2020 embedded VISION summit

Joint Regularization on Activations and Weights for Efficient Neural Network Pruning

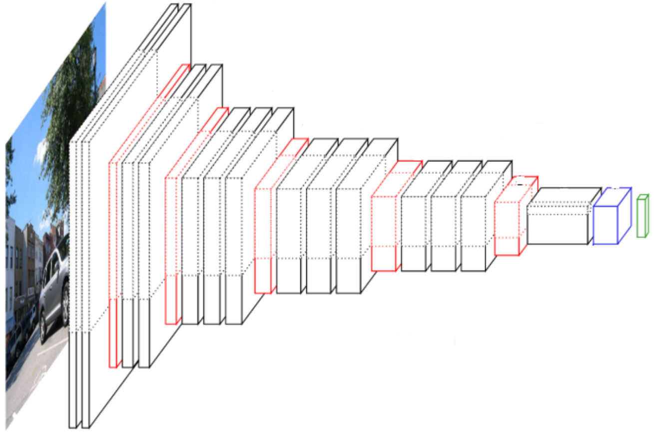
Zuoguan Wang, Senior Algorithm Manager

September 2020

BLACK SESAME TECHNOLOGIES

Neural Network Deployment Challenge

Big network

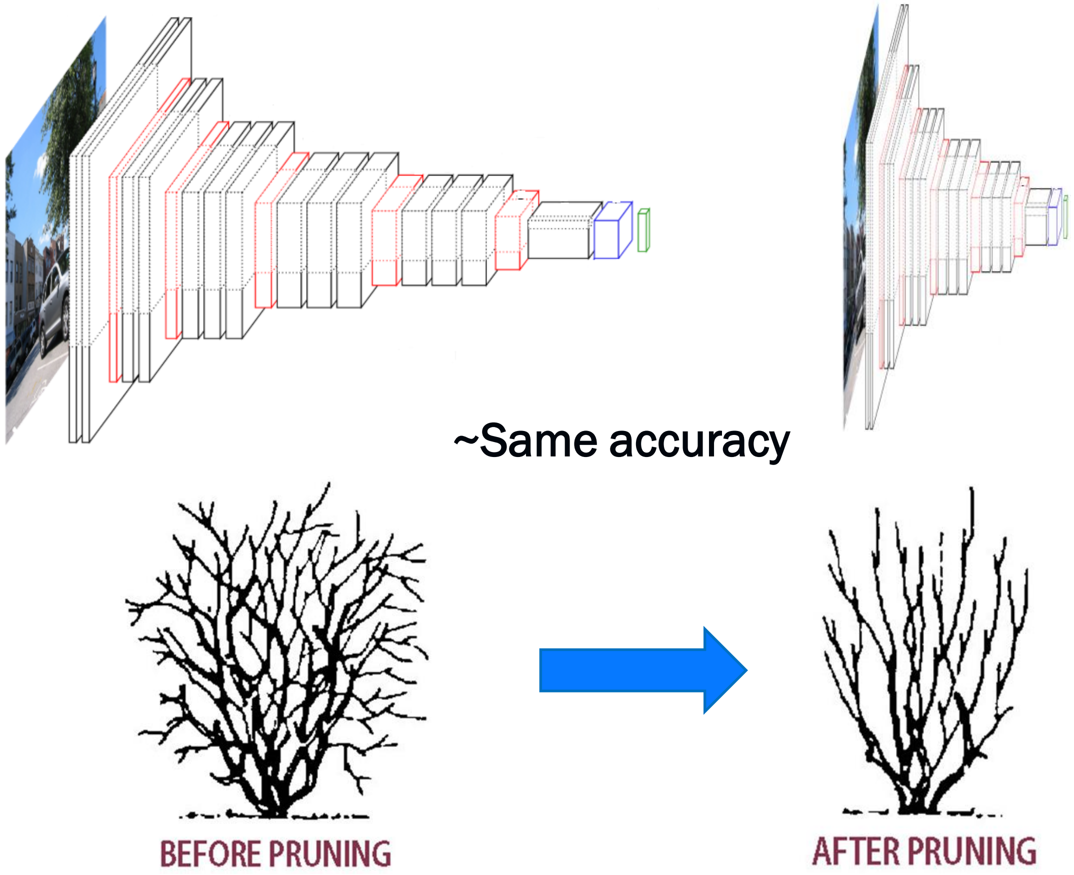


Computation & Memory & Power

Deployment

Embedded system, e.g., autonomous driving

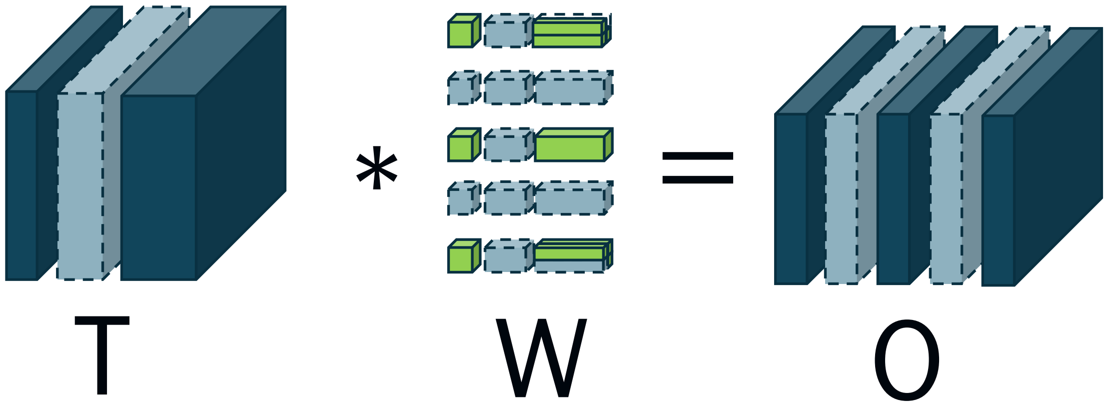
Neural Network Pruning

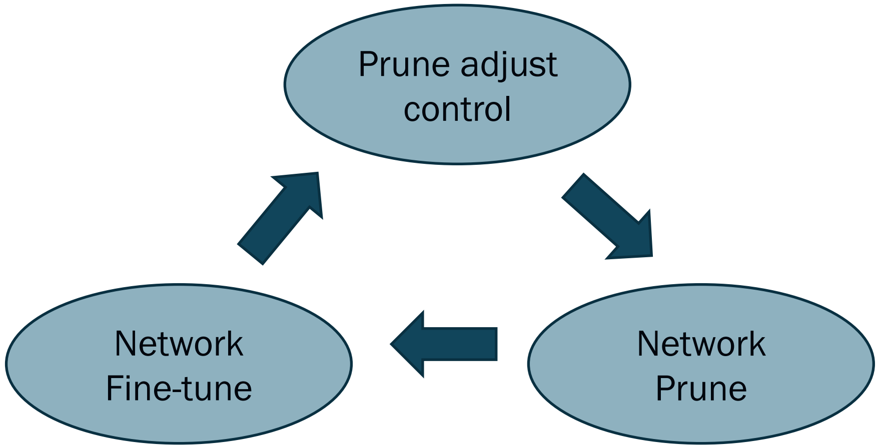


Purposes of network pruning

- Reduce network size
- Improve inference speed
- Deployed on resource constrained platform

Weight Pruning





Magnitude based:

$$m_i = \begin{cases} 1 & if\ |w_i| \geq t \\ 0 & if\ |w_i| < t \end{cases}$$

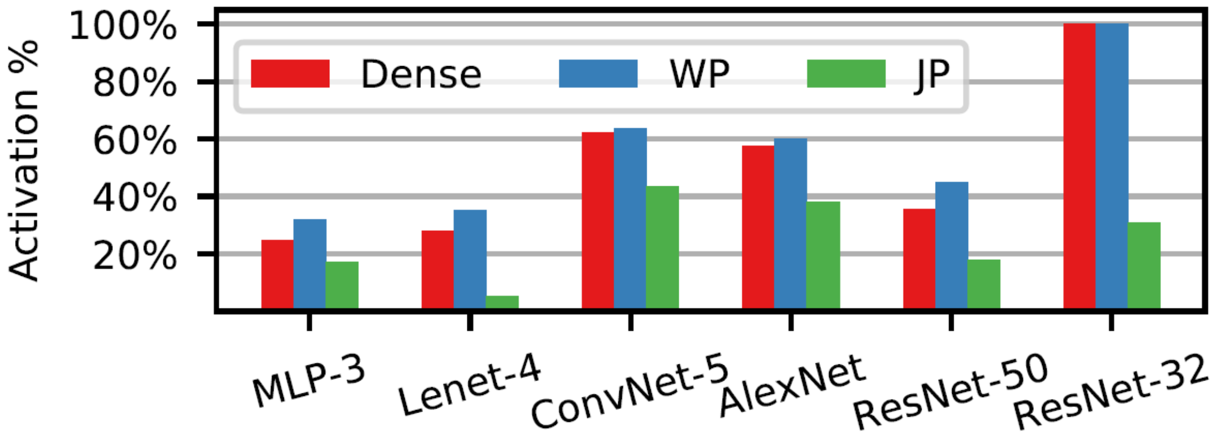
Sensitivity based:

$$m_i = \begin{cases} 1 & if\ \left| w_i \frac{\partial L}{w_i} \right| \geq t \\ 0 & if\ \left| w_i \frac{\partial L}{w_i} \right| < t \end{cases}$$

where m is sparsity mask.

Motivations

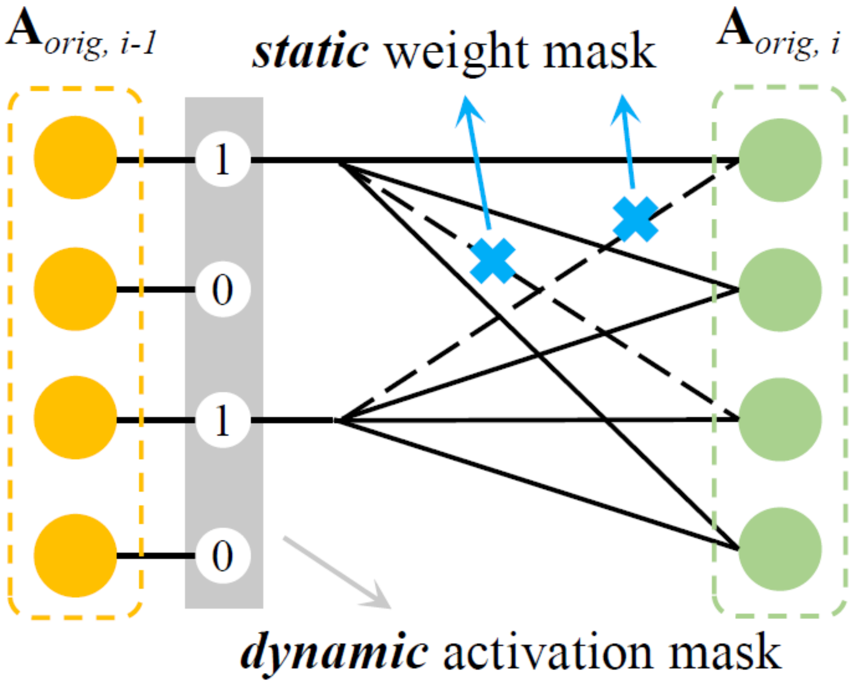
- Activation sparsity could reduce computation as well
- Weight sparsity causes more non-zero activations



* Dense – without pruning
* WP – weight pruning
* JP – weight and activation joint pruning

Joint Regularization for Weights and Activations

Joint Regularization

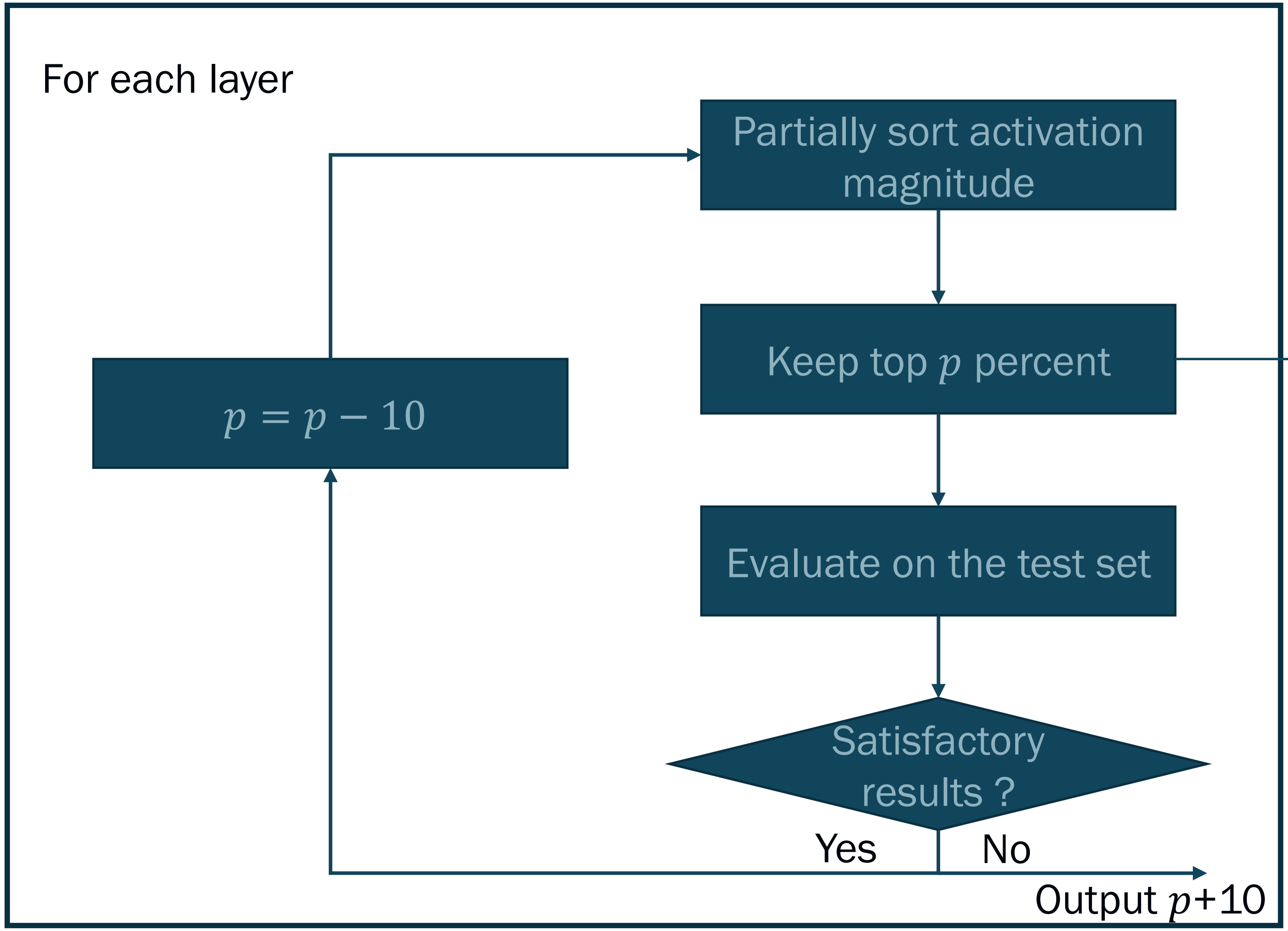


$$Loss = \frac{1}{n}\sum_{k=1}^{n}\mathcal{L}(\mathbf{y}_k, \mathcal{D}(\mathbf{x}_k, \mathbb{S}_W, \mathbb{S}_T)) + \alpha \cdot ||\mathbb{S}_W||_1,$$

$$\mathbb{S}_W^*,\ \mathbb{S}_T^* = \underset{\mathbb{S}_W,\ \mathbb{S}_T}{\operatorname{argmin}}\ \{Loss\}.$$

- Use widely adopted ℓ_1/ℓ_2 regularizers to get **static** weight mask.
- Use projected gradient descent (PGD) method to get **dynamic** ℓ_0 activation mask.
- The **dynamic** mask keeps the N largest elements, while pruning small elements, compared with **static** mask where elements below a given threshold are pruned.

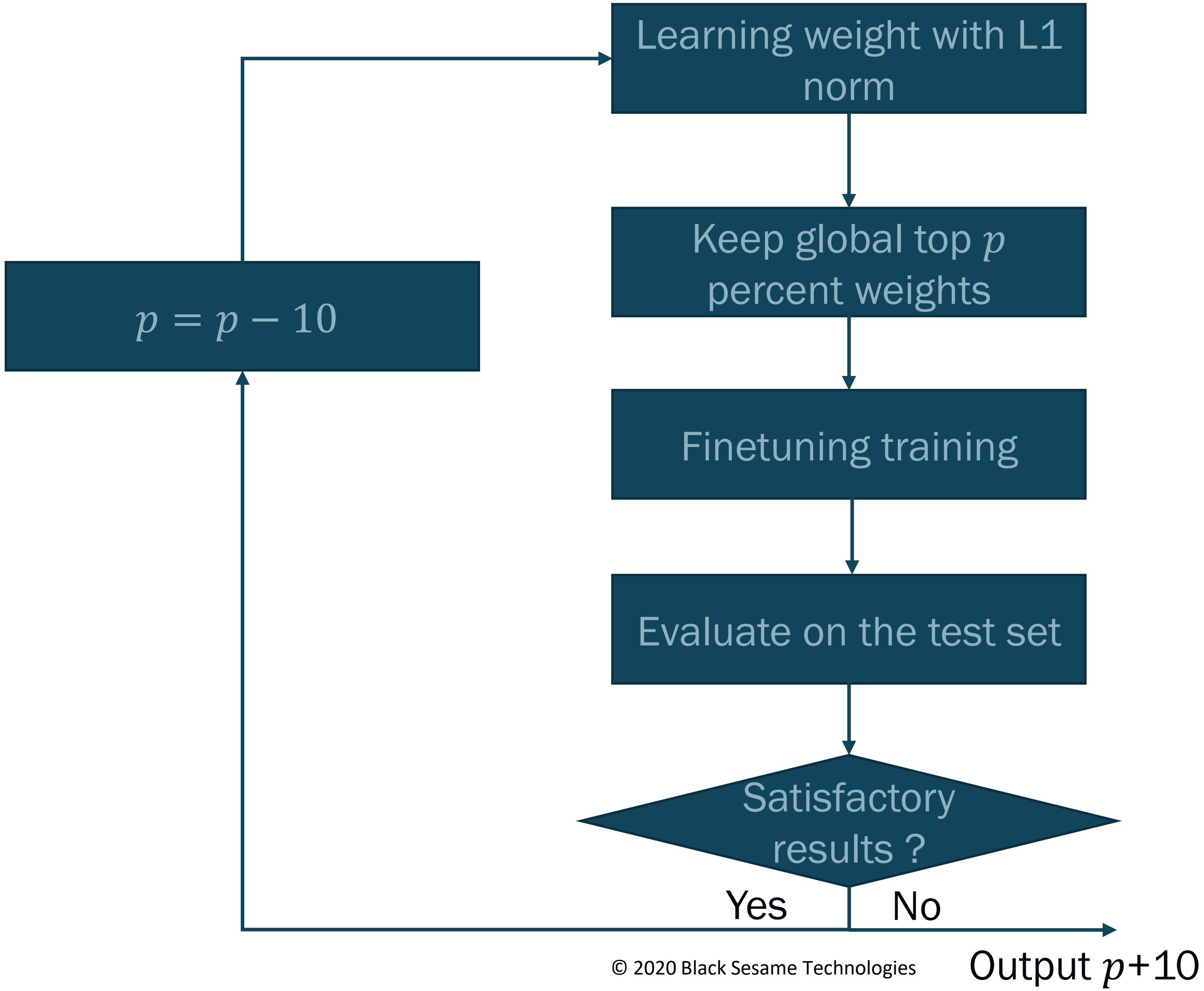
Dynamic Activation Mask



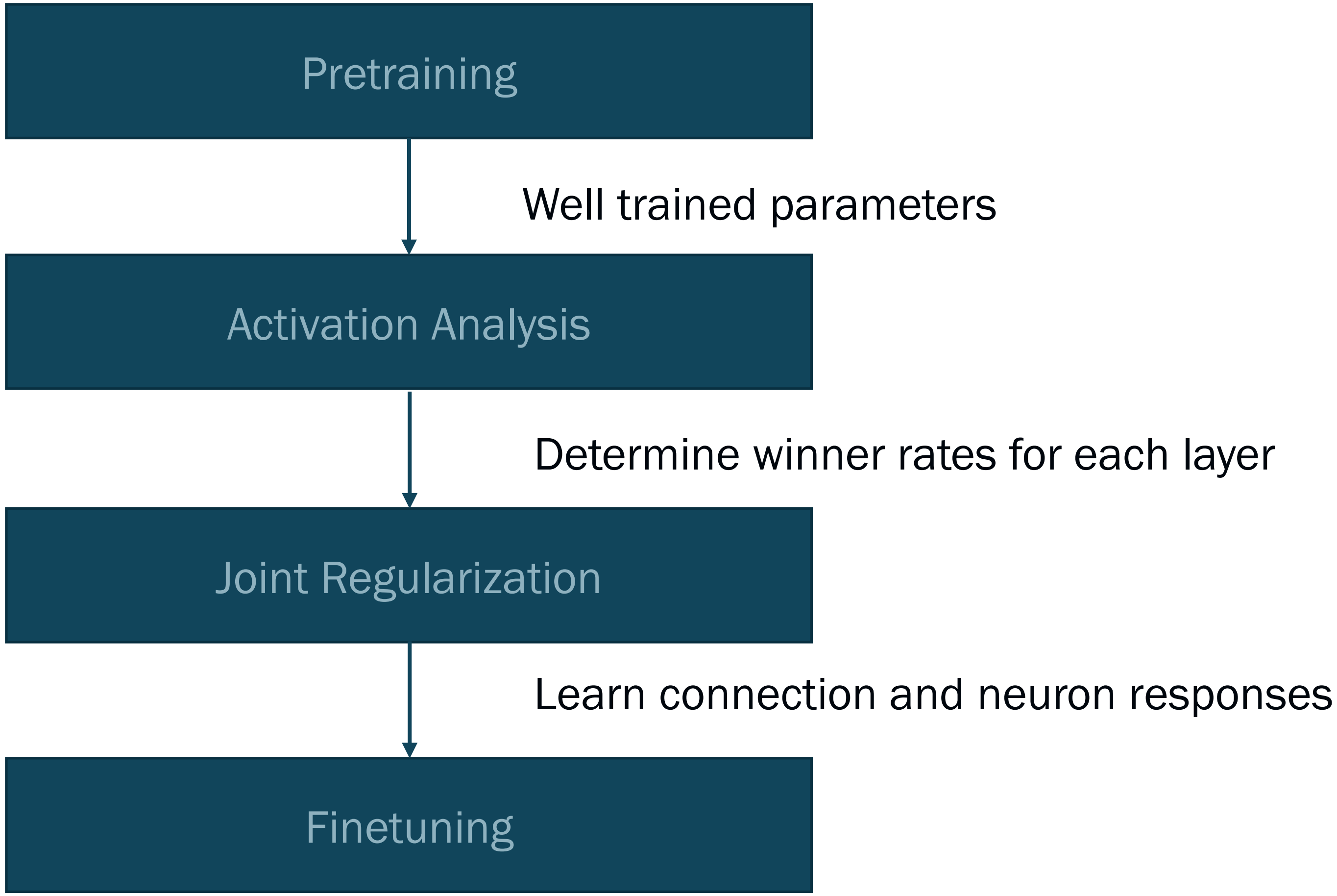
$$a_{m,j} = \begin{cases} a_{orig,j}, & \text{when } \mathbf{abs}(a_{orig,j}) > \theta, \\ 0, & \text{otherwise.} \end{cases}$$

The acceptable results are evaluated against tolerable loss, e.g., 2%

Static Weight Mask



Network Training



Experiments

Applied to Multiple Nets on Multiple Datasets

Compared to the original dense network, within 0.4% accuracy loss:

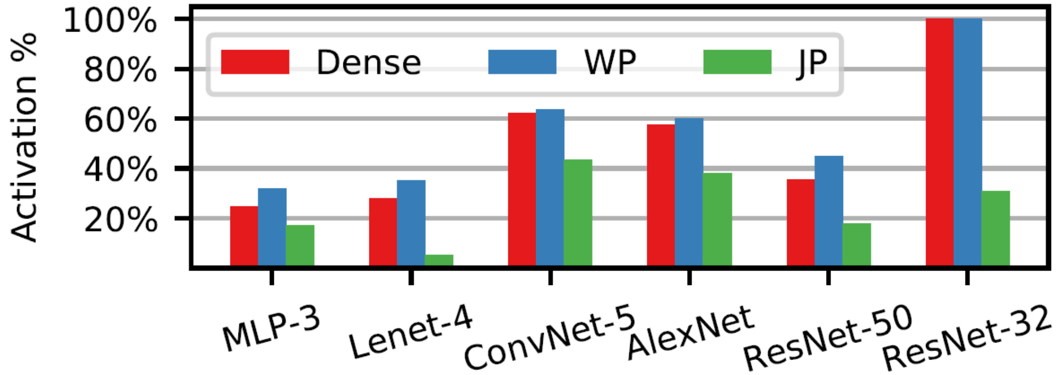
- 1.4x to 5.2x activation compression rate
- 1.6x to 12.3x weight compression rate
- 72.3% to 98.8% MAC reduction

TABLE I
SUMMARY OF JPNETS

Network	MLP-3	Lenet-4	ConvNet-5	AlexNet	ResNet-50	ResNet-32
Dataset	MNIST	MNIST	CIFAR-10	ImageNet	ImageNet	CIFAR-10
Activation Function	ReLU	ReLU	ReLU	ReLU	ReLU	Leaky ReLU
Accuracy Baseline	98.41%	99.4%	86.0%	57.22%	75.6%	95.0%
Accuracy Joint Regularization	98.42%	99.0%	85.9%	57.26%	75.7%	94.6%
Activation Percentage	17.1%	5.5%	43.6%	37.9%	17.7%	30.8%
Weight Compression Rate	10×	12.3×	2.5×	5.3×	1.6×	3.1×
MAC Percentage	3.65%	1.2%	27.7%	25.2%	19.1%	11.5%

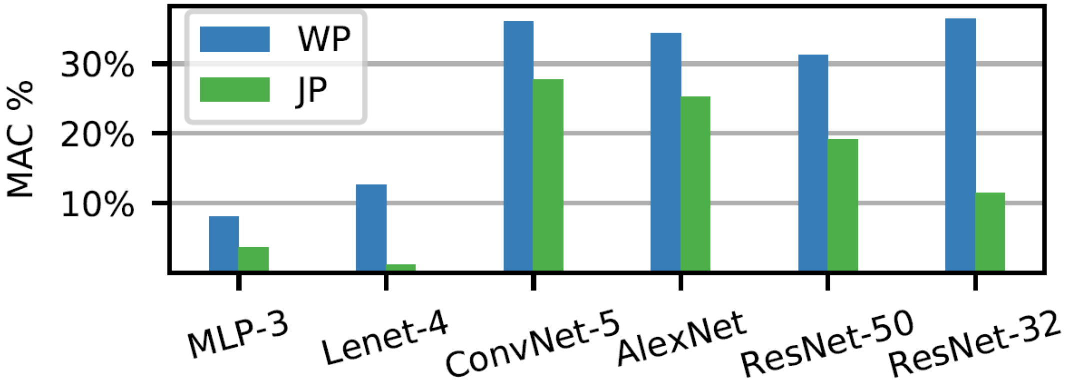
- Activation Percentage: percentage of non-zero activations
- MAC Percentage: percentage relative to dense network

Compared with Weight Pruning Only



(a) Comparison of non-zero activation percentage.

* Percentage relative to total activations



(b) Comparison of MAC percentage.

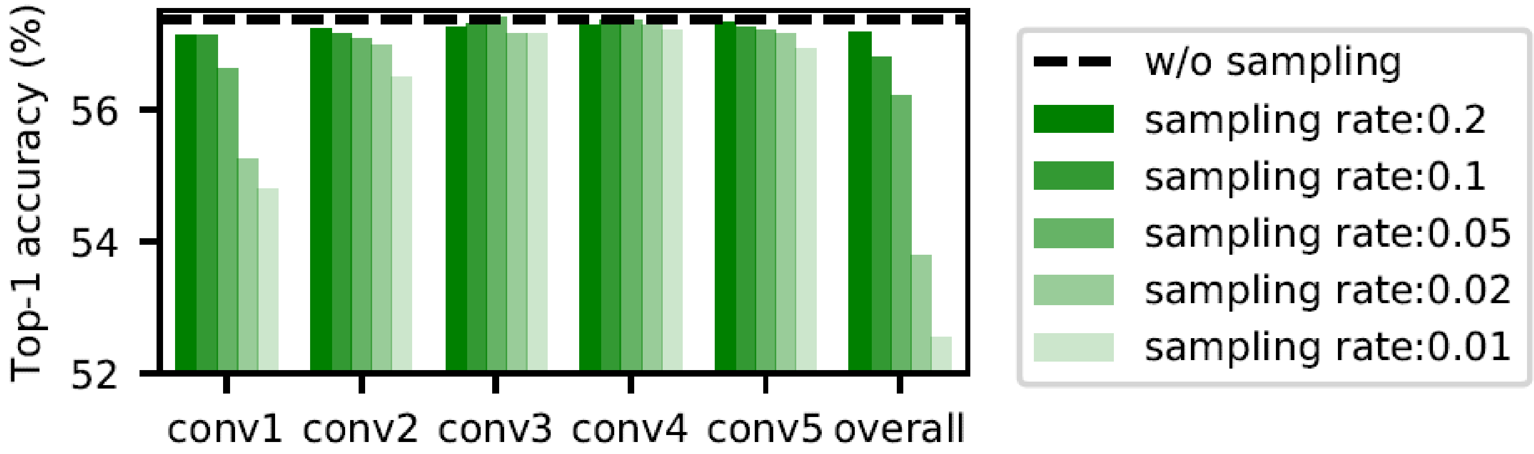
* Percentage relative to dense network

- The weight pruning based on ℓ_1/ℓ_2 regularization tends to increase the non-zero activation percentage compared to the original dense models.
- JP provides a 1.3× ~ 10.5× improvement compared to WP only by remove 7.7% ~ 22.5% more activations.
- JP also works on non-ReLU activation function, like in ResNet-32.

BLACK SESAME TECHNOLOGIES

Sampling Rate for Activation Analysis

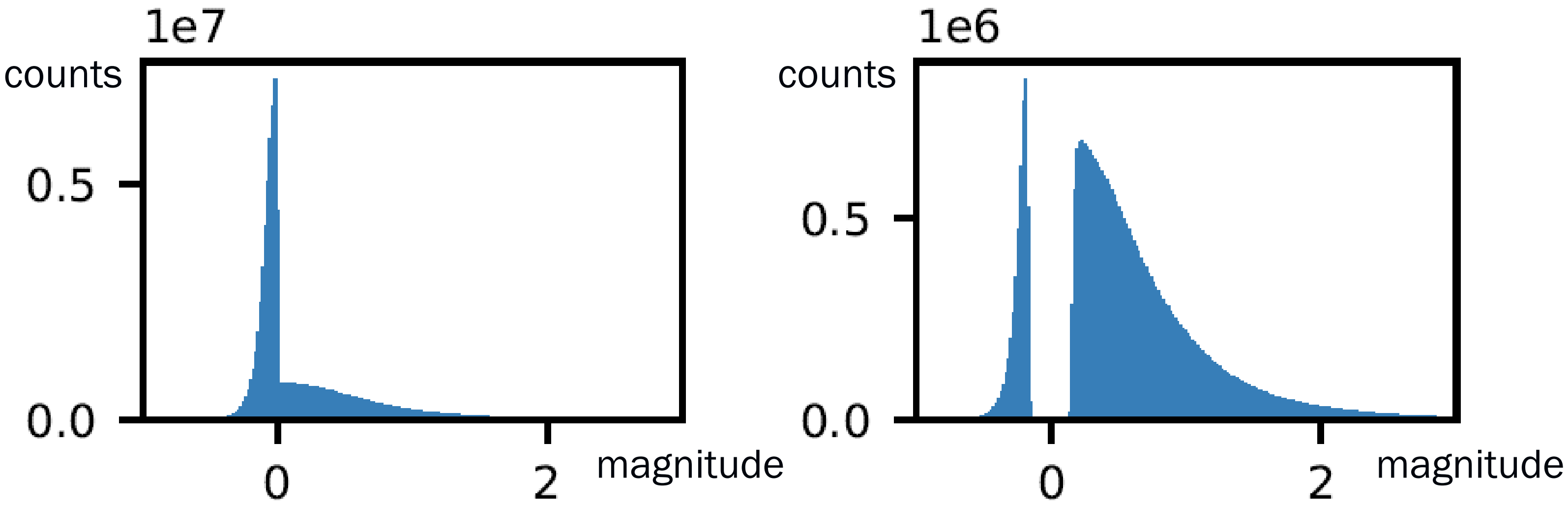
- The identification of activation pruning threshold could be speeded up by predicting on a down sampled activation set
- The effect of sampling on final accuracy is different for layers
- In the AlexNet example, down sampling 10% is a good compromise



The effects of threshold prediction

Activation Distribution

- The activations near zero are pruned out
- The remaining activations have larger magnitude

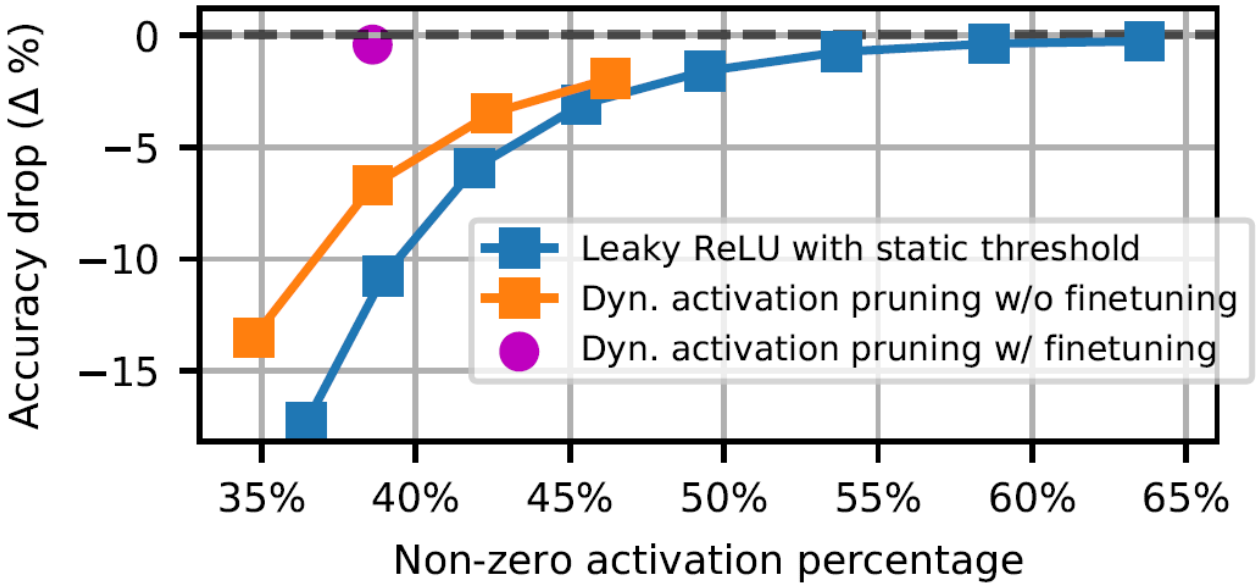


(a) Original. (b) After joint pruning.

Activation distribution of ResNet-32

Compared with Static Activation Pruning

- ResNet-32 with leaky ReLU on CIFAR-10 dataset
- Dynamic activation pruning achieves better performance under the same pruning rate



Comparison to static activation pruning for ResNet-32

Comparison with State of Art Weight Pruning Methods

- Compared with L0, VIB and VD, our method can be easily applied to large models
- Compared with DNS/ADMM, we can obtain minimum prediction error with a comparable computation reduction

COMPARISON WITH THE STATE-OF-THE-ART WEIGHT PRUNING METHODS.

Model	Dataset	Method	Weight %	MAC Reduction	Error
Lenet-4	MNIST	L0	8.9%	5.9×	0.9%
		VIB	0.8%	71.4×	1.0%
		VD	0.4%	80.6×	**0.8%**
		Ours	8.1%	**83.3**×	1.0%
AlexNet*	ImageNet	DNS	32.5%	3.7×	20%
		ADMM	20.5%	**3.8**×	19.8%
		Ours	38.7%	3.7×	**19.6%**

* For AlexNet, we focus on *conv* layers which are the computation bottleneck for inference. The top-5 prediction error is reported in the table.

* Weight % represents percentage of weights after pruning

- **L0**: l_0 regularization
- **VIB**: variational information bottleneck
- **VD**: variational dropout
- **DNS**: dynamic network surgery
- **ADMM**: non-convex problem optimization method

Conclusion

- This work presents a joint regularization algorithm for network pruning, which prunes not only weights but also activations
- Overall, joint regularization outperforms weight only pruning, pointing a promising direction to further network compression
- The model derived from joint regularization requires dedicated DNN accelerators to take the full advantage of the sparsity

Resources

Paper link:

https://arxiv.org/abs/1906.07875

Company website:

http://www.bst.ai/