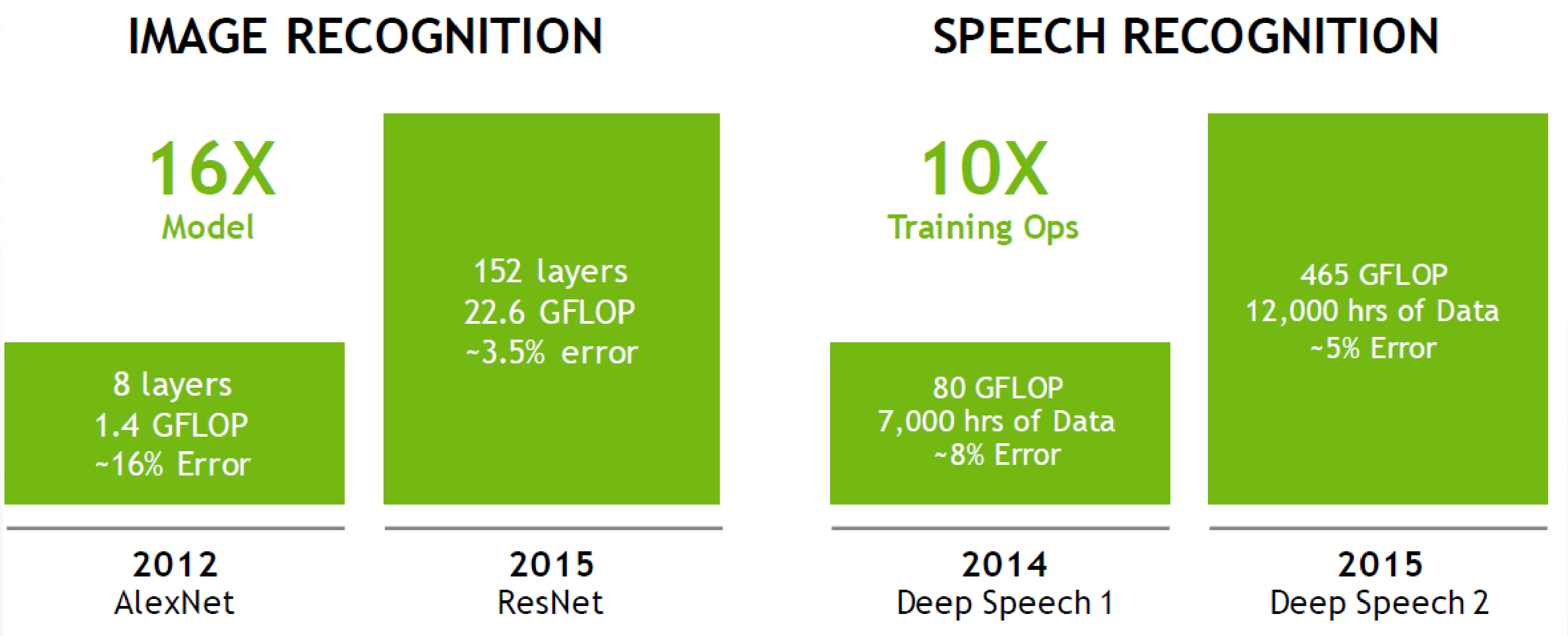
Evaluating the Efficiency of Iterative Network Pruning on ResNet

Hong Tran

1. Introduction

- Many popular deep neural networks models have emerged in the recent years, and it achieves very good state-of-the-art performance.
- The models are gettering larger and larger.



1. Introduction

- The deep neural networks models have following challenges:
  1. **Model Size**
  2. **Training/Inferencing Speed**
  3. **Energy Efficiency**

➢ Energy Consumed: larger model => more memory reference => more energy.

➢ This is a dilemma between a model with good accuracy vs. a lightweight model for edge devices.

	Error rate	Training time
ResNet18:	10.76%	2.5 days
ResNet50:	7.02%	5 days
ResNet101:	6.21%	1 week
ResNet152:	6.16%	1.5 weeks

Model Training Time

Model	MACC	COMP	ADD	DIV	Activations	Params	SIZE(MB)
SimpleNet	1.9G	1.82M	1.5M	1.5M	6.38M	6.4M	24.4
SqueezeNet	861.34M	9.67M	226K	1.51M	12.58M	1.25M	4.7
Inception v4*	12.27G	21.87M	53.42M	15.09M	72.56M	42.71M	163
Inception v3*	5.72G	16.53M	25.94M	8.97M	41.33M	23.83M	91
Incep-Resv2*	13.18G	31.57M	38.81M	25.06M	117.8M	55.97M	214
ResNet-152	11.3G	22.33M	35.27M	22.03M	100.11M	60.19M	230
ResNet-50	3.87G	10.89M	16.21M	10.59M	46.72M	25.56M	97.70
AlexNet	7.27G	17.69M	4.78M	9.55M	20.81M	60.97M	217.00
GoogleNet	16.04G	161.07M	8.83M	16.64M	102.19M	7M	40
NIN	11.06G	28.93M	380K	20K	38.79M	7.6M	29
VGG16	154.7G	196.85M	10K	10K	288.03M	138.36M	512.2

*Inception v3, v4 did not have any Caffe model, so we reported their size related information from MXNet and Tensorflow respectively. Inception-ResNet-V2 would take 60 days of training with 2 Titan X to achieve the reported accuracy. Statistics are obtained using `http://dgschwend.github.io/netscope`

A.2 GENERALIZATION SAMPLES

Model Size and Complexity

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**

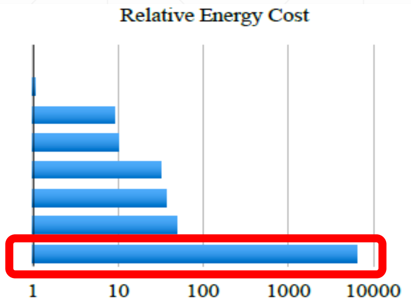


Figure 1: Energy table for 45nm CMOS process [7]. Memory access is 3 orders of magnitude more energy expensive than simple arithmetic.

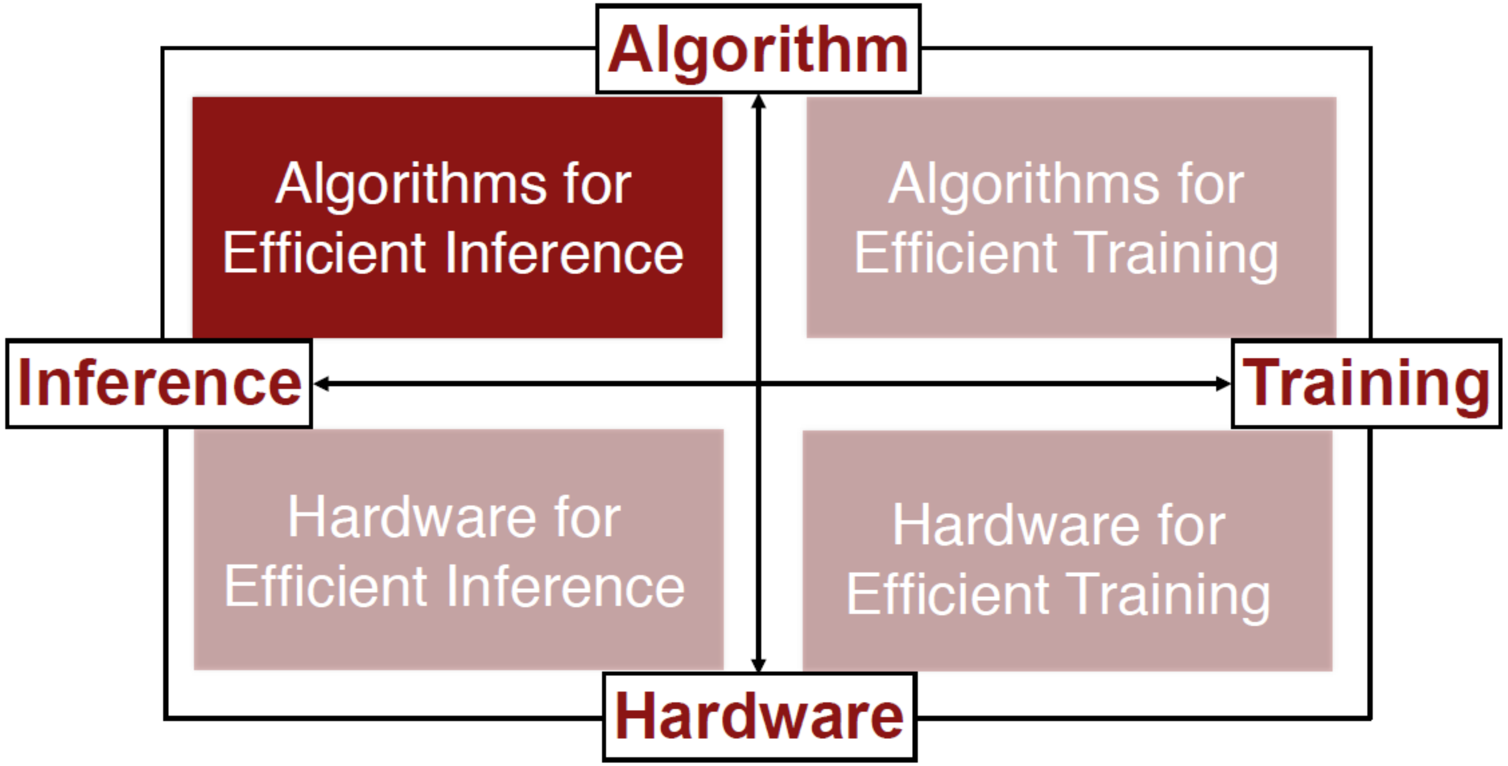
Energy Consumption

1. Introduction

❑ To tackle these challenges, we can analyze an efficient network from both **Algorithm** and **Hardware** viewpoint.

❑ In this project, we only focus on the **Algorithms** for Efficient Inference:

1. **Pruning**
2. **Weight Sharing**
3. Vector Quantization
   a. **k-Means**
   b. Product quantization
   c. Residual quantization
4. Matrix Factorization
5. Low Rank Approximation
6. Binary / Ternary Net



1. Introduction

- To improve the network inference efficiency, the papers from Song Han et al. (2016 [1], 2015 [2]) show that the iterative network pruning method achieves very good result.
- This work demonstrates very notable compression ratio on the Alexnet, LeNet and VGG-16 models.
- However, these models are still shallow comparing to the most recent deeper learning models (for example: ResNet).

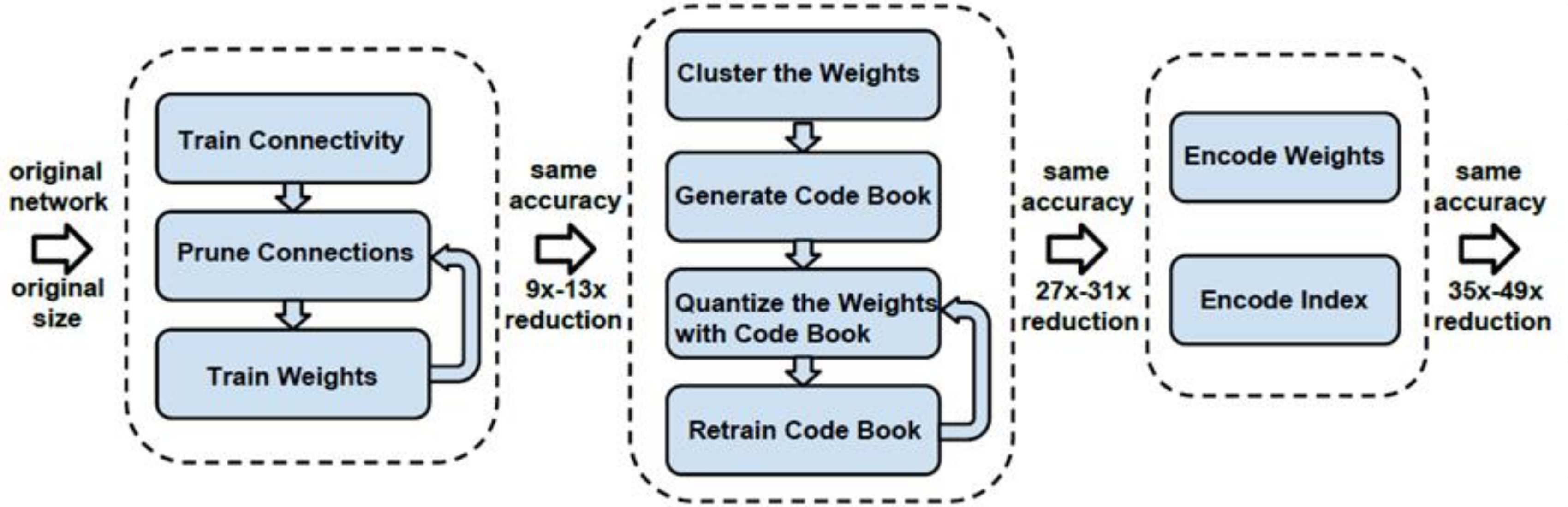
[1] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2016. This is ICLR'16 best paper award presentation.

[2] Han, Song & Pool, Jeff & Tran, John & Dally, William. (2015). Learning both Weights and Connections for Efficient Neural Networks.

2. Project Scope

- This final project targets to:
  - Understand the concept of the Deep Compression proposal in Song Han 2016 paper [2].
  - Implement this concept on Pytorch
    - The paper's author provide a reference Caffe code for simple Alexnet net, but it is not straight forward to understand
    - There is one pytorch reference code in GitHub, but it does not have the weight pruning for convolution layer, Huffman decode layer and also several bugs.
  - Evaluate this implementation on a deep neural networks model (ResNet)
  - Compare this result with the most recent model compression papers (2019).
    - [3] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang and W. Liu, "*Compressing Convolutional Neural Networks via Factorized Convolutional Filters*," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 3972-3981, doi: 10.1109/CVPR.2019.00410.
    - [4] Ma, Xiaolong & Yuan, Geng & Lin, Sheng & Li, Zhengang & Sun, Hao & Wang, Yetang. (2019). *ResNet Can Be Pruned 60x: Introducing Network Purification and Unused Path Removal (P-RM) after Weight Pruning*.
- Dataset will be used: CIFAR-10: This dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

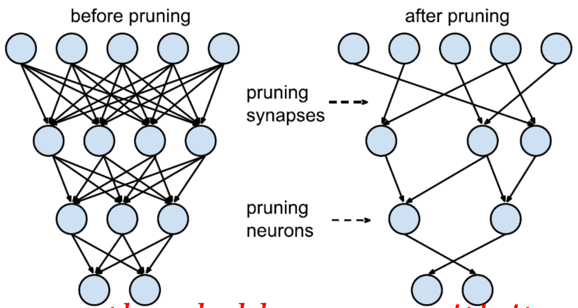
3. Deep Compression



1. Pruning: less number of weights

2. Quantization: less bits per weight

3. Huffman Encoding



$$threshold = \sigma * sensitivity$$

Use k-means clustering to identify the shared weights for each layer of a trained network

Centroid initialization:

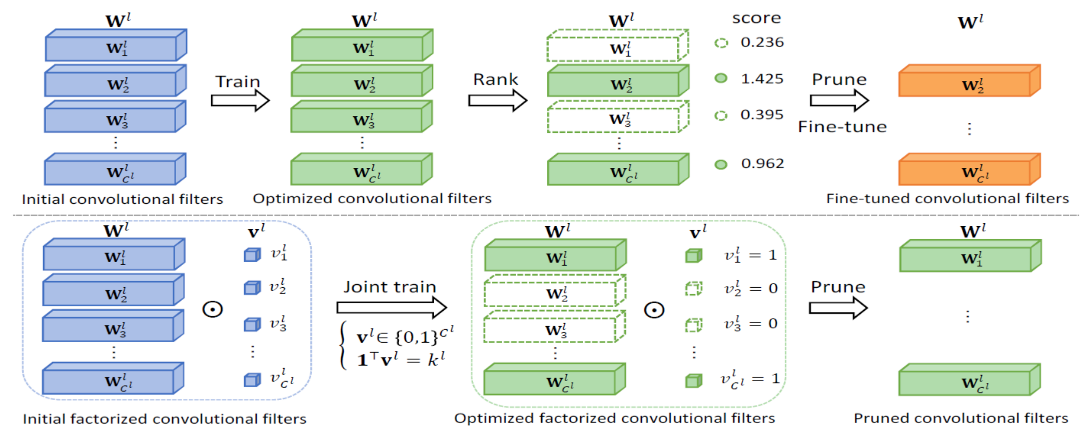
- Forgy (random) initialization
- Density-based initialization
- Linear initialization

* Convert the dense weight matrix to compressed sparse row (CSR) or compressed sparse column (CSC)
* Use Huffman encoding/decoding to shorten the weight representation

3. Most Recent Model Compression Papers

❑Compressing Convolutional Neural Networks via Factorized Convolutional Filters in 2019 [3]

- The workflow of a traditional pruning consists of three sequential stages:
  - Pre-training the original model,
  - Selecting the pre-trained filters via ranking according to a **manually** designed criterion (e.g., the norm of filters), and learning the remained filters via fine-tuning
- In this paper, the authors a factorized convolutional filters (CNN-FCF) that can conduct **filter selection and filter learning simultaneously**, in a unified model.
- The CNN-FCF will update:
  - The standard filter (W) using back-propagation,
  - The binary scalar (V) using the alternating direction method of multipliers (Alternating direction method of multipliers – ADMM) based optimization method.



3. Most Recent Model Compression Papers

- ResNet Can Be Pruned 60x: Introducing Network Purification and Unused Path Removal (P-RM) after Weight Pruning in 2019 [4]

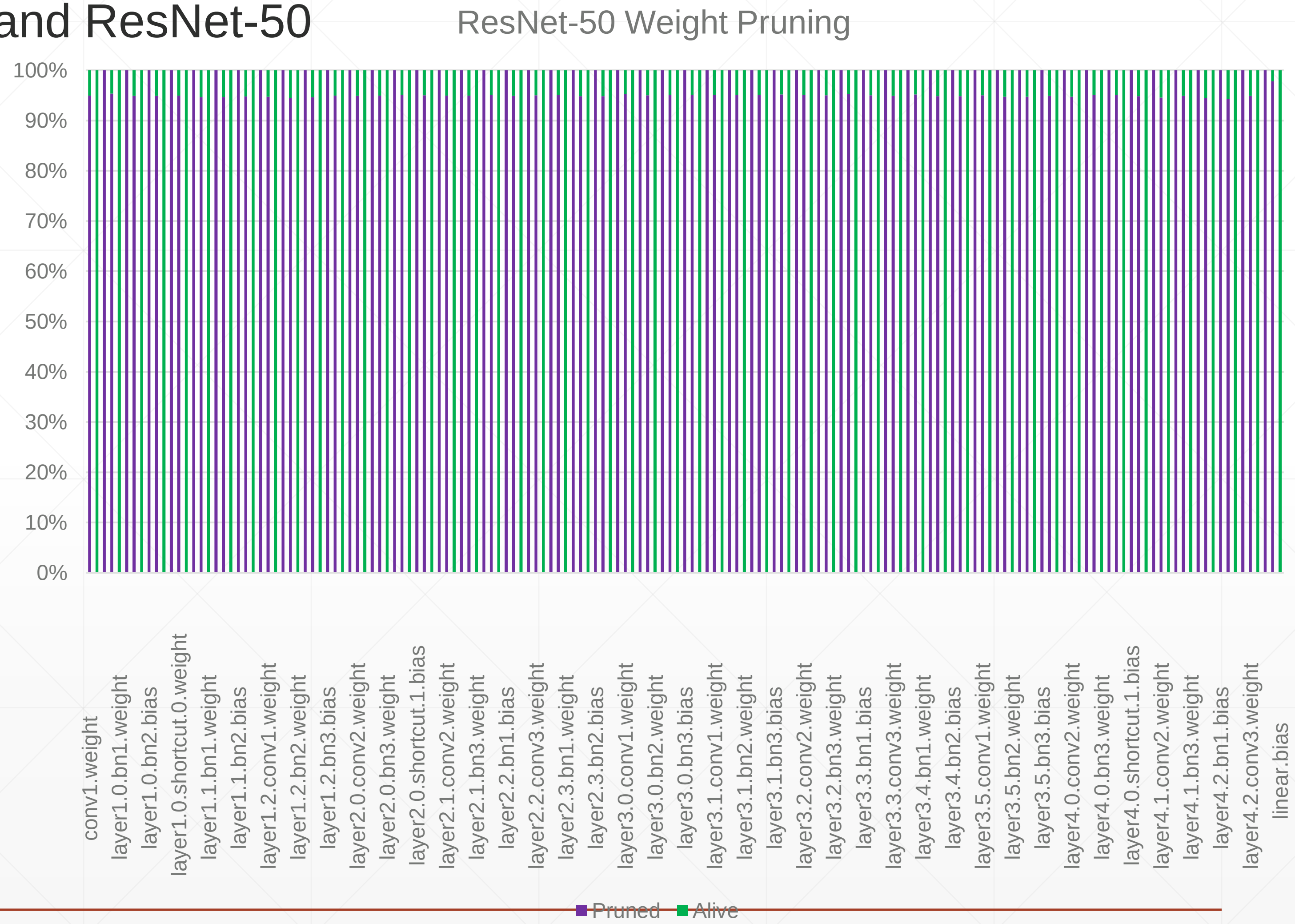
- This paper propose a framework which combines:
  - Structured weight pruning (filter and column prune)
  - Alternating direction method of multipliers (ADMM) algorithm for better prune performance.

4. Experimental Result

➢ Deep Compression on ResNet-18 and ResNet-50

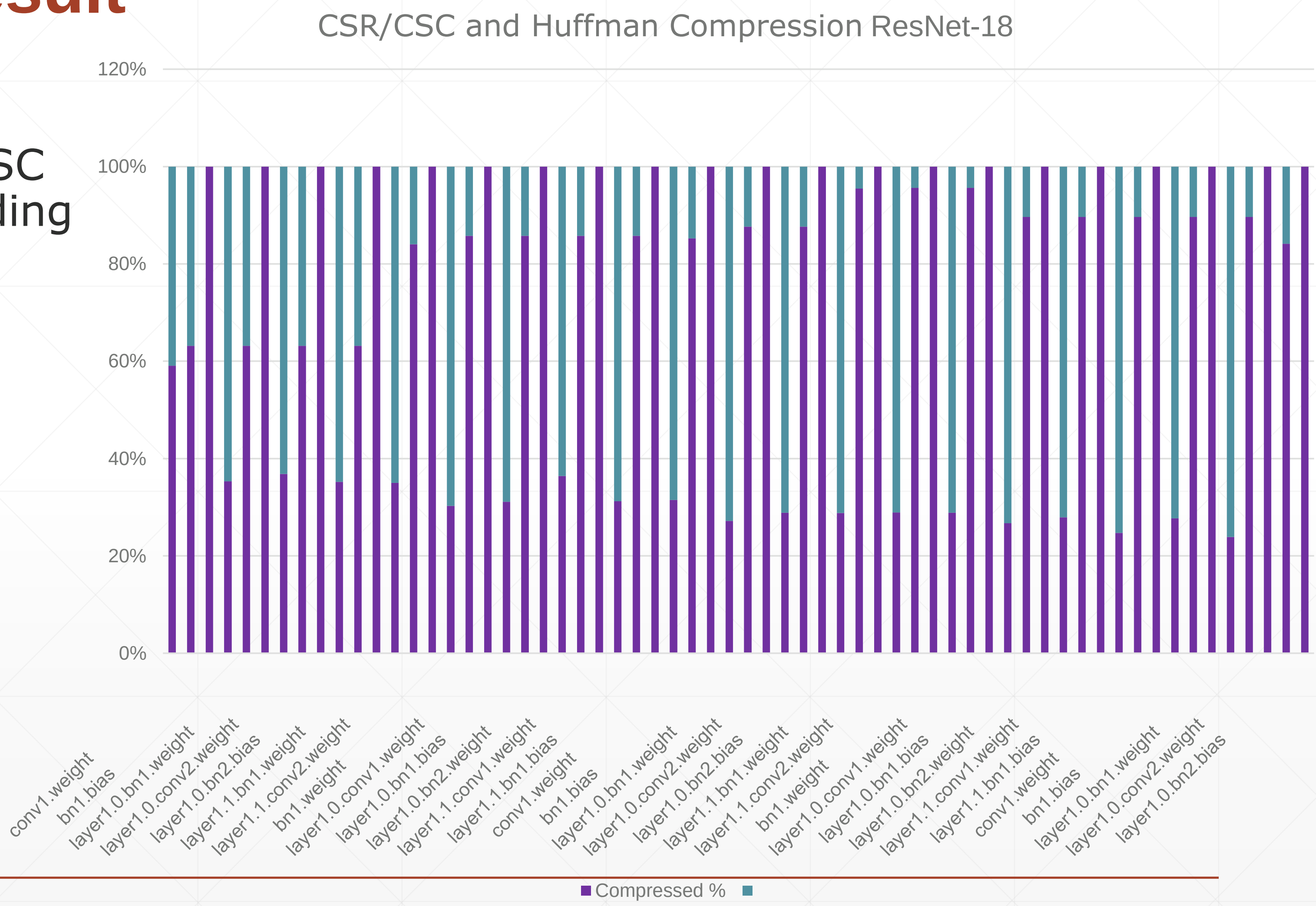
❑ Step 1: Pruning and Retraining:

- Using $threshold = \sigma * 0.02)$



4. Experimental Result

- Step 2: Weight Sharing
- Step 3: Convert to CSR/CSC matrix and Huffman Encoding



4. Experimental Result

- Deep Compression 2016 Final Result:

Model	Initial Accuracy	After Pruning				Weight Sharing	Convert to CSR/CSC matrix and Huffman Encoding		Final Result	
		Alive Param	Compression Rate	Accuracy	Accuracy after Retrain	Accuracy	Accuracy	Compression Rate More	Model size	Accuracy
ResNet-18: Total Param #: 11,173,962	87.59% (81 epoches)	563,354	19.83x (94.96% pruned)	10%	89.95%, ↑2.36% (19 epoches)	89.42%, ↓0.53%	89.42%	↓3.52x, ↓71%	↓65.58x, ↓98.47% # Param: ↓94.96%	89.42%, ↑1.83%
ResNet-50: Total Param #: 23,520,842	94.55% (150 epoches)	1,239,011	18.98x (94.73% pruned)	10%	92.6%, ↓1.95% (100 epoches)	92.27%, ↓0.33%	92.27%	↓3.41x, ↓29.35%	↓57.08x, ↓98.25% # Param: ↓94.73%	92.27%, ↓2.28%

Model	Method	Params.↓%	FLOPs↓%	Ref.%[1]	Acc.↓%
ResNet-20	SNLI [40]	37.22	–[2]	92.00	1.10
	SFP [11]	–	**42.20**	92.20	1.37
	CNN-FCF	**42.75**	41.60	92.20	**1.07**
	SNLI [40]	67.83	–	92.00	3.20
	CNN-FCF	**68.44**	**68.91**	92.20	**2.67**
ResNet-32	SFP [11]	–	41.50	92.63	0.55
	CNN-FCF	**42.71**	**42.21**	92.43	**0.25**
	CNN-FCF	69.46	70.21	92.43	1.69
ResNet-56	Pruning-A [21]	9.40	10.40	93.04	-0.06
	Pruning-B [21]	13.70	27.60	93.04	-0.02
	SFP [11]	–	41.10	93.59	-0.19
	NISP [41]	42.60	**43.61**	–	0.03
	CNN-FCF	**43.09**	42.78	93.14	**-0.24**[3]
	CNN-FCF	69.74	70.90	93.14	1.22
ResNet-110	Pruning-A [21]	2.30	15.90	93.53	0.02
	Pruning-B [21]	32.40	38.60	93.53	0.23
	SFP [11]	–	40.80	93.68	**-0.18**
	NISP [41]	**43.25**	**43.78**	–	0.18
	CNN-FCF	43.19	43.08	93.58	-0.09
	CNN-FCF	69.51	70.81	93.58	0.62

Compressing Convolutional Neural Networks via Factorized Convolutional Filters in 2019

STRUCTURED WEIGHT PRUNING RESULTS ON MULTI-LAYER NETWORK ON MNIST, CIFAR-10 AND IMAGENET ILSVRC-2012 DATASETS

Structured Weight Pruning Statistics					
Method	Original Accuracy	Prune Rate w/o P-RM	Accuracy w/o P-RM	Prune Rate with P-RM	Accuracy with P-RM
MNIST					
SSL		26.10×	99.00%	N/A	N/A
our LeNet-5	99.17%	23/18× 34.46× 45.54×	99.20% 99.06% 98.48%	**39.23×** ***87.93×** **231.82×**	**99.20%** **99.06%** **98.48%**
			*numbers of parameter reduced: **25.2K**		
CIFAR-10					
2PFPCE	92.98%	4.00×	92.76%	N/A	N/A
our VGG-16	93.70%	20.16×	93.36%	**44.67×** ***50.02×**	**93.36%** **92.73%**
AMC	93.53%	1.70×	93.55%	N/A	N/A
our ResNet-18	94.14%	5.83× 15.14×	93.79% 93.20%	**52.07×** ***60.11×**	**93.79%** **93.22%**
			*numbers of parameter reduced on: *VGG-16*: **14.42M**, *ResNet-18*: **10.97M**		
ImageNet ILSVRC-2012					
SSL AlexNet	80.40%	1.40×	80.40%	N/A	N/A
our AlexNet	82.40%	4.69×	81.76%	**5.13×**	**81.76%**
our ResNet-18	89.07%	3.02×	88.41%	**3.33×**	**88.47%**
our ResNet-50	92.86%	2.00×	92.26%	**2.70×**	**92.27%**
			numbers of parameter reduced on: *AlexNet*: **1.66M**, *ResNet-18*: **7.81M**, *ResNet-50*: **14.77M**		

Network Purification and Unused Path Removal (P-RM) after Weight Pruning in 2019

4. Conclusion

- The Deep Compression 2016 achieves a very good pruned rate and testing accuracy comparing to the recent methods (2019).
- The accuracy after compression is slightly increased (1.83%) instead of decreasing. My guess is that the model original accuracy (87.59%) has not reached to the optimized point; therefore, the pruning process might remove some unused neuron and make the model a little bit better (as same as Lasso regularization).
- However, the deep compression might have a big inference timing due to some delay in decoding:
  - Compressed sparse row (CSR) / Compressed sparse column (CSC) format
  - Huffman encoded weight
- This limitation can be improved by using the hardware accleator for the sparse matrix calcuation.

➢ Work Remaining:

- Improve the baseline ResNet-18 testing accuracy
- Conduct more experiement with a deeper model (ResNet-152)
- Experiment and compare the inference timing and the FLOPS complexity between Deep Compression and other models.

Q&A

Thank You