GitHub

ICML | 2018

Deep *k*-Means:

Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

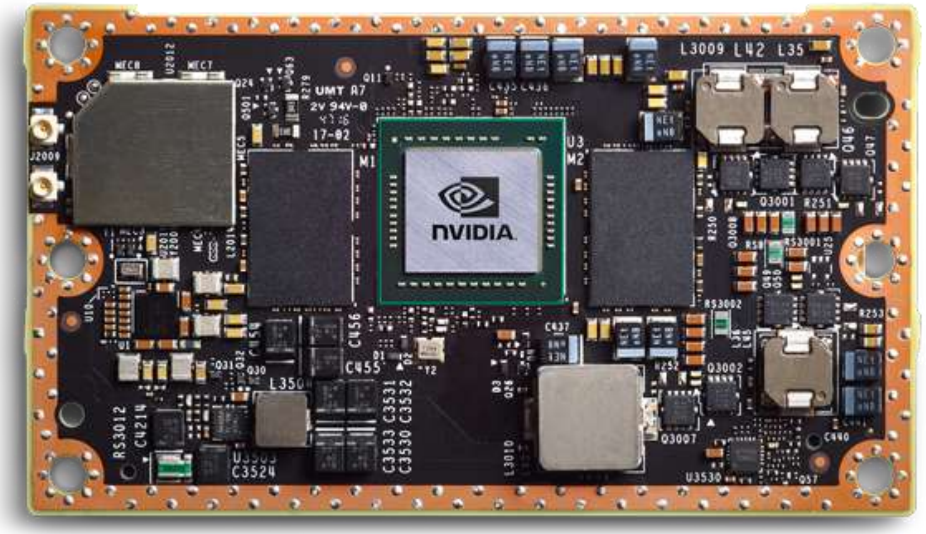
Junru Wu[1] Yue Wang[2] Zhenyu Wu[1] Zhangyang Wang[1]
Ashok Veeraraghavan[2] Yingyan Lin[2]

COMPUTER SCIENCE & ENGINEERING
TEXAS A&M UNIVERSITY

RICE
Electrical and Computer Engineering

Motivation

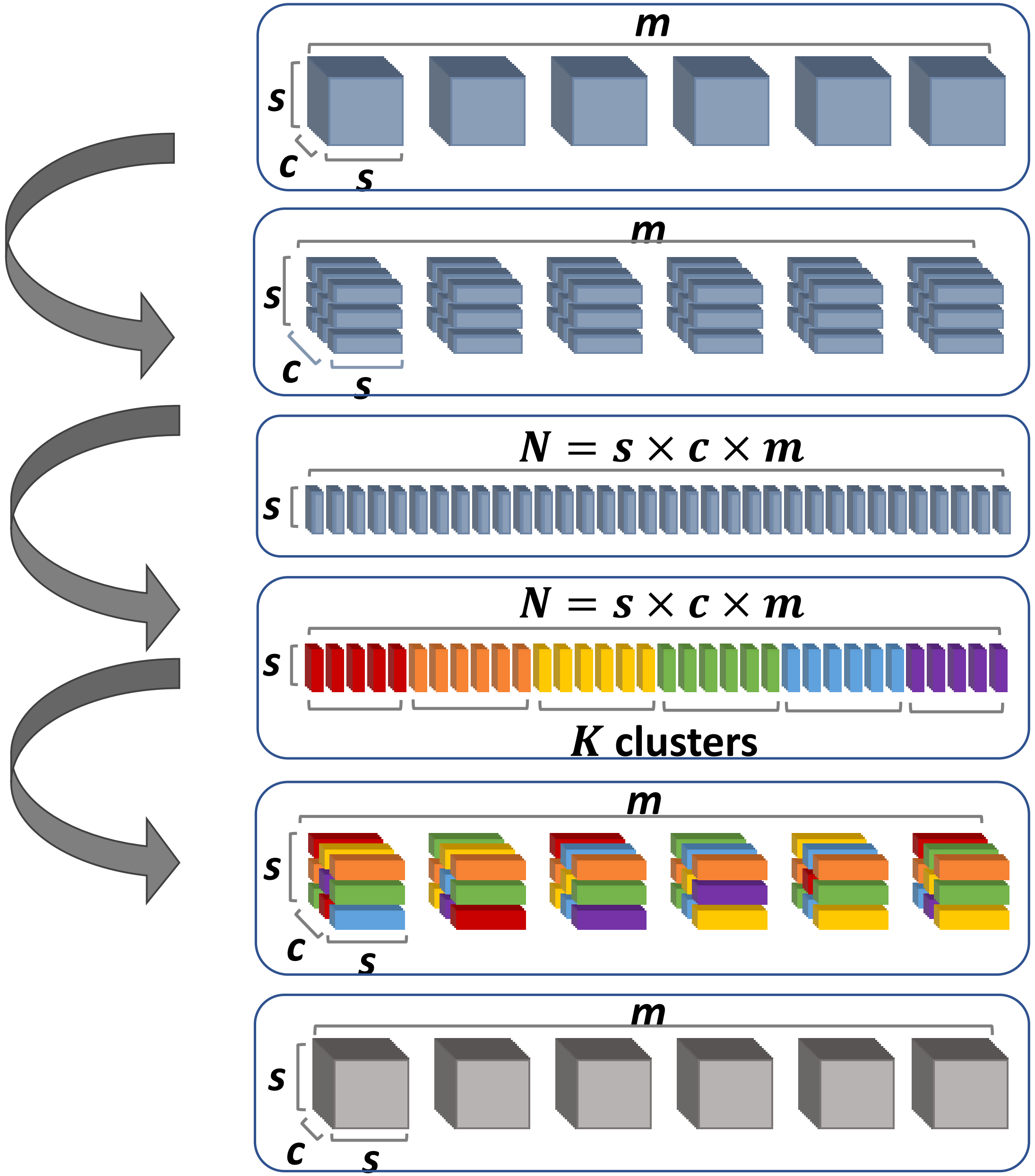
- Deploying CNNs on resource-constrained platforms
- Two important concerns: **Model Size + Energy Efficiency**
- They are often not aligned*, so need to **consider both** in implementation



* Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks, IEEE ISSCC 2016

Parameter Sharing via Row-wise k-Means

1. Reshape into $W \in \mathbb{R}^{s \times N}$ $(N = s \times c \times m)$
2. Apply k-Means on W, cluster N samples into K clusters
3. Reshape back into $W \in m \times \mathbb{R}^{s \times c \times s}$



$\boldsymbol{W} \in \boldsymbol{m} \times \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{c} \times \boldsymbol{s}}$

$\boldsymbol{W} \in \boldsymbol{m} \times \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{c} \times \boldsymbol{s}}$

$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{N}}$

$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{N}}$

$\boldsymbol{W} \in \boldsymbol{m} \times \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{c} \times \boldsymbol{s}}$

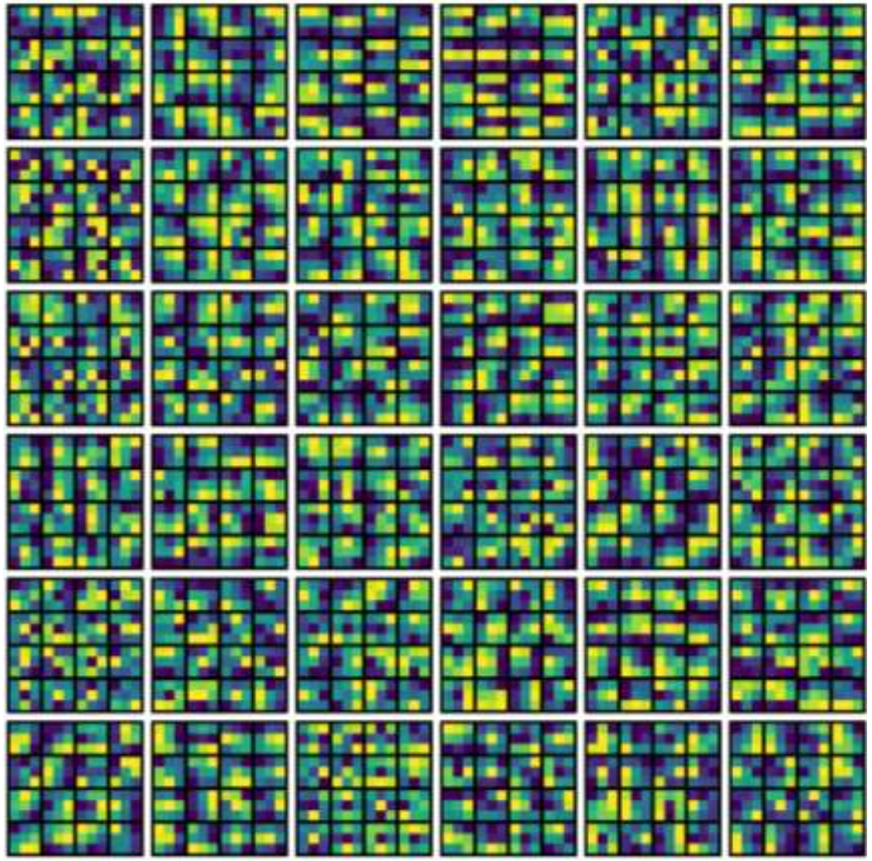
$\boldsymbol{W} \in \boldsymbol{m} \times \mathbb{R}^{\boldsymbol{s} \times \boldsymbol{c} \times \boldsymbol{s}}$

Parameter Sharing via Row-wise k-Means

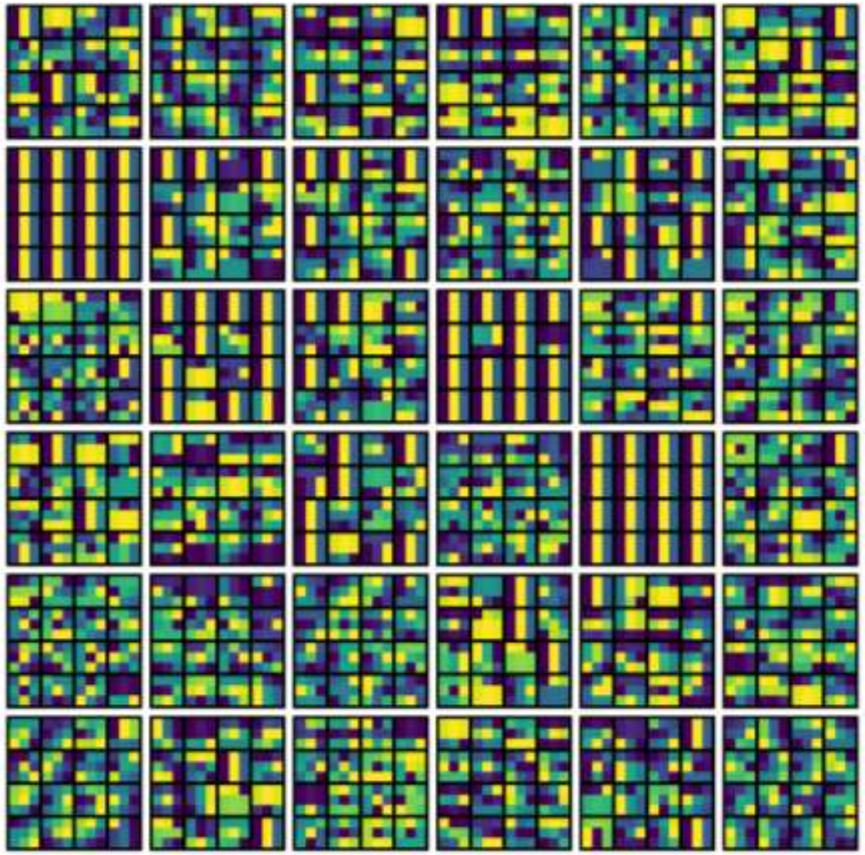
- For a conv layer with m filters each of size $s \times s \times c$
- Original Memory Consumption can be represented as:
  - $MEM_{org} = \underbrace{s \times s \times c \times m}_{\text{Weights}} + \underbrace{m}_{\text{Bias}}$
- Applying K-Means* to assign weights with K clusters, the memory consumption is reduced to:
  - $MEM_{comp} = \underbrace{K \times s}_{\text{Weights}} + \underbrace{\left(-\sum_{i=1}^{N} p_i \log_2 p_i\right)}_{\text{Weight Assignment Indexes}} + \underbrace{m}_{\text{Bias}}$
  - p_i: occurrence probability of samples in the $i\ th$ cluster.

* Compressing deep convolutional networks using vector quantization, ICLR 2015

Filter Visualization on Wide ResNet



Pre-Trained Model



Compressed Model w/o Re-Training

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

Deep k-Means w/o Re-Training

Wide ResNet (top-1)

Model	Δ (%)	CR
Soft Weight-Sharing	-2.02	45
Deep k-Means WR	-16.02	45
Deep k-Means WR	-25.45	47
Deep k-Means WR	-45.08	50

GoogleNet (top-1 + top-5)

Model	$\Delta^\dagger$ %	$\Delta^\ddagger$ %	CR
One-shot (Kim et al., 2015)	N/A	-0.24	1.28
Low-rank (Tai et al., 2015)	N/A	-0.42	2.84
Deep k-Means WR	-1.22	-0.65	1.5
Deep k-Means WR	-3.7	-2.46	2
Deep k-Means WR	-13.72	-10.05	3
Deep k-Means WR	-48.95	-48.82	4

- CR: Compression Ratio, same as defined in (Han et. al., 2015)
- Considerable Performance Drop!
- Design a re-training process that is **more "suitable"** for k-means?

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

k-Means Regularized Re-Training

- Spectrally Relaxation* of k-means ($W \in \mathbb{R}^{s \times N}$ denotes the sample matrix):
  - 1. Rewrite k-means objective: $\min_{W;F\in\mathcal{F}} Tr(W^T W) - Tr(F^T W^T W F),$
    ($F \in \mathbb{R}^{N \times k}$: cluster index matrix with special structure)
  - 2. Since *W* as given: $\max_{F\in\mathcal{F}} Tr(F^T W^T W F)$
  - 3. Relax the structure of *F*: $\max_{F} Tr(F^T W^T W F),\ s.t.\ F^T F = I$

* H Zha, X He, C Ding, M Gu, HD Simon "Spectral relaxation for k-means clustering", NIPS 2001

k-Means Regularized Re-Training

- Spectrally Relaxation of k-means ($W \in \mathbb{R}^{s \times N}$ denotes the sample matrix):
  - 1. Rewrite k-means objective: $\min_{W;F\in\mathcal{F}} Tr(W^TW) - Tr(F^TW^TWF),$
    ($F \in \mathbb{R}^{N \times k}$: cluster index matrix with special structure)
  - 2. Since *W* as given: $\max_{F\in\mathcal{F}} Tr(F^TW^TWF)$
  - **No longer true for W as a variable during re-training!**
  - 3. Relax the structure of *F*: $\max_{F} Tr(F^TW^TWF), \; s.t. \; F^TF = I$

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

k-Means Regularized Re-Training

- Use k-means spectrally relaxation to design a new regularizer, that keeps weights W "suitable" for k-means clustering

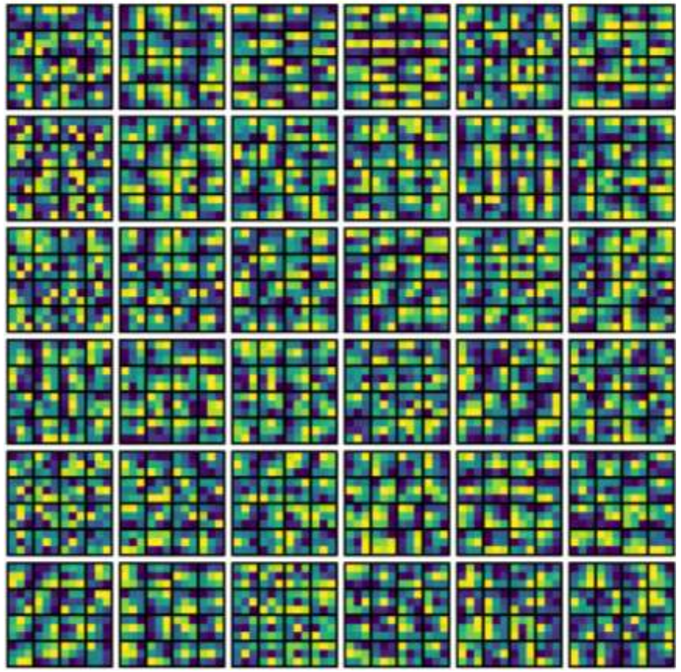
- Assume the original training objective: $\mathrm{E}(W)$
- The new regularized re-training objective:

$$\min_{W,F} E(W) + \frac{\lambda}{2}[Tr(W^T W) - Tr(F^T W^T W F)],$$
$$s.t.\ F^T F = I$$

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

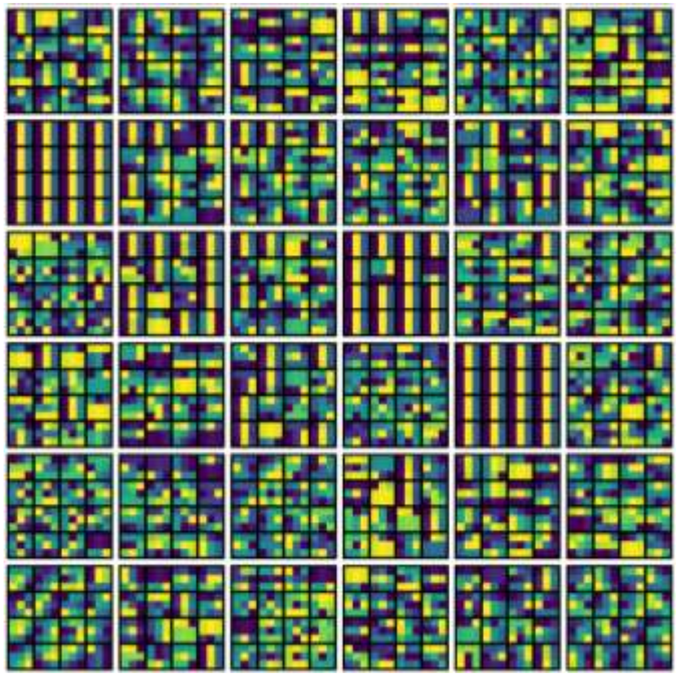
Filter Visualization on Wide ResNet

MMSE: 1.5e-08



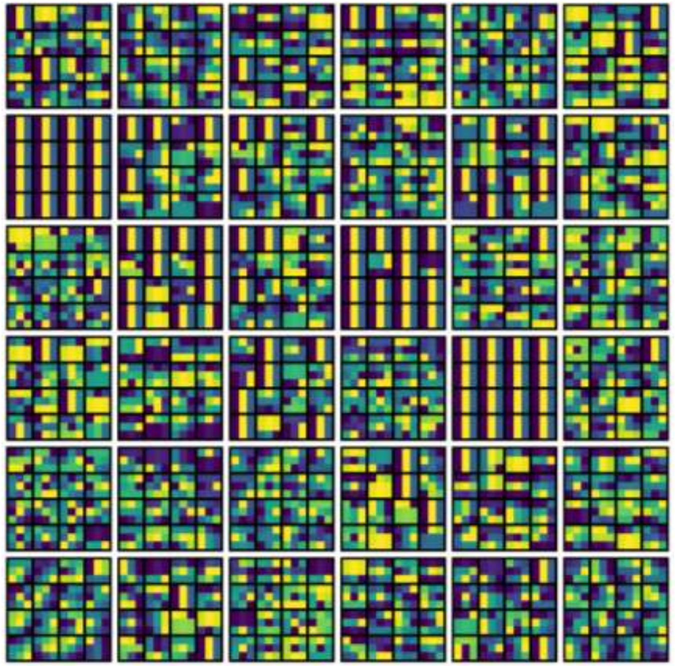
Accuracy: 94.69%

Pre-Trained Model



Accuracy: 92.89%

Pre-Trained Model w/ Re-Training



Accuracy: 93.06%

Compressed Model w/ Re-Training

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

Deep k-Means w/ Re-Training

Wide ResNet

Model	Δ (%)	CR
Soft Weight-Sharing	-2.02	45
Deep k-Means WR	-16.02	45
Deep k-Means WR	-25.45	47
Deep k-Means WR	-45.08	50
Deep k-Means	-1.63	45
Deep k-Means	-2.23	47
Deep k-Means	-4.49	50

Table 3. Compressing Wide ResNet in comparison to soft weight-sharing (Ullrich et al., 2017).

GoogLeNet

Model	$\Delta^{\dagger}$ %	$\Delta^{\ddagger}$ %	CR
One-shot (Kim et al., 2015)	N/A	-0.24	1.28
Low-rank (Tai et al., 2015)	N/A	-0.42	2.84
Deep k-Means WR	-1.22	-0.65	1.5
Deep k-Means WR	-3.7	-2.46	2
Deep k-Means WR	-13.72	-10.05	3
Deep k-Means WR	-48.95	-48.82	4
Deep k-Means	-0.26	0.00	1.5
Deep k-Means	-0.17	+0.06	2
Deep k-Means	-0.36	+0.03	3
Deep k-Means	-1.95	-1.14	4

Table 4. Compressing GoogLeNet on ILSVRC12 († and ‡ are top-1 and top-5 accuracies respectively).

- Minimum Performance Drop!

Wu et al. Deep k-Means: Re-Training and Parameter Sharing with Harder Cluster Assignments for Compressing Deep Convolutions

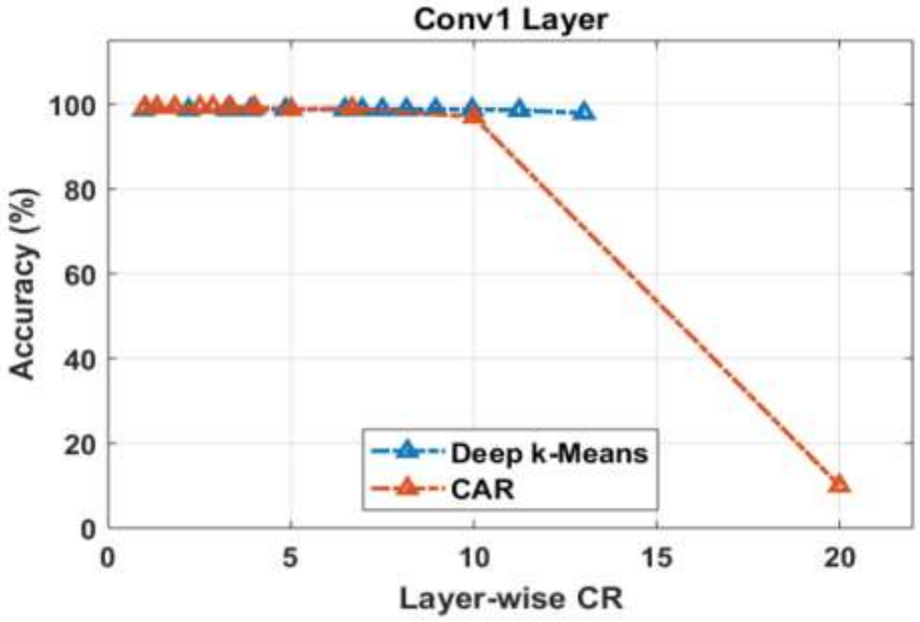
More Experiments on CR

Model	Δ (%)	CR
TT-conv (naive)	-2.4	2.02
TT-conv (naive)	-3.1	2.90
TT-conv	-0.8	2.02
TT-conv	-1.5	2.53
TT-conv	-1.4	3.23
TT-conv	-2.0	4.02
Deep k-Means	+0.05	2
Deep k-Means	-0.04	4

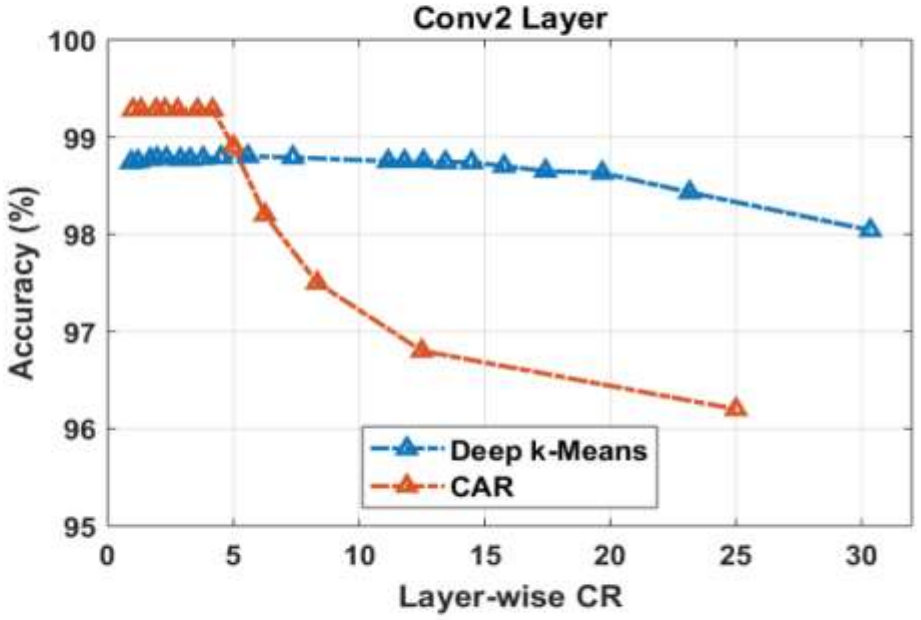
Table 1. Compressing TT-conv-CNN in (Garipov et al., 2016).

Model	Δ (%)	CR
LRD	-8.32	16
HashedNet	-9.79	16
FreshNet	-6.51	16
Deep k-Means WR	-5.95	16
Deep k-Means	-1.30	16

Table 2. Compressing FreshNet-CNN in (Chen et al., 2016a).



(a) Comparison in the first convolutional layer



(b) Comparison in the second convolutional layer

Computational cost *

- Measure the computational resources needed to generate a single decision (1 bit full adders)

$$DB_wB_x + (D-1)(B_x + B_w + [\log_2 D] - 1)$$

- B_w: weight precision
- B_x: activation precision
- D is the dimensional of dot product.

*Charbel Sakr, Yongjune Kim, Naresh R. Shanbhag, "Analytical Guarantees on Numerical Precision of Deep Neural Networks" ICML, 2017

Weight/ Activation Representational Cost *

- Measure the storage complexity and communication costs associated with data movement

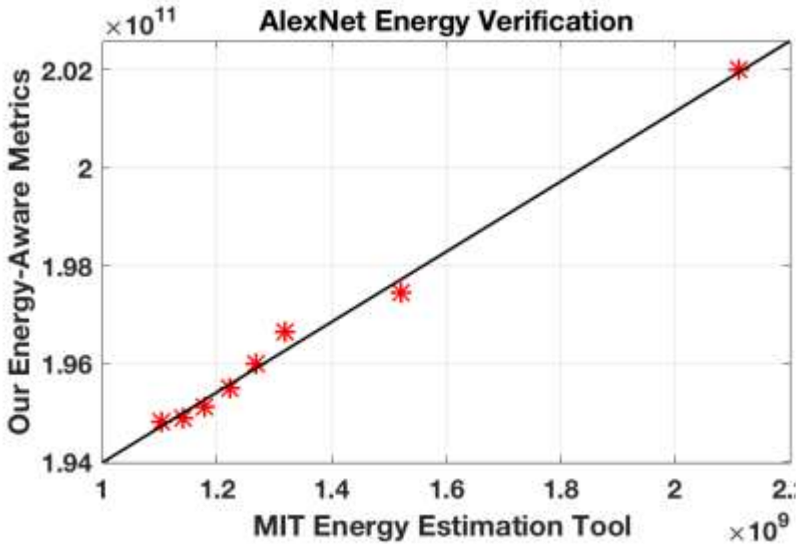
$$N_w|W|B_w + N_x|\chi|B_x$$

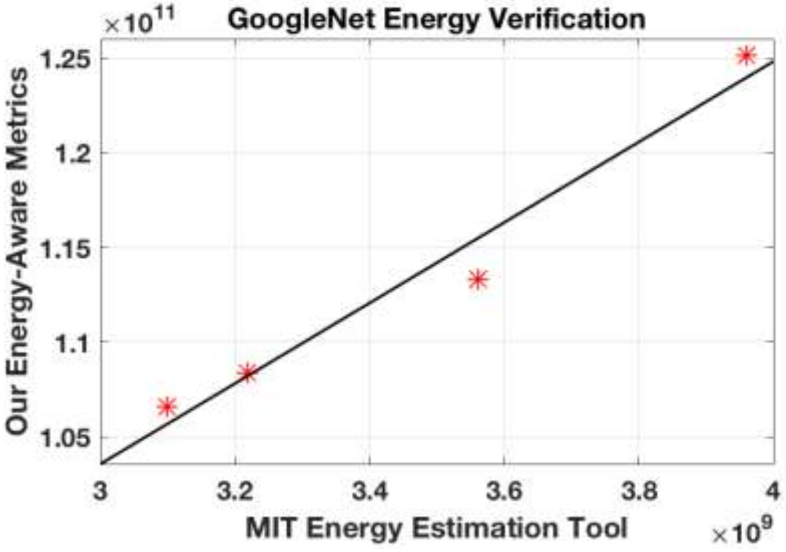
- $N_w,\ N_x$: total number of times weight/ activation is used for convolution
- $|W|$: index sets of weights $|\chi|$: index sets of activation
- B_w: weight precision B_x: activation precision

*Charbel Sakr, Yongjune Kim, Naresh R. Shanbhag, "Analytical Guarantees on Numerical Precision of Deep Neural Networks" ICML, 2017

Verification of Energy-Aware Metrics

- We verify our Energy-Aware Metrics with MIT energy estimation* tool whose results are extrapolated **from actual hardware measurements.**





R^2 Coefficient:

- AlexNet: 0.9931
- GoogLeNet_v1: 0.9675

Highly aligned!

*T.-J. Yang, Y.-H. Chen, V. Sze, "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," CVPR, 2017

Computational Resources Used in Project

High Performance Research Computing

A Resource for Research and Discovery

TEXAS A&M UNIVERSITY.

- Hardware Stack
  - Texas A&M HRPC **Terra GPU Cluster**
    - Intel Xeon E5-2680 v4 2.40GHz 14-core
    - NVIDIA Tesla K80 Accelerator
- Software Stack:
  - CUDA 8.0
  - PyTorch 0.3.1