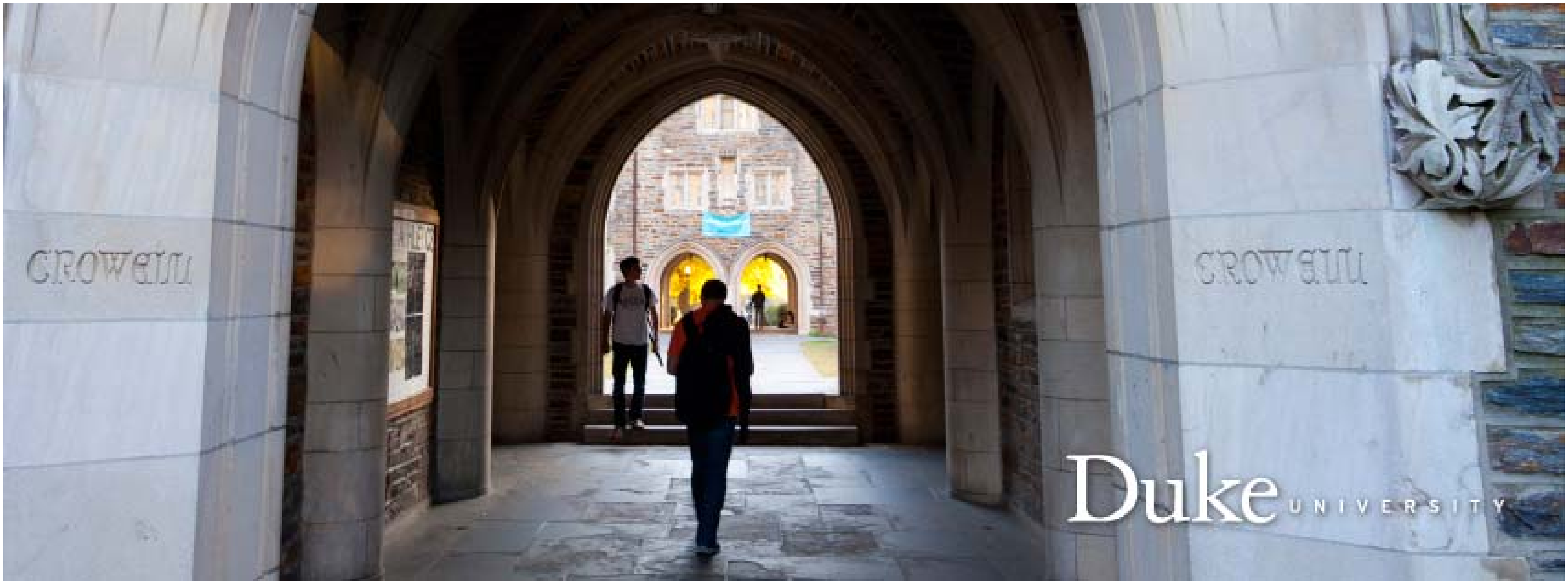


Improving Hardware Efficiency for DNN Applications

Hai (Helen) Li

Electrical and Computing Engineering Department

Duke Center for Evolutionary Intelligence

Technology Landscape

T. Hylton, "Perspectives on Neuromorphic Computing," 2016.

- Sensing
- Display
- Wireless communication & internet
- Computing at the edge

Mobile Phone

Dynamic
Online
Real world

- IoT
- Robotics
- Industrial internet
- Self-driving cars
- Smart Grid
- Secure autonomou networks
- Real-time data to decision

Intelligent Systems

Program

Learn

- Personal computing
- Wired internet

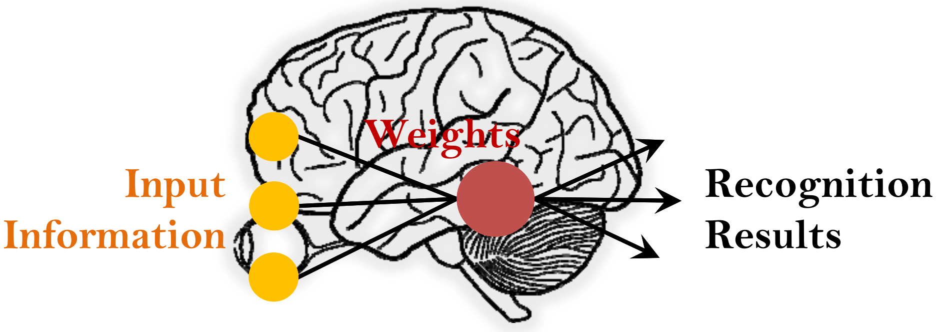
Desktop/Workstation

- Data integration
- Large-scale storage
- Large-scale computing

Static
Offline
Virtual world

Data Center / Cloud

Deep Neural Networks



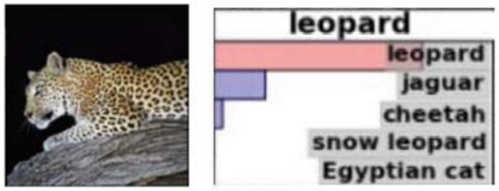
Input Information

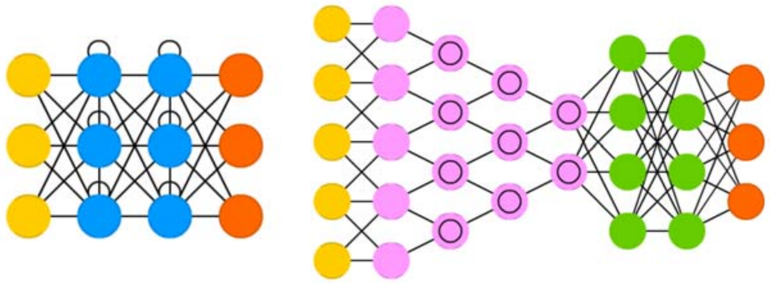
Weights

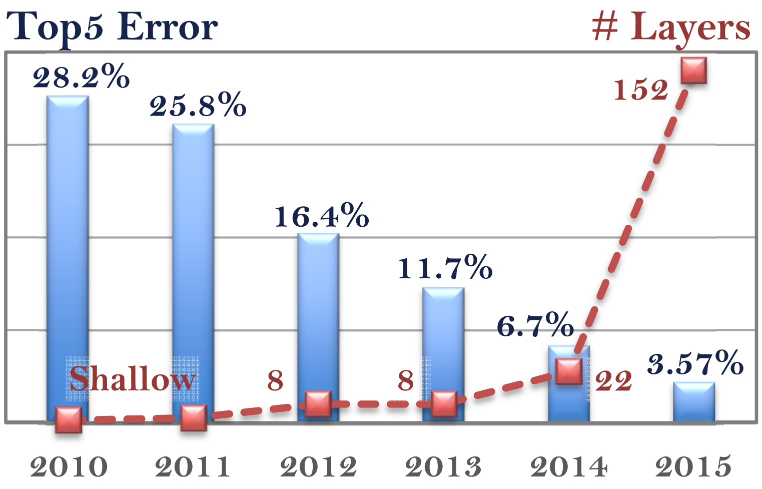
Recognition Results

ImageNet Challenge (ILSVRC)

- Dataset: 1.2M images in 1K categories
- Classification: make 5 guesses







Deeper Model

→ Lower Error Rate

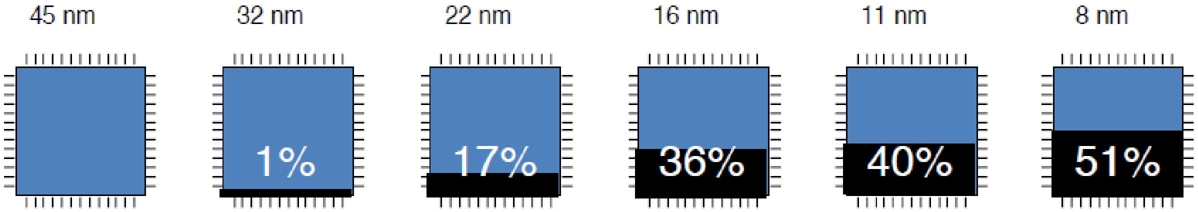
→ Higher Requirement on Computation

Data Explosion & Hardware Development

- **Every minute**, we send over **200 million emails**, click almost **2 million likes on Facebook**, send almost **300K tweets** and **up-load 200K photos to Facebook** as well as **100 hours of video to YouTube**.
- "Data centers consumes up to 1.5% of all the world's electricity…"
- "Google's data centers draw almost 260 MW of power, which is more power than Salt Lake City uses…"

J. Glanz, "Google Details, and Defends, Its Use of Electricity"

Google's "Council Bluffs" data center facilities in Iowa.



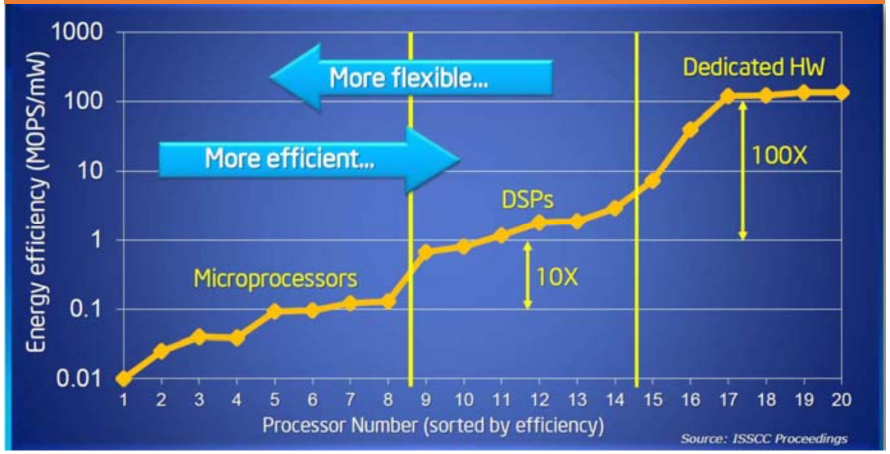
2X transistor count,
But only 40% faster,
50% more efficient…

L. Ceze, "Approximate Overview of Approximate Computing".

Hardware Acceleration for DNNs

- **GPUs**
  - Fast, but high power consumption (~200W)
  - Training DNNs in back-end GPU clusters
- **FPGAs**
  - Massively parallel + low-power (~25W) + reconfigurable
  - Suitable for latency-sensitive real-time inference job
- **ASICs**
  - Fast + energy efficient
  - Long development cycle
- **Novel architectures and emerging devices**

Efficiency vs. Flexibility



Source: ISSCC Proceedings

Mismatch: Software vs. Hardware

	Software	Hardware
Model/Component scale	Large	Small/Moderate
Reconfigurability	Easy	Hard
Accuracy vs. Power	Accuracy	Tradeoff
Training implementation	Easy	Hard
Precision vs. **Limited programmability**	Double (high) precision	Low precision (often a few bits)
Connectivity realization	Easy	Hard

Our work: Improve Efficiency for DNN Applications thorugh Software/Hardware Co-Design Framework

Outline

Our work: Improve Efficiency for DNN Applications Through Software/Hardware Co-Design

- Introduction
- Research Spotlights
  - Structured Sparsity Regularization (NIPS'16)
  - Local Distributed Mobile System for DNN
  - ApesNet for Image Segmentation
- Conclusion

Related Work

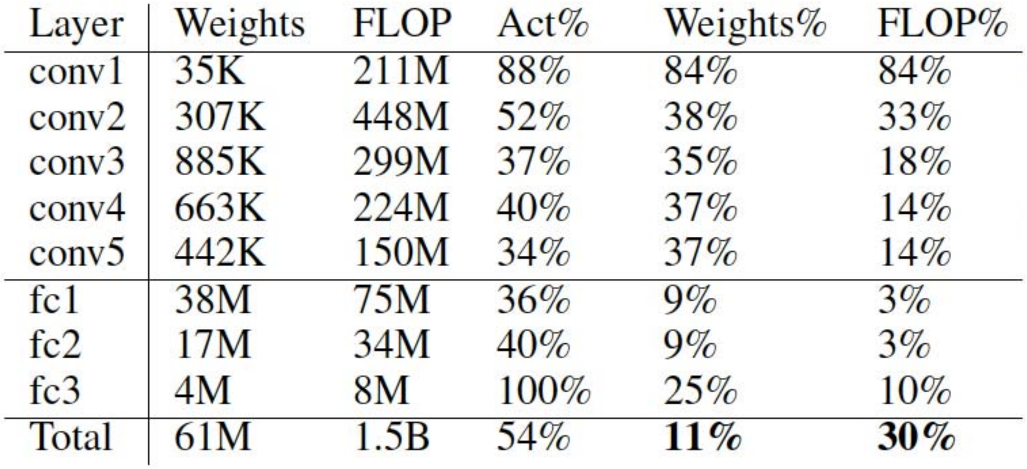
- State-of-the-art methods to reduce the number of parameters
  - Weight regularization (L1-norm)

AlexNet, B. Liu, *et al.*, CVPR 2015

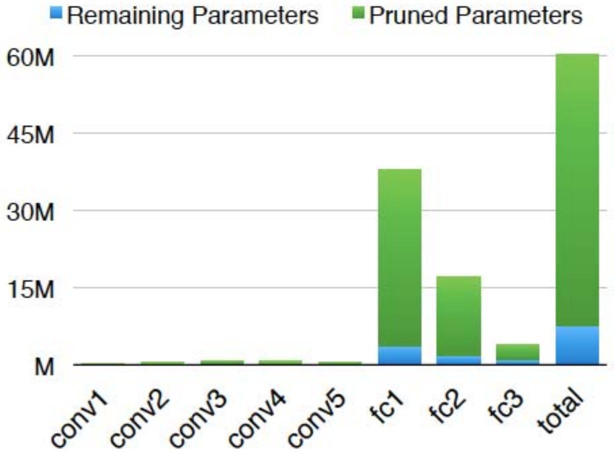
Layer	*conv1*	*conv2*	*conv3*	*conv4*	*conv5*
Sparsity	0.927	0.95	0.951	0.942	0.938
Theoretical speedup	2.61	7.14	16.12	12.42	10.77

  - Connection pruning

AlexNet, S. Han, *et al.*, NIPS 2015

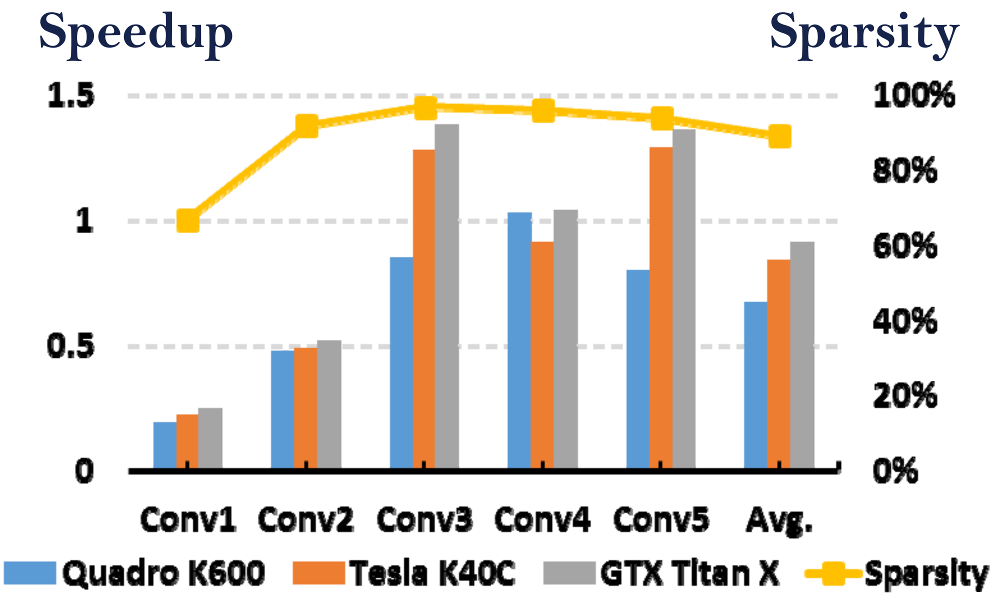


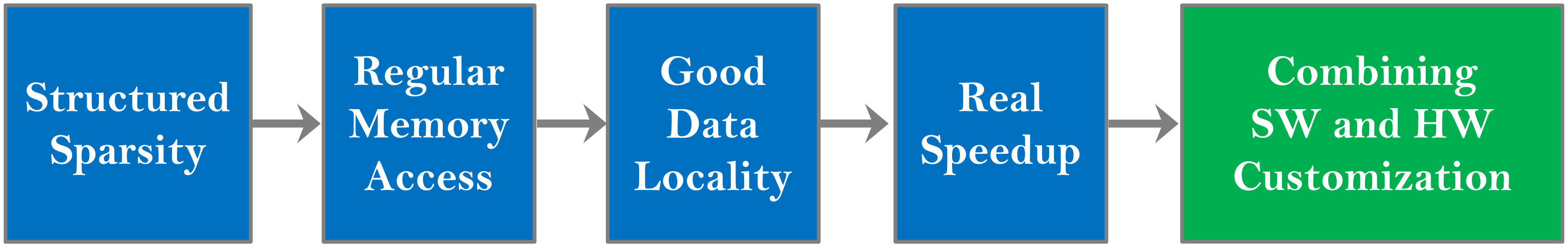
				Remained	
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**



Theoretical Speedup ≠ Practical Speedup

- Forwarding speedups of AlexNet on GPU platforms and the sparsity.
- Baseline is GEMM of cuBLAS.
- The sparse matrixes are stored in the format of *compressed sparse row* (CSR) and accelerated by cuSPARSE.




Structured Sparsity → Regular Memory Access → Good Data Locality → Real Speedup → Combining SW and HW Customization

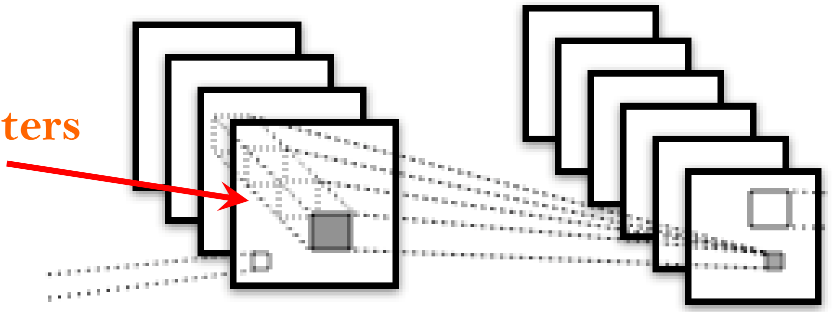
B. Liu, et al., "Hardcoding nonzero weights in source code," CVPR'15

S. Han, et al., "Customizing an EIE chip accelerator for compressed DNN," ISCA'17

Computation-efficient Structured Sparsity

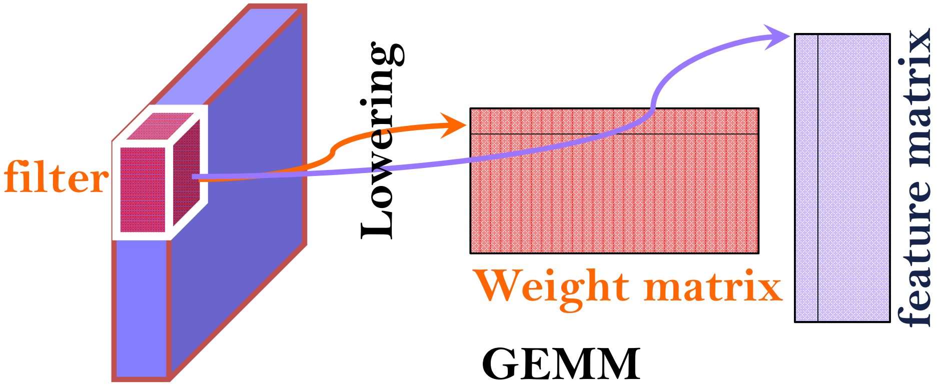
Example 1: Removing 2D filters in convolution (2D-filter-wise sparsity)

3D filter =
stacked 2D filters



http://deeplearning.net/

Example 2: Removing rows/columns in GEMM (row/column-wise sparsity)



feature map

filter

Lowering

Weight matrix

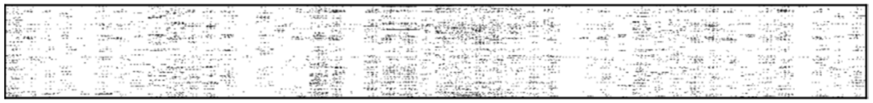
feature matrix

GEMM

GEneral Matrix Matrix Multiplication

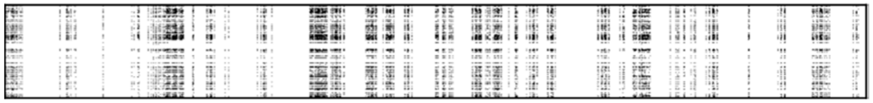
Non-structured sparsity

conv2_1: weight sparsity (col:8.7% row:19.5% elem:94.6%)



Structured sparsity

conv2_1: weight sparsity (col:75.2% row:21.9% elem:91.5%)



5.17X speedup

Structured Sparsity Regularization

- Group Lasso regularization in ML model

$$\arg\min_{\mathbf{w}} \left\{ E\left(\mathbf{w}\right) \right\} = \arg\min_{\mathbf{w}} \left\{ E_D\left(\mathbf{w}\right) + \lambda_g \cdot R_g\left(\mathbf{w}\right) \right\}$$

□□

$$\arg\min_{\mathbf{w}} \left\{ E\left(\mathbf{w}\right) \right\} = \arg\min_{\mathbf{w}} \left\{ E_D\left(\mathbf{w}\right) \right\}$$

$$s.t.\ R_g\left(\mathbf{w}\right) \le \eta_g$$

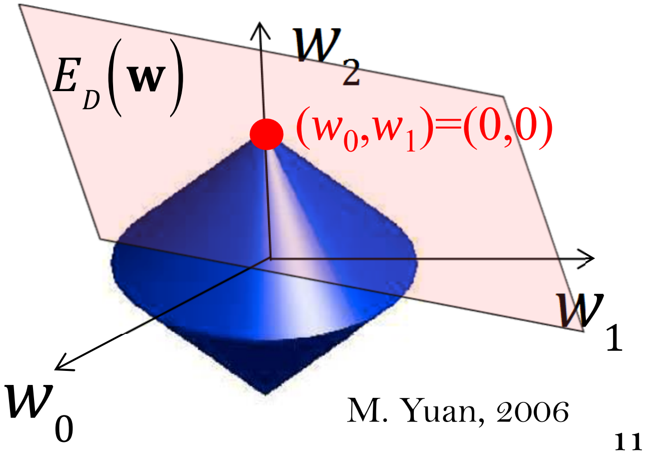
Many groups will be zeros

$$R_g(\boldsymbol{w}) = \sum_{g=1}^{G} ||\boldsymbol{w}^{(g)}||_g,$$

$$||\boldsymbol{w}^{(g)}||_g = \sqrt{\sum_{i=1}^{|\boldsymbol{w}^{(g)}|} \left(w_i^{(g)}\right)^2}$$

Example:

$$R_g\left(\underbrace{\boxed{w_0, w_1}}_{\text{group 1}}, \underbrace{\boxed{w_2}}_{\text{group 2}}\right) = \sqrt{w_0^2 + w_1^2} + \sqrt{w_2^2} \le \eta_g$$



M. Yuan, 2006

SSL: Structured Sparsity Learning

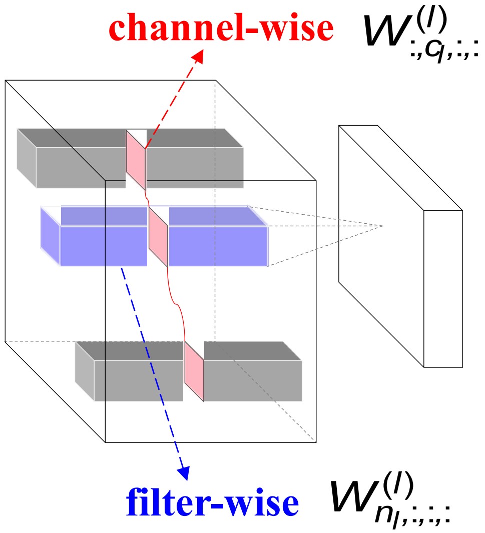
- Group Lasso regularization in DNNs

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda \cdot R(\boldsymbol{W}) + \lambda_g \cdot \sum_{l=1}^{L} R_g\left(\boldsymbol{W}^{(l)}\right)$$

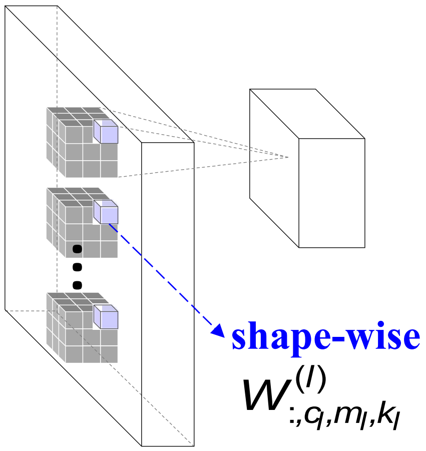
$$R_g(\boldsymbol{w}) = \sum_{g=1}^{G} ||\boldsymbol{w}^{(g)}||_g$$

- The learned structured sparsity is determined by the way of splitting groups

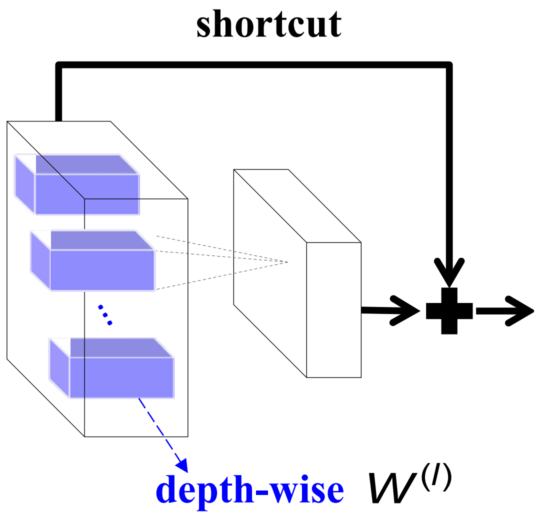
Penalize unimportant filters and channels


channel-wise $W^{(l)}_{:,c_l,:,:}$
filter-wise $W^{(l)}_{n_l,:,:,:}$

Learn filter shapes



Learn the depth of layers



SSL: Structured Sparsity Learning

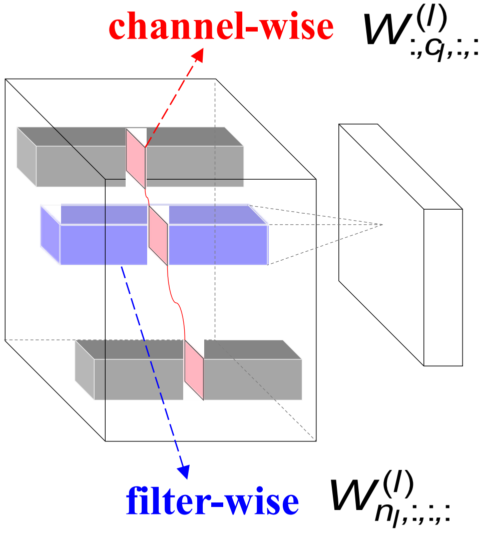
- Group Lasso regularization in DNNs

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda_n \cdot \sum_{l=1}^{L}\left(\sum_{n_l=1}^{N_l} ||\boldsymbol{W}^{(l)}_{n_l,:,:,:}||_g\right) + \lambda_c \cdot \sum_{l=1}^{L}\left(\sum_{c_l=1}^{C_l} ||\boldsymbol{W}^{(l)}_{:,c_l,:,:}||_g\right).$$

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda_s \cdot \sum_{l=1}^{L}\left(\sum_{c_l=1}^{C_l}\sum_{m_l=1}^{M_l}\sum_{k_l=1}^{K_l} ||\boldsymbol{W}^{(l)}_{:,c_l,m_l,k_l}||_g\right).$$

$$E(\boldsymbol{W}) \;=\; E_D(\boldsymbol{W}) + \lambda_d \cdot \textstyle\sum_{l=1}^{L} ||\boldsymbol{W}^{(l)}||_g.$$

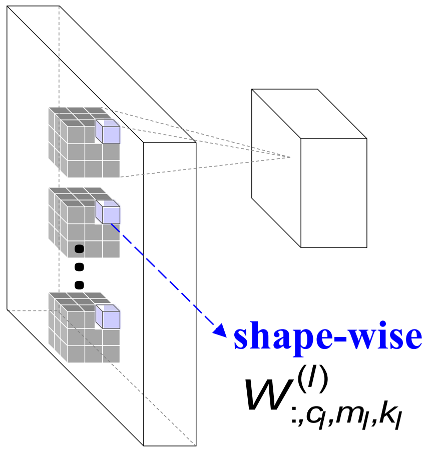
filters and channels



channel-wise $W^{(l)}_{:,c_l,:,:}$

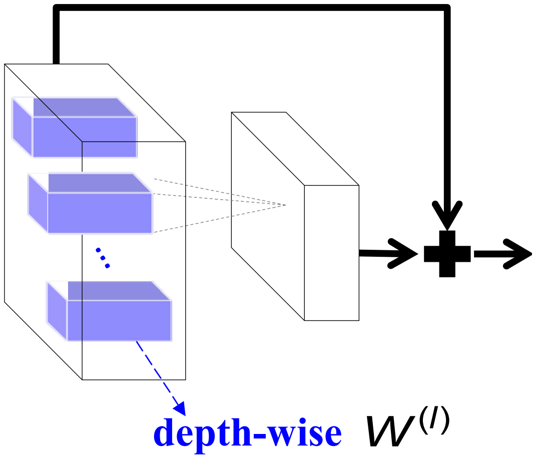
filter-wise $W^{(l)}_{n_l,:,:,:}$

Learn filter shapes



shape-wise $W^{(l)}_{:,c_l,m_l,k_l}$

Learn the depth of layers



shortcut

depth-wise $W^{(l)}$

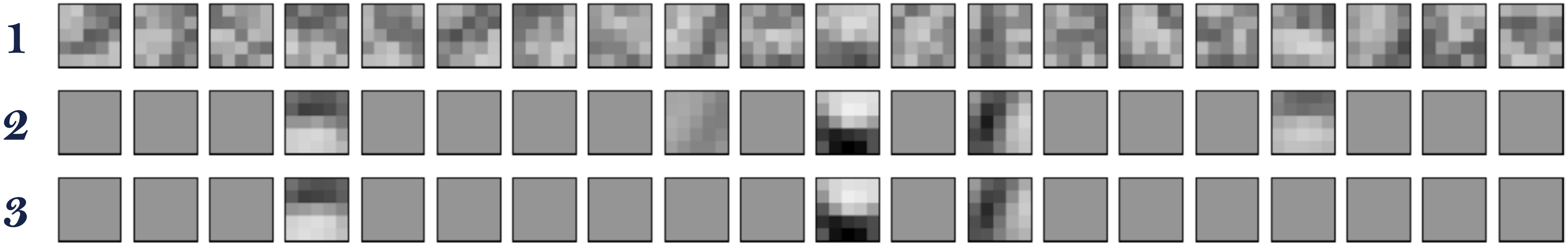
Penalizing Unimportant Filters and Channels

LeNet on MNIST

LeNet #	Error	Filter #	Channel #	FLOP	Speedup
Baseline - 1	0.9%	20 – 50	1 – 20	100% – 100%	1.00× – 1.00×
2	0.8%	5 – 19	1 – 4	25% – 7.6%	1.64× – 5.23×
3	1.0%	3 - 12	1 – 3	15% – 3.6%	1.99× – 7.44×

The data is represented in the order of *conv1 – conv2*.

Conv1 filters (gray level 128 represents zero)



SSL obtains FEWER but more natural patterns

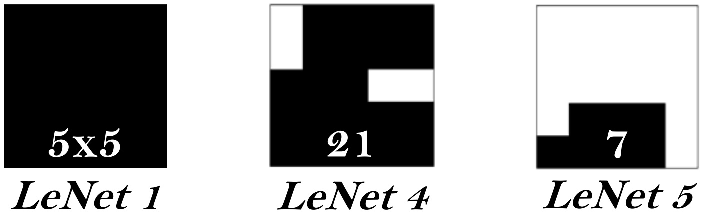
Learning Smaller Filter Shapes

LeNet on MNIST

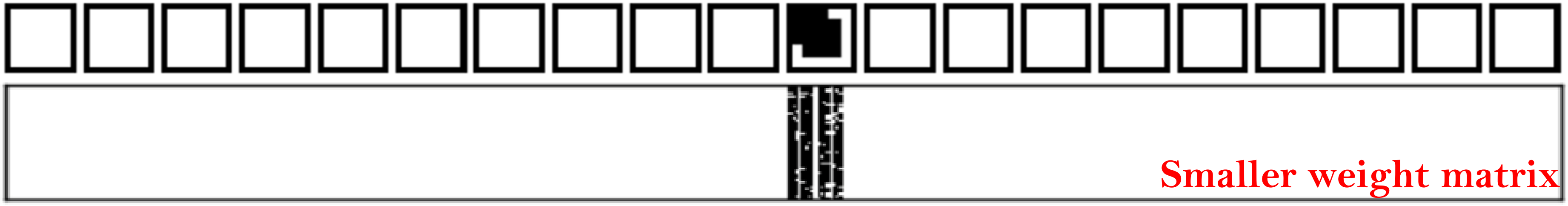
LeNet #	Error	Filter #	Channel #	FLOP	Speedup
Baseline - 1	0.9%	25 – 500	1 – 20	100% – 100%	1.00× – 1.00×
4	0.8%	21 – 41	1 – 2	8.4% – 8.2%	2.33× – 6.93×
5	1.0%	7 - 14	1 – 1	1.4% – 2.8%	5.19× – 10.82×

The size of filters after removing zero shape fibers, in the order of *conv1 – conv2*.

Learned shapes of *conv1* filters:



Learned shape of *conv2* filters @ *LeNet 5*   3D 20x5x5 filters is regularized to 2D filters!



Smaller weight matrix

SSL obtains SMALLER filters w/o accuracy loss

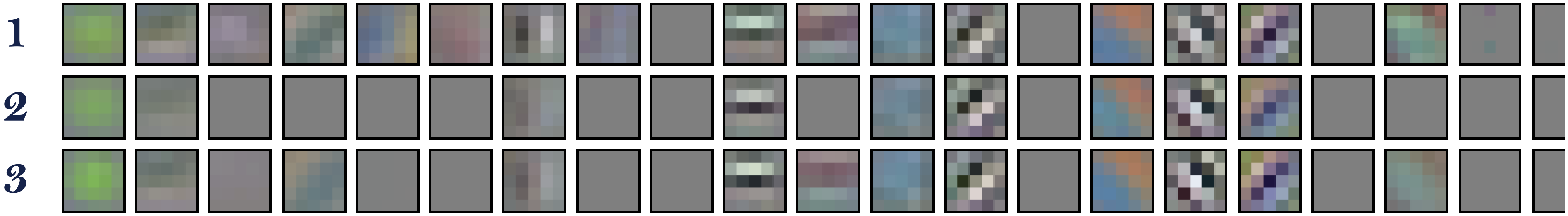
Learning Smaller Dense Weight Matrix

ConvNet on CIFAR-10

ConvNet #	Error	Row Sparsity	Column Sparsity	Speedup
Baseline 1	17.9%	12.5%–0%–0%	0%–0%–0%	1.00×–1.00×–1.00×
4	17.9%	50.0%–28.1%–1.6%	0%–59.3%–35.1%	1.43×–3.05×–1.57×
5	16.9%	31.3%–0%–1.6%	0%–42.8%–9.8%	1.25×–2.01×–1.18×

Row/column sparsity is represented in the order of *conv1–conv2–conv3*.

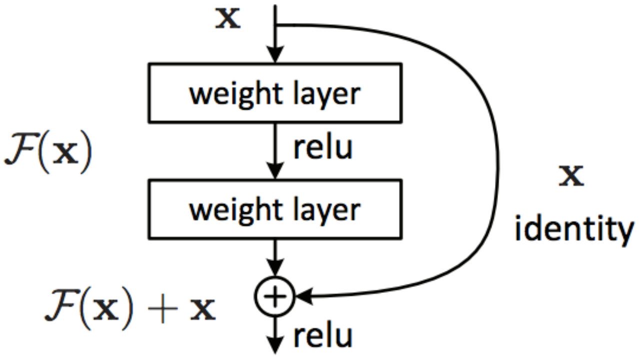
The learned *conv1* filters



SSL can efficiently learn DNNs with smaller but dense weight matrix which has good locality

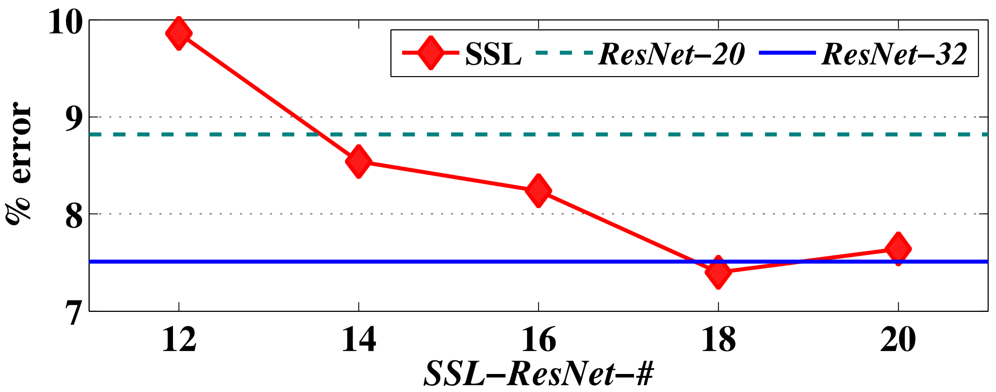
Regularizing The Depth of DNNs

Baseline, K. He, CVPR'16

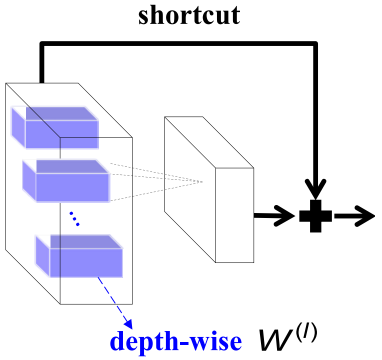


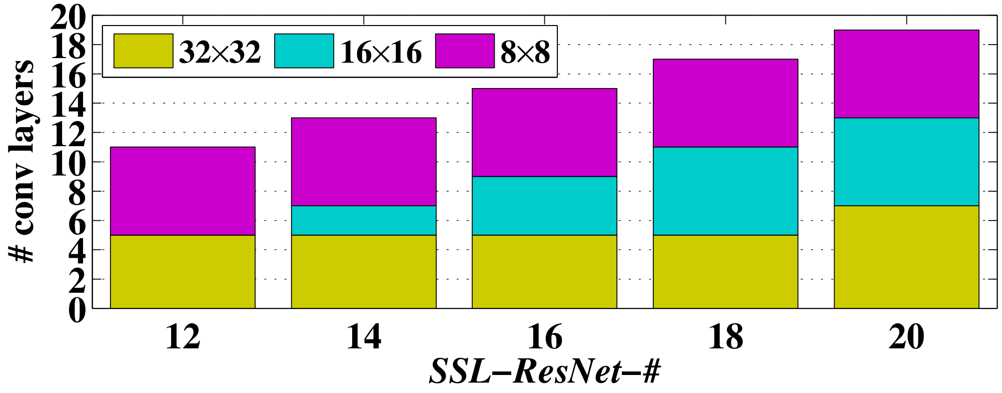
ResNet on CIFAR-10

	# layers	Error	# layers	Error
ResNet	20	8.82%	32	7.51%
SSL-ResNet	14	8.54%	18	7.40%



Depth-wise SSL



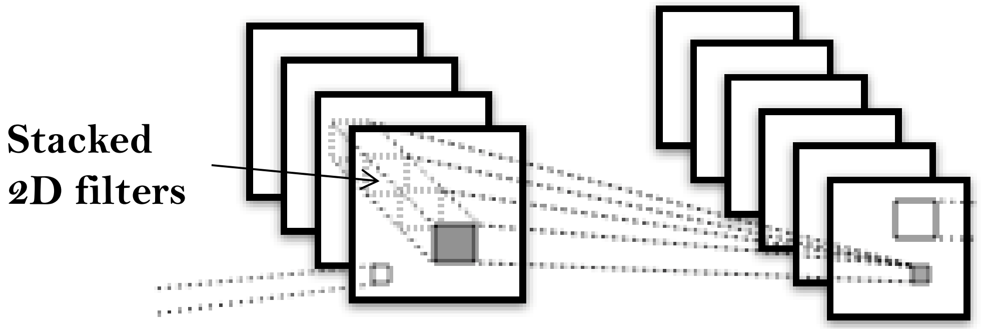


Experiments – AlexNet@ImageNet

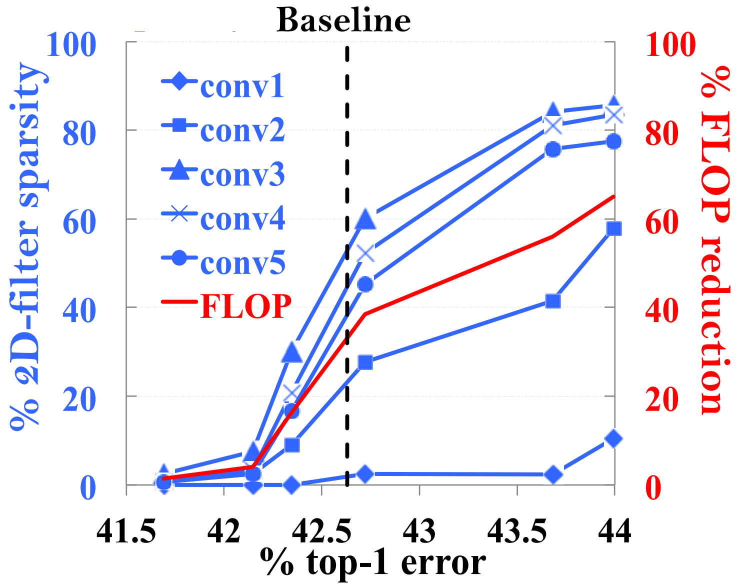
3D convolution = sum of 2D convolutions:

$$\mathbf{F}^{(l+1)}_{c_{l+1},y_{l+1},x_{l+1}} = \sum_{c_l=1}^{C_l}\sum_{m_l=1}^{M_l}\sum_{k_l=1}^{K_l} \mathbf{F}^{(l)}_{c_l,(y_{l+1}+m_l-1),(x_{l+1}+k_l-1)} \cdot \mathbf{W}^{(l)}_{n_l,c_l,m_l,k_l}$$

Learning 2D-filter-wise sparsity



http://deeplearning.net/

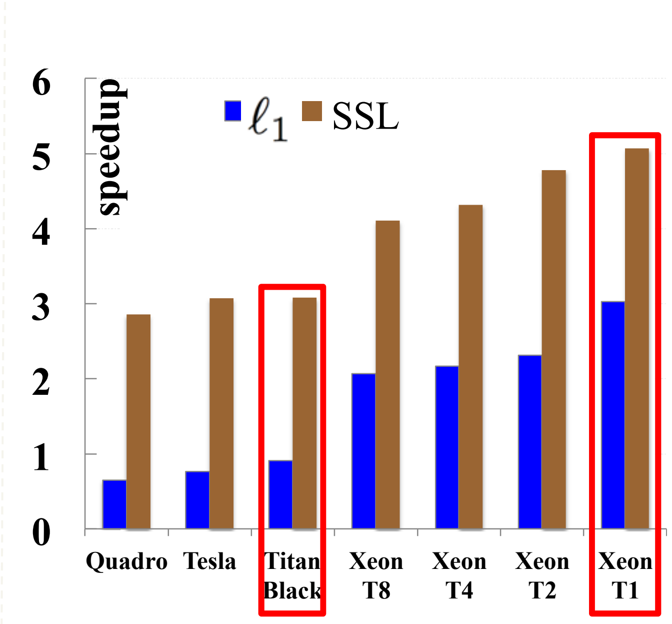


- SSL saves 30-40% (60-70%) FLOPs with 0 (<1.5%) accuracy loss by structurally removing 2D filters.
- Deeper layers have higher sparsity.

Experiments – AlexNet@ImageNet

Higher speedups than non-structured speedups

- 5.1X/3.1X layer-wise speedup on CPU/GPU with 2% accuracy loss
- 1.4X layer-wise speedup on both CPU and GPU w/o accuracy loss



2% loss

	Method	Top1 Error	Statistics	*conv1*	*conv2*	*conv3*	*conv4*	*conv5*
1	L1	**44.67%**	Sparsity	67.6%	92.4%	97.2%	96.6%	94.3%
			CPU	0.80×	2.91×	4.84×	3.83×	2.76×
			GPU	0.25×	0.52×	1.38×	1.04×	1.36×
2	SSL	**44.66%**	Col. Sparsity	0.0%	63.2%	76.9%	84.7%	80.7%
			Row Sparsity	9.4%	12.9%	40.6%	46.9%	0.0%
			CPU	1.05×	3.37×	6.27×	9.73×	4.93×
			GPU	1.00×	2.37×	4.94×	4.03×	3.05×
3	**Pruning***	42.80%	**Sparsity**	**16.0%**	**62.0%**	**65.0%**	**63.0%**	**63.0%**
4	L1	**42.51%**	Sparsity	14.7%	76.2%	85.3%	81.5%	76.3%
			CPU	0.34×	0.99×	1.30×	1.10×	0.93×
			GPU	0.08×	0.17×	0.42×	0.30×	0.32×
5	SSL	**42.53%**	Sparsity	0.00%	20.9%	39.7%	39.7%	24.6%
			CPU	1.00×	1.27×	1.64×	1.68×	1.32×
			GPU	1.00×	1.25×	1.63×	1.72×	1.36×

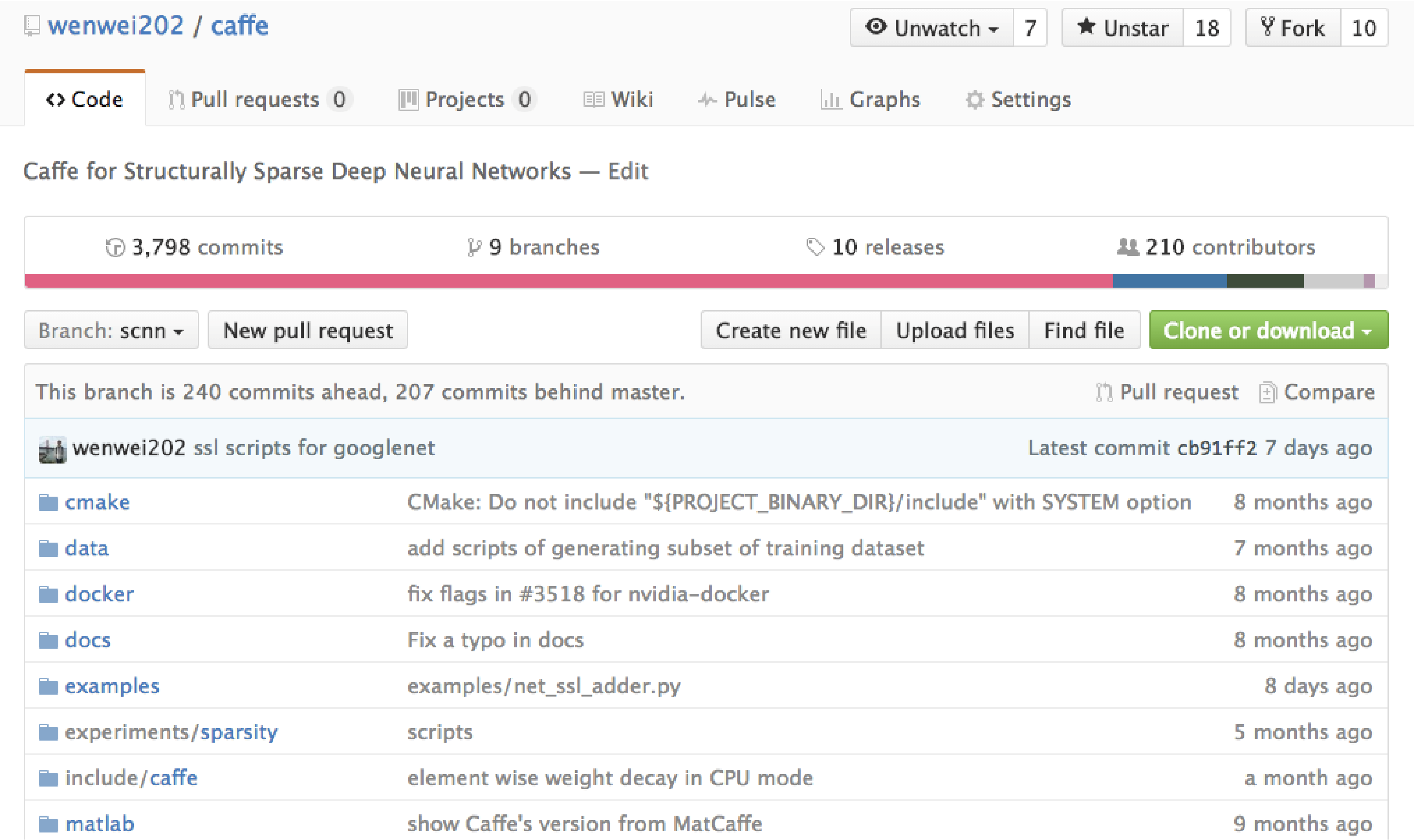
2% loss

Loss free

Duke UNIVERSITY

Open Source

- Source code in Github, and trained model in model zoo
- https://github.com/wenwei202/caffe/tree/scnn



wenwei202 / caffe

Caffe for Structurally Sparse Deep Neural Networks — Edit

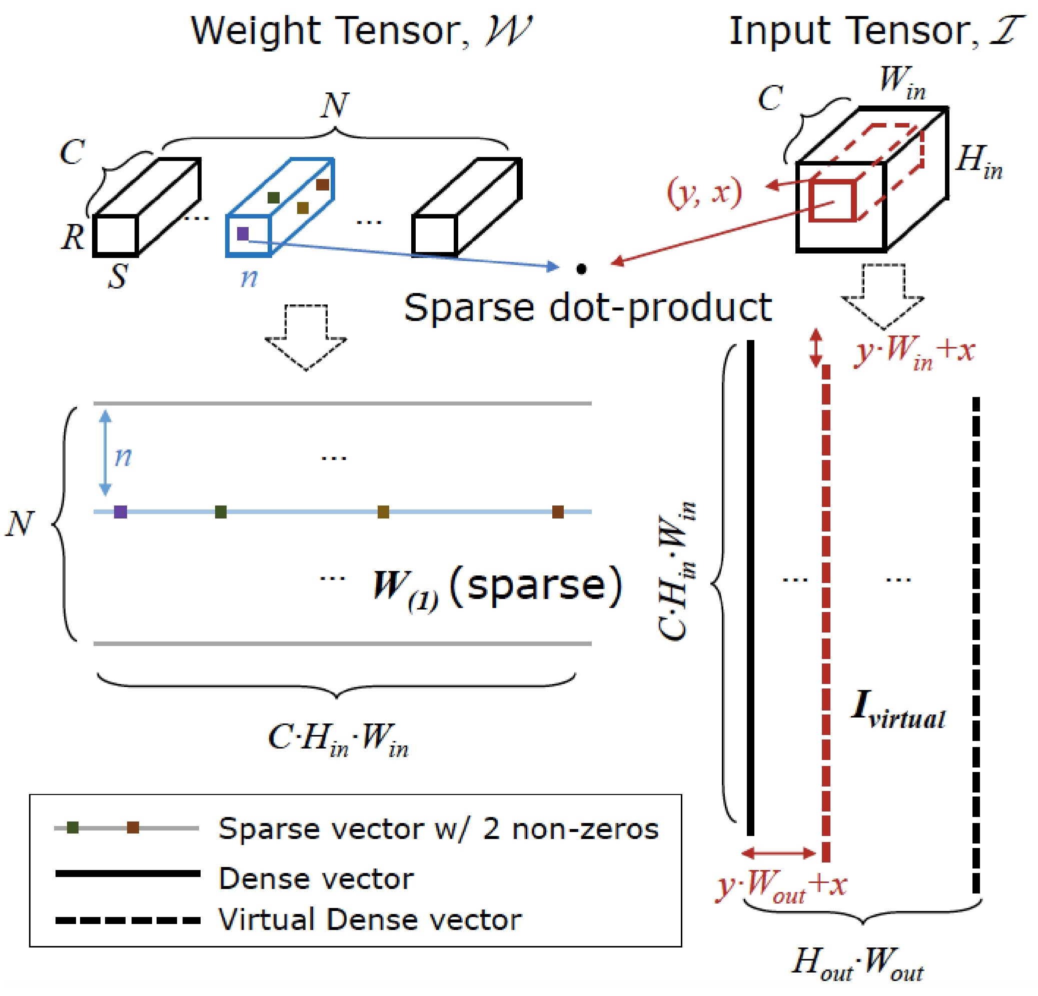
3,798 commits | 9 branches | 10 releases | 210 contributors

This branch is 240 commits ahead, 207 commits behind master.

wenwei202 ssl scripts for googlenet — Latest commit cb91ff2 7 days ago

cmake	CMake: Do not include "${PROJECT_BINARY_DIR}/include" with SYSTEM option	8 months ago
data	add scripts of generating subset of training dataset	7 months ago
docker	fix flags in #3518 for nvidia-docker	8 months ago
docs	Fix a typo in docs	8 months ago
examples	examples/net_ssl_adder.py	8 days ago
experiments/sparsity	scripts	5 months ago
include/caffe	element wise weight decay in CPU mode	a month ago
matlab	show Caffe's version from MatCaffe	9 months ago

Collaborated Work with Intel – ICLR 2017



- ✓ **Direct Sparse Convolution**: an efficient implementation of convolution with sparse filters
- ✓ **Performance Model**: A predictor of the speedup vs. sparsity level
- ✓ **Guided Sparsity Learning**: A performance-model-supervised pruning process that avoids pruning layers, which are unbeneficial for speedup but harmful for accuracy

Jongsoo Park, *et al*, "Faster CNNs with Direct Sparse Convolutions and Guided Pruning," ICLR 2017.

Outline

Our work: Improve Efficiency for DNN Applications Through Software/Hardware Co-Design

- Introduction
- Research Spotlights
  - Structured Sparsity Regularization
  - Local Distributed Mobile System for DNN (DATE'17)
  - ApesNet for Image Segmentation
- Conclusion

Duke
UNIVERSITY

Neural Network on Mobile Platforms

Advantages:

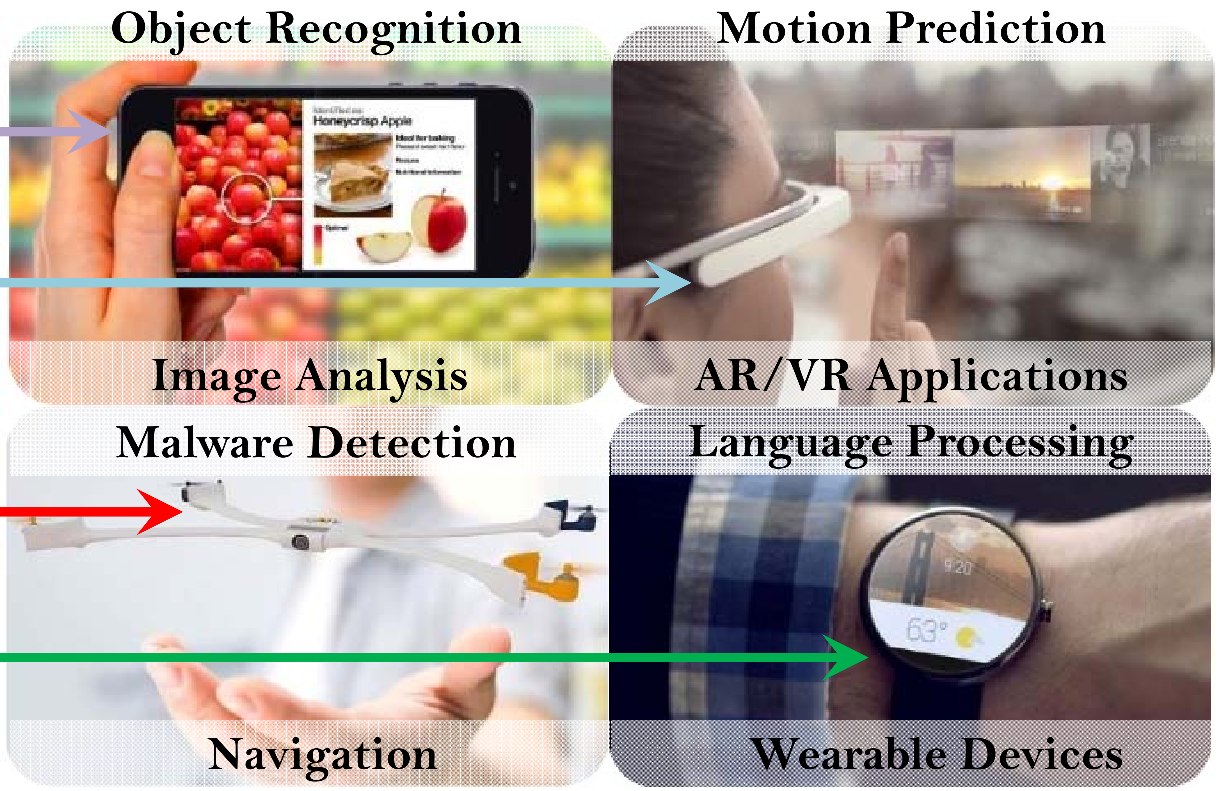
Applications:

(1) A brain-inspired chip from IBM

(2) Google glass

(3) Nixie

(4) Android wear smart watch

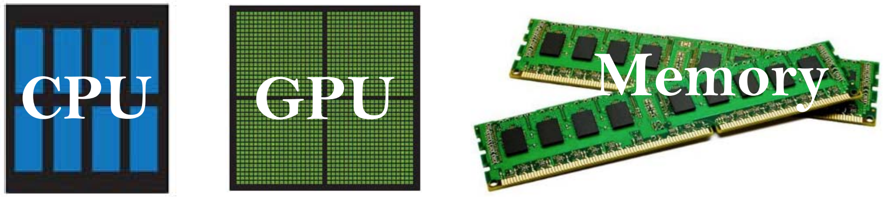


Challenges and Preliminary Analysis

Challenges:

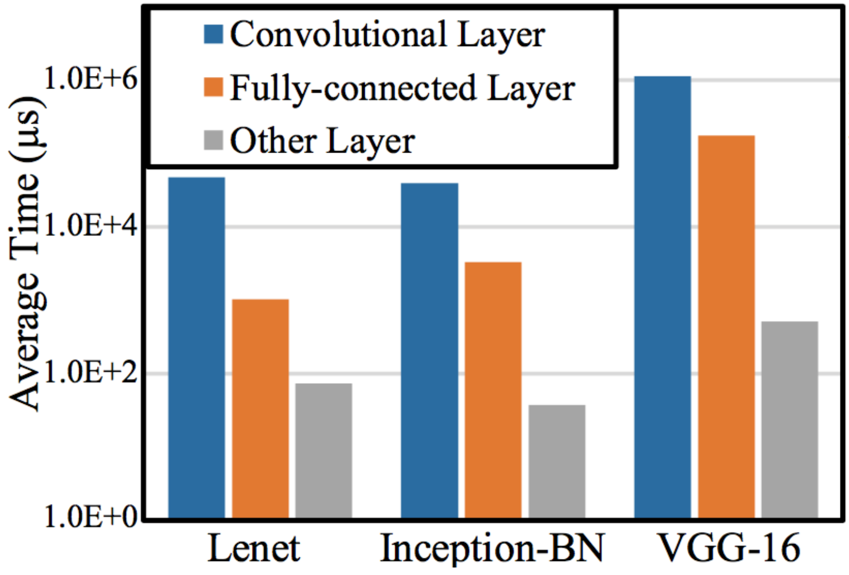
Security

Battery Capacity

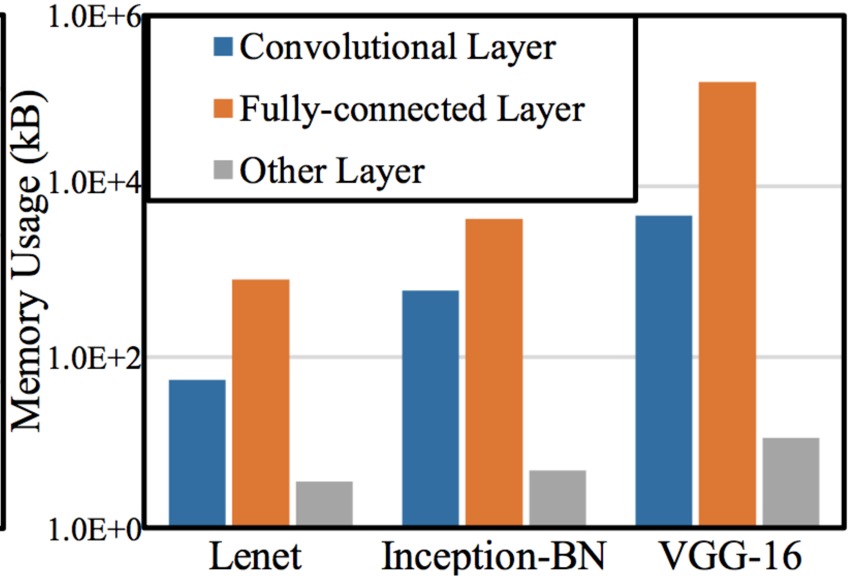


Hardware Performance

Layer Analysis of DNNs on Smartphones:



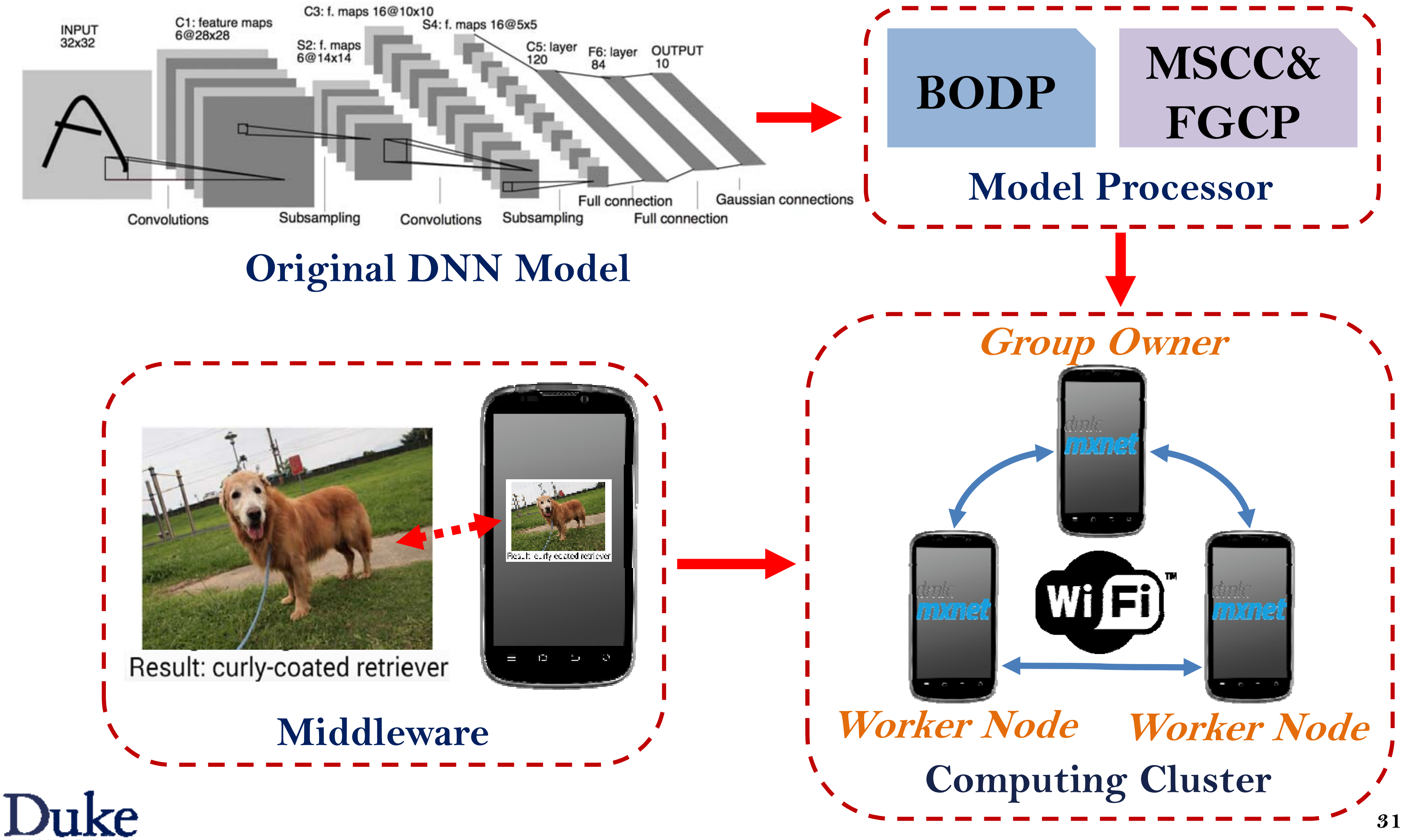
Computing-intensive



Memory-intensive

Local Distributed Mobile System for DNN

System Overview of MoDNN:



Original DNN Model

BODP

MSCC& FGCP

Model Processor

Middleware

Group Owner

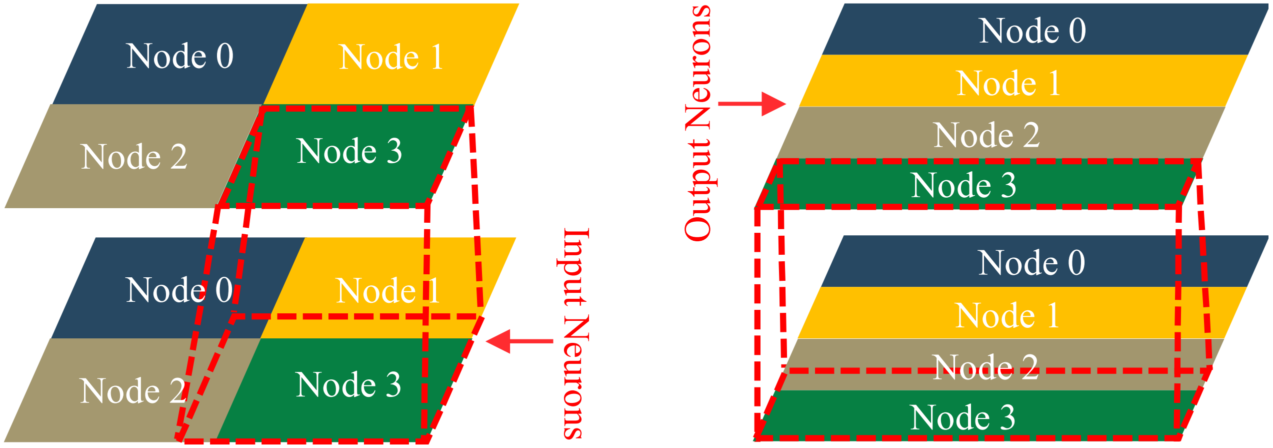
Worker Node

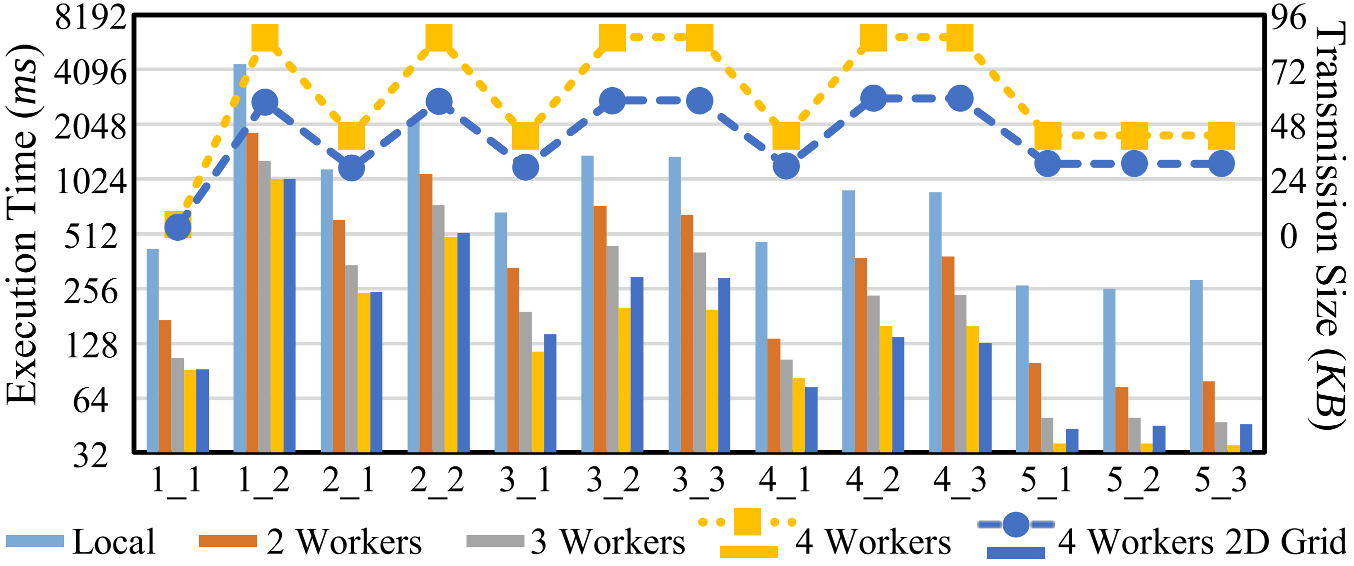
Worker Node

Computing Cluster

Optimization of Convolutional Layers

Biased One-dimensional Partition (BODP)

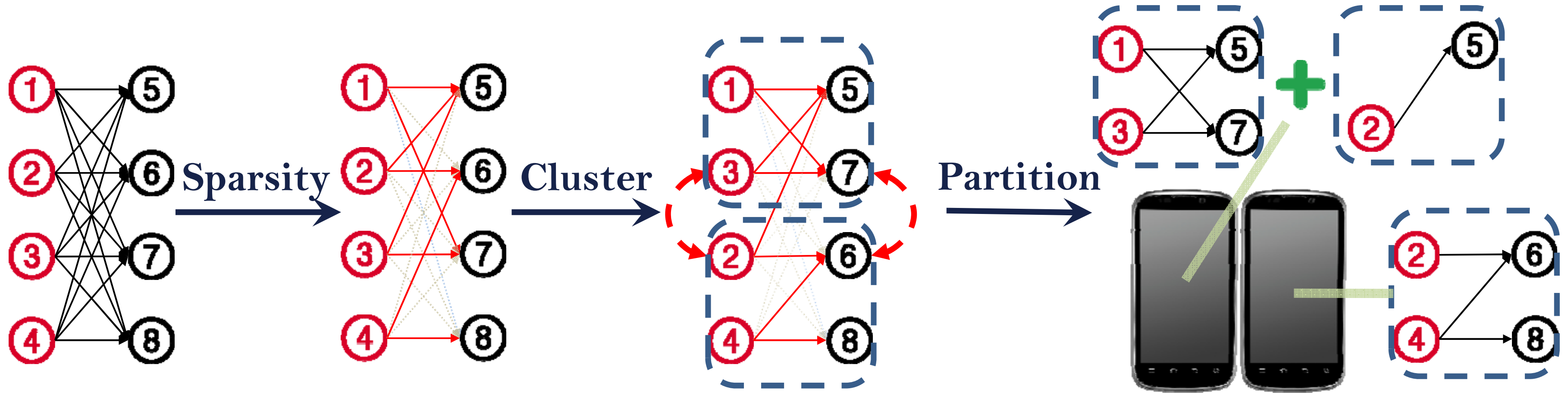




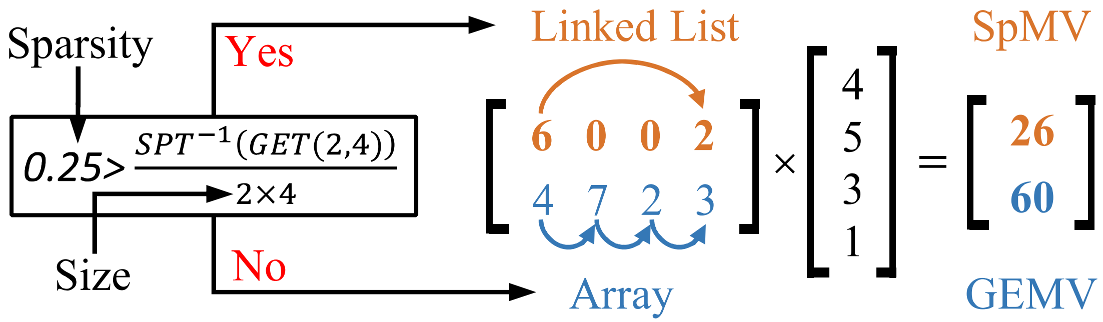
Execution Time Reduction of 16 layers in VGG

Duke UNIVERSITY

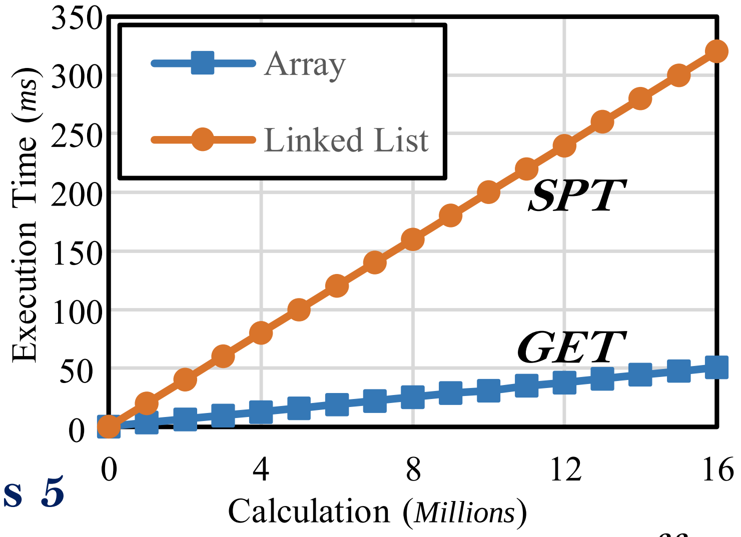
Optimization of Fully-connected Layers



Schematic diagram of the partition scheme for fully-connected layers


Sparsity
Yes
$0.25 > \frac{SPT^{-1}(GET(2,4))}{2\times 4}$
Size
No
Linked List
Array
SpMV
GEMV
$\begin{bmatrix} 6 & 0 & 0 & 2 \\ 4 & 7 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 4 \\ 5 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 26 \\ 60 \end{bmatrix}$

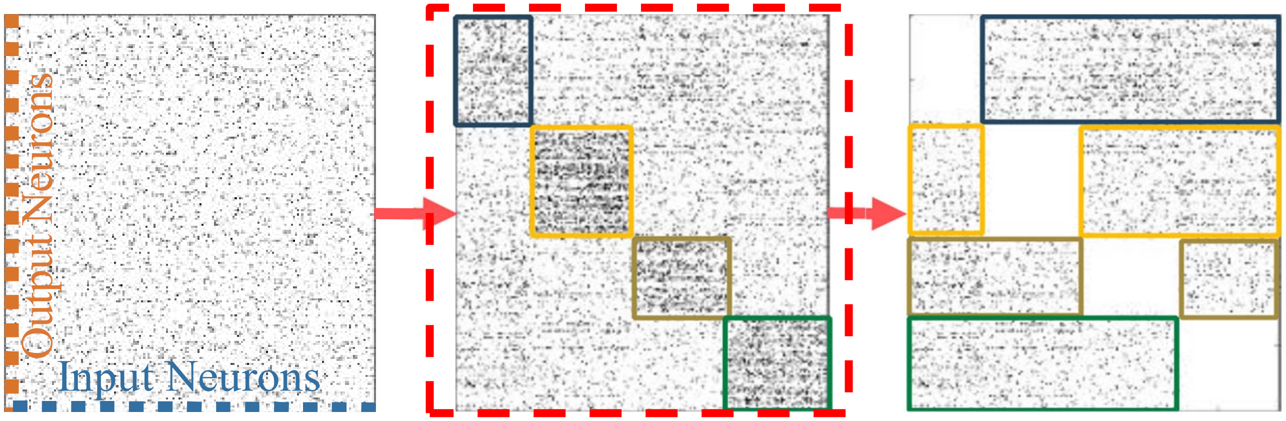
Speed-oriented decision: SPMV and GEMV



Performance characterization of Nexus 5

Optimization of Sparse Fully-connected Layers

Modified Spectral Co-Clustering (MSCC)



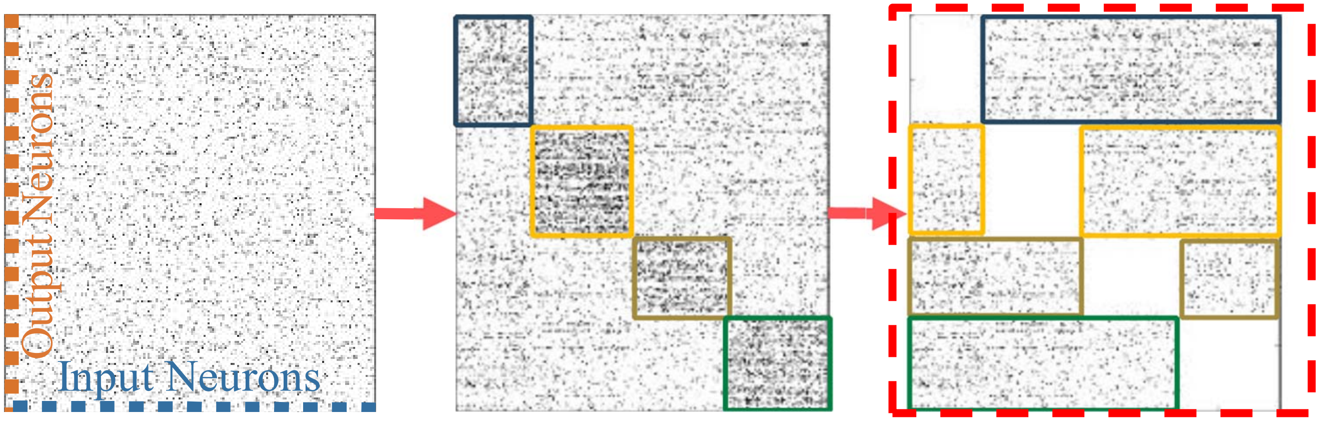
(a) Original (b) Clustered (MSCC) (c) Outliers (FGCP)

Target: Higher Computing Efficiency & Lower Transmission Amount

- Apply spectral co-clustering to original sparse matrix
- Decide the execution method for each cluster
- Initialize the estimated time with the execution time of each cluster and their corresponding outliers.

Optimization of Sparse Fully-connected Layers

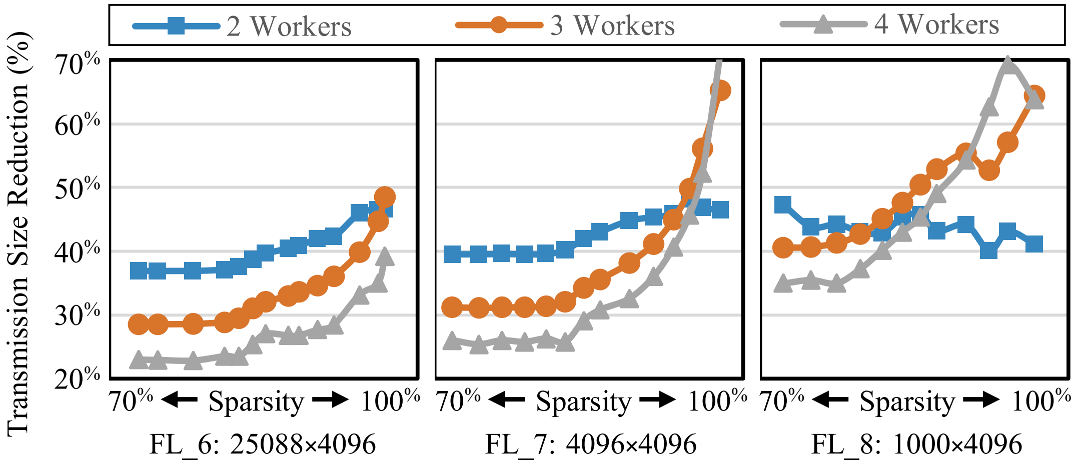
Fine-Grain Cross Partition (FGCP)



(a) Original (b) Clustered (MSCC) (c) Outliers (FGCP)

Target: Workload Balance & Lower Transmission Amount

- Choose the node of highest time
- Find the sparsest line
- Offload it to GO



Transmission size decrease ratio of MoDNN to baseline.

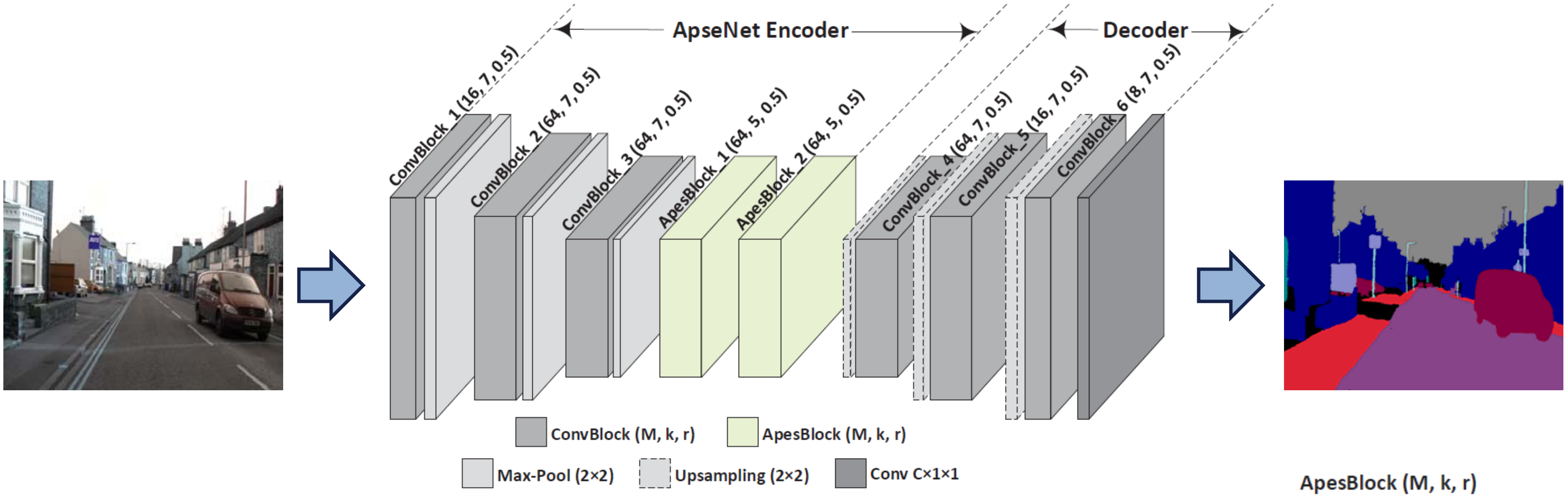
Outline

Our work: Improve Efficiency for DNN Applications Through Software/Hardware Co-Design

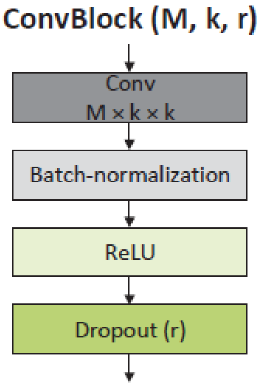
- Introduction
- Research Spotlights
  - Structured Sparsity Regularization
  - Local Distributed Mobile System for DNN
  - ApesNet for Image Segmentation (ESWEEK'16)
- Conclusion

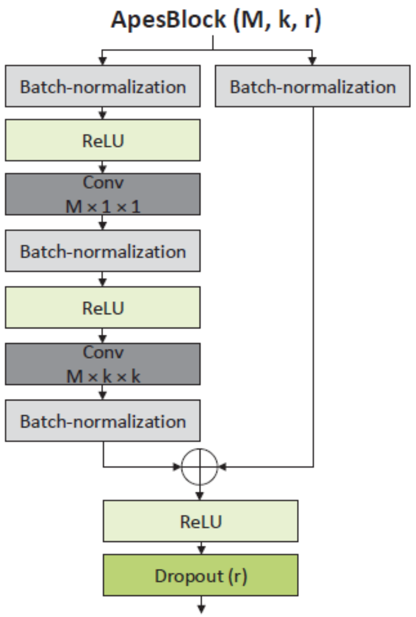
Duke UNIVERSITY

ApesNet Design Concept

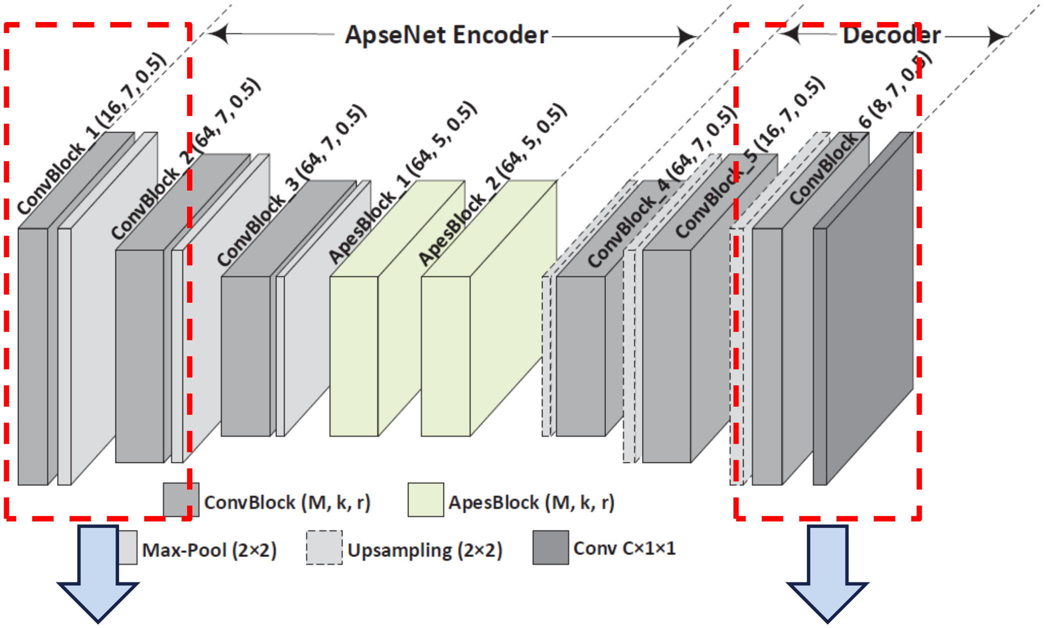


- ConvBlock & ApesBlock
- Asymmetric encoder & decoder
- *Thin* ConvBlock w/ large feature map
- *Thick* ApesBlock w/ small kernel



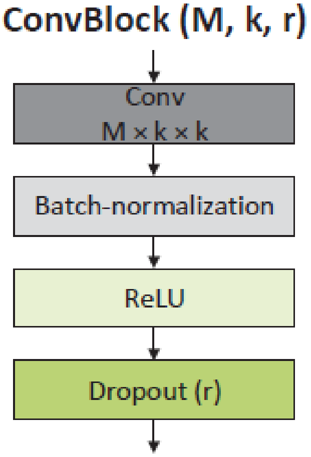


ConvBlock



16 feature maps

8 feature maps



ConvBlock (M, k, r)

Conv M × k × k

Batch-normalization

ReLU

Dropout (r)

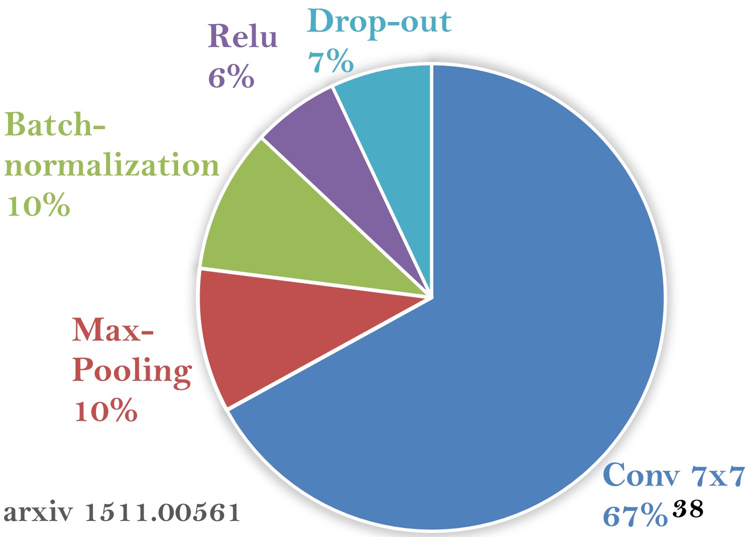
M: # Feature map

k: Kernel size

r: Dropout ratio

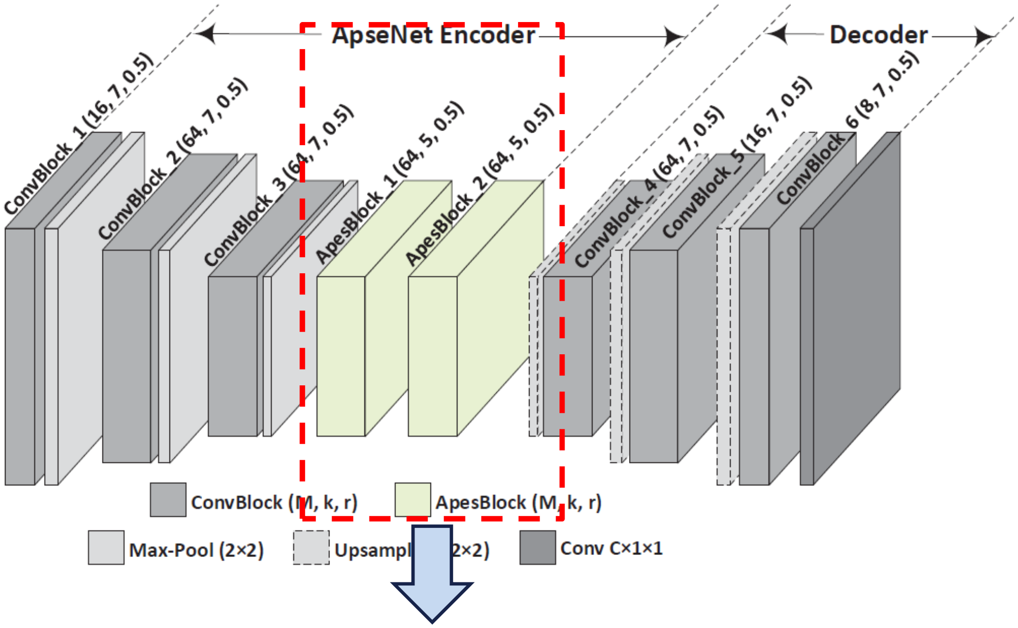
	#Feature map	#Feature map
SegNet	64	64
Deconv-Net	64	64
AlexNet	96	-
VGG16	64	-

Preliminary Result on Running Time



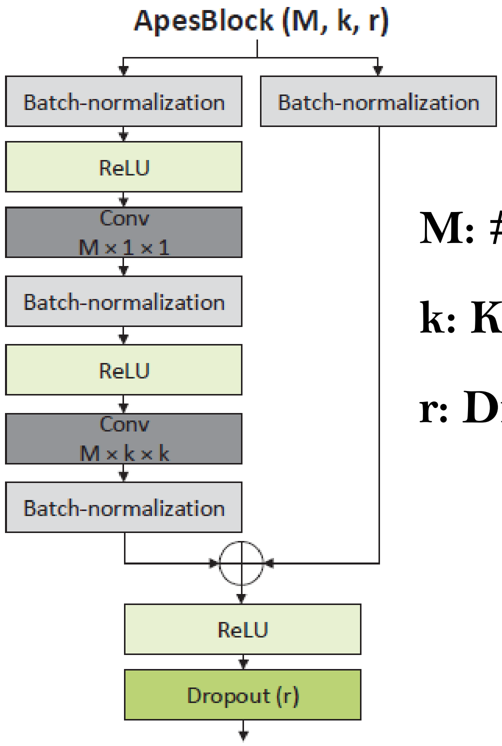
VGG: arxiv 1409.1556; Deconv-Net: arxiv 1505.04366; SegNet: arxiv 1511.00561

ApesBlock



Two ApesBlock: 5x5 kernel, 64 feature maps

	# Parameters
64 Conv 8x8	0.4M
64 Conv 8x8	0.2M
4096 Full-connected	4.2M
Max-Pooling	-

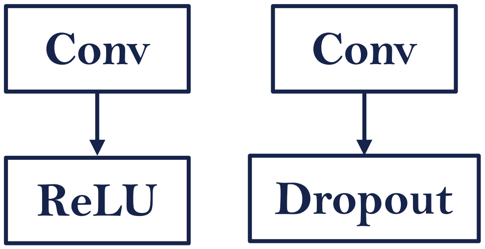


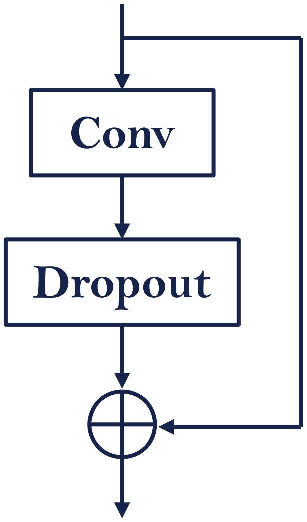
M: # Feature map

k: Kernel size

r: Dropout ratio

Previous models

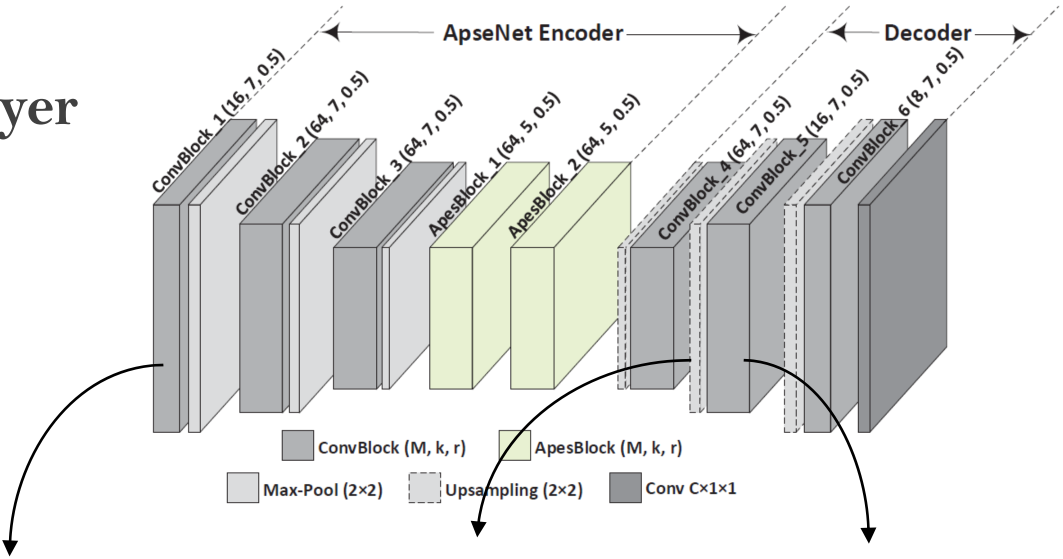




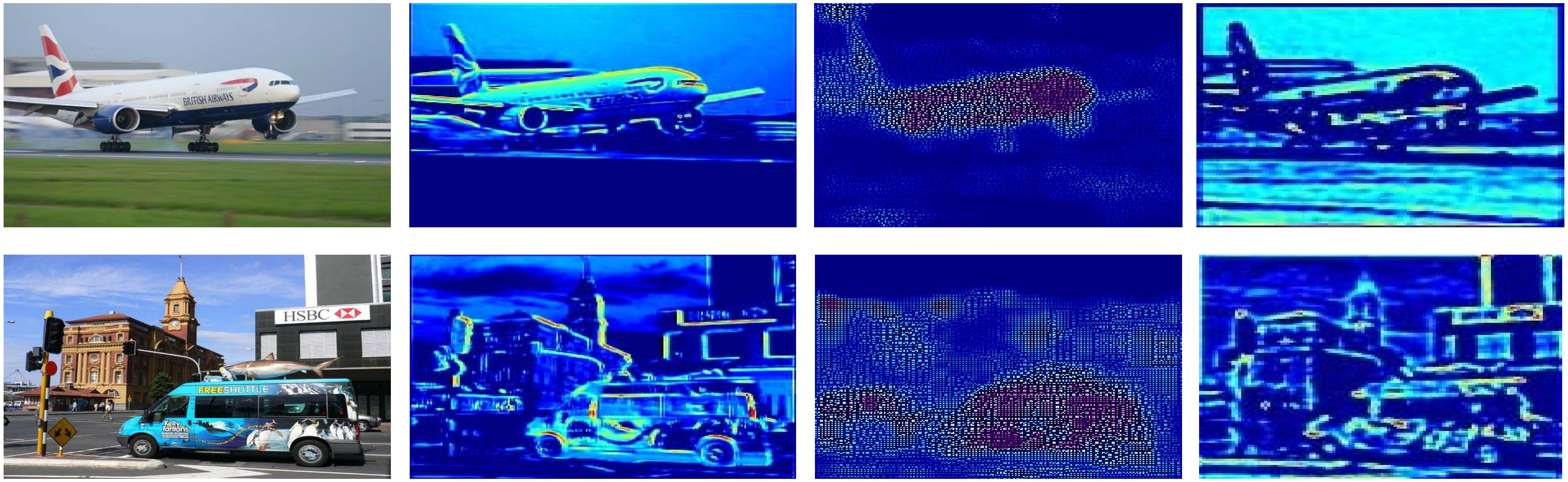
Neuron Visualization of ApesNet

Feature response at each layer (Neuron visualization)

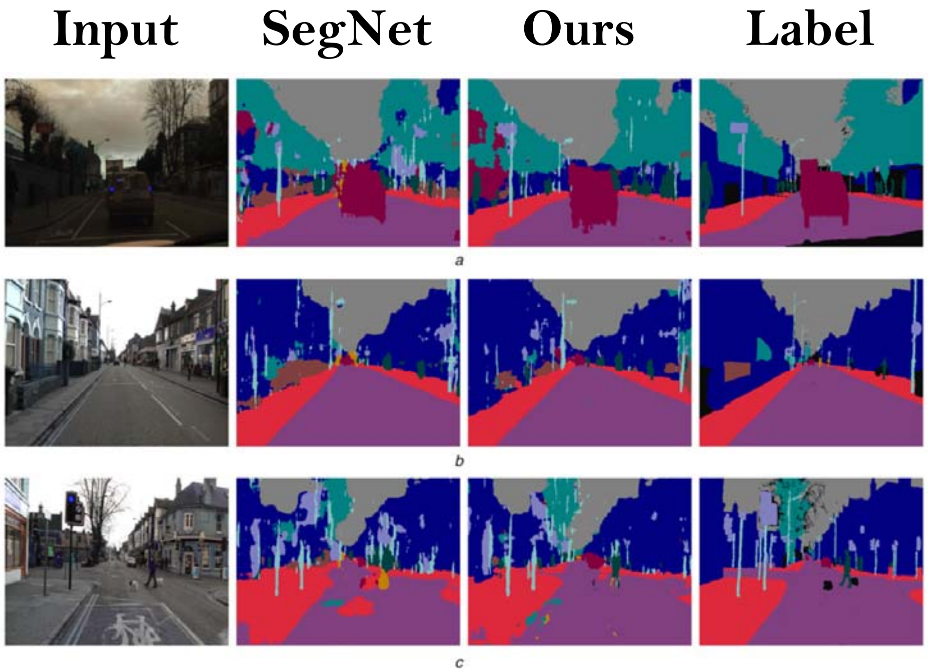
- Encoder: Convolution
- Decoder: Up-sampling
- Decoder: Convolution



Convolution (En) **Up-sampling** **Convolution (De)**



Evaluation and Comparison



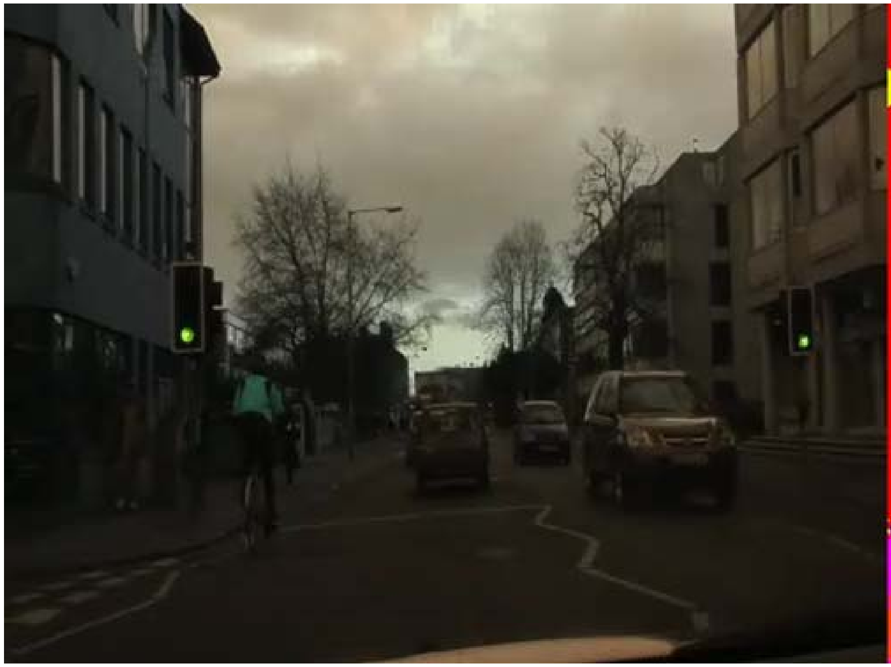
	SegNet-Basic	ApesNet
Model size	5.40MB	3.40MB
GTX 760M	181ms	73ms
GTX 860M	170ms	70ms
TITAN X (Kepler)	63ms	40ms
Tesla K40	58ms	39ms
GTX 1080	48ms	33ms

CamVid Dataset

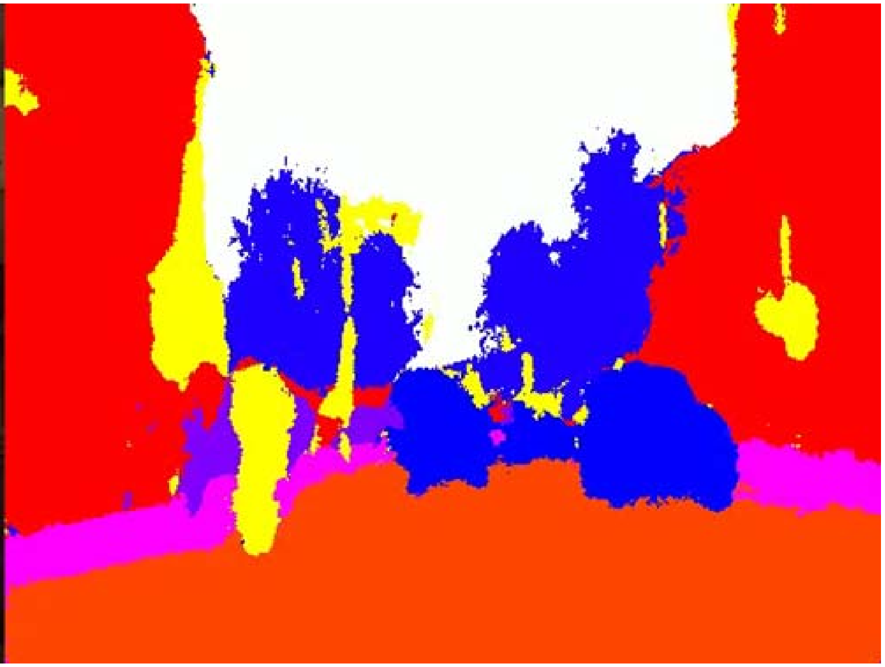
	Class Avg.	Mean IoU	Bike	Tree	Sky	Car	Sign	Road	Pedestrian
SegNet-Basic	62.9%	46.2%	75.1%	**83.1%**	88.3%	80.2%	36.2%	91.4%	56.2%
ApesNet	**69.3%**	**48.0%**	**76.0%**	80.2%	**95.7%**	**84.2%**	**52.3%**	**93.9%**	**59.9%**

ApesNet: An Efficient DNN for Image Segmentation

Input



ApesNet Output



Outline

Our work: Improve Efficiency for DNN Applications Through Software/Hardware Co-Design

- Introduction
- Research Spotlights
  - Structured Sparsity Regularization
  - Local Distributed Mobile System for DNN
  - ApesNet for Image Segmentation
- Conclusion

Conclusion

- DNNs demonstrate great success and potentials in various types of applications;
- Many software optimization techniques cannot obtain the theoretical speedup in real implementations;
- The mismatch between the software requirement and hardware capability is more severe as problem scale increases;
- New approaches that coordinate the software and hardware co-design and optimization are necessary.
- New devices and novel architecture could play an important role too.

Thanks to Our Sponsors, Collaborators, & Students

NSF

DARPA

AIR FORCE RESEARCH LABORATORY

U.S. DEPARTMENT OF ENERGY

QUALCOMM®

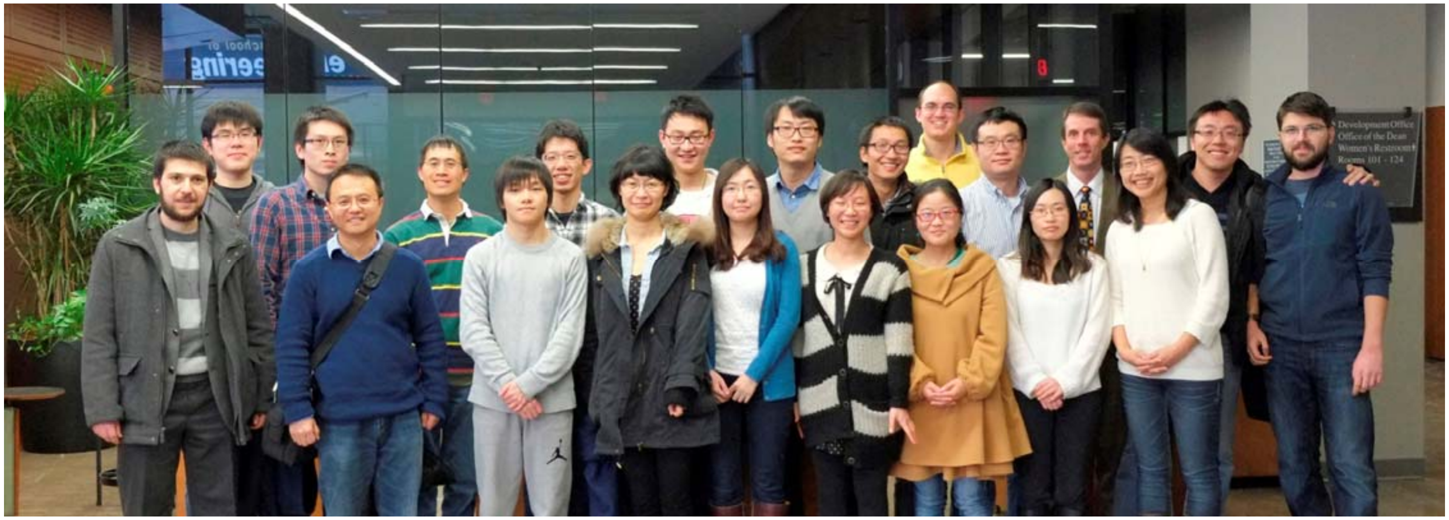
LG

CISCO

LABS hp

intel®

SAMSUNG



We welcome industrial collaborators.