Efficient and Scalable Deep Learning

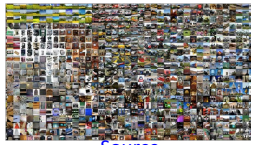
Wei Wen

Computational Evolutionary Intelligence Lab

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

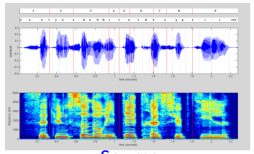
Source

[The Wikipedia Corpus](#) 4.4M Articles & 1.9B Words

[Yelp Reviews](#) 6M Reviews

[WMT 18](#) 30M Sentences

Speech



Source

[Deep Speech 2](#)

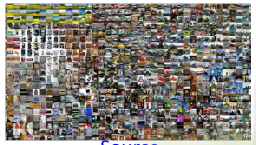
11,940 Hours of English and 9,400 Hours of Mandarin

More in (Dar 2018)

Big data made deep learning successful.

The Scale of Training Data

Image



Source

ImageNet — Open Images Dataset — MS COCO

9 M — Objects

Video

Source

1.9 Billion Frame Features — 1.1 Million Videos

Feature Engineering → A machine learning model

Text

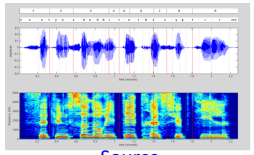
Source

[The Wikipedia Corpus](): 4.4M Articles & 1.9B Words

[Yelp Reviews](): 6M Reviews

[WMT 18](): 30M Sentences

Speech



Source

[Deep Speech 2]()

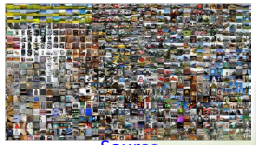
11,940 Hours of English and 9,400 Hours of Mandarin

More in (Dar 2018)

Big data made deep learning successful.

The Scale of Training Data

Image



Source

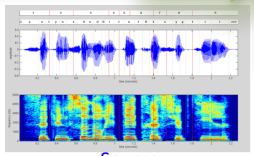
Video

Source

Text

Source

Speech



Source

More in (Dar 2018)

Feature Engineering

A machine learning model

Deep Learning

ImageNet — Open Images Dataset — MS COCO

9 M … Objects

1.9 Billion Frame Features — 1.1 Million Videos

The Wikipedia Corpus — Yelp Reviews — WMT 18

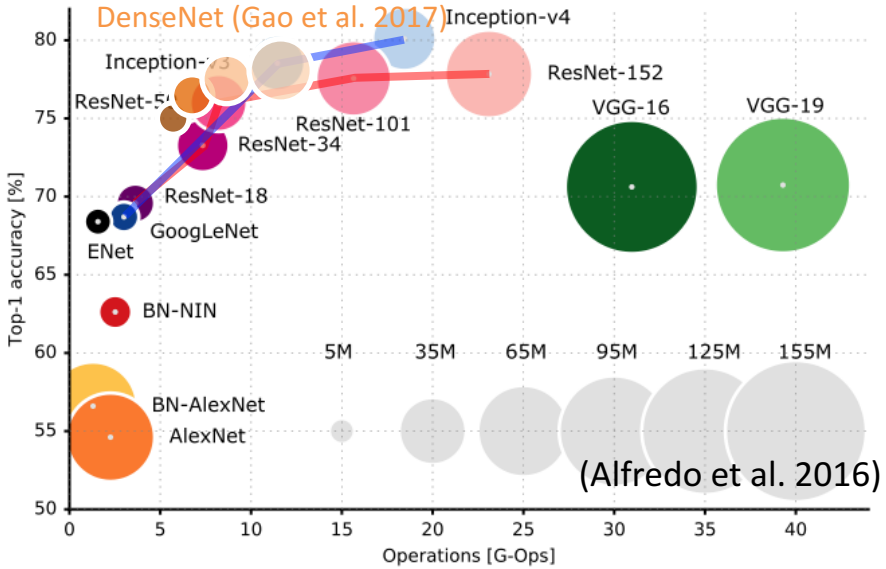
… M Sentences

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

Big data made deep learning successful.

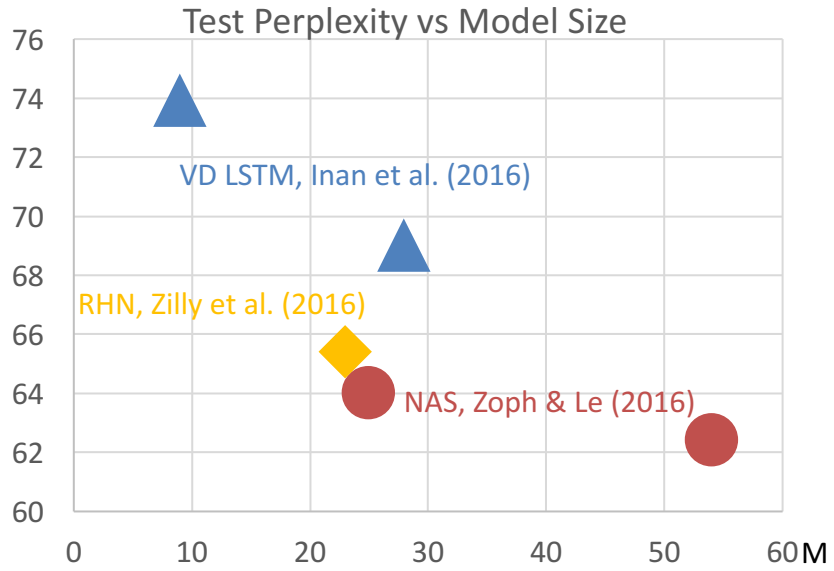
The Scale of Deep Learning Models

Image Classification on ImageNet



(Alfredo et al. 2016)

Language Modeling on Penn Treebank



More accurate prediction by paying computation cost using larger models.

Challenges of Deploying Deep Learning Models


Wired

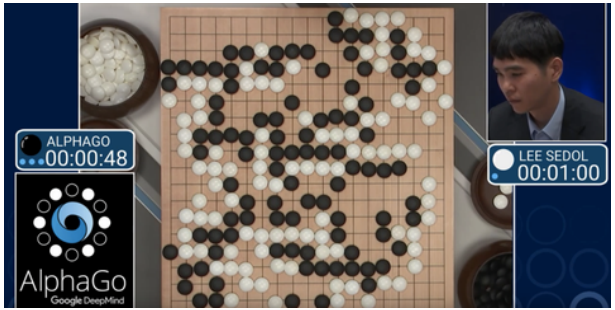
OSXDaily


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)

Inference on the edge:

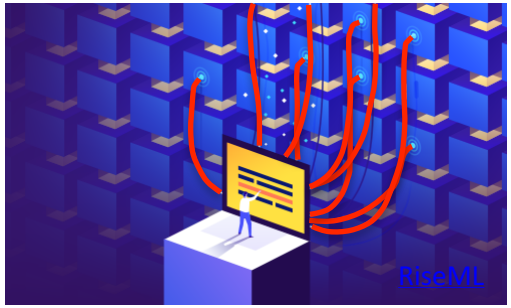
- Limited computing capability
- Limited memory
- Limited battery energy

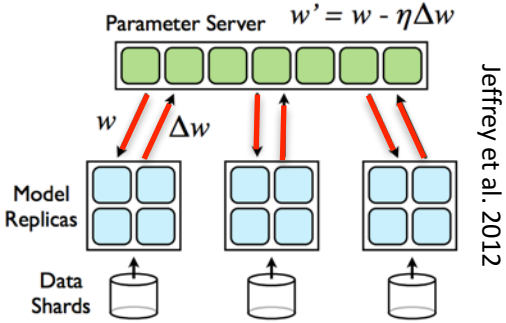
Inference in the cloud:

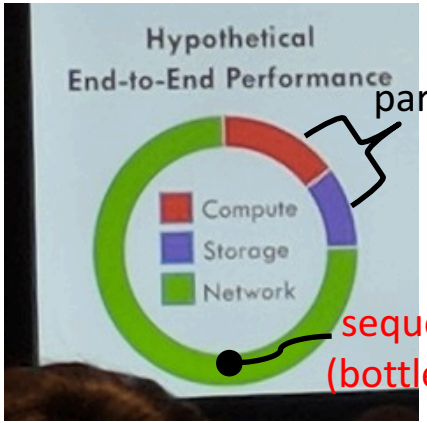
- A challenge: Real-time response to trillions of AI service requests
- Facebook (K. Hazelwood@ISCA'18)
  - 200+ Trillion predictions per day
  - 5+ Billion language translations per day

We must compress and accelerate deep learning models.

Challenges of Training Deep Learning Models

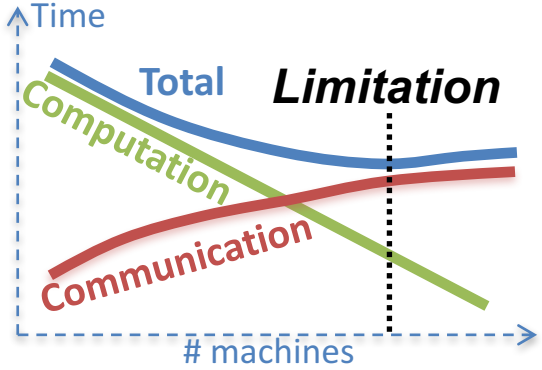




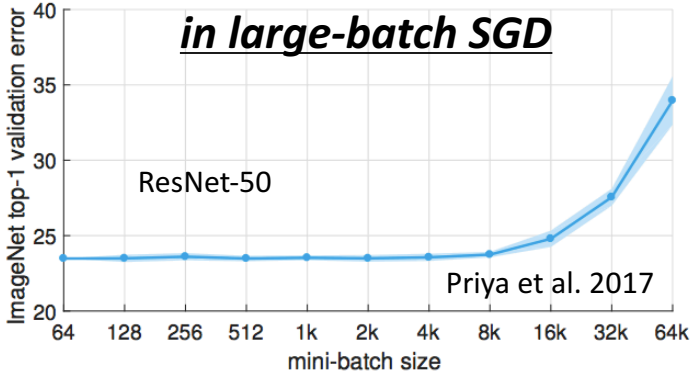


(K. Hazelwood@ISCA'18) Picture: leiphone

(1) Communication bottleneck



(2) Bad generalization 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

Research Highlights

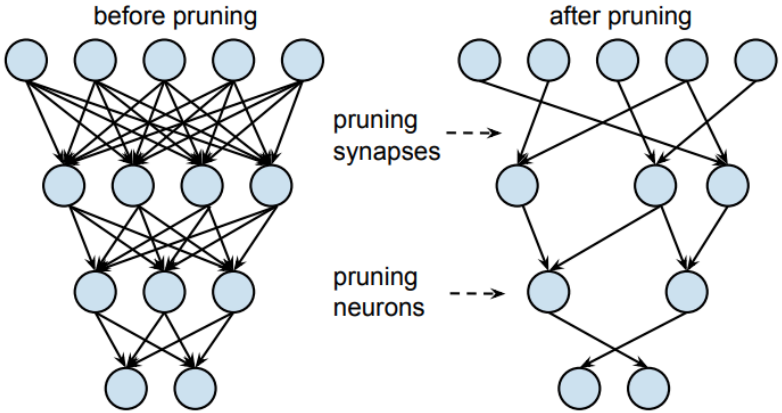
- Efficient inference of deep learning models
  - *Structurally sparse DNNs* (*NIPS 2016 & ICLR 2018*)
  - Lower-rank DNNs (ICCV 2017)
  - A compact DNN considering domain adaptation (*CVPR 2017*)
  - Direct sparse convolution (*ICLR 2017*)
  - Neuromorphic chip on the edge (DAC 2015 & DAC 2016; best 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

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]

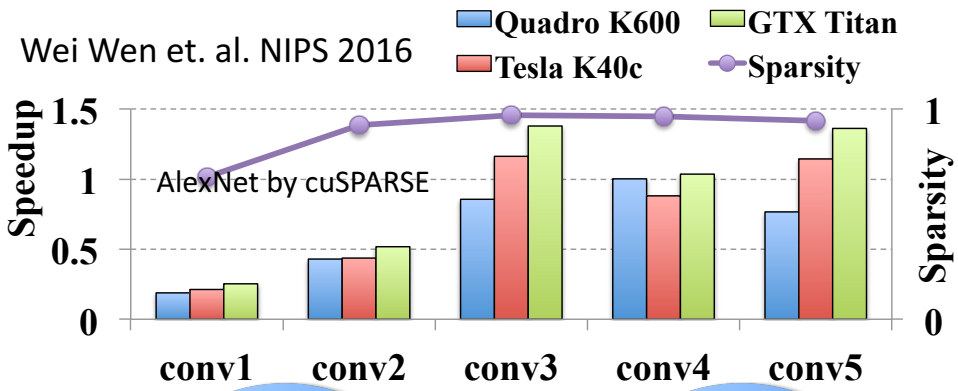
Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, Hai Li, "*Learning Intrinsic Sparse Structures within Long Short-Term Memory*", ICLR, 2018. [paper][code]

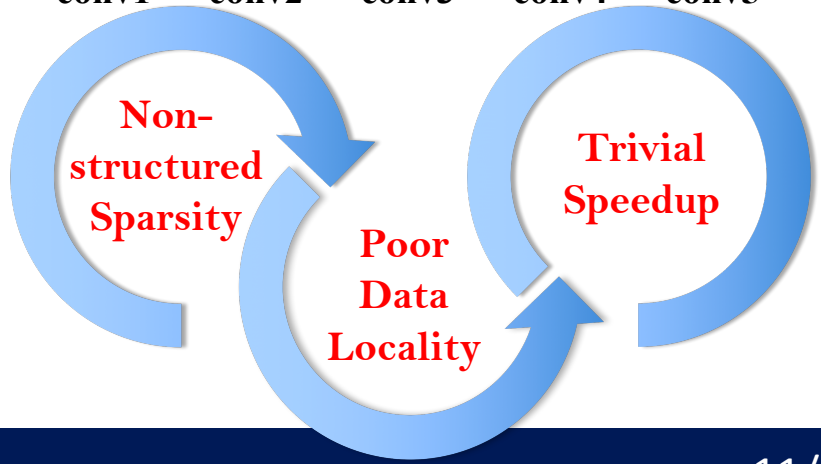
Inefficiency of Sparse Convolutional Neural Networks



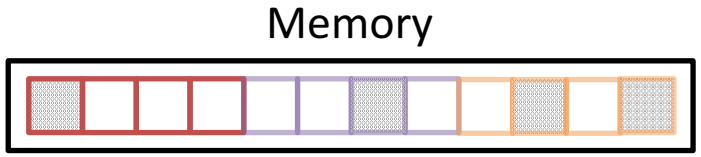
Significantly reduces the storage size of DNNs [1]
Good speedup with customized hardware [2]

[1] S. Han et. al. NIPS 2015
[2] S. Han et. al. ISCA 2016.

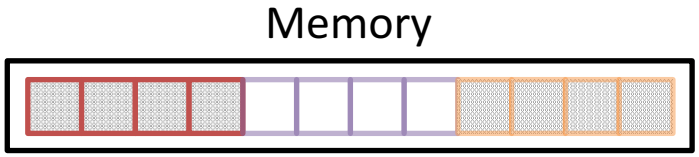




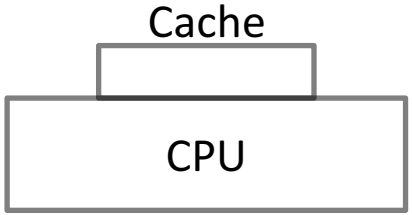
Inefficiency of Sparse Convolutional Neural Networks

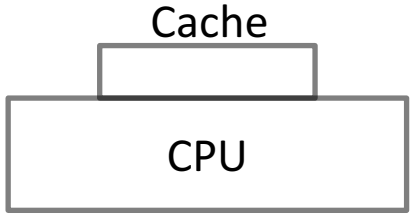


Non-structured sparse access



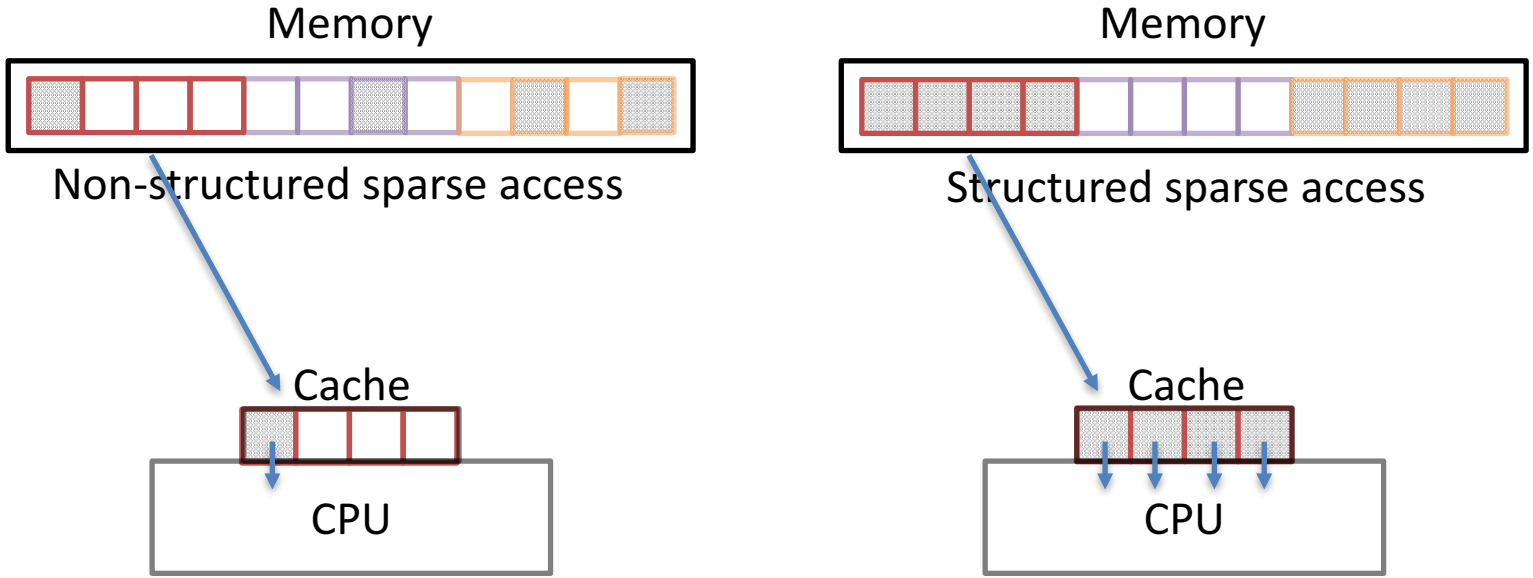
Structured sparse access





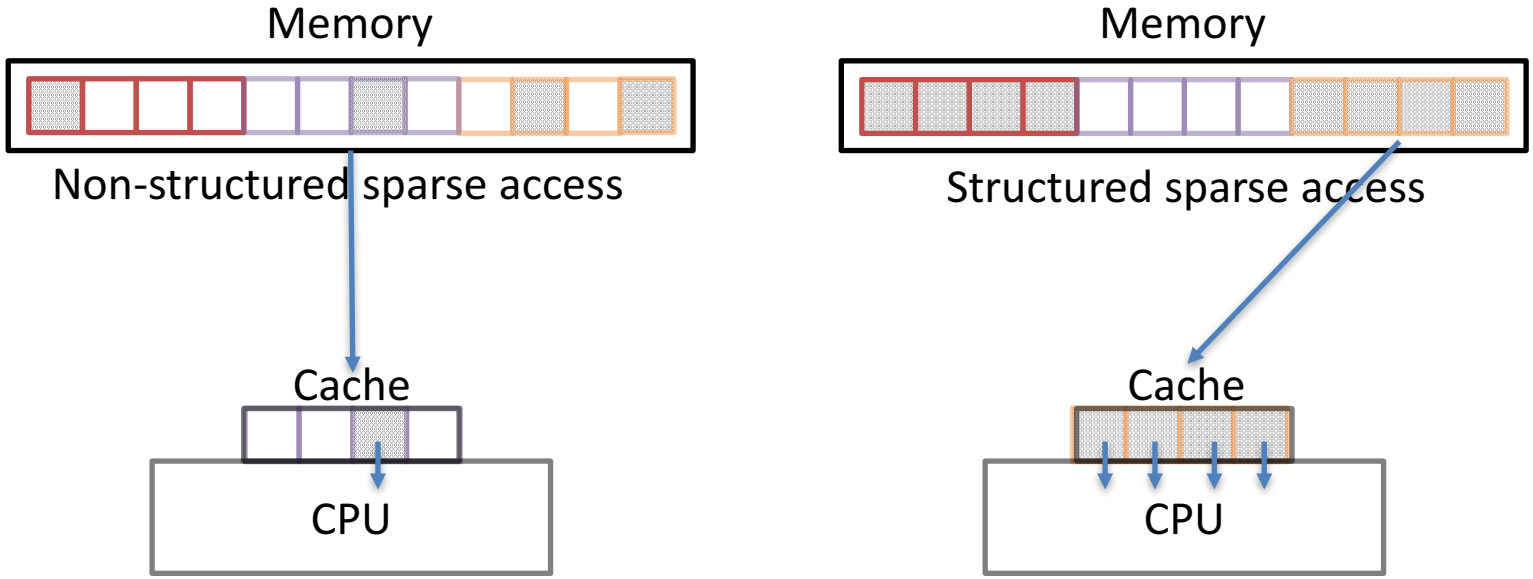
A toy example.

Inefficiency of Sparse Convolutional Neural Networks



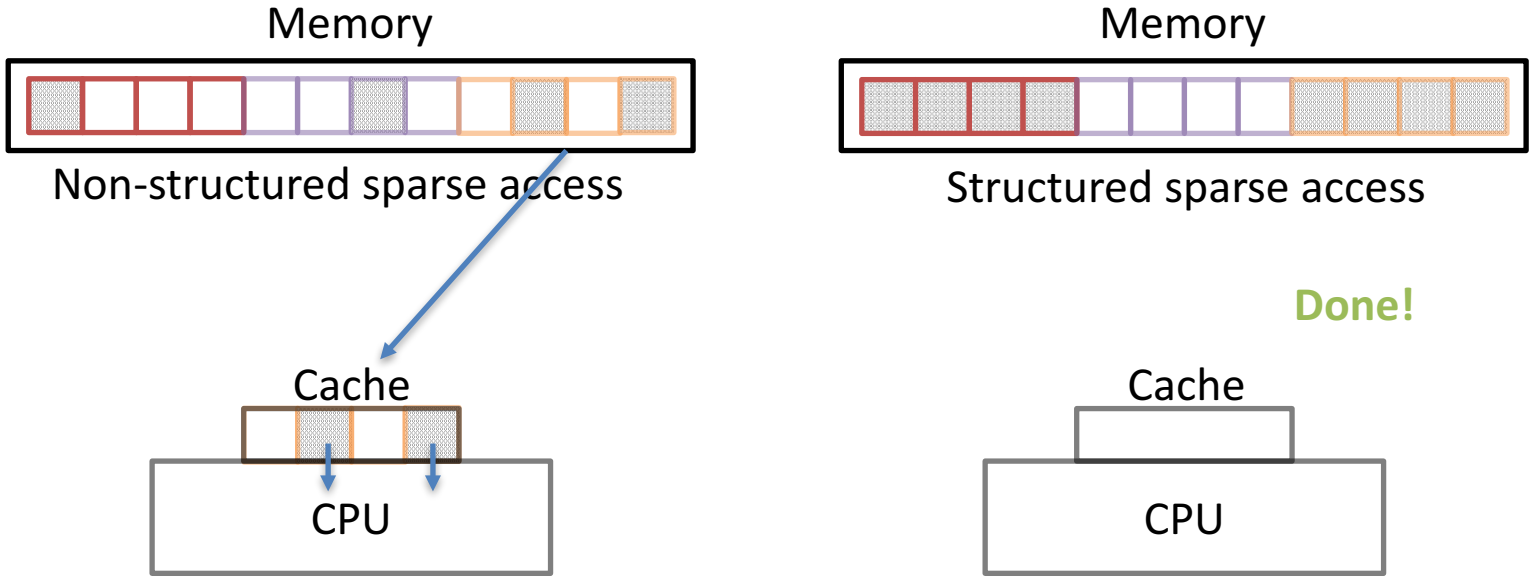
A toy example.

Inefficiency of Sparse Convolutional Neural Networks



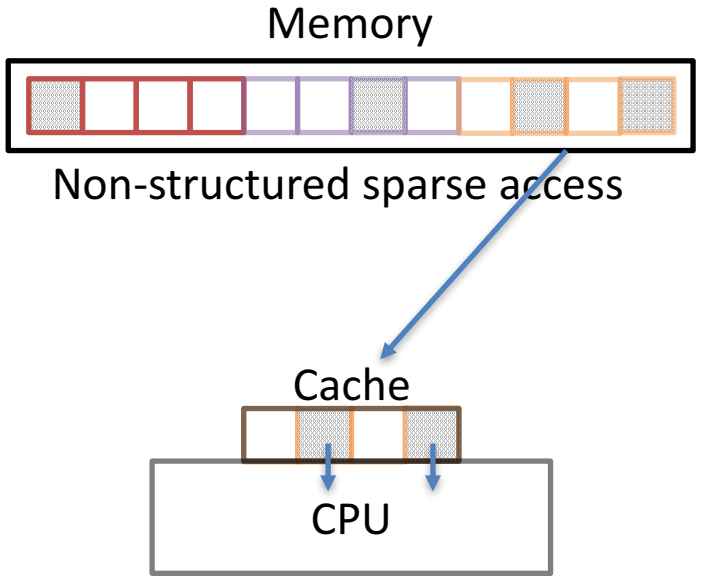
A toy example.

Inefficiency of Sparse Convolutional Neural Networks

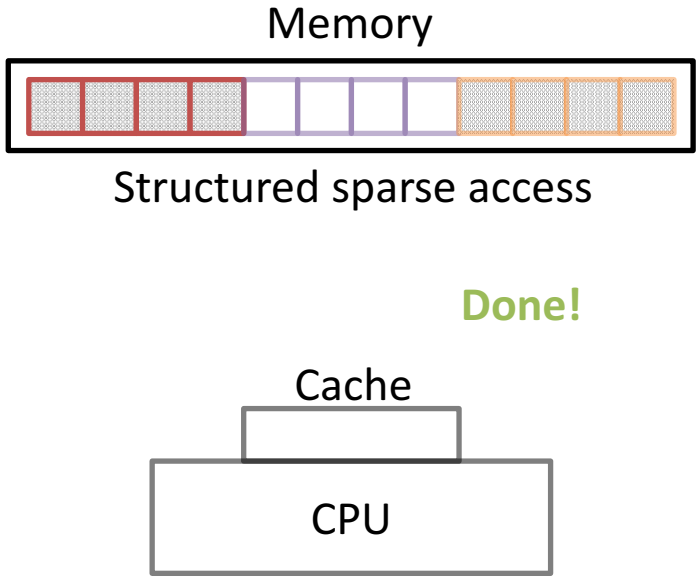


A toy example.

Inefficiency of Sparse Convolutional Neural Networks



Breaks regularity & breaks hardware parallelism



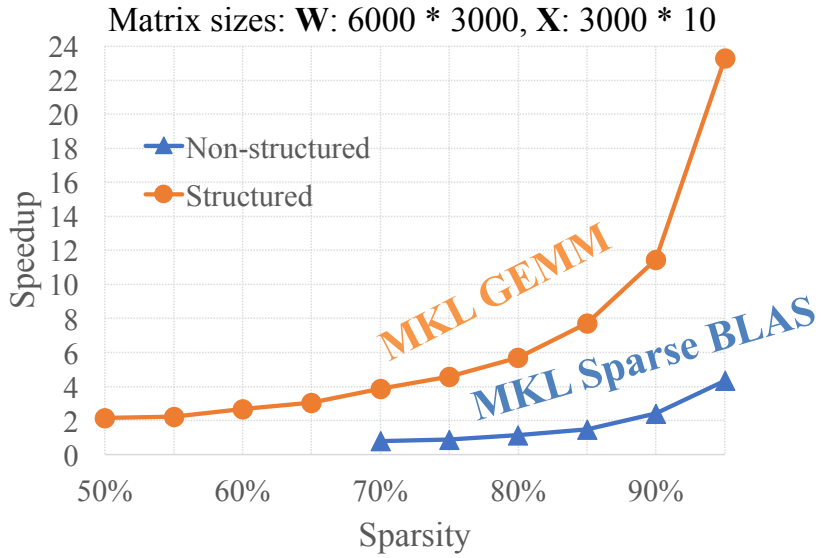
Keeps regularity & keeps hardware parallelism

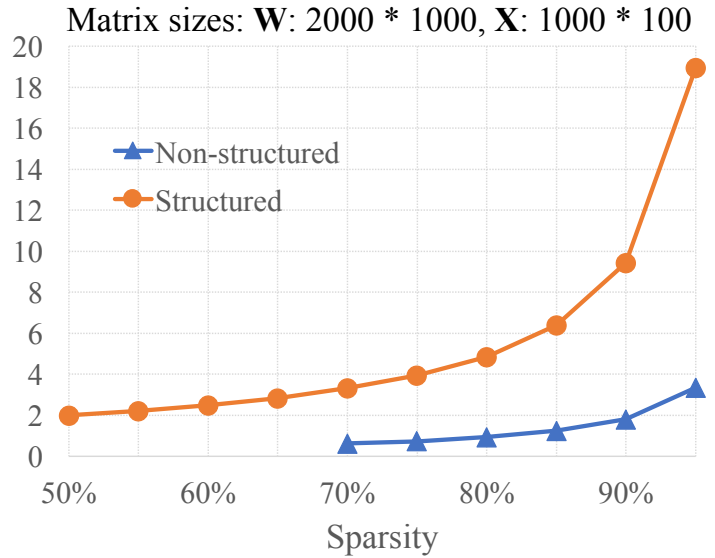
A toy example.

Efficiency of Structured Sparsity

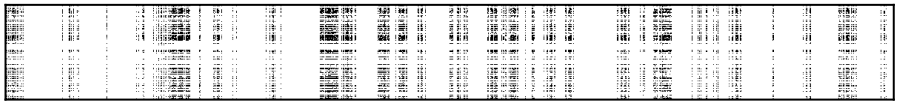
W. Wen et. al. ICLR 2018

Tested on CPUs





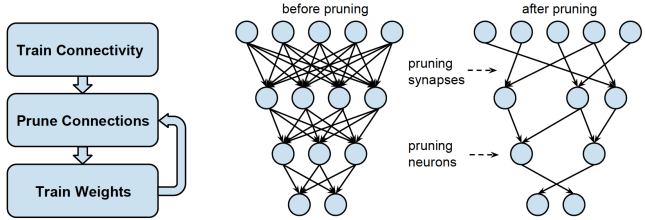
Structured sparsity



Non-structured sparsity

Structurally Sparse DNNs

S. Han et. al. NIPS 2015

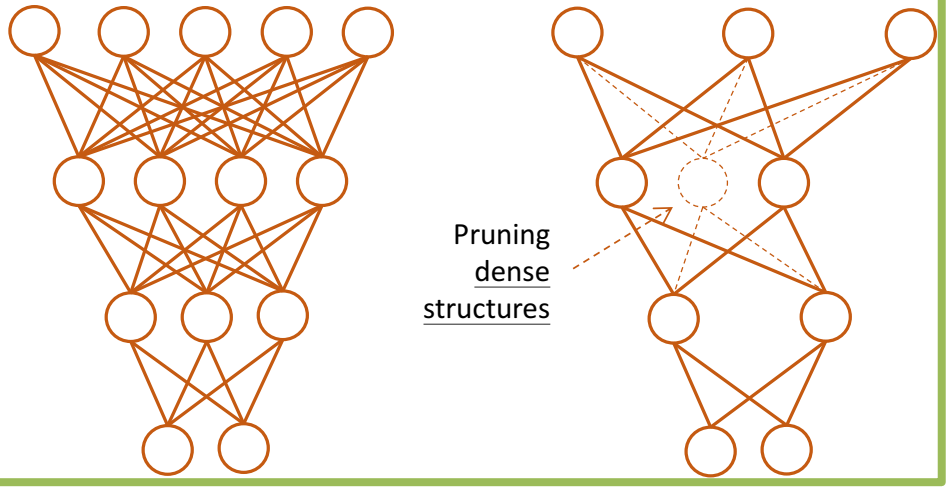


Faster training speed &
Faster inference speed

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

Train weights with Group Lasso Regularization

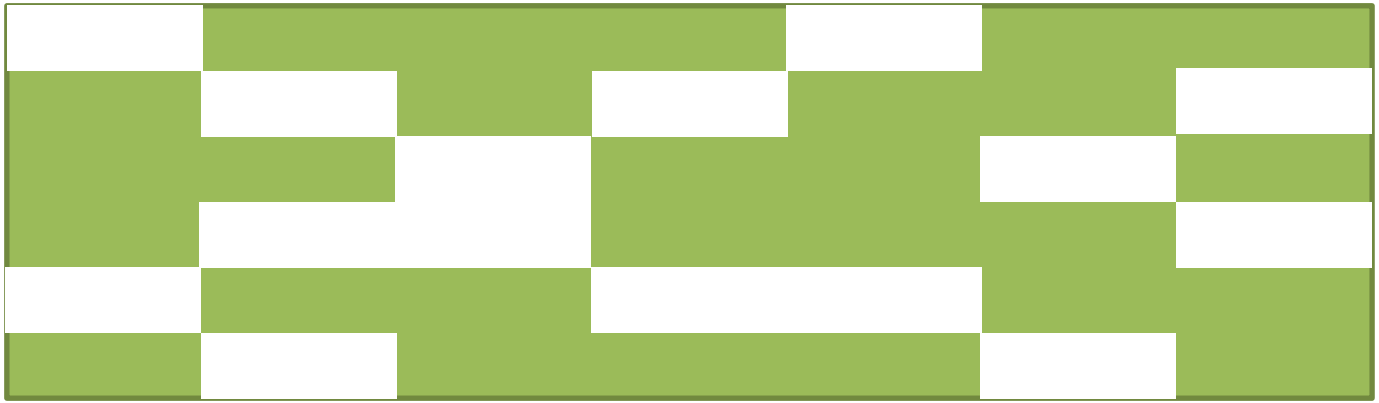
One-shot pruning from scratch
(with the same epochs)



Structurally Sparse Deep Neural Networks

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

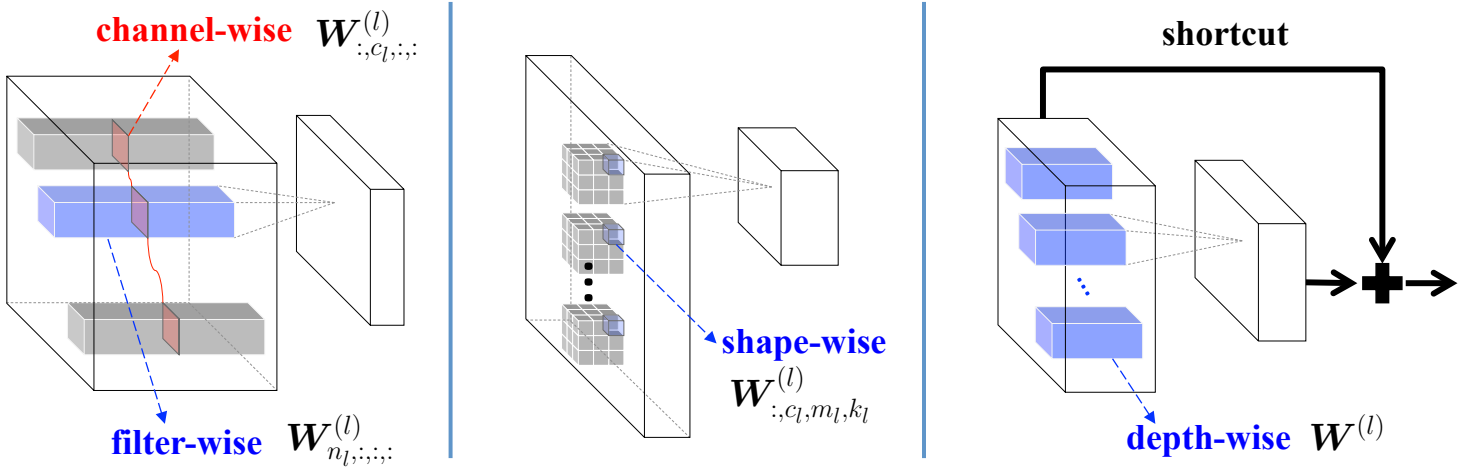
  Weights/connections are removed group by group.
  A group can be a rectangle block, a row, a column or even the whole matrix



Structurally Sparse Deep Neural Networks

In a nutshell, a group can be ***any*** form of a weight block, depending on what sparse structure you want to learn

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) -> group 1: (w_1, w_2, w_3), group 2: (w_4, w_5)

Step 2: Add group Lasso on each group $\mathbf{w}^{(g)}$ $||\boldsymbol{w}^{(g)}||_g = \sqrt{\sum_{i=1}^{|\boldsymbol{w}^{(g)}|} \left(w_i^{(g)}\right)^2}$ i.e. vector length

sqrt($w_1^2 + w_2^2 + w_3^2$), sqrt($w_4^2 + w_5^2$)

Step 3: Sum group Lasso over all groups as a regularization: $R_g(\boldsymbol{w}) = \sum_{g=1}^{G} ||\boldsymbol{w}^{(g)}||_g$,

sqrt($w_1^2 + w_2^2 + w_3^2$) + sqrt($w_4^2 + w_5^2$)

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

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:

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

A group of weights

Regular gradients to minimize error

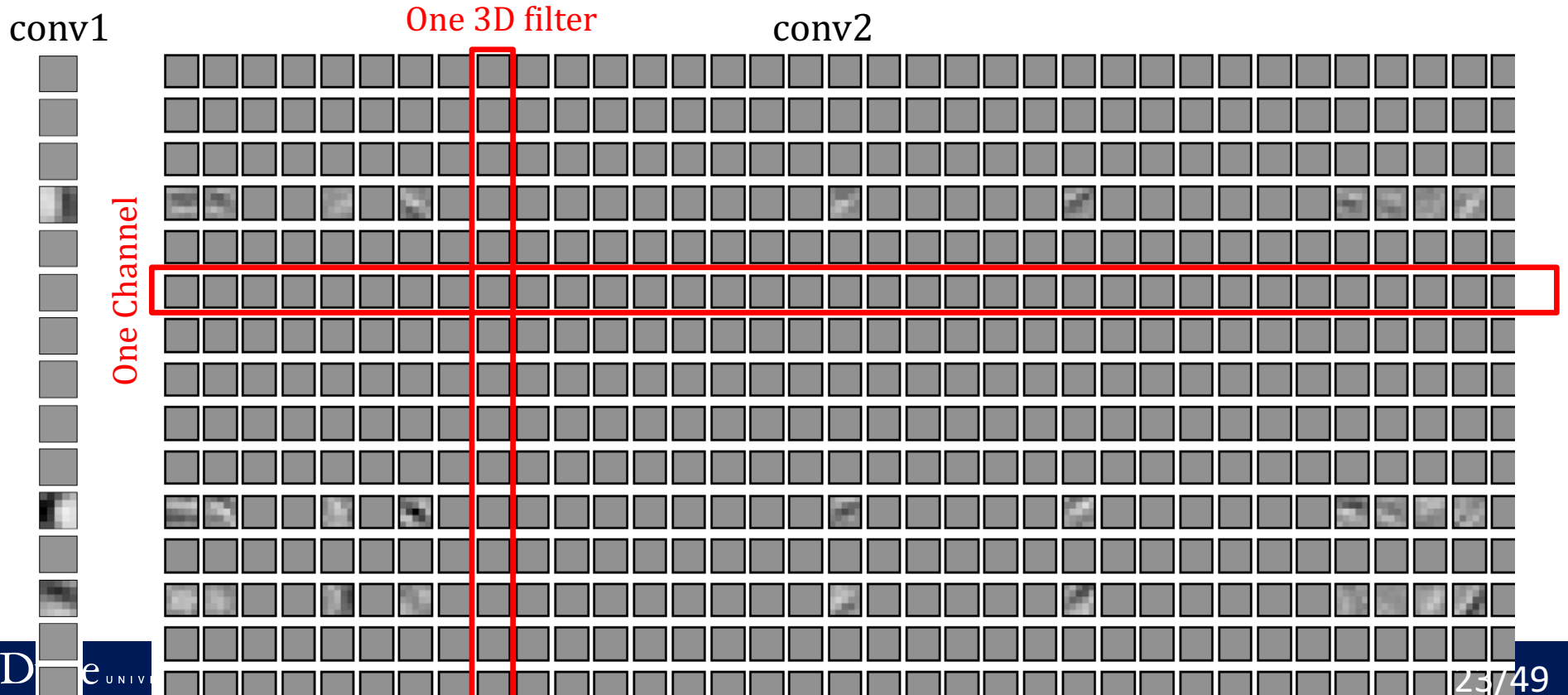
Additional gradients to learn sparsity

$\mathbf{w}_k^{(n)}$

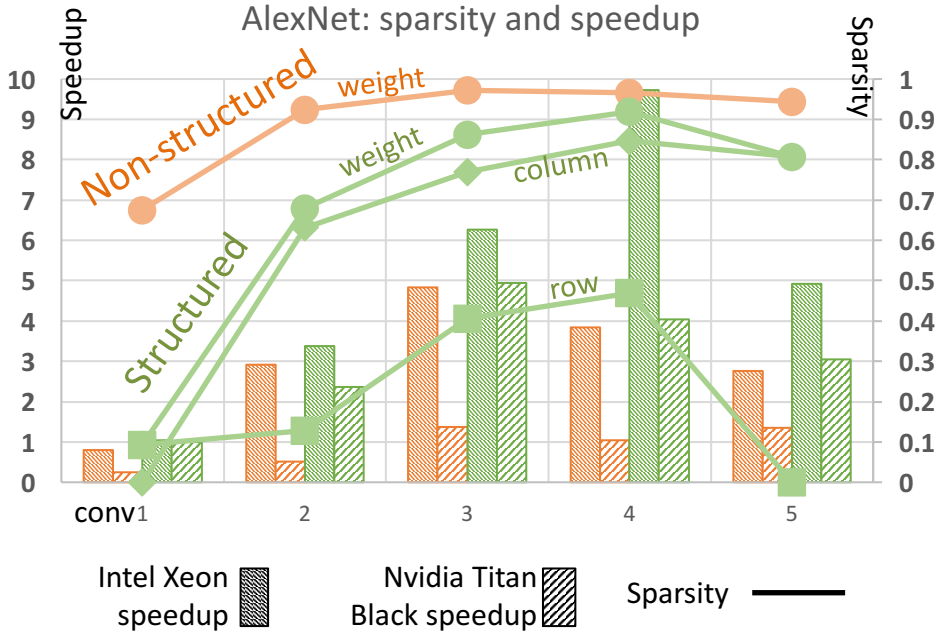
$-\mathbf{w}_k^{(n)} / \|\mathbf{w}_k^{(n)}\|_2$

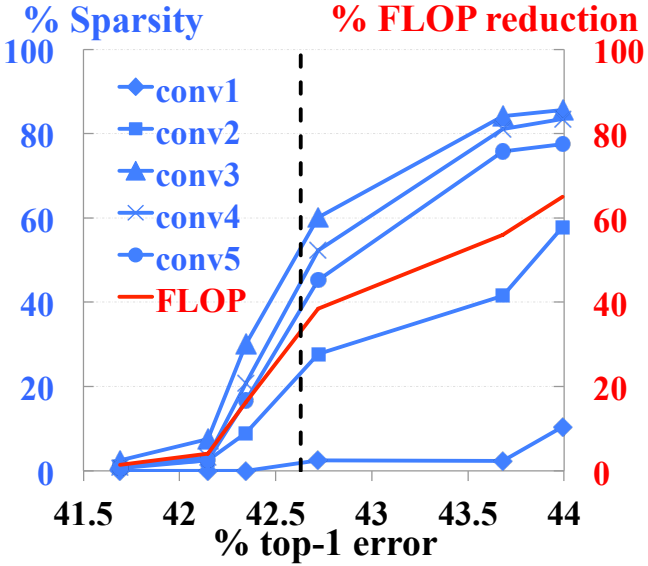
Many groups are pushed to zeros, but not all

Structurally Sparse *LeNet* – Removing Channels and Filters



Structurally Sparse *AlexNet*

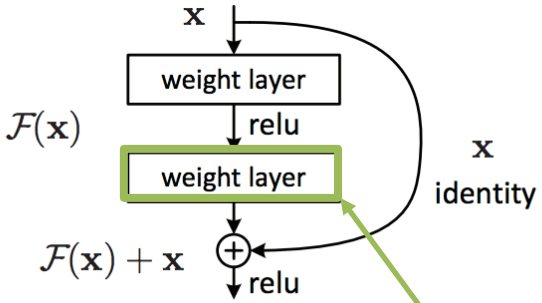




Removing Columns and Rows:

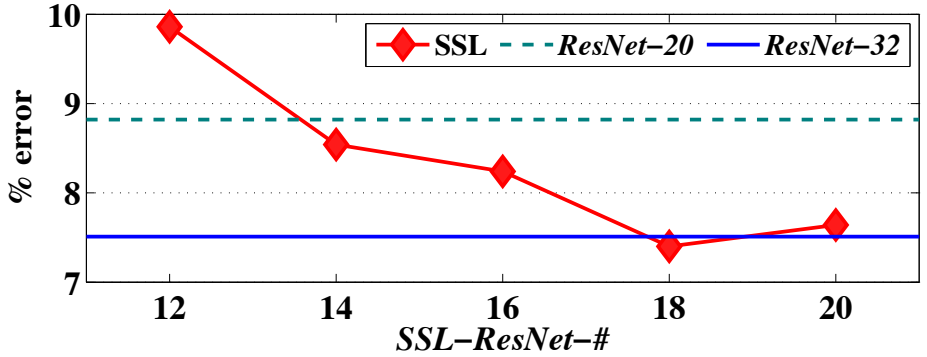
Removing 2D convolution:

Structurally Sparse *ResNets* – Removing Layers



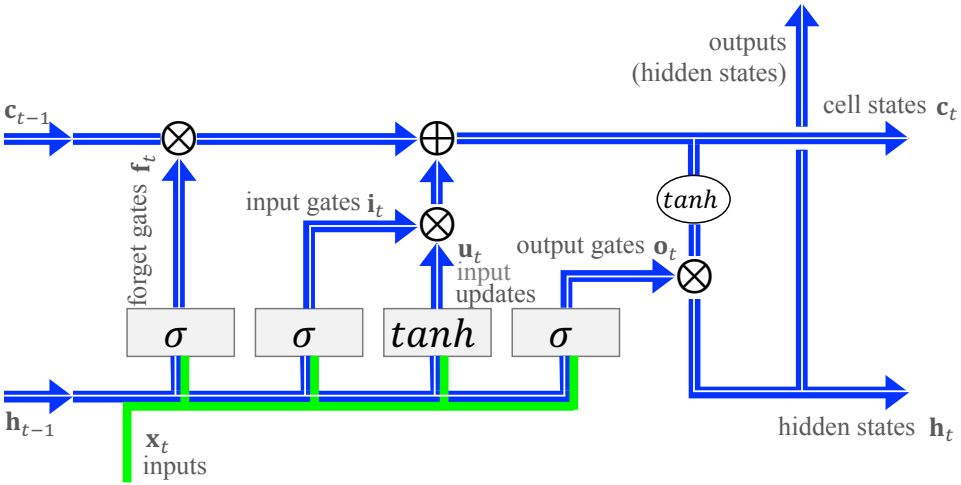
K. He et al. CVPR 2016.

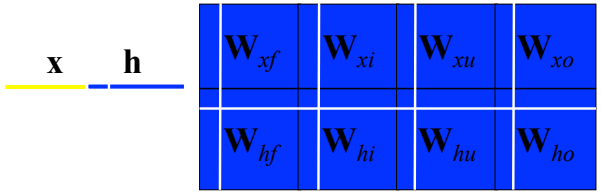
One layer is one group.



	# 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





Weights matrices in LSTM

Weights in next layer(s)

$$\mathbf{i}_t = \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_{xi} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{hi} + \mathbf{b}_i\right)$$
$$\mathbf{f}_t = \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_{xf} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{hf} + \mathbf{b}_f\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_{xo} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{ho} + \mathbf{b}_o\right)$$
$$\mathbf{u}_t = tanh\left(\mathbf{x}_t \cdot \mathbf{W}_{xu} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{hu} + \mathbf{b}_u\right)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{u}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot tanh\left(\mathbf{c}_t\right)$$

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

Removing one group = reducing hidden size by one

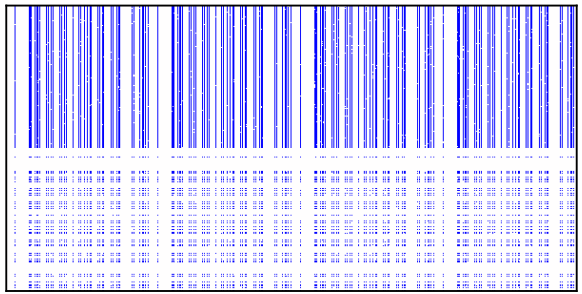
Structurally Sparse *LSTMs*

Method	Dropout keep ratio	Perplexity (validate, test)	ISS # in (1st , 2nd) LSTM	Weight #	Total time*	Speedup	Mult-add reduction†
baseline	0.35	(82.57, 78.57)	(1500, 1500)	66.0M	157.0ms	1.00×	1.00×
ours ISS	0.60	(82.59, 78.65)	(373, 315)	21.8M	14.82ms	10.59×	7.48×
		(80.24, 76.03)	(381, 535)	25.2M	22.11ms	7.10×	5.01×
direct design	0.55	(90.31, 85.66)	(373, 315)	21.8M	14.82ms	10.59×	7.48×

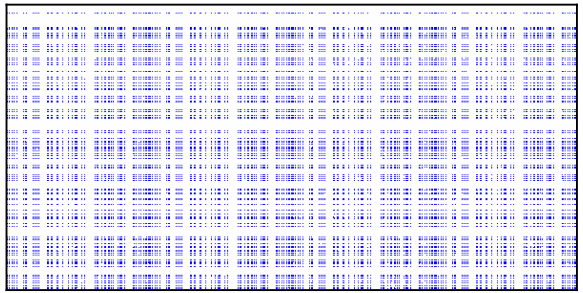
* Measured with 10 batch size and 30 unrolled steps.

† The reduction of multiplication-add operations in matrix multiplication. Defined as (original Mult-add)/(left Mult-add)

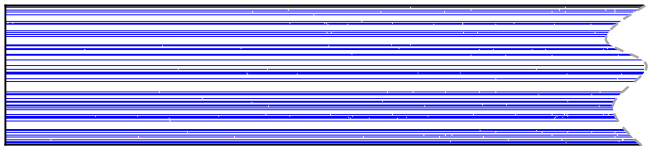
LSTM 1



LSTM 2



Output



Learned structurally sparse LSTMs

Deployment of Structurally Sparse *LSTMs*

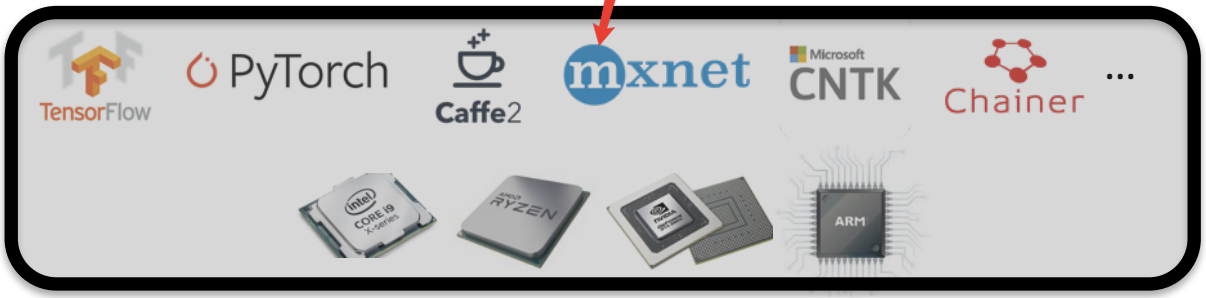
Create a smaller LSTM with the learned hidden sizes

↓

Initialize the smaller LSTM by the non-zero parameters

↓

Deploy



Faster inference
Easier deployment

A black box
(No software or hardware customization)

Generalizing to Different Recurrent Structures

Recurrent Highway Networks (Zilly et al. 2016), a STOA language modeling on Penn Treebank dataset

Table 2: Learning ISS sparsity from scratch in RHNs.

	Method	λ	Perplexity (validate, test)	RHN width	Parameter #
	baseline	0.0	(67.9, 65.4)	830	23.5M
Ours	ISS*	0.004	(67.5, 65.0)	726	18.9M
Ours	ISS*	0.005	**(68.1, 65.4)**	**517**	**11.1M**
Ours	ISS*	0.006	(70.3, 67.7)	403	7.6M
Ours	ISS*	0.007	(74.5, 71.2)	328	5.7M

Machine Question Answering on SQuAD dataset. **BiDAF** model, a compact model with **2.7M** parameters

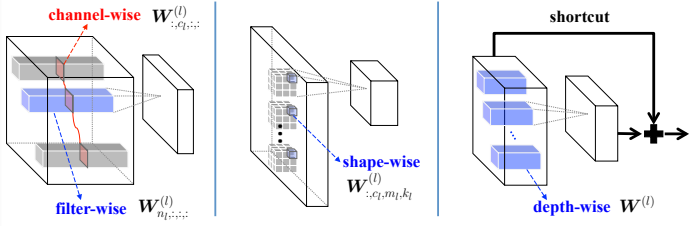
Table 4: Remaining ISS components in BiDAF by training from scratch.

EM	F1	ModFwd1	ModBwd1	ModFwd2	ModBwd2	OutFwd	OutBwd	weight #	Total time*
67.98	77.85	100	100	100	100	100	100	2.69M	6.20ms
67.36	77.16	87	81	87	92	74	96	2.29M	5.83ms
66.32	76.22	51	33	42	58	37	26	1.17M	4.46ms
65.36	75.78	20	33	40	38	31	16	0.95M	3.59ms
64.60	74.99	23	22	35	35	25	14	0.88M	2.74ms

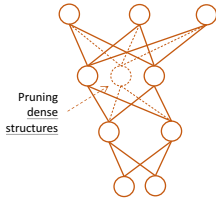
* Measured with batch size 1.

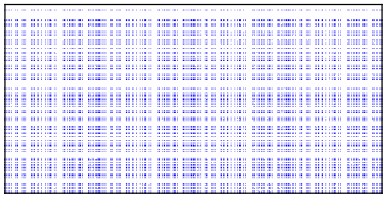
Research Summary on Efficient Deep Learning

This presentation

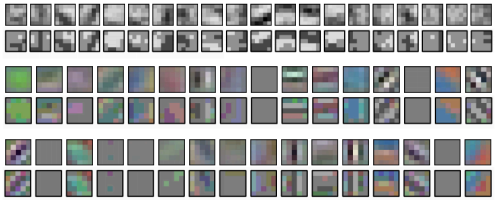


Wen et al. NIPS 2016

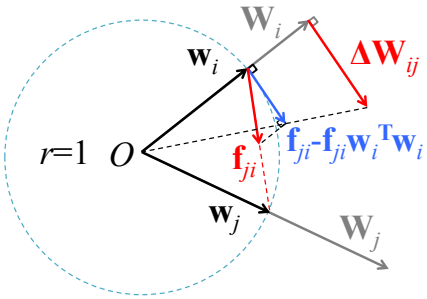




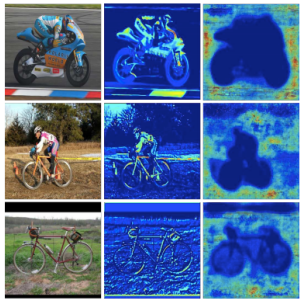
Wen et al. ICLR 2018



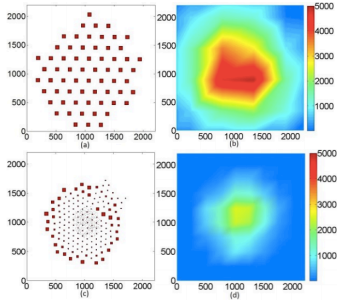
Wang et al. ASP-DAC 2017
(Best Paper Award)



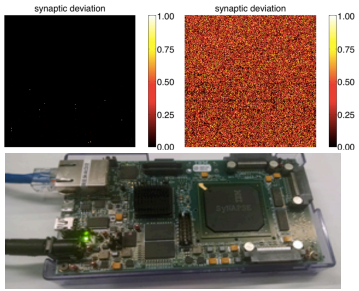
Wen et al. ICCV 2017



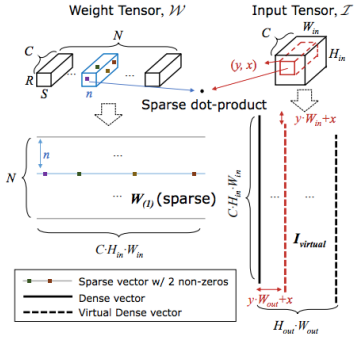
Wu et al. CVPR 2017



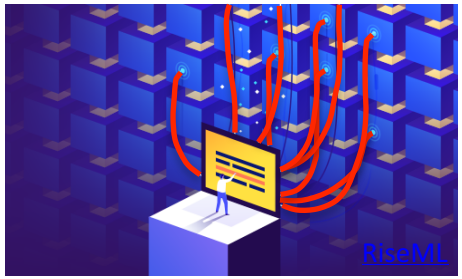
Wen et al. DAC 2015
(Best Paper Nomination)

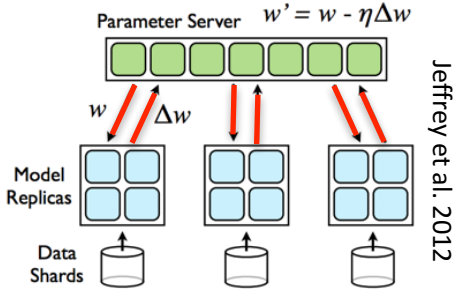


Wen et al. DAC 2016
(Best Paper Nomination)



Park et al. ICLR 2017





TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning

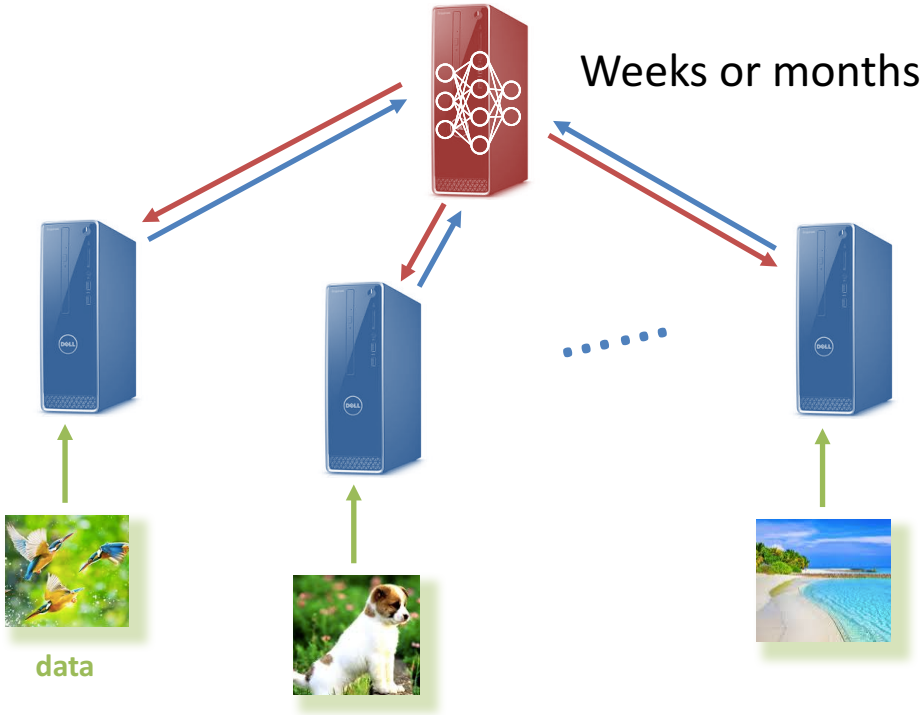
NIPS 2017 (Oral)

Wei Wen[1], Cong Xu[2], Feng Yan[3], Chunpeng Wu[1],
Yandan Wang[4], Yiran Chen[1], Hai (Helen) Li[1]

Duke University[1], Hewlett Packard Labs[2],
University of Nevada - Reno[3], University of Pittsburgh[4]

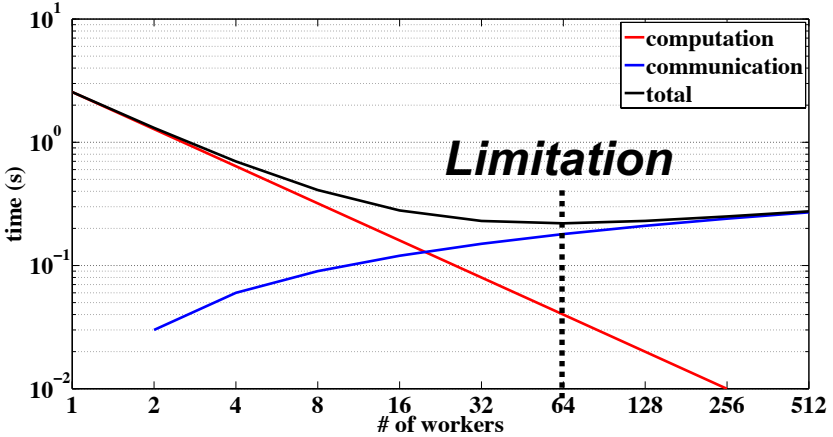
https://github.com/wenwei202/terngrad

Background - Distributed Deep Learning

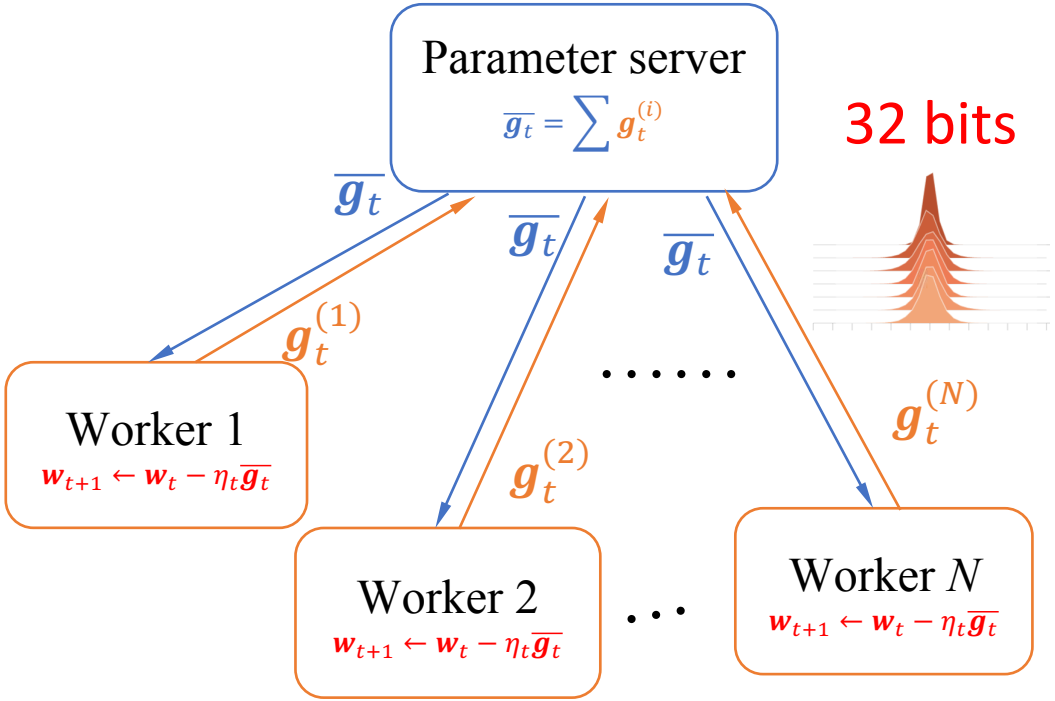


Weeks or months -> hours

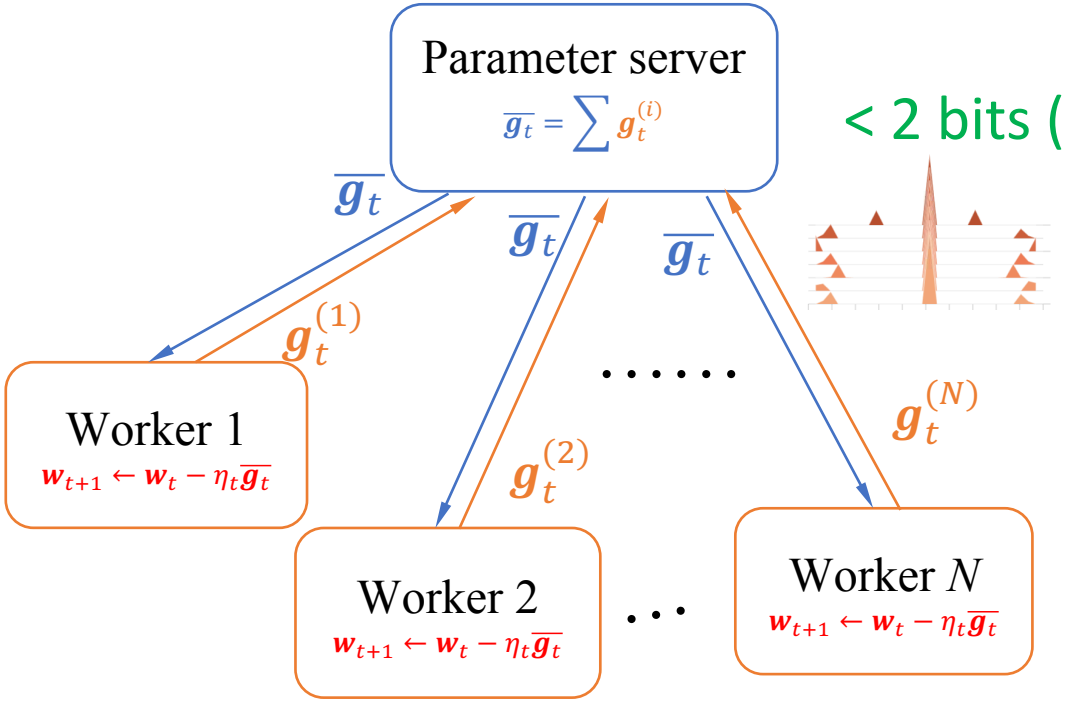
Communication Bottleneck!



Gradient Quantization for Communication Reduction



Gradient Quantization for Communication Reduction



< 2 bits (>16x reduction)

ternary gradients (TernGrad)

It's challenging because it loses precision in the learning which relies on small updates

Stochastic Gradients without Bias

Batch Gradient Descent

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

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

SGD

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

I is randomly drawn from [1,n]

$$\mathrm{E}\left\{g_t^{(I)}\right\} = \nabla C(\boldsymbol{w})$$ No bias

TernGrad

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

$$\mathrm{E}\left\{ternarize\left(g_t^{(I)}\right)\right\} = \nabla C(\boldsymbol{w})$$ No bias

TernGrad is Simple

$$\tilde{\boldsymbol{g}}_t = ternarize(\boldsymbol{g}_t) = s_t \cdot sign\left(\boldsymbol{g}_t\right) \circ \boldsymbol{b}_t$$

$$s_t \triangleq ||\boldsymbol{g}_t||_\infty \triangleq max\left(abs\left(\boldsymbol{g}_t\right)\right)$$

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

Example:

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

s_t: 1.20

Signs: [1, -1, …, 1]

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

$\boldsymbol{b}_t$: [0,1,…,1]

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

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

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

No bias

Convergence

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

Assumption 1:

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

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):

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

Standard SGD *almost-truly* converges

Assumption 3 (gradient bound):

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

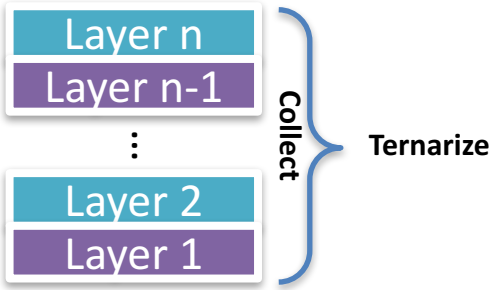
TernGrad almost-truly converges

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

Stronger gradient bound in *TernGrad*

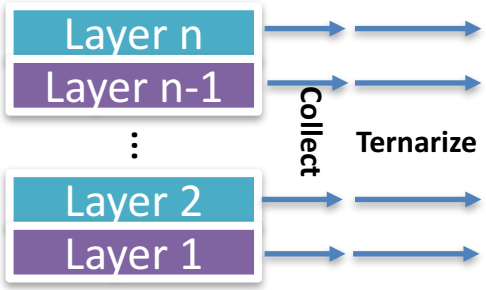
Closing Bound Gap

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

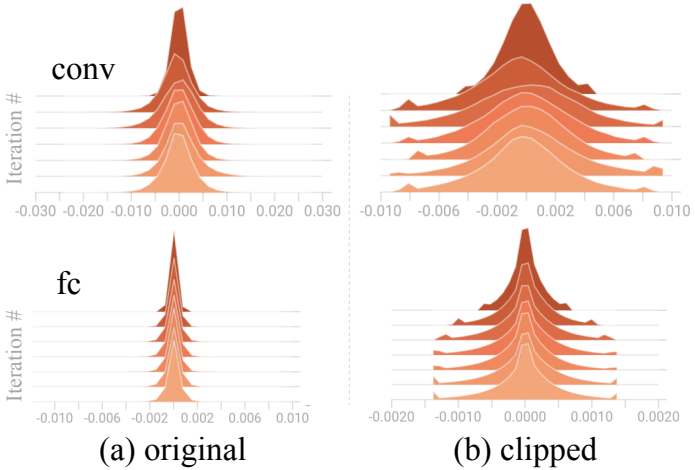
Approach:

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



(a) original (b) clipped

Gradient clipping

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

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

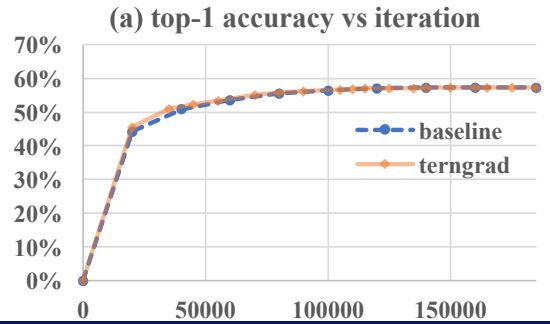
AlexNet

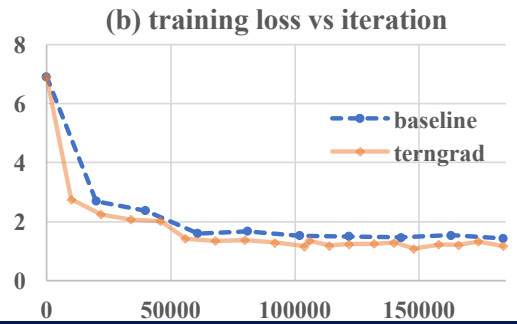
base LR	mini-batch size	workers	iterations	gradients	weight decay	DR†	top-1	top-5
0.01	256	2	370K	floating	0.0005	0.5	57.33%	80.56%
				TernGrad	0.0005	0.2	57.61%	80.47%
				TernGrad-noclip ‡	0.0005	0.2	54.63%	78.16%
0.02	512	4	185K	floating	0.0005	0.5	57.32%	80.73%
				TernGrad	0.0005	0.2	57.28%	80.23%
0.04	1024	8	92.5K	floating	0.0005	0.5	**56.62%**	80.28%
				TernGrad	0.0005	0.2	**57.54%**	80.25%

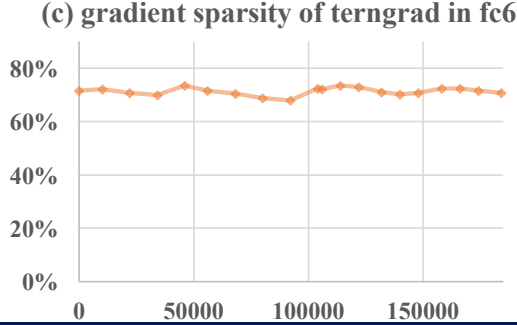
Keskar, et al., 2017

† DR: dropout ratio, the ratio of dropped neurons. ‡ *TernGrad* without gradient clipping.

4 workers







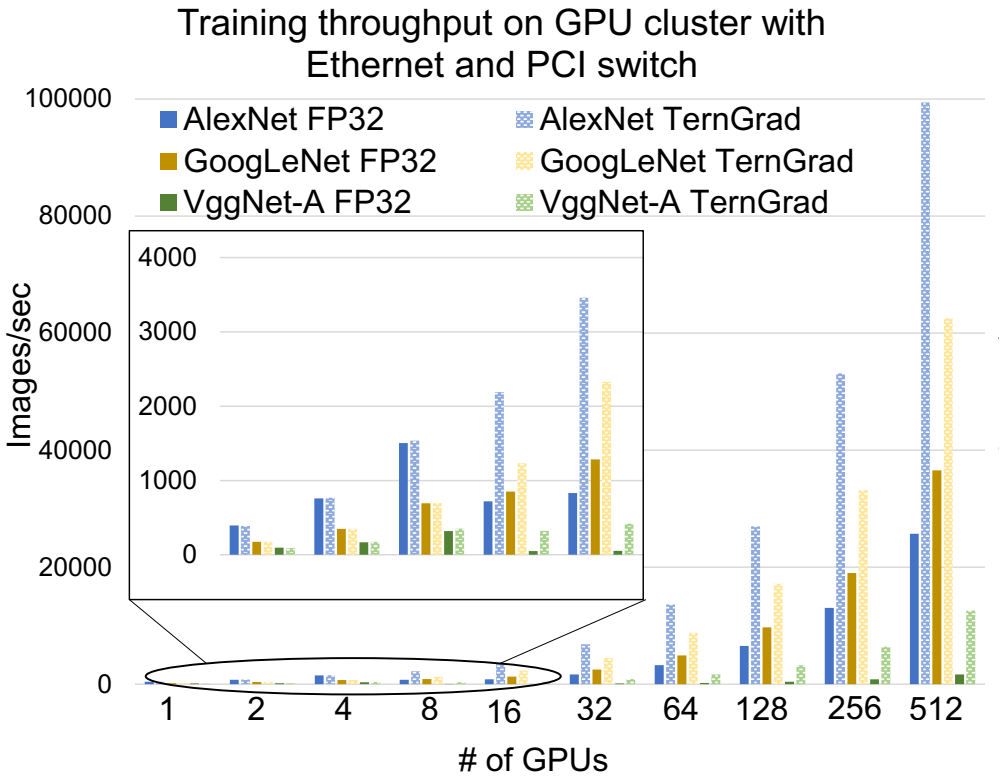
Scaling to Large-scale Deep Learning

GoogLeNet accuracy loss is <2% on avgerage.

base LR	mini-batch size	workers	iterations	gradients	weight decay	DR	top-5
0.04	128	2	600K	floating	4e-5	0.2	88.30%
				TernGrad	1e-5	0.08	86.77%
0.08	256	4	300K	floating	4e-5	0.2	87.82%
				TernGrad	1e-5	0.08	85.96%
0.10	512	8	300K	floating	4e-5	0.2	89.00%
				TernGrad	2e-5	0.08	86.47%

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

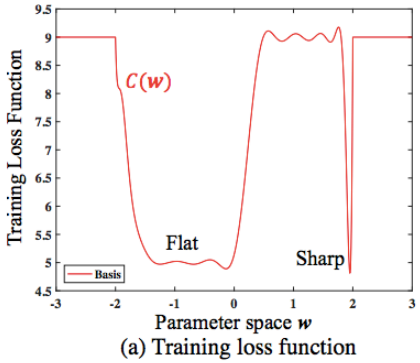
Performance Model



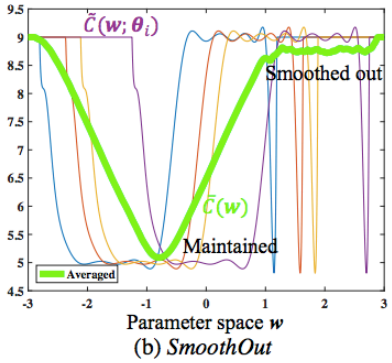
TernGrad gives higher speedup when

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

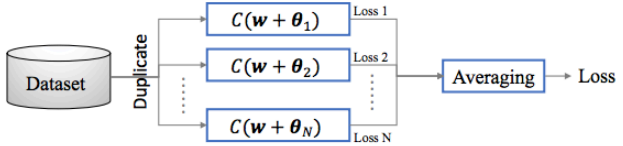
Close Bad Generalization in Large-batch SGD



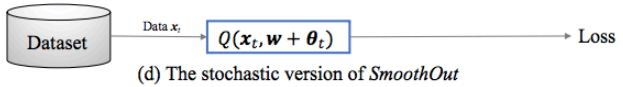
(a) Training loss function



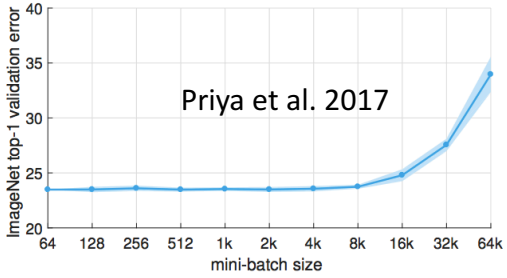
(b) *SmoothOut*

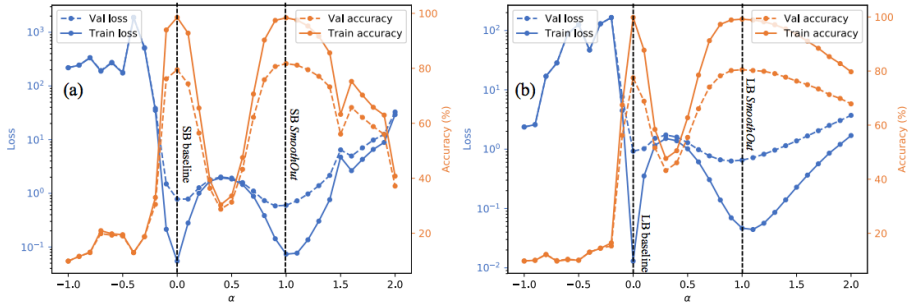


(c) The first version of *SmoothOut*



(d) The stochastic version of *SmoothOut*





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]

Research Adopted in Industry

Structured Sparsity Learning

Ref. Ref.

Ref.

TernGrad

Ref.

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, Beiye 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**, Beiye 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