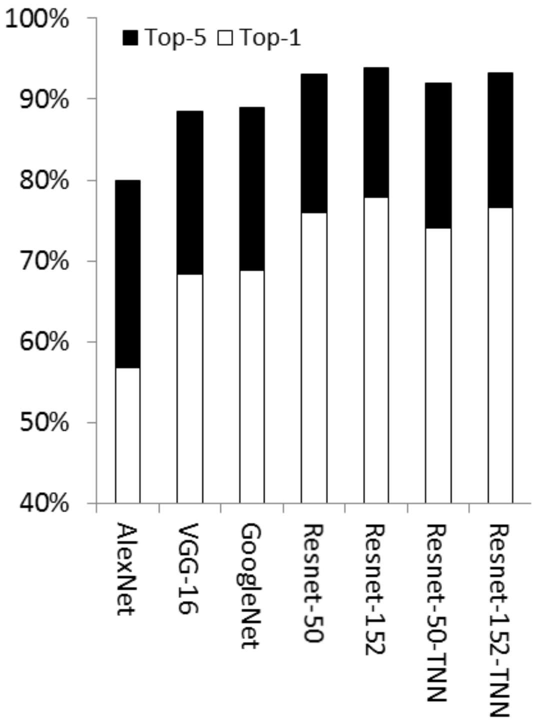
Ternary NN: neural net with weights of +1,-1,0





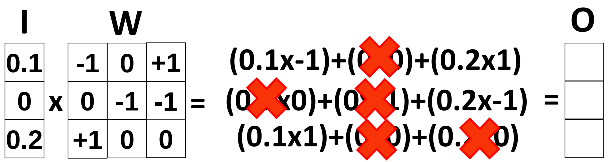
Skip computation on zero values, and no multiply

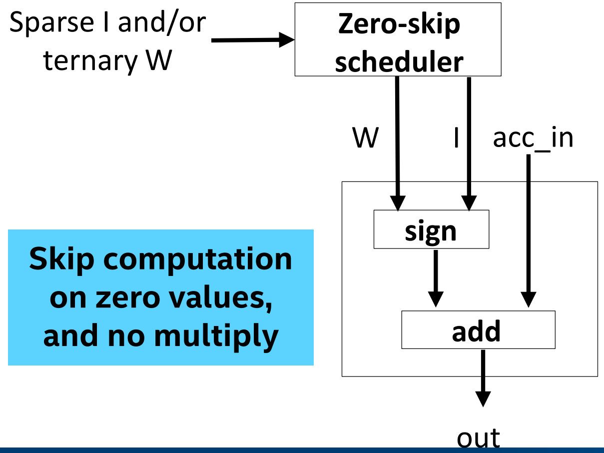
ImageNet Accuracy



Study 2: Ternary ResNet Case Study

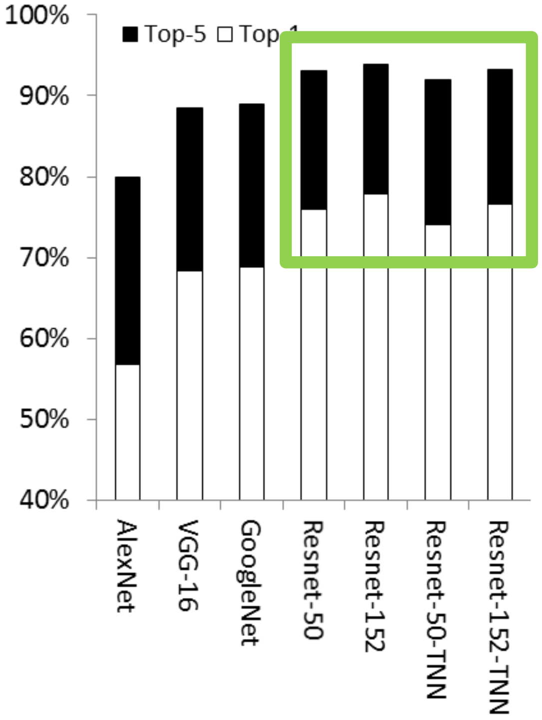
Ternary NN: neural net with weights of +1,-1,0





Skip computation on zero values, and no multiply

ImageNet Accuracy



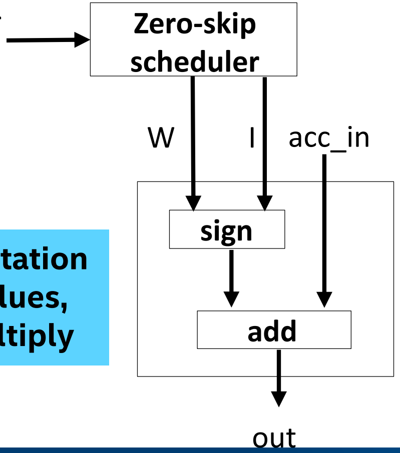
Ternary ResNet offers state-of-the-art accuracy [ICASSP'17]

Study 2: Ternary ResNet Case Study

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

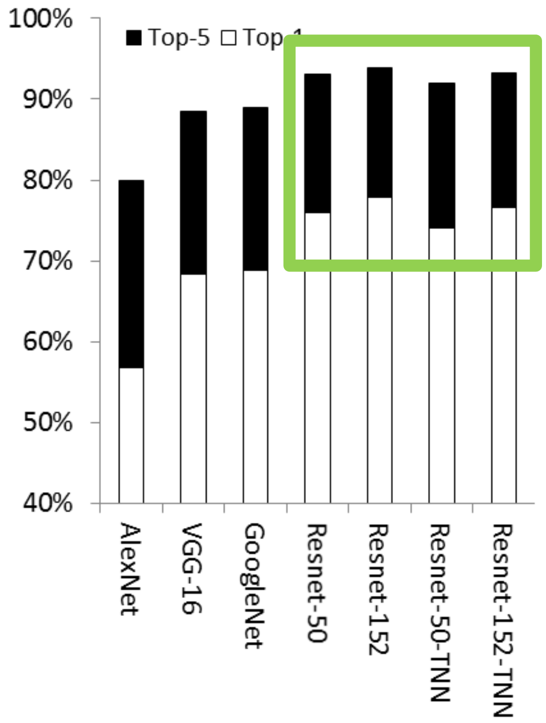
$$\underset{I}{\begin{bmatrix} 0.1 \\ 0 \\ 0.2 \end{bmatrix}} \times \underset{W}{\begin{bmatrix} -1 & 0 & +1 \\ 0 & -1 & -1 \\ +1 & 0 & 0 \end{bmatrix}} = \begin{matrix} (0.1\text{x-}1)+(\cancel{0\text{x}0})+(0.2\text{x}1) \\ (\cancel{0\text{x}0})+(\cancel{0\text{x-}1})+(0.2\text{x-}1) \\ (0.1\text{x}1)+(\cancel{0\text{x}0})+(\cancel{0.2\text{x}0}) \end{matrix} = \underset{O}{\begin{bmatrix} \ \\ \ \\ \ \end{bmatrix}}$$

Sparse I and/or ternary W



Skip computation on zero values, and no multiply

ImageNet Accuracy

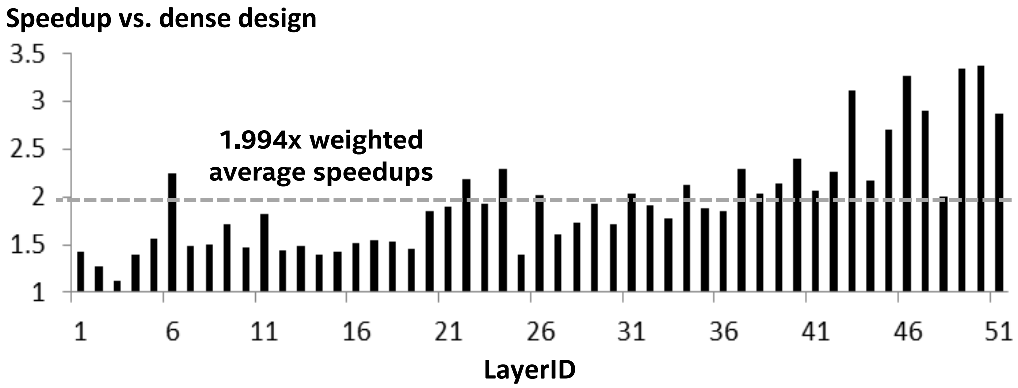


Ternary ResNet offers state-of-the-art accuracy [ICASSP'17]

We target Resnet-50-TNN in this study

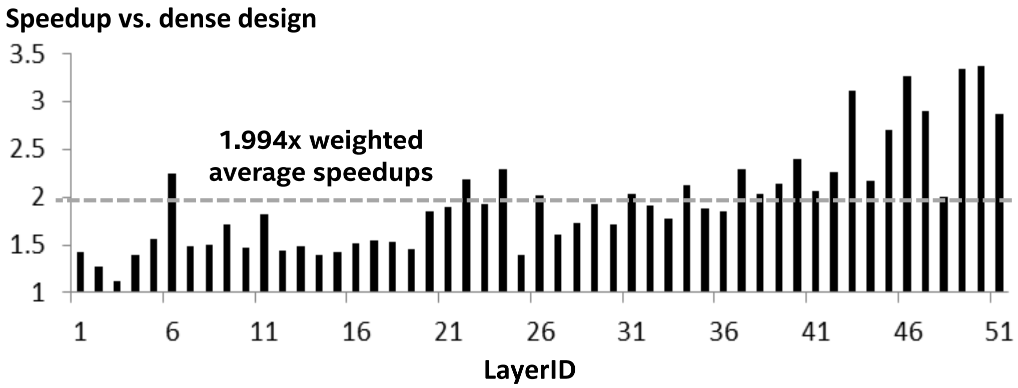
Results

Per layer speedup varies depends on sparsity

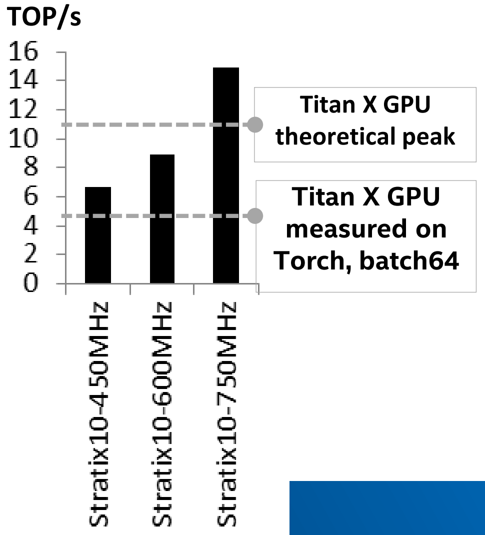


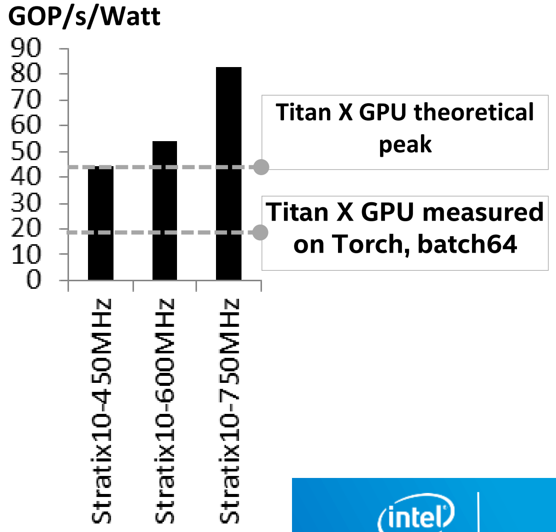
Results

Per layer speedup varies depends on sparsity



S10 FPGA performs better, across all frequency targets





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?

intel®

experience what's inside™