

On Matrix Sparsification and Quantization for Efficient and Scalable Deep Learning

Wei Wen

*Electrical and Computer Engineering
Duke University*

The Scale of Training Data

Image



[Source](#)

[ImageNet](#)

1.2 Million

[Open Images Dataset](#)

9 Million

[MS-COCO](#)

1.5 Million Objects

Video



[Source](#)

[YouTube-8M](#)

1.9 Billion Frame Features

[Sports-1M](#)

1.1 Million Videos

Text



[Source](#)

[The Wikipedia Corpus](#)

4.4M Articles & 1.9B Words

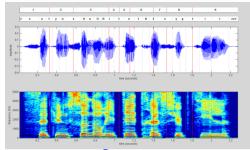
[Yelp Reviews](#)

6M Reviews

[WMT 18](#)

30M Sentences

Speech



[Source](#)

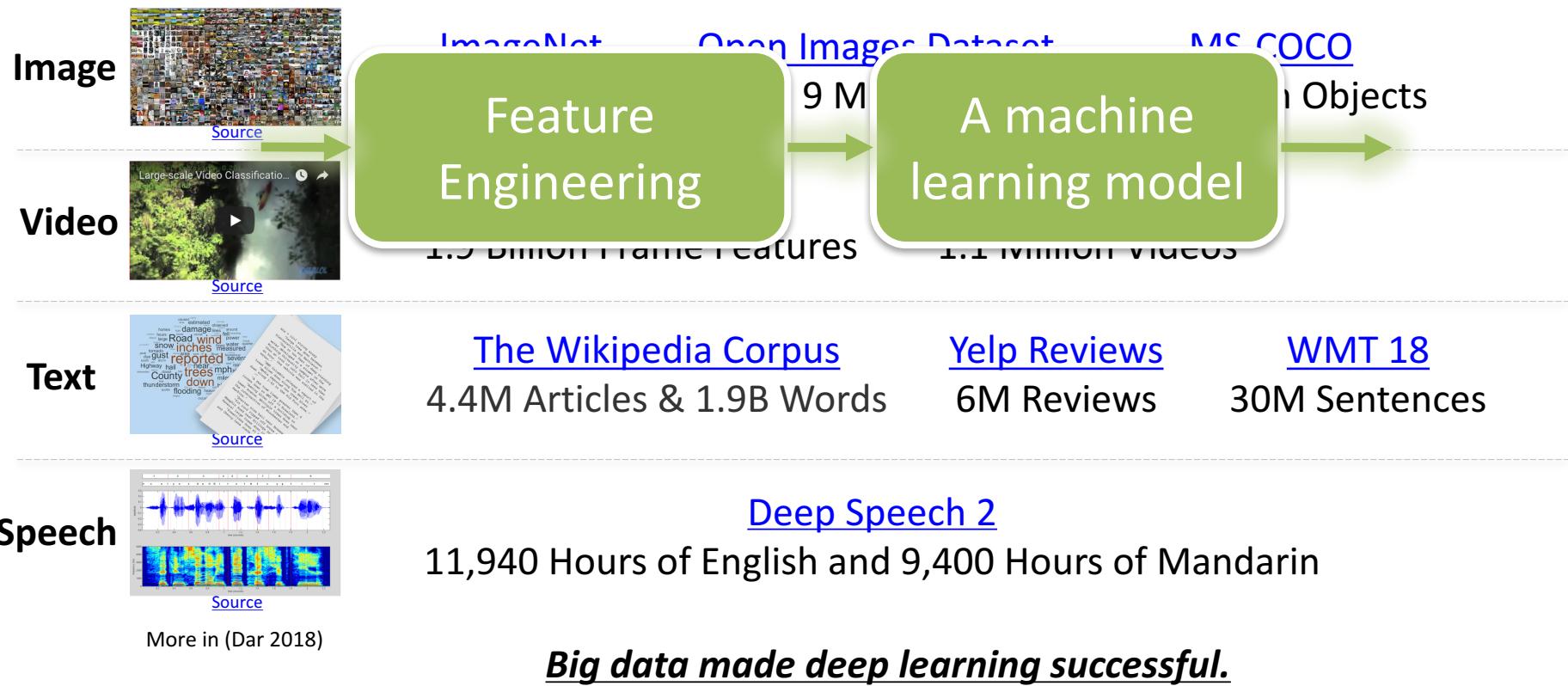
[Deep Speech 2](#)

11,940 Hours of English and 9,400 Hours of Mandarin

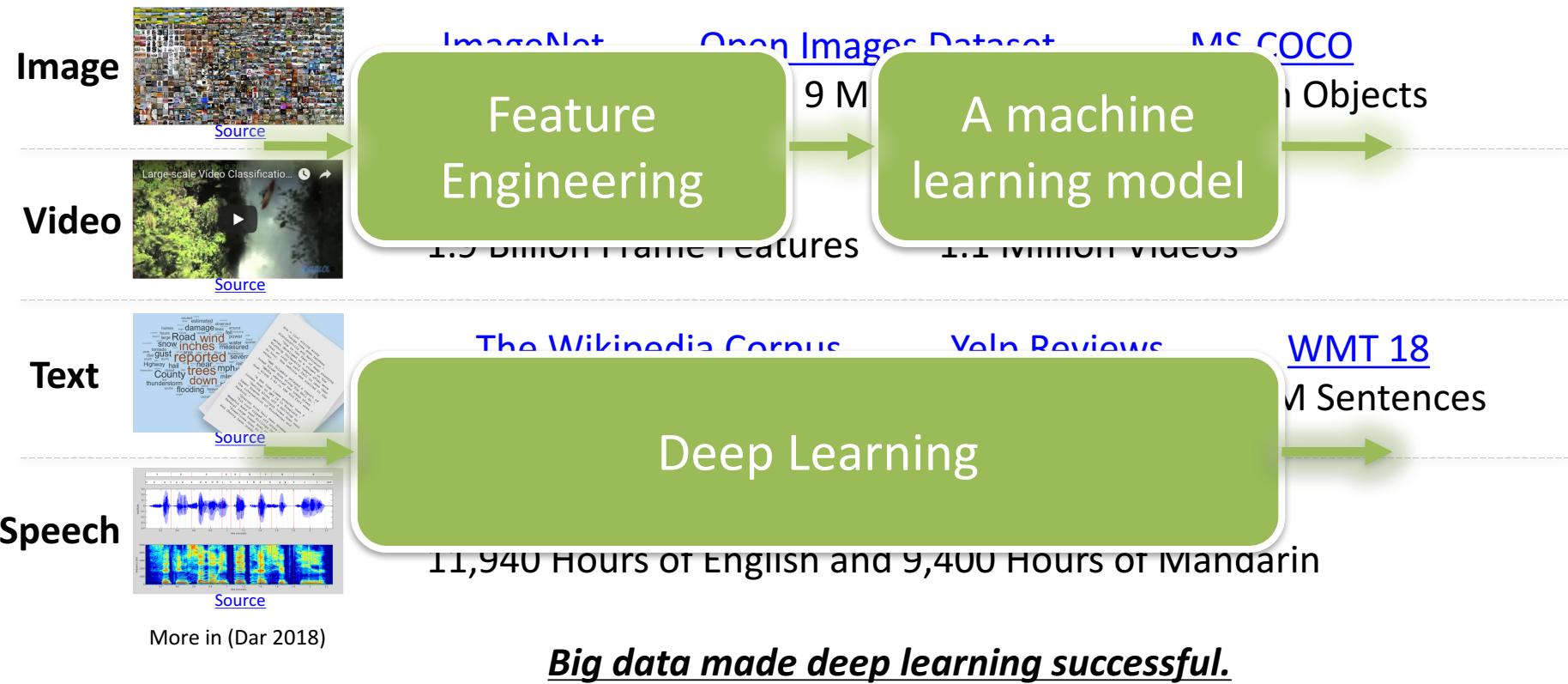
More in (Dar 2018)

Big data made deep learning successful.

The Scale of Training Data

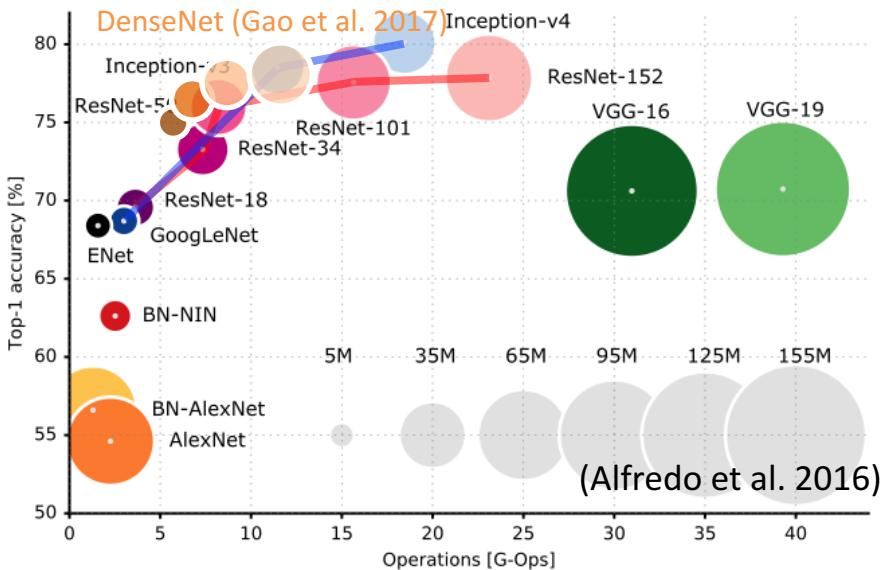


The Scale of Training Data

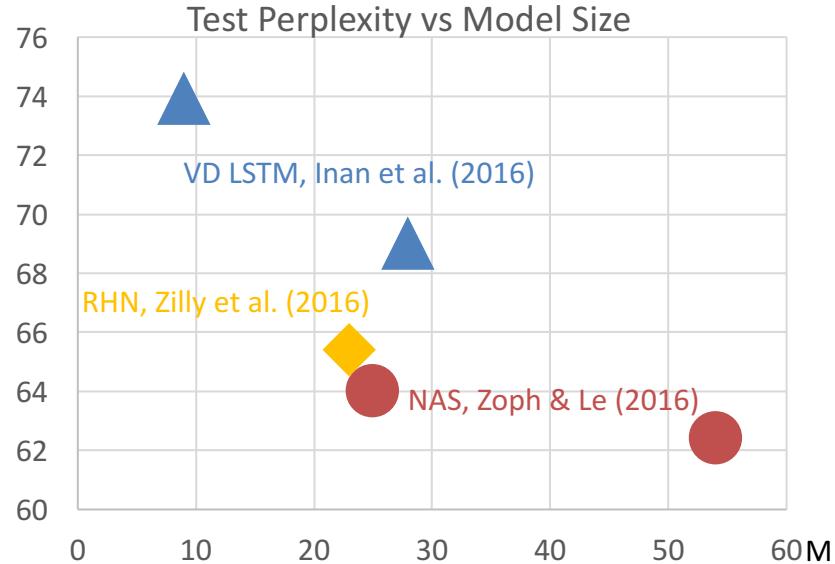


The Scale of Deep Learning Models

Image Classification on ImageNet



Language Modeling on Penn Treebank



More accurate prediction by paying computation cost using larger models.

Challenges of Deploying Deep Learning Models



[Wired](#)



[OSXDaily](#)



[Fortune](#)



[Google Cloud AI](#)



[Wired](#)

"Prototypes use around 2,500 watts"



[PopularMechanics](#)

A machine: 48 CPUs and 8 GPUs; a distributed version: 1,202 CPUs and 176 GPUs (Silver et al. 2016)

We must compress and accelerate deep learning models.

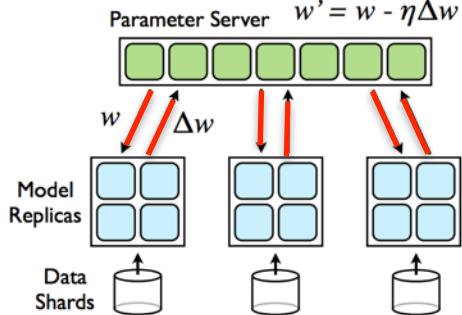
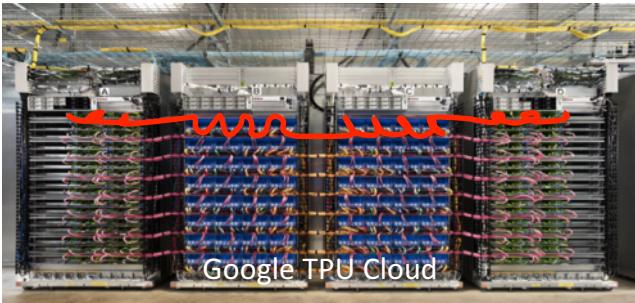
Inference on the edge:

- Limited computing capability
- Limited memory
- Limited battery energy

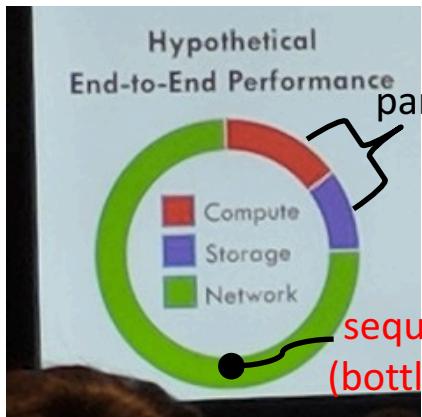
Inference in the cloud:

- A challenge: Real-time response to trillions of AI service requests
- Facebook (K. Hazelwood@ISCA'18)
 - 200+ Trillion predictions per day
 - 5+ Billion language translations per day

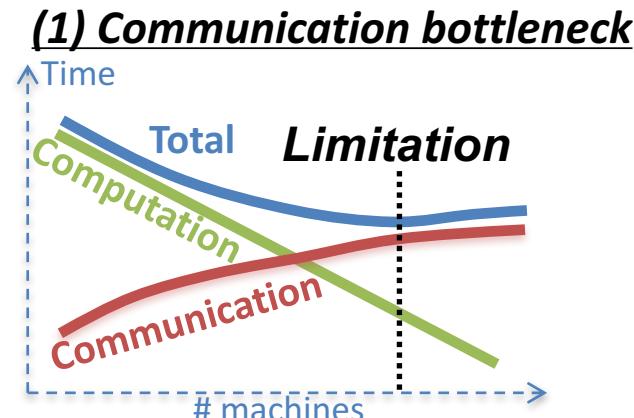
Challenges of Training Deep Learning Models



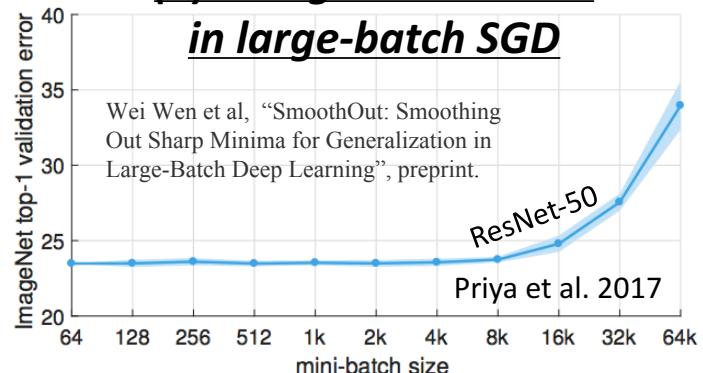
Jeffrey et al. 2012



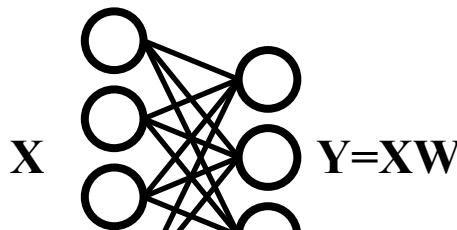
(K. Hazelwood@ISCA'18) Picture: leiphone



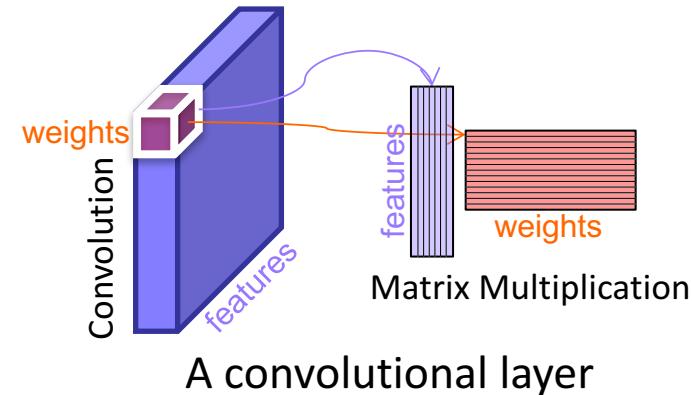
**(2) Bad generalization
in large-batch SGD**



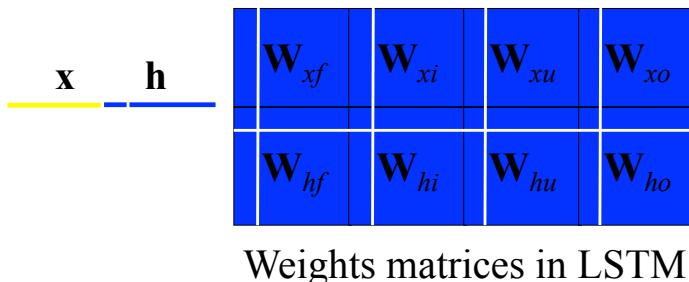
Matrix Computation and Communication in Deep Learning



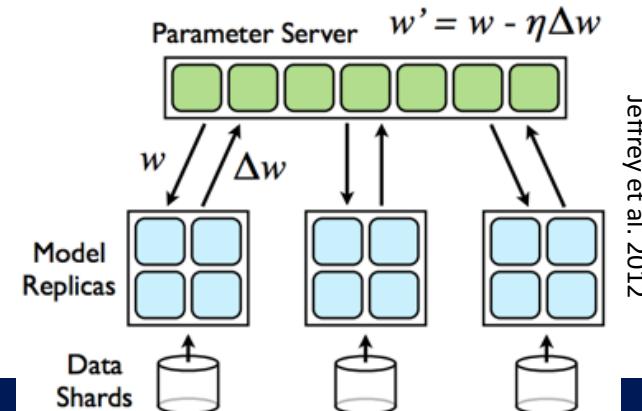
A fully-connected layer



A convolutional layer



Weights matrices in LSTM



Jeffrey et al. 2012

Research Highlights

- Efficient inference of deep learning models
 - Structurally sparse DNNs (NIPS 2016 & ICLR 2018)
 - Lower-rank DNNs (ICCV 2017)
 - A compact DNN considering domain adaptation (CVPR 2017)
 - Direct sparse convolution (ICLR 2017)
 - Neuromorphic chip on the edge (DAC 2015 & DAC 2016; best paper nominations)
- Scalable distributed deep learning
 - TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning, NIPS 2017 (oral)
 - SmoothOut: closing generalization gap in large-batch SGD

Research Highlights

- Efficient inference of deep learning models
 - Structurally sparse DNNs (NIPS 2016 & ICLR 2018)
 - Lower-rank DNNs (ICCV 2017)
 - A compact DNN considering domain adaptation (CVPR 2017)
 - Direct sparse convolution (ICLR 2017)
 - Neuromorphic chip on the edge (DAC 2015 & DAC 2016; best paper nominations)
- Scalable distributed deep learning
 - TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning, NIPS 2017 (oral)
 - SmoothOut: closing generalization gap in large-batch SGD



[Wired](#)



Face ID



[OSXDaily](#)



[Fortune](#)



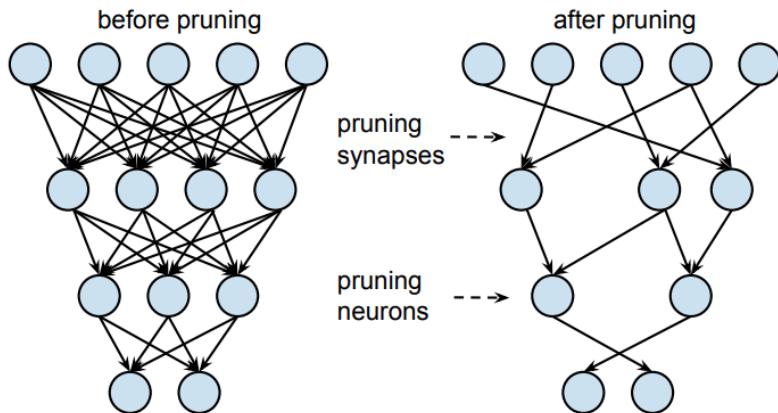
Google Cloud AI

Structurally Sparse Deep Neural Networks

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li, “[Learning Structured Sparsity in Deep Neural Networks](#)”, NIPS, 2016. [[paper](#)][[code](#)][[poster](#)]

Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, Hai Li, “[Learning Intrinsic Sparse Structures within Long Short-Term Memory](#)”, ICLR, 2018. [[paper](#)][[code](#)]

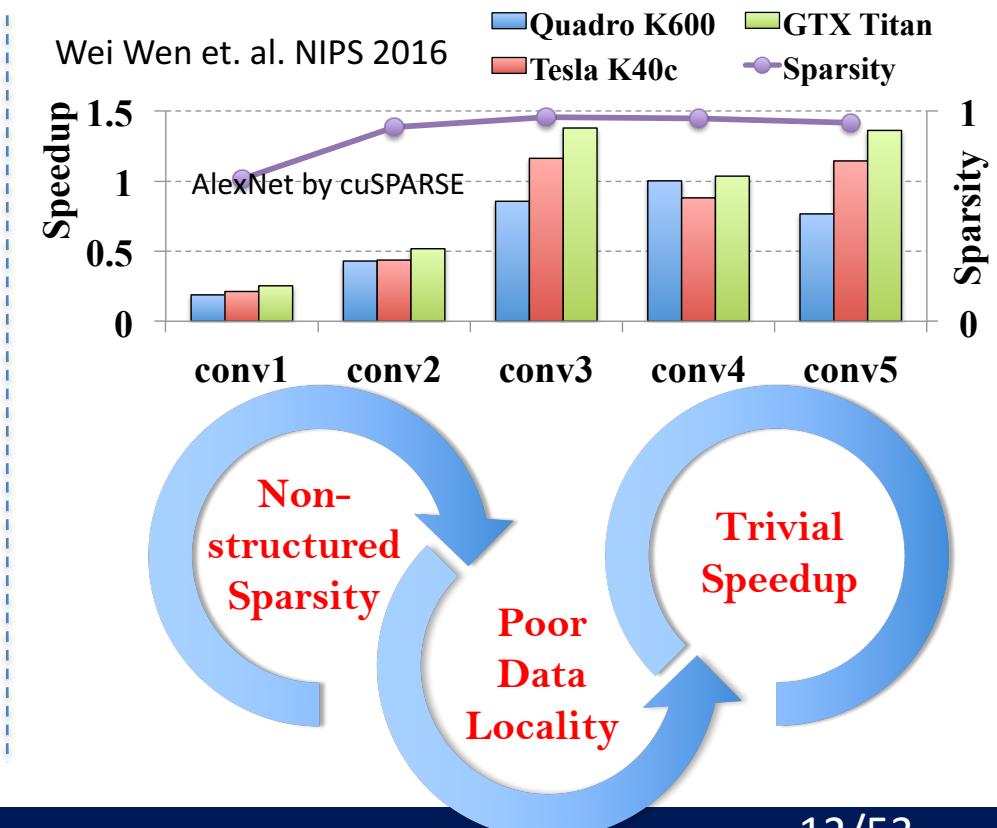
Inefficiency of Sparse Convolutional Neural Networks



Significantly reduces the storage size of DNNs [1]
Good speedup with customized hardware [2]

[1] S. Han et. al. NIPS 2015

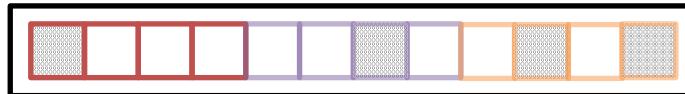
[2] S. Han et. al. ISCA 2016.



Inefficiency of Non-structured Sparse Computation

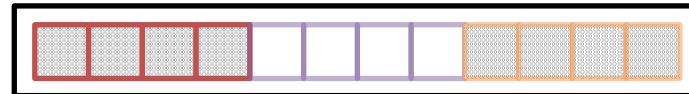
Dense feature matrix * sparse weight matrix

Features in a Memory



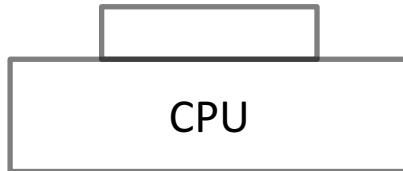
Non-structured sparse access

Features in a Memory

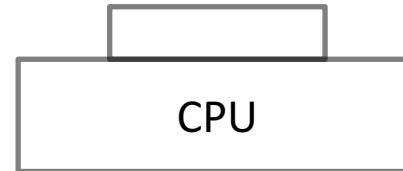


Structured sparse access

Cache



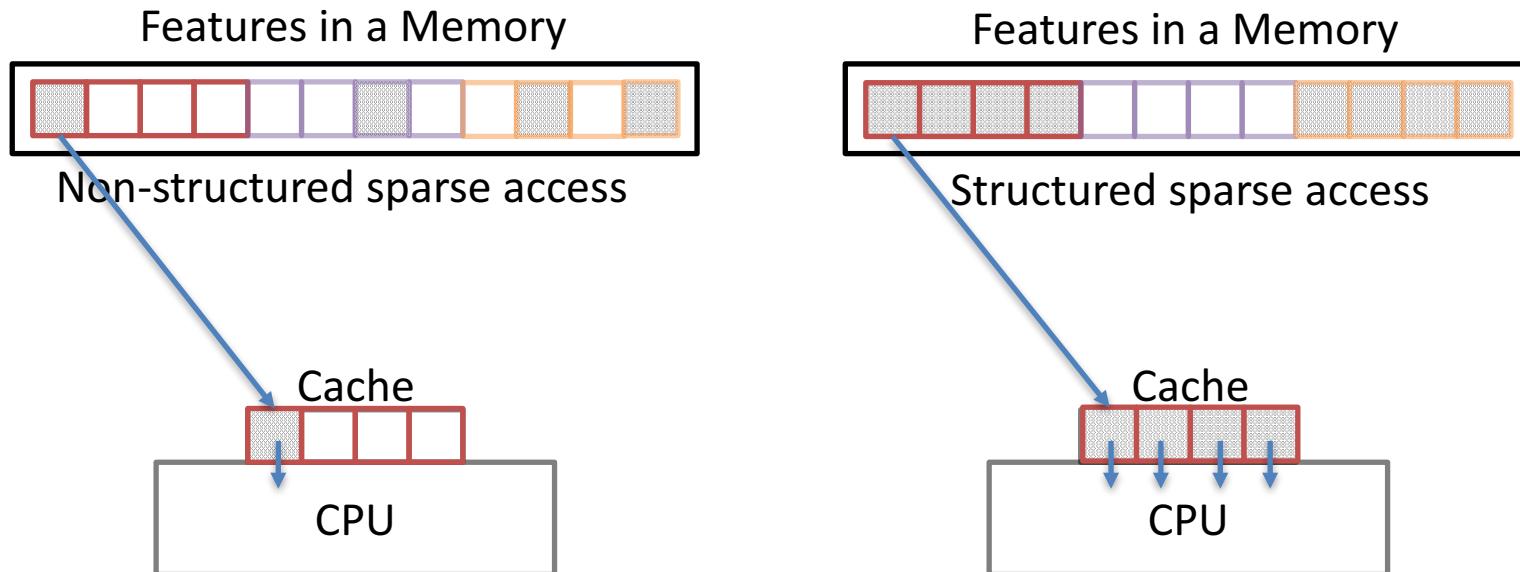
Cache



A toy example.

Inefficiency of Non-structured Sparse Computation

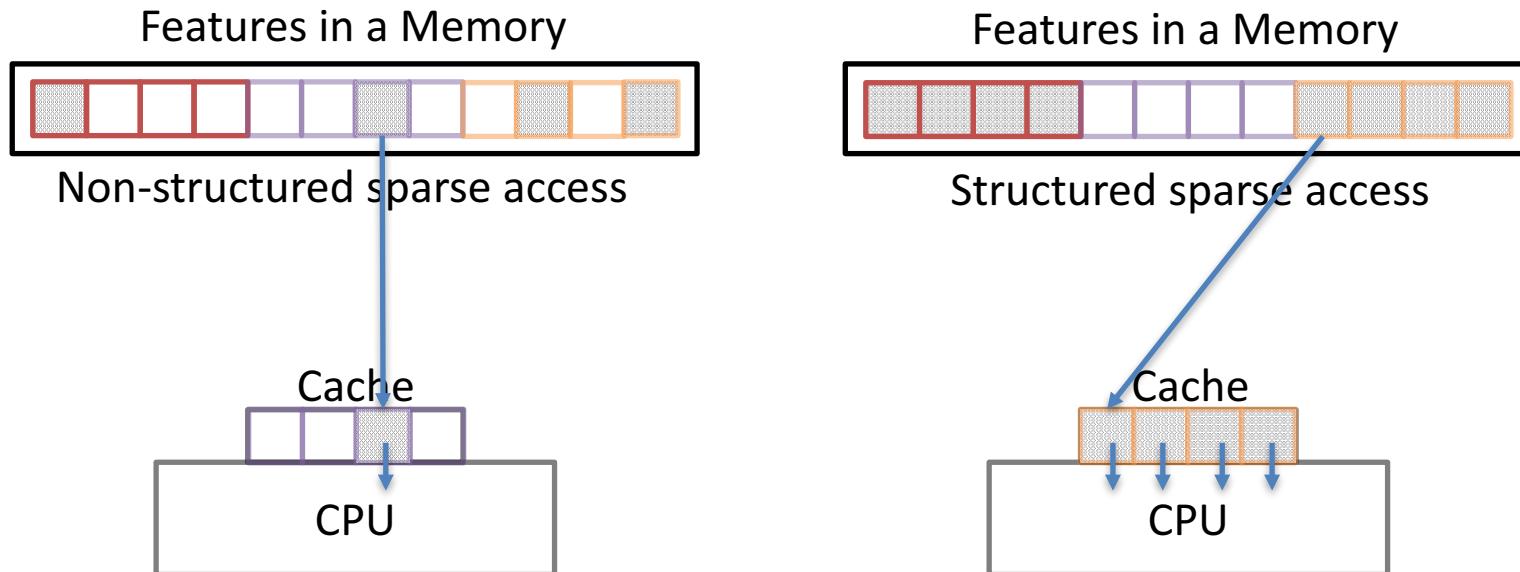
Dense feature matrix * sparse weight matrix



A toy example.

Inefficiency of Non-structured Sparse Computation

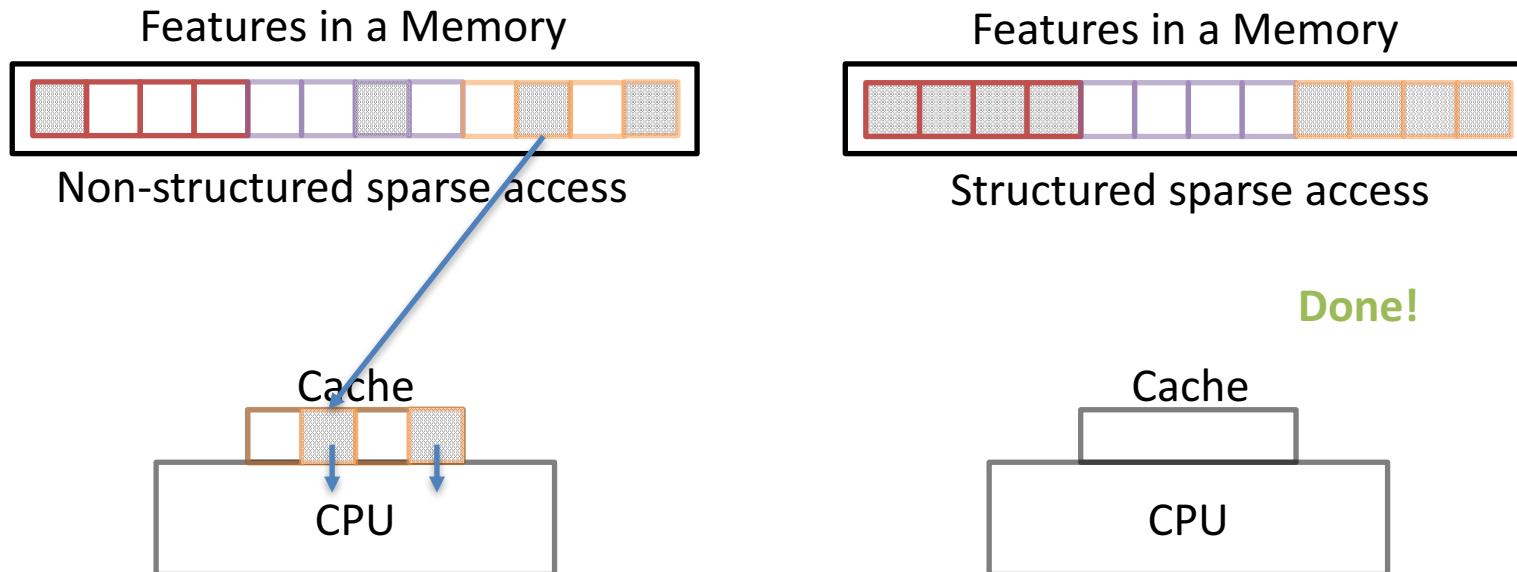
Dense feature matrix * sparse weight matrix



A toy example.

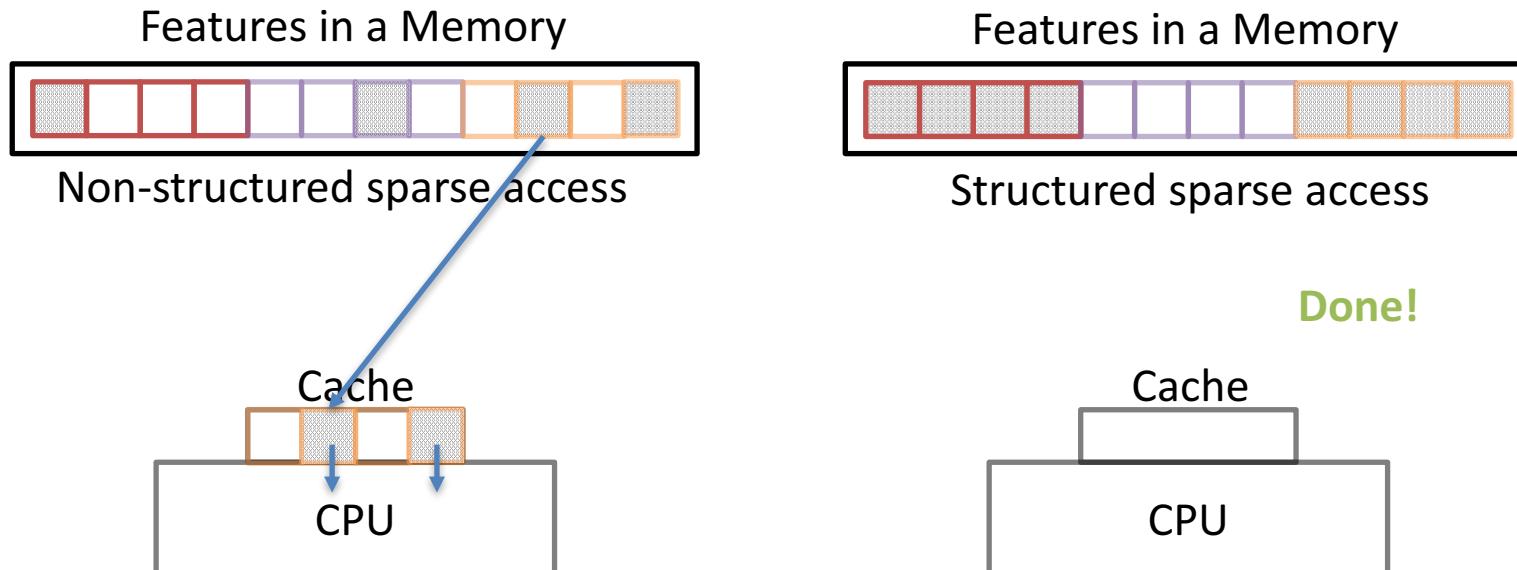
Inefficiency of Non-structured Sparse Computation

Dense feature matrix * sparse weight matrix



Inefficiency of Non-structured Sparse Computation

Dense feature matrix * sparse weight matrix



**Breaks regularity & breaks
hardware parallelism**

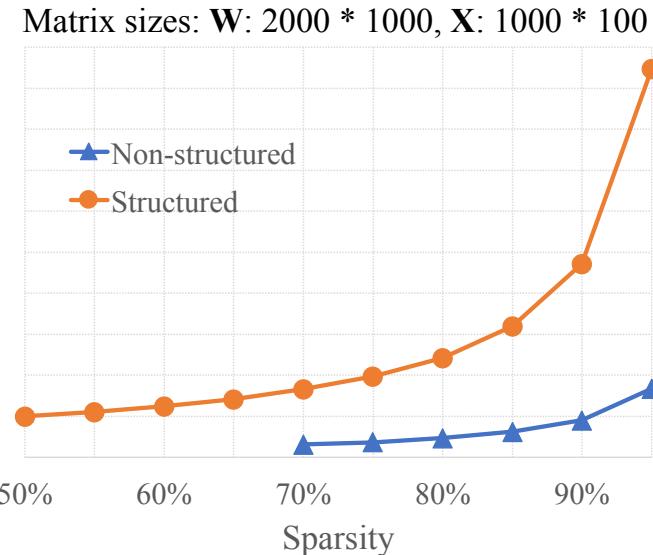
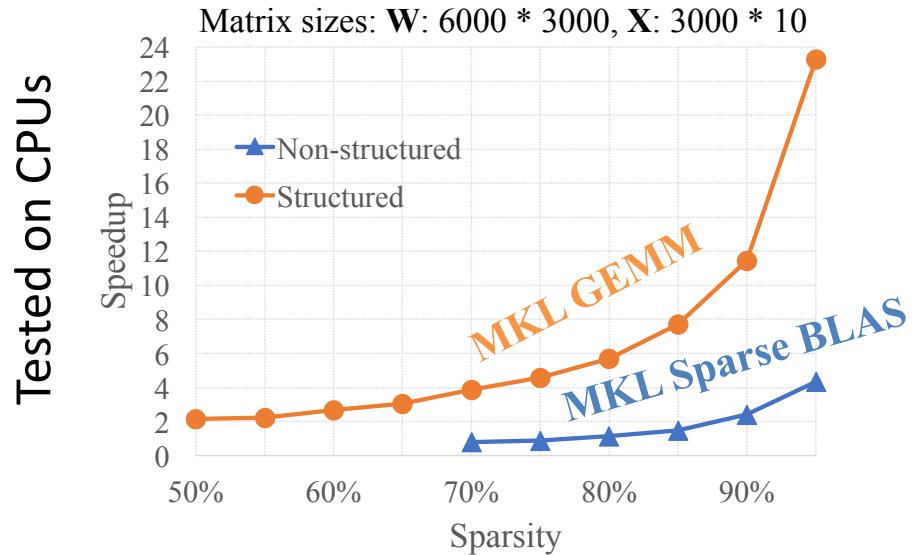
A toy example.

**Keeps regularity & keeps
hardware parallelism**

Done!

Efficiency of Structured Sparsity

W. Wen et. al. ICLR 2018



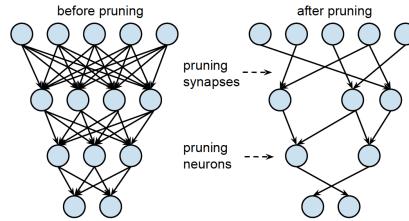
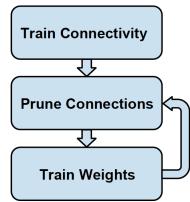
Structured sparsity



Non-structured sparsity



Structurally Sparse DNNs

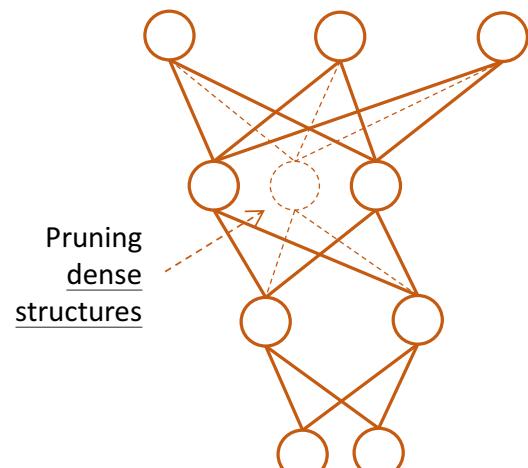
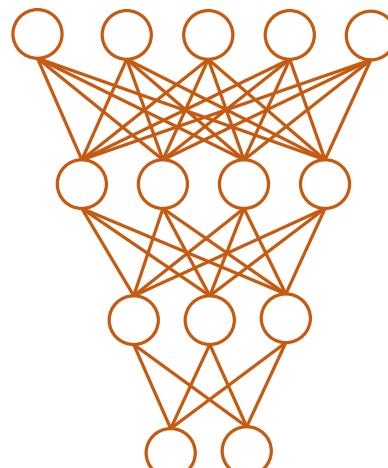


Faster training speed &
Faster inference speed

Wei Wen et. al. NIPS 2016
Wei Wen et. al. ICLR 2018

Train weights with
Group Lasso
Regularization

One-shot pruning from scratch
(with the same epochs)

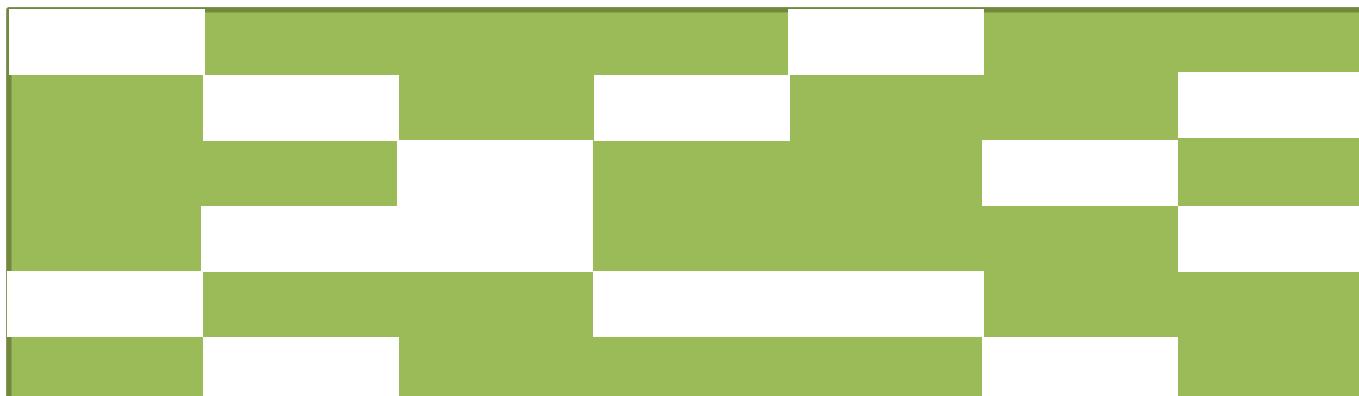


Structurally Sparse Deep Neural Networks

- What is Structurally Sparse Deep Neural Networks (SSDNNs)?

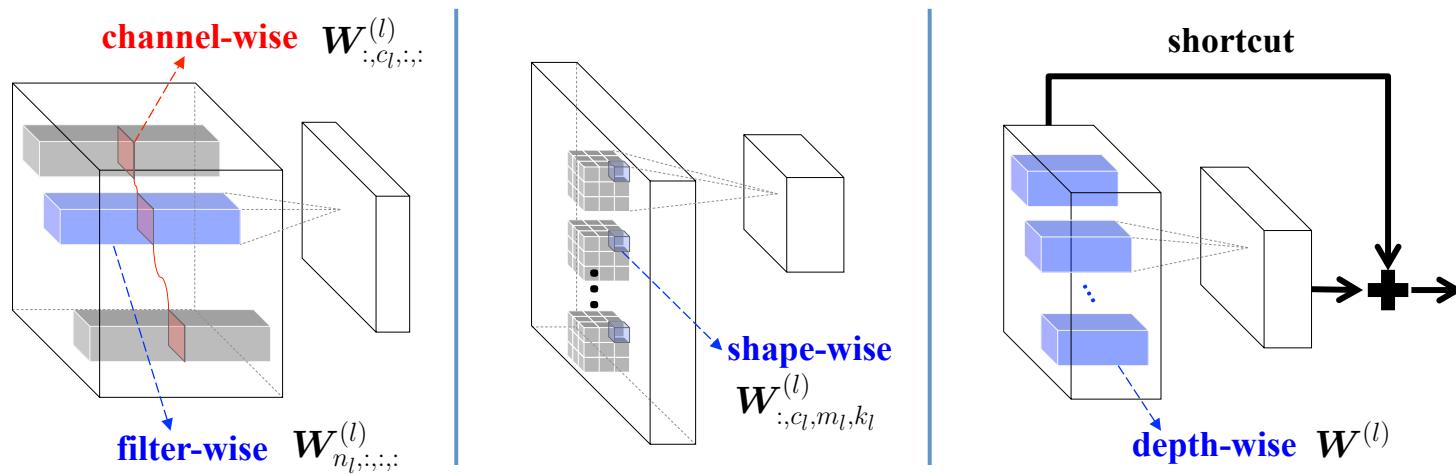
Weights/connections are removed group by group.

A group can be a rectangle block, a row, a column or even the whole matrix



Structurally Sparse Deep Neural Networks

In CNNs, a group of weights can be a channel, a 3D filter, a 2D filter, a filter shape fiber (i.e. a weight column), and even a layer (in ResNets).



Group Lasso Regularization is All you Need for SSDNNs

Step 1: Weights are split to G groups $\mathbf{w}^{(1..G)}$

e.g. $(w_1, w_2, w_3, w_4, w_5) \rightarrow$ group 1: (w_1, w_2, w_3) , group 2: (w_4, w_5)

Step 2: Add group Lasso on each group $\mathbf{w}^{(g)}$ $\|\mathbf{w}^{(g)}\|_g = \sqrt{\sum_{i=1}^{|w^{(g)}|} (w_i^{(g)})^2}$ i.e. vector length
 $\text{sqrt}(w_1^2 + w_2^2 + w_3^2), \text{sqrt}(w_4^2 + w_5^2)$

Step 3: Sum group Lasso over all groups as a regularization: $R_g(\mathbf{w}) = \sum_{g=1}^G \|\mathbf{w}^{(g)}\|_g$,
 $\text{sqrt}(w_1^2 + w_2^2 + w_3^2) + \text{sqrt}(w_4^2 + w_5^2)$

Step 4: SGD optimizing $\arg \min_{\mathbf{w}} \{E(\mathbf{w})\} = \arg \min_{\mathbf{w}} \{E_D(\mathbf{w}) + \lambda_g \cdot R_g(\mathbf{w})\}$

We refer to our method as ***Structured Sparsity Learning (SSL)***

Structured Sparsity Learning (SSL)

- Why can SSL learn to remove weights group by group?

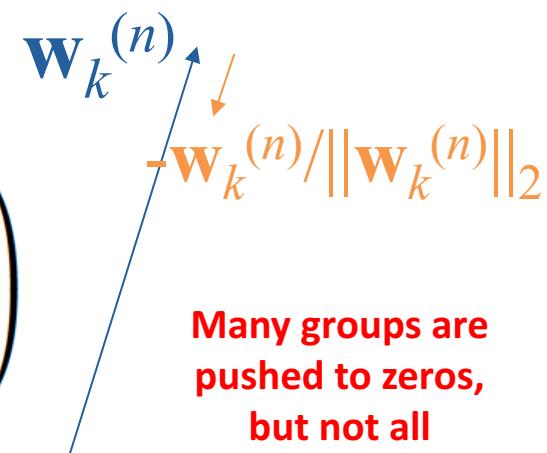
The gradient-based explanation:

$\boxed{\mathbf{w}_k^{(n)}}$ ← $\mathbf{w}_k^{(n)} - \eta \cdot \left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_k^{(n)}} + \lambda \cdot \frac{\mathbf{w}_k^{(n)}}{\|\mathbf{w}_k^{(n)}\|_2} \right)$

A group of weights

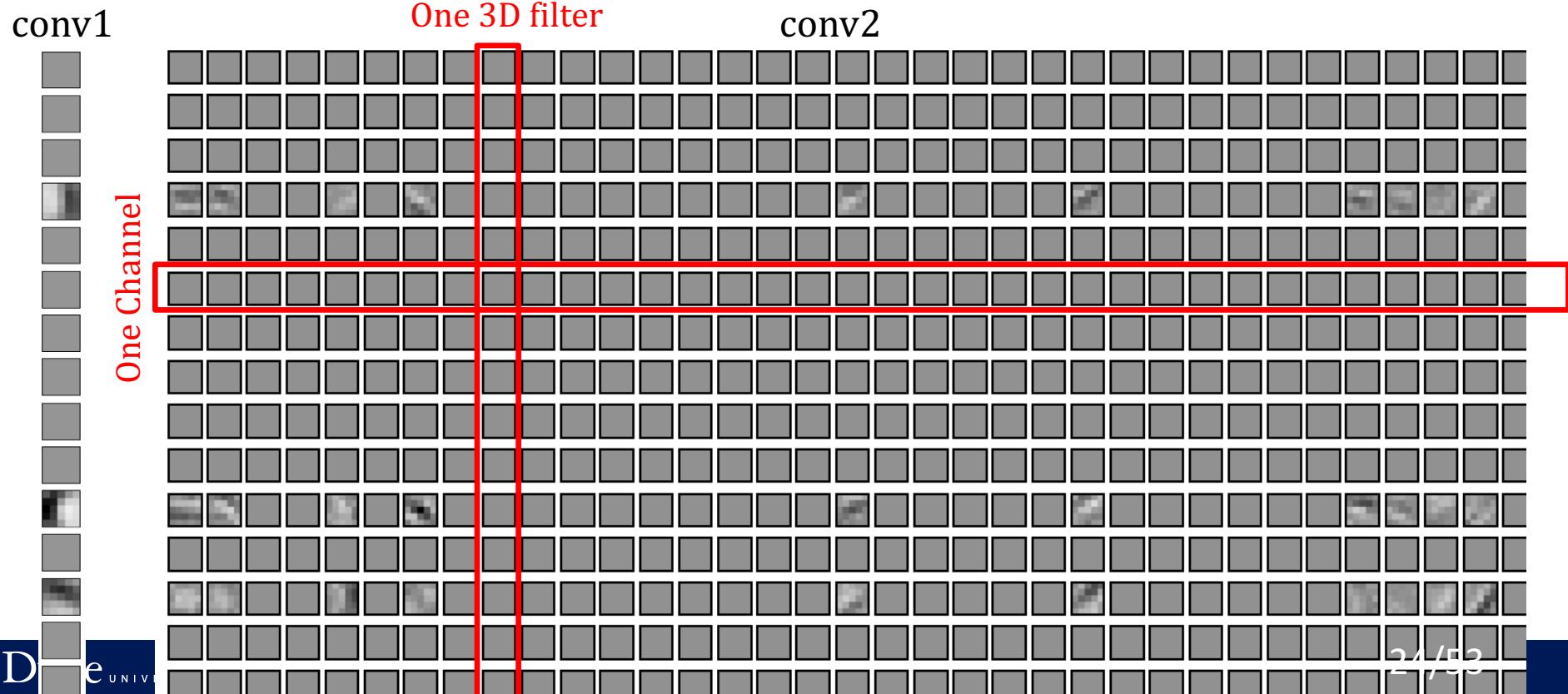
Regular gradients
to minimize error

Additional gradients
to learn sparsity

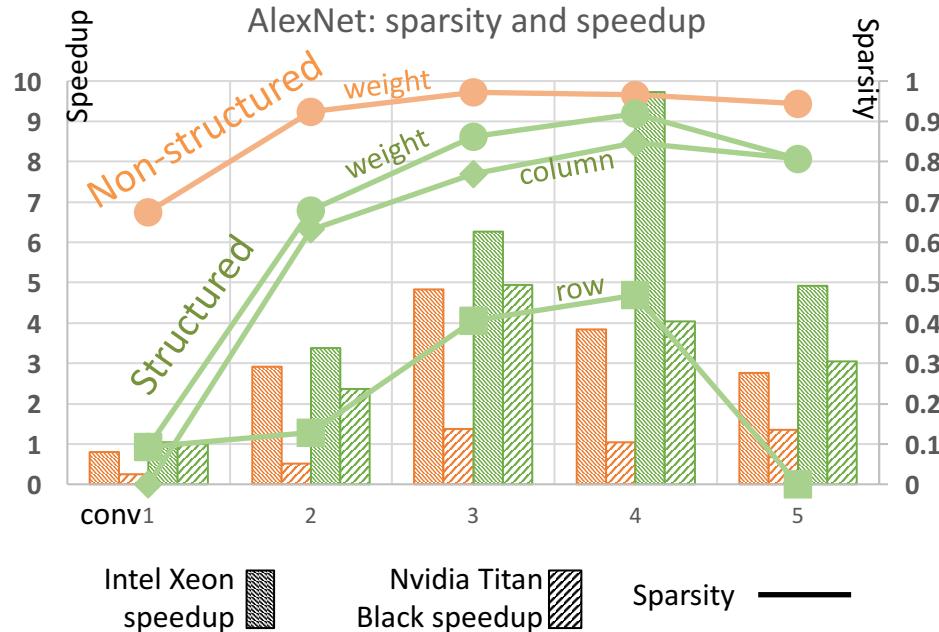


Many groups are pushed to zeros,
but not all

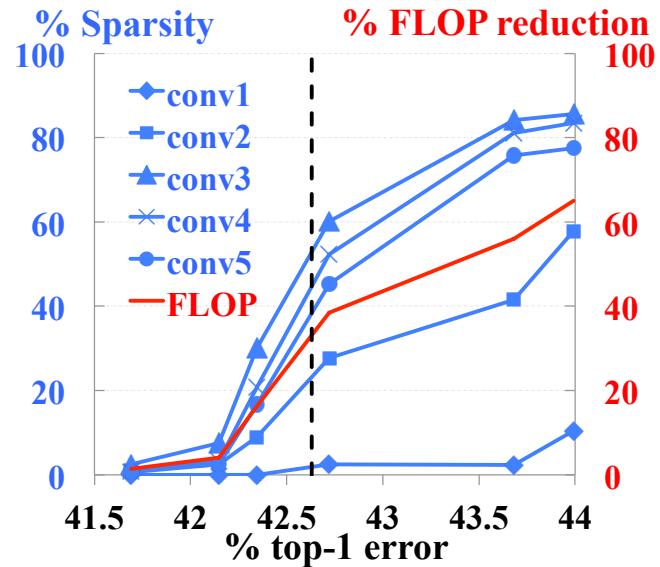
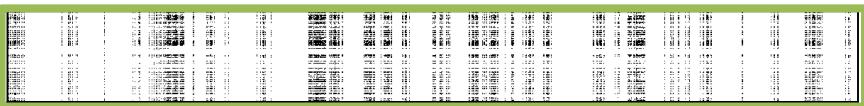
Structurally Sparse LeNet – Removing Channels and Filters



Structurally Sparse *AlexNet*



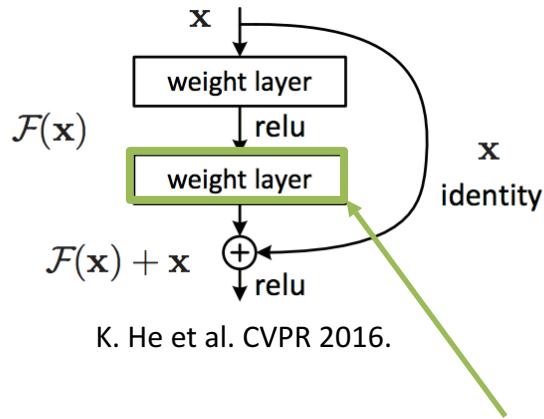
Removing Columns and Rows:



Removing 2D convolutions:



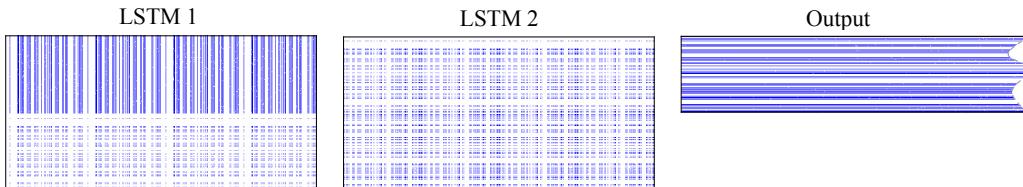
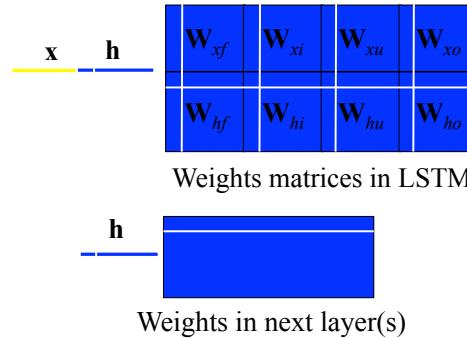
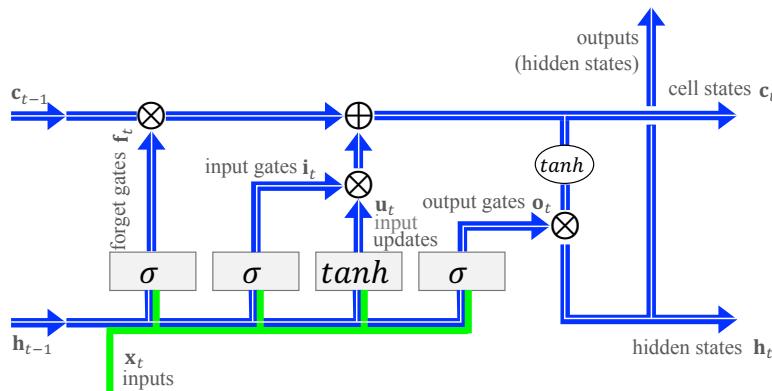
Structurally Sparse ResNets – Removing Layers



One layer is
one group.

	# layers	error	# layers	error
ResNet	20	8.82%	32	7.51%
SSL-ResNet	14	8.54%	18	7.40%
Reduced	6/20		14/32	

Structurally Sparse LSTMs: Removing Hidden Structures



White strips (ISS: Intrinsic Sparse Structures) = one group of weights in SSL

Deployment of Structurally Sparse DNNs

Create a smaller DNN with the learned structure



Initialize the smaller DNN by the non-zero parameters



Deploy



Faster inference
Easier deployment

A black box
(No software or
hardware
customization)



[Wired](#)



[OSXDaily](#)



[Fortune](#)



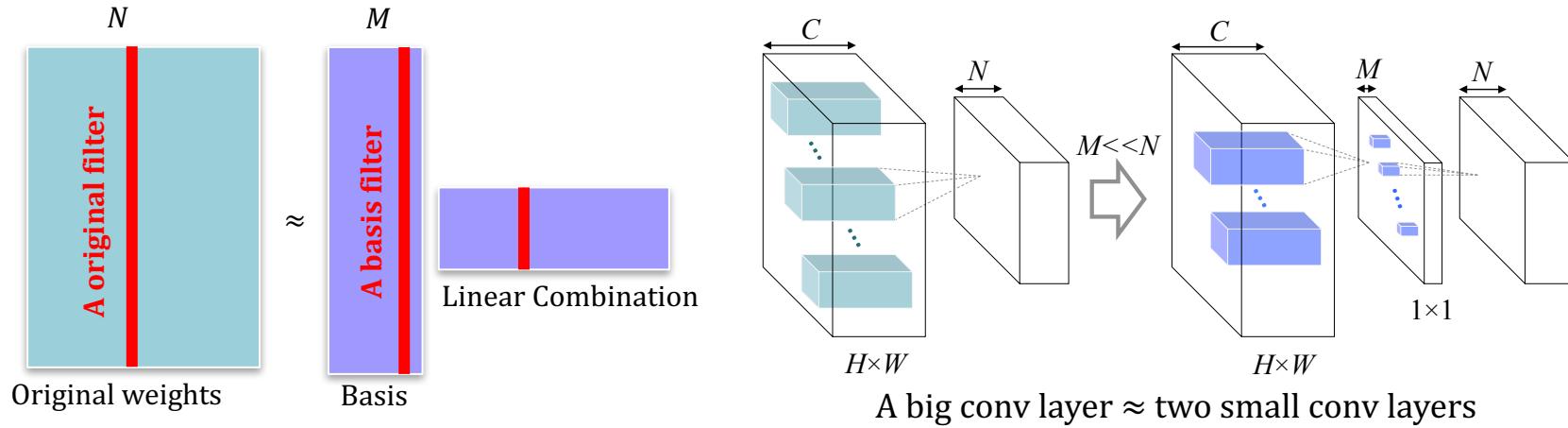
[Google Cloud AI](#)

Lower-rank Deep Neural Networks

Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li, “Coordinating Filters for Faster Deep Neural Networks”, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. [\[paper\]](#)[\[code\]](#)[\[poster\]](#)

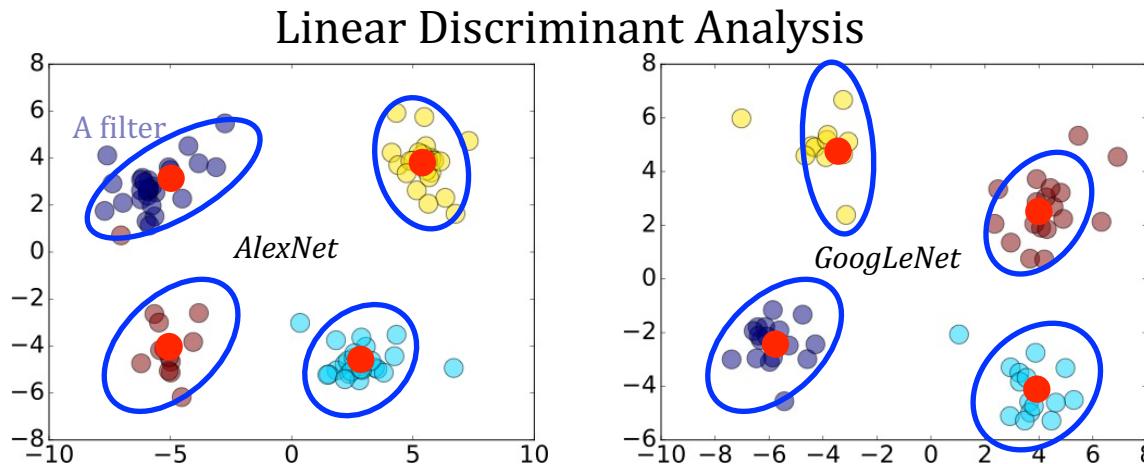
Low-rank Deep Neural Networks

- ✓ Filters are redundant and highly correlated with each other
- ✓ Decompose filters (weight matrices) to low-rank space



Lower-rank Deep Neural Networks

Motivation: Coordinate filters to lowerer-rank space -> more compact DNNs



Coordinate a cluster of filters closer to each other -> more accurate approximation
merge multiple clusters -> lower rank

Lower-rank Deep Neural Networks

- ✓ Introduce additional gradients in Stochastic Gradient Descent (SGD):

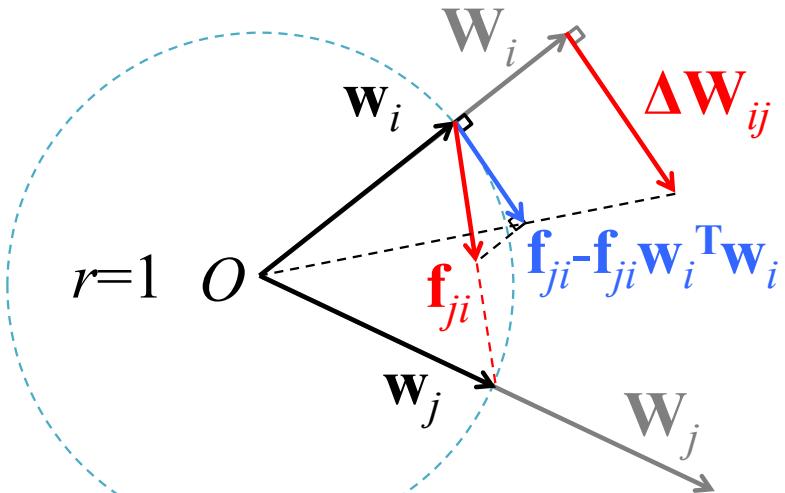
Force regularization:

$$\Delta \mathbf{W}_i = \boxed{\sum_{j=1}^N \Delta \mathbf{W}_{ij}} = \|\mathbf{W}_i\| \sum_{j=1}^N (\mathbf{f}_{ji} - \mathbf{f}_{ji} \mathbf{w}_i^T \mathbf{w}_i)$$

Forces from all other filters

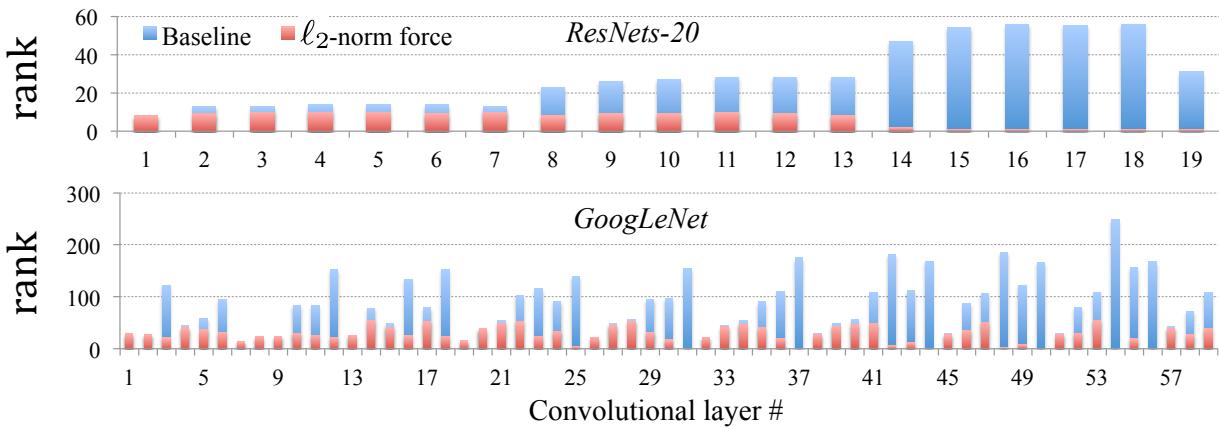
$$\mathbf{f}_{ji} = f(\mathbf{w}_j - \mathbf{w}_i) = \mathbf{w}_j - \mathbf{w}_i \quad L_2\text{-norm force}$$

or $= \frac{\mathbf{w}_j - \mathbf{w}_i}{\|\mathbf{w}_j - \mathbf{w}_i\|}$ $L_1\text{-norm force}$

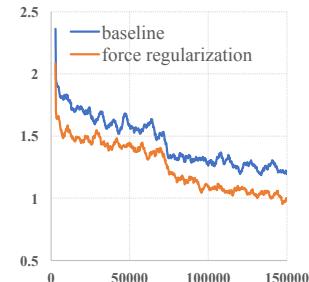


Lower-rank Deep Neural Networks

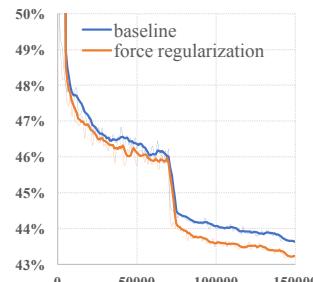
$\leq 5\%$ PCA reconstruction error



(a) Training data loss



(b) Top-1 validation error



Lower-rank Deep Neural Networks

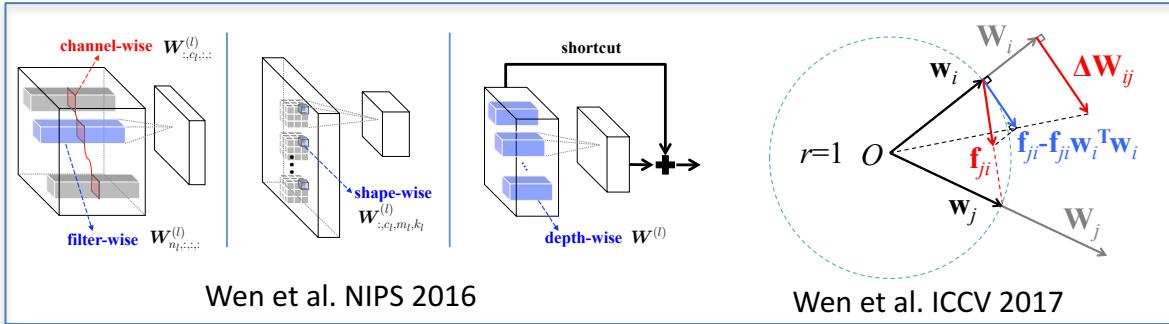
Table 6. Comparison of speedup factor on *AlexNet* by state-of-the-art DNN acceleration methods.

Method	Top-5 err.	conv1	conv2	conv3	conv4	conv5	total
<i>AlexNet in Caffe</i>	19.97%	1.00×	1.00×	1.00×	1.00×	1.00×	1.00 ×
<i>cp-decomposition</i> [1]	20.97% (+1.00%)	–	4.00×	–	–	–	1.27 ×
<i>one-shot</i> [2]	21.67% (+1.70%)	1.48×	2.30×	3.84×	3.53×	3.13×	2.52 ×
<i>SSL</i> [3]	19.58% (-0.39%)	1.00×	1.27×	1.64×	1.68×	1.32×	1.35 ×
	21.63% (+1.66%)	1.05×	3.37×	6.27×	9.73×	4.93×	3.13 ×
<i>ours</i>	20.14% (+0.17%)	2.61×	6.06×	2.48×	2.20×	1.58×	2.69 ×
	21.68% (+1.71%)	2.65×	6.22×	4.81×	4.00×	2.92×	4.05 ×

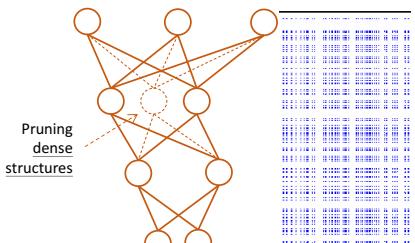
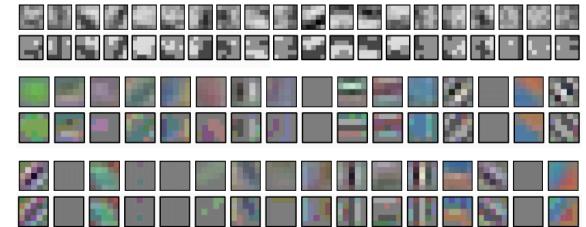
[1] V. Lebedev, et al. ICLR 2015; [2] Y.-D. Kim, et al. ICLR 2016; [3] W. Wen, et al. NIPS 2016

Research Summary on Efficient Deep Learning

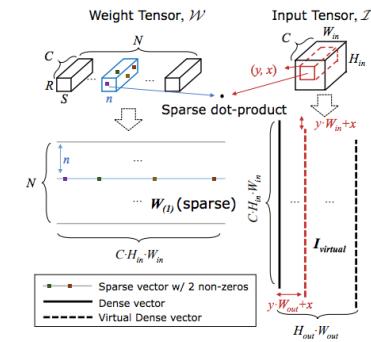
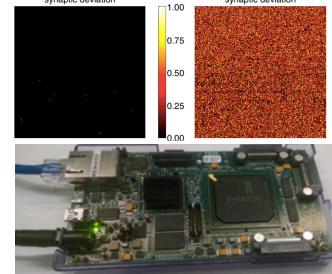
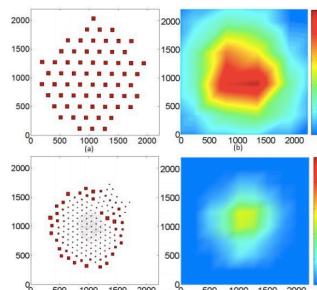
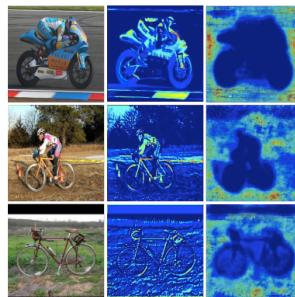
This presentation



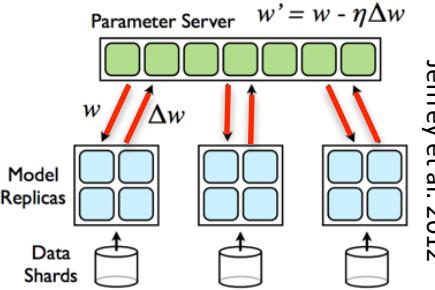
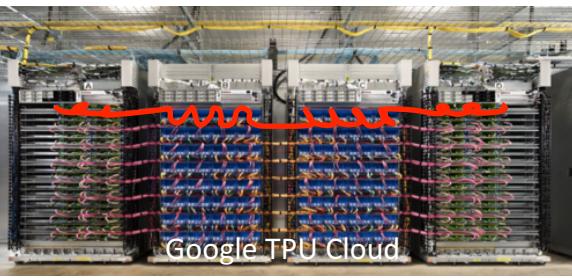
Wang et al. ASP-DAC 2017
(Best Paper Award)



Wen et al. ICLR 2018



Park et al. ICLR 2017



Jeffrey et al. 2012

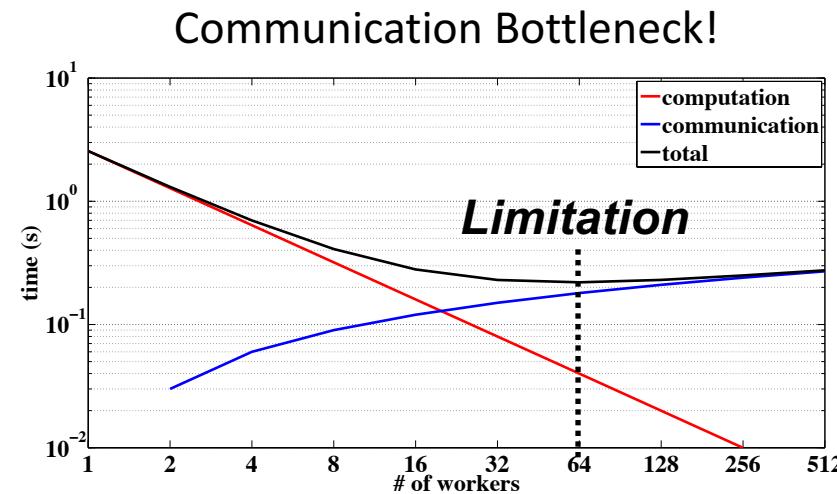
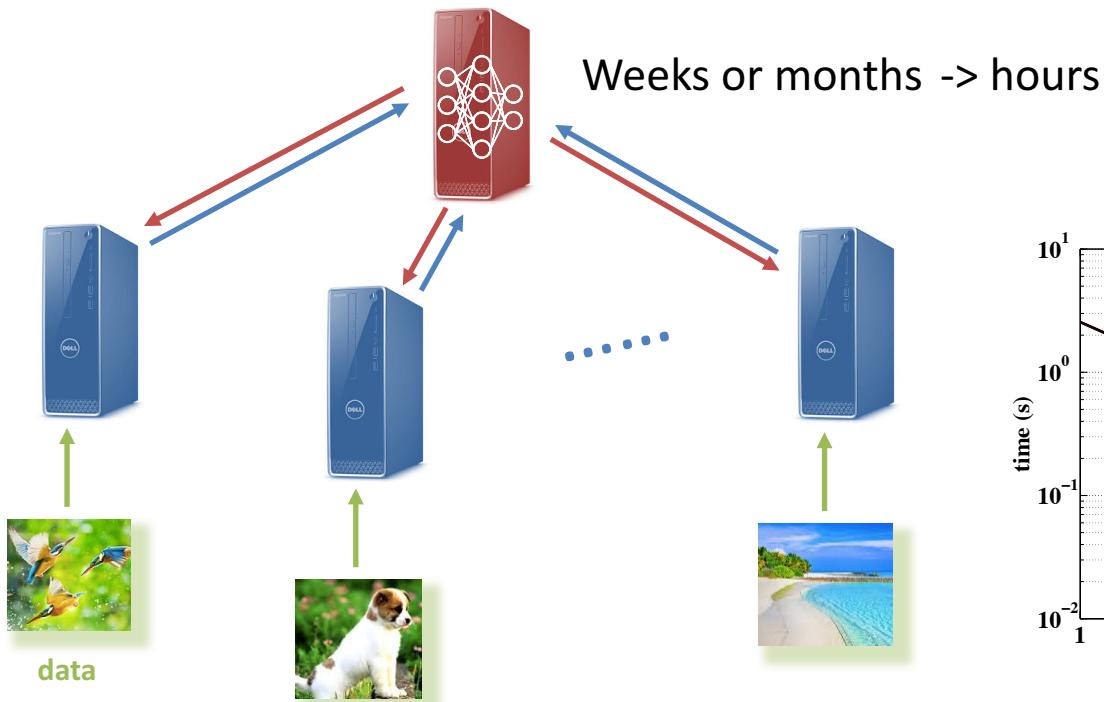
TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning

NIPS 2017 (Oral)

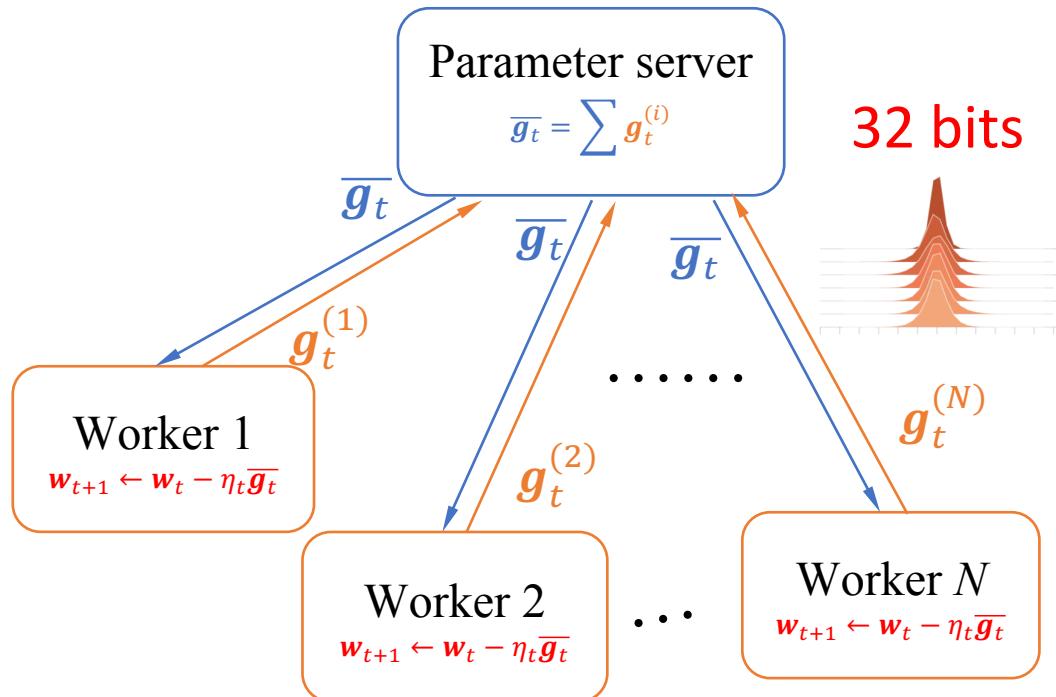
Wei Wen¹, Cong Xu², Feng Yan³, Chunpeng Wu¹,
Yandan Wang⁴, Yiran Chen¹, Hai (Helen) Li¹

Duke University¹, Hewlett Packard Labs²,
University of Nevada - Reno³, University of Pittsburgh⁴
<https://github.com/wenwei202/terngrad>

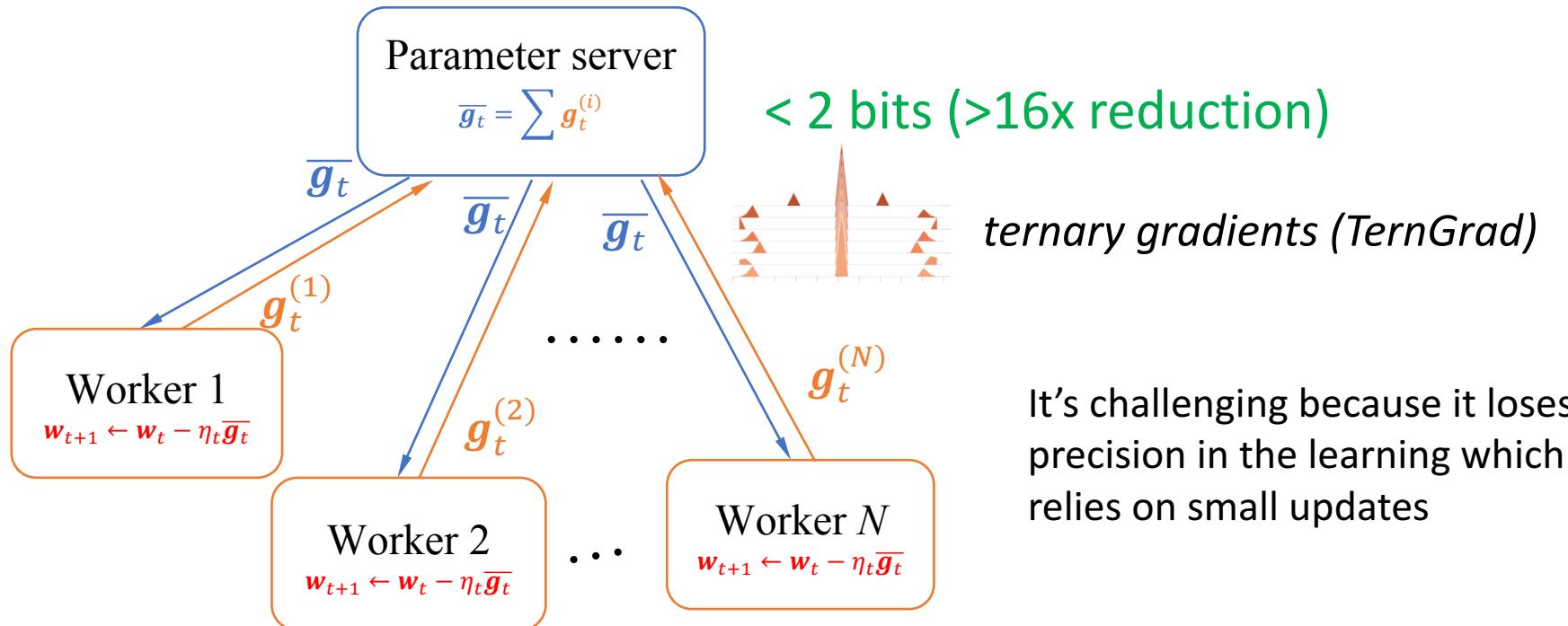
Background - Distributed Deep Learning



Gradient Quantization for Communication Reduction



Gradient Quantization for Communication Reduction



Stochastic Gradients without Bias

Batch Gradient Descent

$$C(\mathbf{w}) \triangleq \frac{1}{n} \sum_{i=1}^n Q(\mathbf{z}_i, \mathbf{w})$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta_t}{n} \sum_{i=1}^n g_t^{(i)}$$

SGD

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \cdot g_t^{(I)}$$

I is randomly drawn from $[1, n]$

$$\mathbb{E}\{g_t^{(I)}\} = \nabla C(\mathbf{w})$$

No bias

TernGrad

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \cdot \text{ternarize}(g_t^{(I)})$$

$$\mathbb{E}\{\text{ternarize}(g_t^{(I)})\} = \nabla C(\mathbf{w})$$

No bias

TernGrad is Simple

$$\tilde{\mathbf{g}}_t = \text{ternarize}(\mathbf{g}_t) = s_t \cdot \text{sign}(\mathbf{g}_t) \circ \mathbf{b}_t$$

$$s_t \triangleq \|\mathbf{g}_t\|_\infty \triangleq \max(\text{abs}(\mathbf{g}_t))$$

$$\begin{cases} P(b_{tk} = 1 \mid \mathbf{g}_t) = |g_{tk}| / s_t \\ P(b_{tk} = 0 \mid \mathbf{g}_t) = 1 - |g_{tk}| / s_t \end{cases}$$

Example:

$$\mathbf{g}_t^{(i)}: [0.30, -1.20, \dots, 0.9]$$

$$s_t: 1.20$$

$$\text{Signs: } [1, -1, \dots, 1]$$

$$P(b_{tk} = 1 | g_t): [\frac{0.3}{1.2}, \frac{1.2}{1.2}, \dots, \frac{0.9}{1.2}]$$

$$\mathbf{b}_t: [0, 1, \dots, 1]$$

$$\widetilde{\mathbf{g}_t^{(i)}}: [0, -1, \dots, 1] * 1.20$$

$$\mathbf{E}_{\mathbf{z}, \mathbf{b}} \{\tilde{\mathbf{g}}_t\} = \mathbf{E}_{\mathbf{z}, \mathbf{b}} \{s_t \cdot \text{sign}(\mathbf{g}_t) \circ \mathbf{b}_t\}$$

$$= \mathbf{E}_{\mathbf{z}} \{s_t \cdot \text{sign}(\mathbf{g}_t) \circ \mathbf{E}_{\mathbf{b}} \{\mathbf{b}_t | \mathbf{z}_t\}\} = \mathbf{E}_{\mathbf{z}} \{\mathbf{g}_t\} = \nabla_{\mathbf{w}} C(\mathbf{w}_t)$$

No bias

Convergence

Standard SGD almost truly converges under assumptions (Fisk 1965, Metivier 1981&1983, Bottou 1998)

Assumption 1:

$C(\mathbf{w})$ has a single minimum \mathbf{w}^* and $\forall \epsilon > 0, \inf_{\|\mathbf{w} - \mathbf{w}^*\|^2 > \epsilon} (\mathbf{w} - \mathbf{w}^*)^T \nabla_{\mathbf{w}} C(\mathbf{w}) > 0$

Assumption 2:

Learning rate γ_t decreases neither very fast nor very slow

$$\begin{cases} \sum_{t=0}^{+\infty} \gamma_t^2 < +\infty \\ \sum_{t=0}^{+\infty} \gamma_t = +\infty \end{cases}$$

Assumption 3 (gradient bound):

$$\mathbf{E} \{ \| \mathbf{g} \|^2 \} \leq A + B \| \mathbf{w} - \mathbf{w}^* \|^2$$

Standard SGD *almost-truly* converges

Assumption 3 (gradient bound):

$$\mathbf{E} \{ \| \mathbf{g} \|_\infty \cdot \| \mathbf{g} \|_1 \} \leq A + B \| \mathbf{w} - \mathbf{w}^* \|^2$$

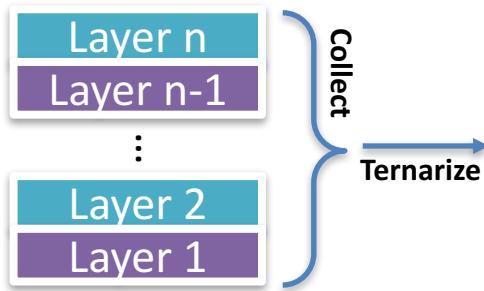
TernGrad *almost-truly* converges

$$\mathbf{E} \{ \| \mathbf{g} \|^2 \} \leq \mathbf{E} \{ \| \mathbf{g} \|_\infty \cdot \| \mathbf{g} \|_1 \} \leq A + B \| \mathbf{w} - \mathbf{w}^* \|^2$$

Stronger gradient bound in *TernGrad*

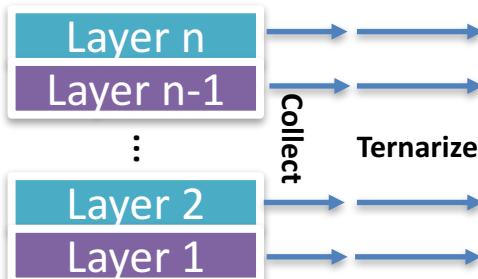
Closing Bound Gap

Two methods to push the gradient bound of *TernGrad* closer to the bound of standard SGD



Closing Bound Gap

Two methods to push the gradient bound of *TernGrad* closer to the bound of standard SGD



Layer-wise ternarizing

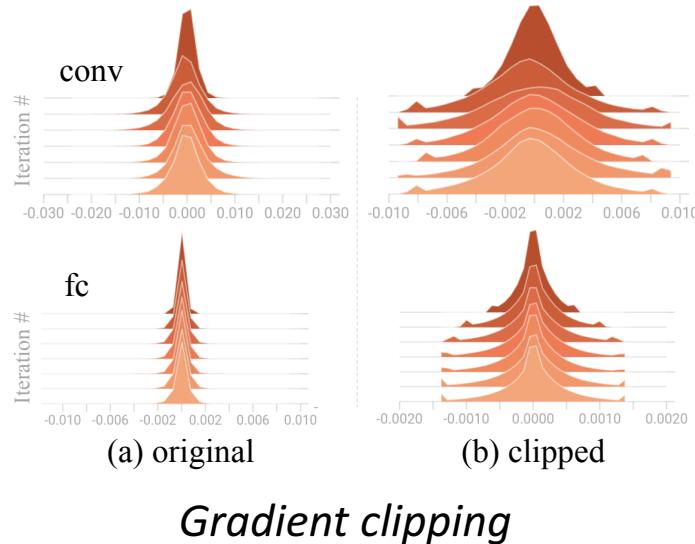
A general approach:

1. Split gradients to buckets
2. Do TernGrad bucket by bucket

When bucket size == 1, TernGrad is floating SGD

Closing Bound Gap

Two methods to push the gradient bound of *TernGrad* closer to the bound of standard SGD



$$f(g_i) = \begin{cases} g_i & |g_i| \leq c\sigma \\ sign(g_i) \cdot c\sigma & |g_i| > c\sigma \end{cases}$$

$c=2.5$ works well for all tested datasets, DNNs and optimizers.

Suppose Gaussian distribution:

1. Change length 1.0%-1.5%
2. Change direction 2° -3°
3. Small bias with variance reduced

Scaling to Large-scale Deep Learning

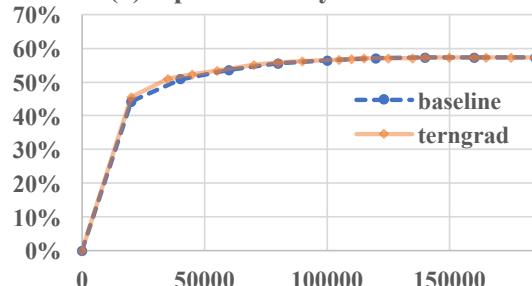
AlexNet

base LR	mini-batch size	workers	gradients	top-1
0.01	256	2	floating	57.33%
			<i>TernGrad</i>	57.61%
			<i>TernGrad-noclip</i>	54.63%
0.02	512	4	floating	57.32%
			<i>TernGrad</i>	57.28%
0.04	1024	8	floating	56.62%
			<i>TernGrad</i>	57.54%

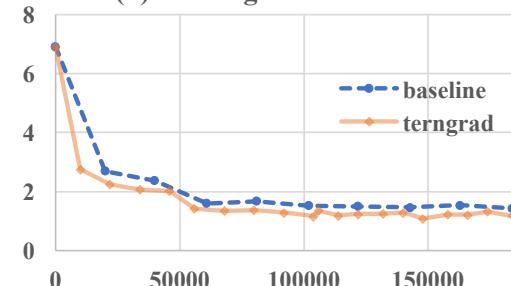
Keskar, et al., 2017

4 workers

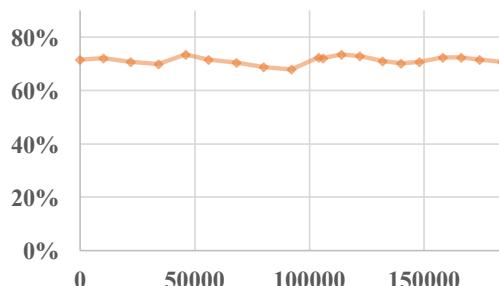
(a) top-1 accuracy vs iteration



(b) training loss vs iteration



(c) gradient sparsity of terngrad in fc6



Scaling to Large-scale Deep Learning

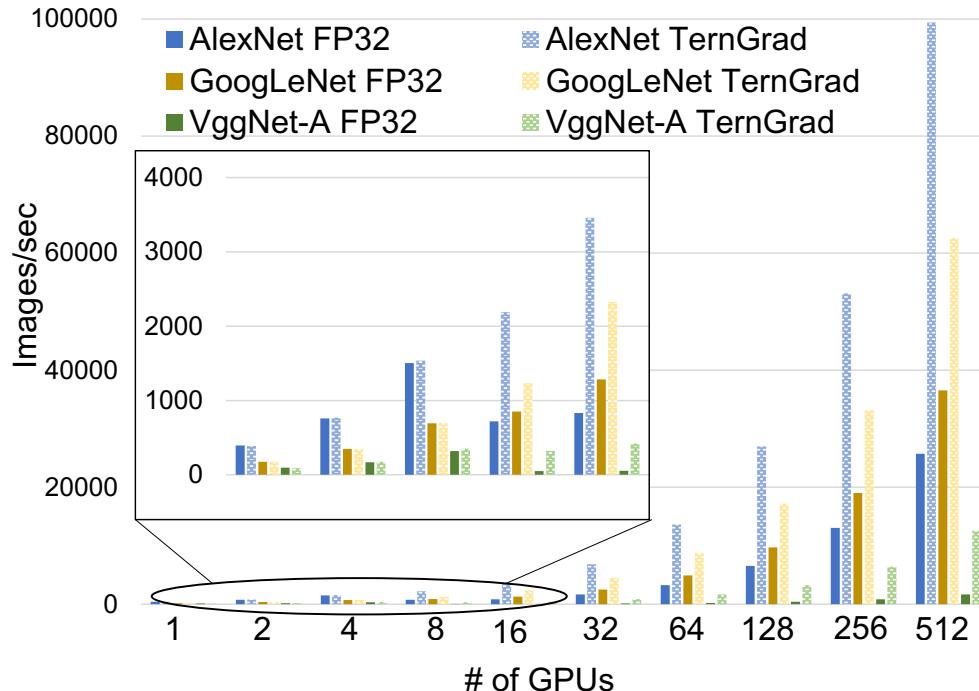
GoogLeNet accuracy loss is <2% on average.

base LR	mini-batch size	workers	gradients	top-5
0.04	128	2	floating	88.30%
			<i>TernGrad</i>	86.77%
0.08	256	4	floating	87.82%
			<i>TernGrad</i>	85.96%
0.10	512	8	floating	89.00%
			<i>TernGrad</i>	86.47%

(Hyper-parameters are tuned for standard SGD and fixed in *TernGrad*)
(Tune hyper-parameters specifically for *TernGrad* may reduce accuracy gap)

Performance Model

Training throughput on GPU cluster with Ethernet and PCI switch

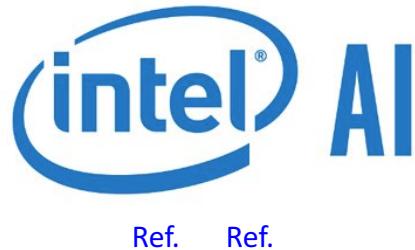


TernGrad gives higher speedup when

1. using more workers
2. having smaller communication bandwidth
 1. Ethernet > InfiniBand
 2. when more network traffic
3. training DNNs with more fully-connected layers
 1. VggNet > GoogLeNet
 2. LSTMs > CNNs
4. using GPU based distributed systems vs. CPU based ones

Research Adopted in Industry

Structured Sparsity Learning



TernGrad



All are available in <https://github.com/wenwei202/>

Conclusion

- AI systems are landing into the cloud and upon the edge
- Efficiency challenges and solutions:
 - Lightweight computing engine restricts the deployment of cumbersome Deep Neural Networks on the edge
 - We propose Structurally Sparse Deep Neural Networks to simplify the “rocket” (model) to lighten the load on the “engine” (computing hardware)
 - Low-rank decomposition and neuromorphic computing systems are promising solutions
 - Solutions can be generalized to solve similar challenges in large-scale cloud AI services
- Scalability challenges and solutions:
 - Communication bottleneck limits the scalability of distributed deep learning
 - We propose *TernGrad* to quantize gradients to reduce communication volume
 - The existence of sharp minima limits mini-batch size and thus parallelism
 - We propose *SmoothOut* to escape sharp minima

Related Publications

- **Wei Wen**, Yandan Wang, Feng Yan, Cong Xu, Yiran Chen, Hai Li, “SmoothOut: Smoothing Out Sharp Minima for Generalization in Large-Batch Deep Learning”, preprint. [[paper](#)][[code](#)]
- **Wei Wen**, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, Hai Li, “Learning Intrinsic Sparse Structures within Long Short-Term Memory”, *the 6th International Conference on Learning Representations (ICLR)*, 2018. [[poster](#)][[paper](#)][[code](#)]
- **Wei Wen**, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li, “TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning”, *the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. (*Oral*, $40/3240=1.2\%$. Available in [PyTorch](#)/[Caffe2](#)). [[paper](#)][[video](#)][[slides](#)][[code](#)][[poster](#)]
- **Wei Wen**, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li, “Coordinating Filters for Faster Deep Neural Networks”, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. [[paper](#)][[code](#)][[poster](#)]
- **Wei Wen**, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li, “Learning Structured Sparsity in Deep Neural Networks”, *the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016. Acceptance Rate: $568/2500=22.7\%$. ([Integrated into Intel Nervada](#)) [[paper](#)][[code](#)][[poster](#)]
- Jongsoo Park, Sheng Li, **Wei Wen**, Ping Tak Peter Tang, Hai Li, Yiran Chen, Pradeep Dubey, “Faster CNNs with Direct Sparse Convolutions and Guided Pruning”, *the 5th International Conference on Learning Representations (ICLR)*, 2017. [[paper](#)][[code](#)][[media](#)]
- Chunpeng Wu, **Wei Wen**, Tariq Afzal, Yongmei Zhang, Yiran Chen, Hai Li, “A Compact DNN: Approaching GoogLeNet-Level Accuracy of Classification and Domain Adaptation”, *CVPR*, 2017. [[paper](#)]
- Yandan Wang, **Wei Wen**, Linghao Song, Hai Li, “Classification Accuracy Improvement for Neuromorphic Computing Systems with One-level Precision Synapses”, *ASP-DAC*, 2017. (**Best Paper Award**). [[paper](#)]
- **Wei Wen**, Chunpeng Wu, Yandan Wang, Kent Nixon, Qing Wu, Mark Barnell, Hai Li, Yiran Chen, “A New Learning Method for Inference Accuracy, Core Occupation, and Performance Co-optimization on TrueNorth Chip”, *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016. Acceptance Rate: $152/876=17.4\%$. (**Best Paper Nomination**, $16/876=1.83\%$). [[paper](#)]
- **Wei Wen**, Chi-Ruo Wu, Xiaofang Hu, Beiyi Liu, Tsung-Yi Ho, Xin Li, Yiran Chen, “An EDA Framework for Large Scale Hybrid Neuromorphic Computing Systems”, *52nd ACM/EDAC/IEEE Design Automation Conference(DAC)*, 2015. Acceptance Rate: $162/789=20.5\%$. (**Best Paper Nomination**, $7/789=0.89\%$). [[paper](#)]
- Yandan Wang, **Wei Wen**, Beiyi Liu, Donald Chiarulli, Hai Li, “Group Scissor: Scaling Neuromorphic Computing Design to Big Neural Networks”, *54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017. Acceptance Rate: 24%. [[paper](#)]
- Jongsoo Park, Sheng R. Li, **Wei Wen**, Hai Li, Yiran Chen, Pradeep Dubey, “Holistic SparseCNN: Forging the Trident of Accuracy, Speed, and Size”, arXiv 1608.01409, 2016. (in [Intel Developer Forum 2016](#), pages 41-43). [[paper](#)][[code](#)]

References

- Pranav Dar, 25 Open Datasets for Deep Learning Every Data Scientist Must Work With, <https://www.analyticsvidhya.com/blog/2018/03/comprehensive-collection-deep-learning-datasets/>. 2018
- Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).
- Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser et al. "Mastering the game of Go with deep neural networks and tree search." *nature*529, no. 7587 (2016): 484.
- Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In *CVPR*, vol. 1, no. 2, p. 3. 2017.
- Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior et al. "Large scale distributed deep networks." In *Advances in neural information processing systems*, pp. 1223-1231. 2012.
- Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, large minibatch SGD: training imagenet in 1 hour." *arXiv preprint arXiv:1706.02677* (2017).



THANKS! Q&A?

Wei Wen, Duke University
@wei_wen_
www.pittnuts.com