Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon [1]

X Dong

University of California, San Diego

doxin@ucsd.edu

October 27, 2017

[1] To appear in NIPS 2017, Long Beach, CA, USA

Introduction

The size of deep neural networks has been being tremendously increased, from LeNet-5 with less than 1M parameters to VGG-16 with 133M parameters.

So, can we compress the matrix of parameter via **sparsity**?

1. Experimental results indicate redundancy of deep neural network
2. Memory may have relation with vanishment of specific synapses

Related Works

Pruning methods have been widely used for model compression in early neural networks and modern deep neural networks.

- Classical methods: OBD and OBS

$$\delta E^l = \left(\frac{\partial E^l}{\partial \mathbf{\Theta}_l}\right)^\top \delta \mathbf{\Theta}_l + \frac{1}{2}\delta \mathbf{\Theta}_l{}^\top \mathbf{H}_l \delta \mathbf{\Theta}_l + O\left(\|\delta \mathbf{\Theta}_l\|^3\right) \quad (1)$$

- Norm methods: P-norm inducing sparsity
- Intuitive methods: set threshold, prune parameters and retrain

Layer-Wise Error

Focus on one layer l in deep neural network.

Suppose every time we aim to find a parameter $\mathbf{\Theta}_{l_{[q]}}$ to set to be zero such that the change δE^l is minimal.

$$\min_q \frac{1}{2}\delta\mathbf{\Theta}_l{}^\top \mathbf{H}_l \delta\mathbf{\Theta}_l, \quad \text{s.t.} \quad \mathbf{e}_q^\top \delta\mathbf{\Theta}_l + \mathbf{\Theta}_{l_{[q]}} = 0, \tag{2}$$

where $\mathbf{e}_q = [0, \cdots, 1, \cdots, 0]$ whose q-th element is 1 and otherwise 0.

Via the Lagrange multipliers method,

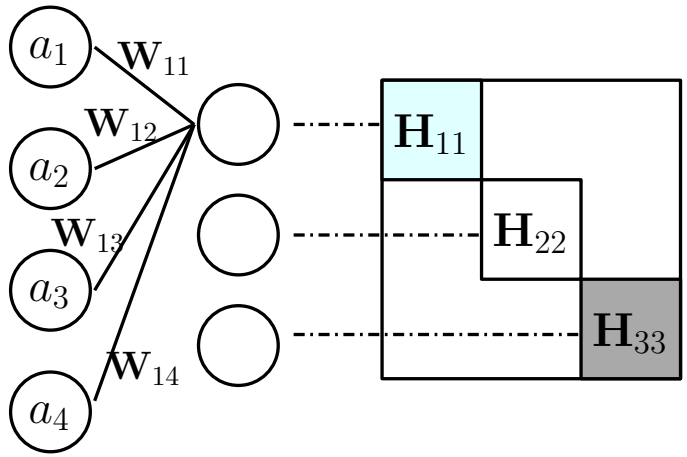
$$\delta\mathbf{\Theta}_l = -\frac{\mathbf{\Theta}_{l_{[q]}}}{[\mathbf{H}_l^{-1}]_{qq}}\mathbf{H}_l^{-1}\mathbf{e}_q, \quad \text{and} \quad L_q = \delta E^l = \frac{1}{2}\frac{(\mathbf{\Theta}_{l_{[q]}})^2}{[\mathbf{H}_l^{-1}]_{qq}}, \tag{3}$$

Layer-Wise pruning

1. Size of Hessian matrix will be reduced a lot.

i) Lemma 3.1

$$\begin{aligned} \varepsilon^l &= \frac{1}{\sqrt{n}} \|\hat{\mathbf{Y}}^l - \mathbf{Y}^l\|_F \\ &\leq \sqrt{E(\hat{\mathbf{Z}}^l)} = \sqrt{\frac{1}{n} \left\|\hat{\mathbf{Z}}^l - \mathbf{Z}^l\right\|_F^2} \end{aligned}$$



ii) $\mathbf{H} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{H}^j \approx \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m_l} \frac{\partial z_{ij}}{\partial \mathbf{\Theta}_l}\left(\frac{\partial z_{ij}}{\partial \mathbf{\Theta}_l}\right)^\top$

(n is number of samples and m_l is number of output units.)

iii) Hessian is the outer product of input!

Layer-Wise pruning

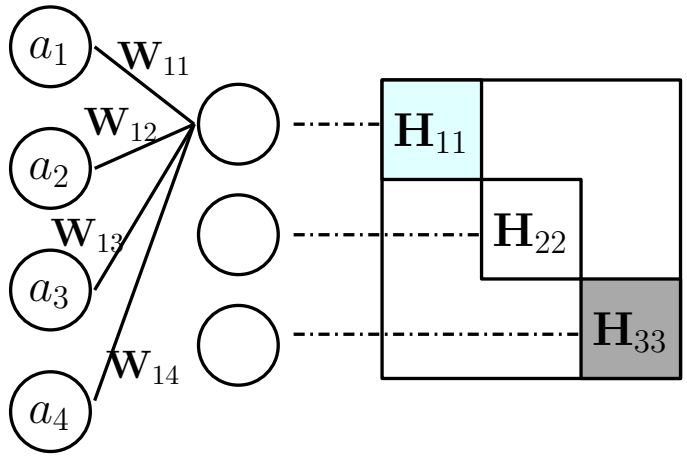
2. More efficient computation.

i) For the first output unit, only

$$\frac{\partial z_{1j}}{\partial W_{1h}} \neq 0, \quad where \;\; h = \{1, 2, 3, 4\}$$

Hessian of this unit has nonzero elements only at area in blue.

ii) $\mathbf{H}_{11} = \mathbf{H}_{11} = \mathbf{H}_{11} = \mathbf{\Psi}$

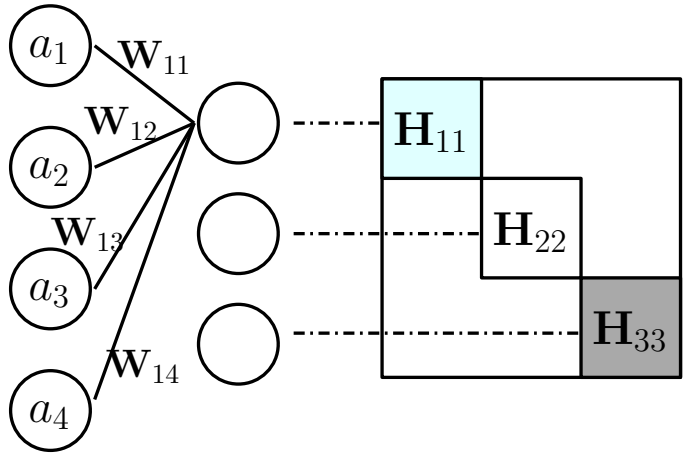


Layer-Wise pruning

3. Compute Inverse

i) Sum of matrices' inverse can be computed via Woodbury matrix identity:

$$\mathbf{\Psi}_{j+1}^{-1} = \Psi_j^{-1} - \frac{\mathbf{\Psi}_j^{-1}\mathbf{y}_j^{l-1}\left(\mathbf{y}_j^{l-1}\right)^\top \mathbf{\Psi}_j^{-1}}{n + \left(\mathbf{y}_{j+1}^{l-1}\right)^\top \mathbf{\Psi}_j^{-1}\mathbf{y}_{j+1}^{l-1}}$$



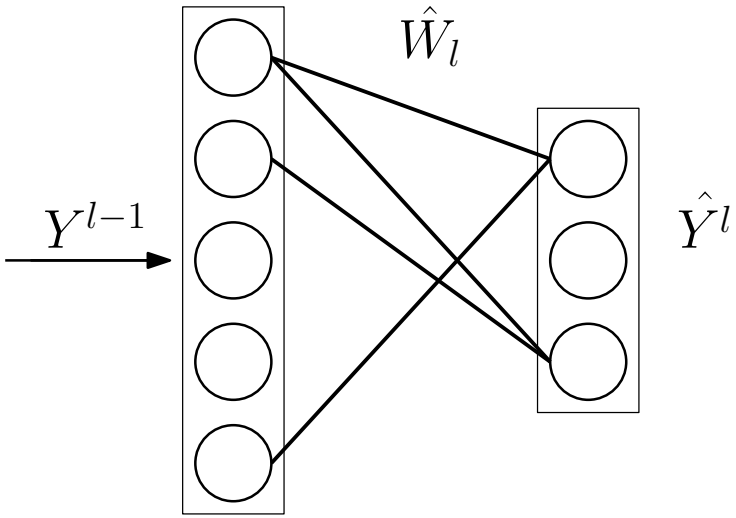
L-OBS Workflow

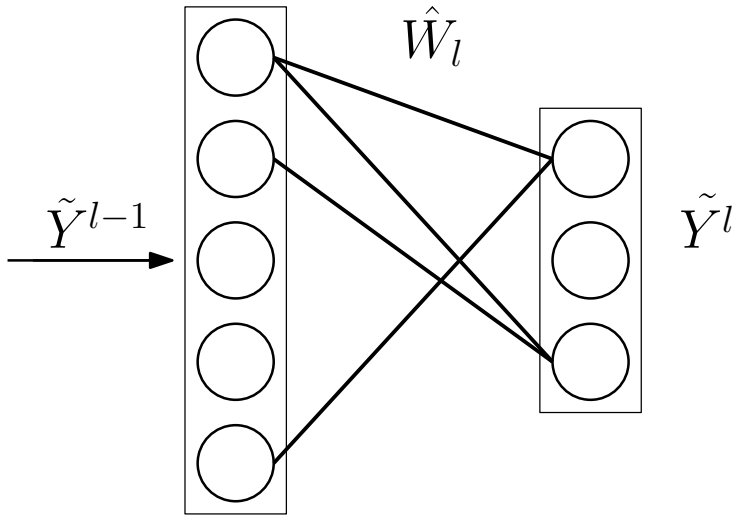
According to above analysis, the procedure of our pruning algorithm for a fully-connected layer l is as follows.

1. Get layer input $\mathbf{y}^{l-1}$ from a well-trained deep network.
2. Calculate the Hessian matrix $\mathbf{H}_{ii}$ and its pseudo-inverse
3. Compute optimal parameter change $\delta\mathbf{\Theta}_l$ and the sensitivity L_q. Set tolerable error threshold ϵ.
4. Pick up parameters $\mathbf{\Theta}_{l_{[q]}}$'s with the smallest sensitivity scores.
5. If $\sqrt{L_q} \leq \epsilon$, prune the parameter $\mathbf{\Theta}_{l_{[q]}}$'s and get new parameter values via $\hat{\mathbf{\Theta}} = \mathbf{\Theta} + \delta\mathbf{\Theta}$, then repeat Step 4; otherwise stop pruning.

Layer-Wise Error Propagation and Accumulation

Layer-wise errors will accumulate.
Then, error of the ultimate network output may **explode**.





$$\tilde{\mathbf{Y}}^l = \sigma(\hat{\mathbf{W}}_l^\top \tilde{\mathbf{Y}}^{l-1}), \quad \tilde{\mathbf{Y}}^1 = \hat{\mathbf{Y}}^1 = \sigma(\hat{\mathbf{W}}_1^\top \mathbf{X}), \text{ and } \hat{\mathbf{Y}}^l = \sigma(\hat{\mathbf{W}}_l^\top \mathbf{Y}^{l-1}).$$

Layer-Wise Error Propagation and Accumulation

Theorem (3.2)

If each layer has its own layer-wise error ε^l, $1 \leq l \leq L$, *then the accumulated error of ultimate network output* $\tilde{\varepsilon}^L = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^L - \mathbf{Y}^L\|_F$ *obeys:*

$$\tilde{\varepsilon}^L \leq \sum_{k=1}^{L-1} \left(\prod_{l=k+1}^{L} \|\hat{\mathbf{\Theta}}_l\|_F \sqrt{\delta E^k} \right) + \sqrt{\delta E^L}$$

where $\tilde{\mathbf{Y}}^l$ *denotes 'accumulated pruned output' of layer* l, $\tilde{\mathbf{Y}}^l = \sigma(\hat{\mathbf{W}}_l^\top \tilde{\mathbf{Y}}^{l-1})$ *and* $\tilde{\mathbf{Y}}^1 = \sigma(\hat{\mathbf{W}}_1^\top \mathbf{X})$.

$$\tilde{\varepsilon}^L \leq \sum_{k=1}^{L-1} \left(\prod_{l=k+1}^{L} \|\hat{\mathbf{\Theta}}_l\|_F \sqrt{\delta E^k} \right) + \sqrt{\delta E^L}$$

Theorem (3.2) shows that:

- Layer-wise error for a layer l will be scaled by continued multiplication of parameters' Frobenius Norm over layers after l when it propagates to final output.

$$\tilde{\varepsilon}^L \leq \sum_{k=1}^{L-1} \left(\prod_{l=k+1}^{L} \|\hat{\mathbf{\Theta}}_l\|_F \sqrt{\delta E^k} \right) + \sqrt{\delta E^L}$$

Theorem (3.2) shows that:

- Layer-wise error for a layer l will be scaled by continued multiplication of parameters' Frobenius Norm over layers after l when it propagates to final output.
- The final error of ultimate network output is bounded by **weighted sum** of layer-wise errors.

$$\tilde{\varepsilon}^L \leq \sum_{k=1}^{L-1} \left(\prod_{l=k+1}^{L} \|\hat{\mathbf{\Theta}}_l\|_F \sqrt{\delta E^k} \right) + \sqrt{\delta E^L}$$

Theorem (3.2) shows that:

- Layer-wise error for a layer l will be scaled by continued multiplication of parameters' Frobenius Norm over layers after l when it propagates to final output.
- The final error of ultimate network output is bounded by **weighted sum** of layer-wise errors.

Theorem (3.2) is proved via induction and triangle inequality.

Proof of Theorem 3.2

Relation of Y, $\hat{Y}$ and $\tilde{Y}$.

1. $\tilde{\varepsilon}^{l+1} = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$ by definition.

Proof of Theorem 3.2

Relation of Y, $\hat{Y}$ and $\tilde{Y}$.

1. $\tilde{\varepsilon}^{l+1} = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$ by definition.
2. $\|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F \leq \|\tilde{\mathbf{Y}}^{l+1} - \hat{\mathbf{Y}}^{(l+1)}\|_F + \|\hat{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$.

Proof of Theorem 3.2

Relation of Y, $\hat{Y}$ and $\tilde{Y}$.

1. $\tilde{\varepsilon}^{l+1} = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$ by definition.
2. $\|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F \leq \|\tilde{\mathbf{Y}}^{l+1} - \hat{\mathbf{Y}}^{(l+1)}\|_F + \|\hat{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$.
3. $\|\tilde{\mathbf{Y}}^{l+1} - \hat{\mathbf{Y}}^{l+1}\|_F \leq \sqrt{n} \|\hat{\mathbf{\Theta}}^{l+1}\|_F \tilde{\varepsilon}^{l}$.

Proof of Theorem 3.2

Relation of Y, $\hat{Y}$ and $\tilde{Y}$.

1. $\tilde{\varepsilon}^{l+1} = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$ by definition.
2. $\|\tilde{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F \leq \|\tilde{\mathbf{Y}}^{l+1} - \hat{\mathbf{Y}}^{(l+1)}\|_F + \|\hat{\mathbf{Y}}^{l+1} - \mathbf{Y}^{(l+1)}\|_F$.
3. $\|\tilde{\mathbf{Y}}^{l+1} - \hat{\mathbf{Y}}^{l+1}\|_F \leq \sqrt{n} \|\hat{\mathbf{\Theta}}^{l+1}\|_F \tilde{\varepsilon}^{l}$.
4. When $l = 1$, $\tilde{\varepsilon}^{1} = \frac{1}{\sqrt{n}} \|\tilde{\mathbf{Y}}^{1} - \mathbf{Y}^{(1)}\|_F = \frac{1}{\sqrt{n}} \|\hat{\mathbf{Y}}^{1} - \mathbf{Y}^{(1)}\|_F$

Redundancy of Networks

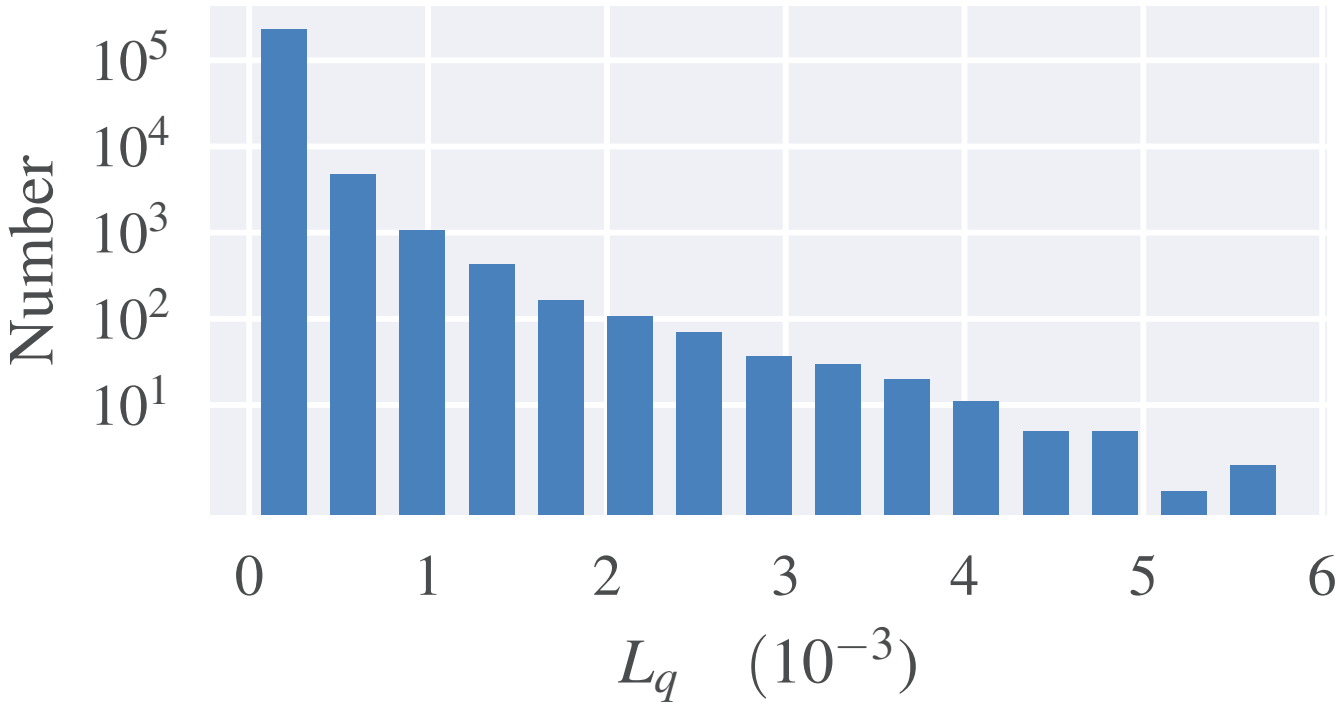


Figure: Distribution of sensitivity of parameters in LeNet-300-100's first layer. More than 90% of parameters' sensitivity is smaller than 0.001.

Redundancy of Networks

Figure: Test accuracy on MNIST using LeNet-300-100 when continually prune the first layer until pruning ratio is 100%.

Retraining Pattern

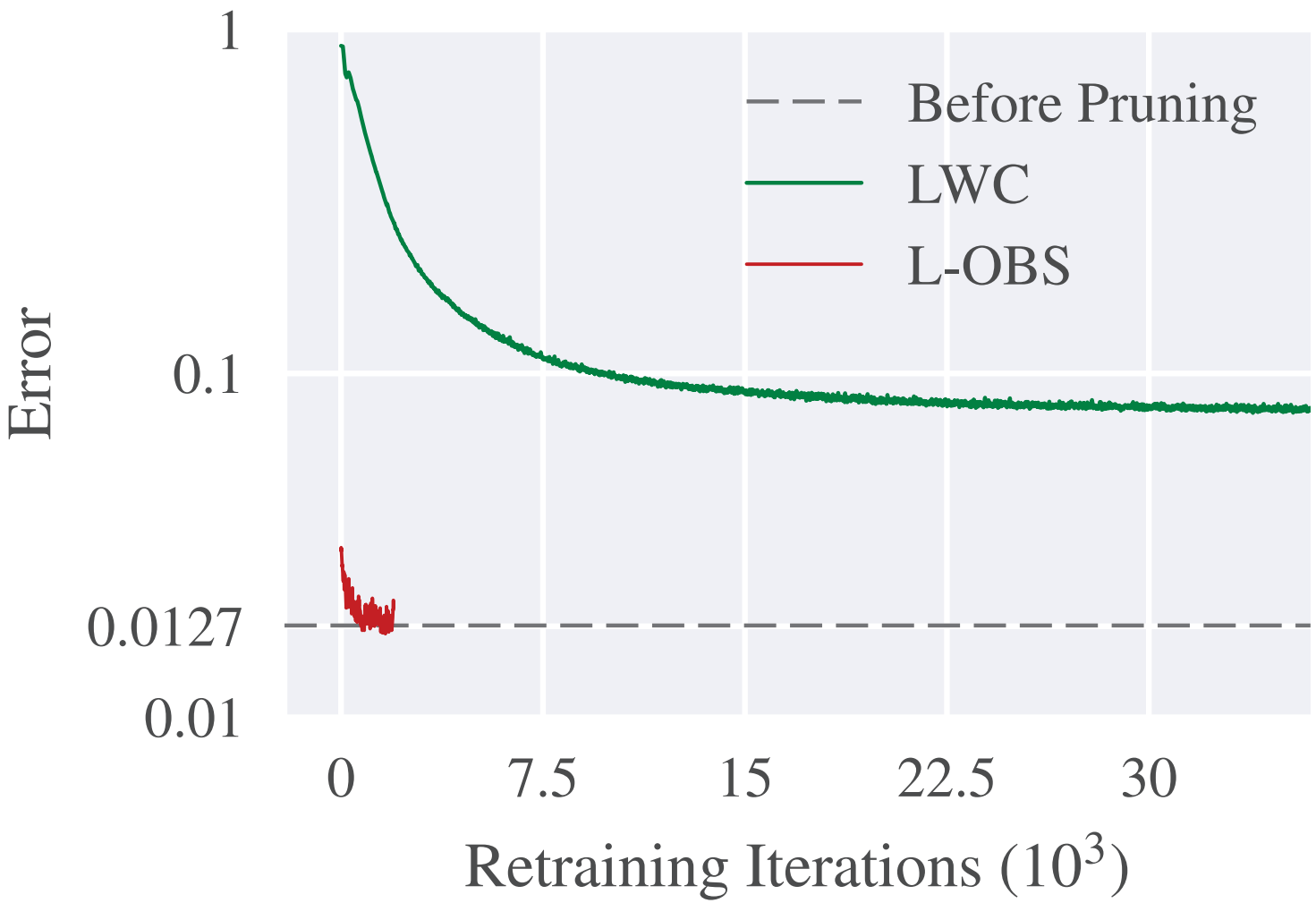


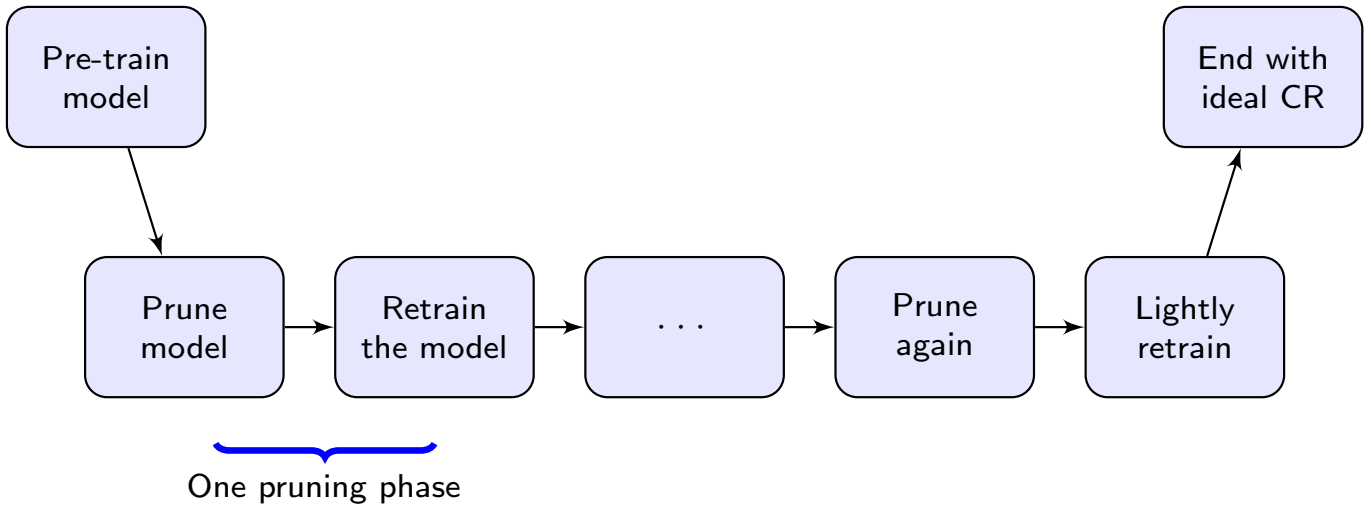
Figure: Retraining pattern of LWC and L-OBS. L-OBS has a better start point and totally resume original performance after 740 iterations for LeNet-5.

Overall Results

Method	Networks	Original Error	CR	Pruned Error	Re-Error	Re-Iters.
Random	LeNet-300-100	1.76%	8%	85.72%	2.25%	3.5×10^5
OBD	LeNet-300-100	1.76%	8%	86.72%	1.96%	8.1×10^4
LWC	LeNet-300-100	1.76%	8%	81.32%	1.95%	1.4×10^5
DNS	LeNet-300-100	1.76%	1.8%	-	1.99%	3.4×10^4
L-OBS	LeNet-300-100	1.76%	7%	**3.10%**	1.82%	**510**
L-OBS (iterative)	LeNet-300-100	1.76%	**1.5%**	2.43%	1.96%	**643**
OBD	LeNet-5	1.27%	8%	86.72%	2.65%	2.9×10^5
LWC	LeNet-5	1.27%	8%	89.55%	1.36%	9.6×10^4
DNS	LeNet-5	1.27%	**0.9%**	-	1.36%	4.7×10^4
L-OBS	LeNet-5	1.27%	7%	**3.21%**	1.27%	**740**
L-OBS (iterative)	LeNet-5	1.27%	**0.9%**	2.04%	1.66%	**841**
LWC	CIFAR-Net	18.57%	9%	87.65%	19.36%	1.62×10^5
L-OBS	CIFAR-Net	18.57%	9%	**21.32%**	18.76%	**1020**
DNS	AlexNet (Top-1 err. / Top-5 err.)	43.30 / 20.08%	**5.7%**	-	43.91 / 21.88%	7.30×10^5
LWC	AlexNet (Top-1 err. / Top-5 err.)	43.30 / 20.08%	11%	76.14 / 57.68%	44.06 / 20.64%	1.04×10^6
L-OBS	AlexNet (Top-1 err. / Top-5 err.)	43.30 / 20.08%	11%	**50.04 / 26.87%**	43.11 / 20.01%	$\mathbf{1.81 \times 10^4}$
DNS	VGG-16 (Top-1 err. / Top-5 err.)	31.66 / 10.12%	7.5%	-	63.38% / 38.69%	1×10^6
LWC	VGG-16 (Top-1 err. / Top-5 err.)	31.66 / 10.12%	7.5%	73.61 / 52.64%	32.43 / 11.12%	3.75×10^9
L-OBS	VGG-16 (Top-1 err. / Top-5 err.)	31.66 / 10.12%	7.5%	**37.32 / 14.82%**	32.02 / 10.97%	$\mathbf{8.63 \times 10^4}$

Iterative L-OBS

To achieve **better compression ratio**, L-OBS can be quite flexibly adopted to its iterative version-*doing pruning and light retraining alternately*.



Better compression ratio, more pruning cost.

Two Compression Modes

- If you want Fast-Compression:
  prune AlexNet to 16% of its original size without substantively impacting accuracy (pruned top-5 error 20.98%) even **without any retraining**.

- If you want Heavy-Compression:
  Iterative L-OBS (two iterations of pruning and retraining) achieves overall compression ratio 7.5% without loss of accuracy on VGG-16.