Supplementary Material

A. Algorithm of Kron-OBD and Kron-OBS

We present the algorithm of Kron-OBD and Kron-OBS in this section, and the derivation can be referred in the next section B.

Algorithm 2 Structured pruning algorithms Kron-OBD and Kron-OBS. For simplicity, we focus on a single layer. $\boldsymbol{\theta}_i$ below denotes the parameters of filter $\mathcal{F}_i$, which is a vector.

Require: pruning ratio p and training data $\mathcal{D}$
Require: model parameters (pretrained) $\boldsymbol{\theta} = \mathrm{vec}(\mathbf{W})$
Compute Kronecker factors $\mathbf{A} = \mathbb{E}[\mathbf{a}\mathbf{a}^\top]$ and $\mathbf{S} = \mathbb{E}[\{\nabla_{\mathbf{s}}\mathcal{L}\}\{\nabla_{\mathbf{s}}\mathcal{L}\}^\top]$
for all filter i **do**
  $\Delta\mathcal{L}_i = \frac{1}{2}\mathbf{S}_{ii}\boldsymbol{\theta}_i^\top\mathbf{A}\boldsymbol{\theta}_i$ or $\Delta\mathcal{L}_i = \frac{1}{2}\frac{\boldsymbol{\theta}_i^\top\mathbf{A}\boldsymbol{\theta}_i}{[\mathbf{S}^{-1}]_{ii}}$
end for
Compute p_{th} percentile of $\Delta\mathcal{L}$ as τ
for all filter i **do**
  if $\Delta\mathcal{L}_i \leq \tau$ **then**
    $\boldsymbol{\theta}_i \leftarrow \mathbf{0}$ or $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\mathbf{S}^{-1}\mathbf{e}_i \otimes \boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}}$
  end if
end for
Finetune the network on $\mathcal{D}$ until converge

B. Derivation of Kron-OBD and Kron-OBS

Derivation of Kron-OBD. Assuming that the weight of a conv layer is $\boldsymbol{\theta} = \mathrm{vec}(\mathbf{W})$, where $\mathbf{W} \in \mathbb{R}^{n\times m}$, $n = c_{\mathrm{in}}k^2$ and $m = c_{\mathrm{out}}$, and the two Kronecker factors are $\mathbf{S} \in \mathbb{R}^{m\times m}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$. Then the Fisher information matrix of $\boldsymbol{\theta}$ can be approximated by $\mathbf{F} = \mathbf{S} \otimes \mathbf{A}$. Substituting the Hessian with K-FAC Fisher in eqn. (8), we get:

$$\Delta\mathcal{L} = \frac{1}{2}\Delta\boldsymbol{\theta}^\top(\mathbf{S}\otimes\mathbf{A})\Delta\boldsymbol{\theta} = \frac{1}{2}\mathrm{Tr}\left(\Delta\mathbf{W}^\top\mathbf{A}\Delta\mathbf{W}\mathbf{S}\right) = \frac{1}{2}\sum_{i,j}\mathbf{S}_{ij}\Delta\boldsymbol{\theta}_i^\top\mathbf{A}\Delta\boldsymbol{\theta}_j \tag{19}$$

where $\Delta\boldsymbol{\theta}_i$ represents the change in $\boldsymbol{\theta}_i^*$, and $\boldsymbol{\theta}_i^* \in \mathbb{R}^n$ is the weight of i-th filter $\mathcal{F}_i$, *i.e.*, i-th column of $\mathbf{W}$. Under the assumption that each filter is independent to each other, and thus $\mathbf{S}$ is diagonal. So, we can get the importance of each filter and the corresponding change in weights are:

$$\Delta\mathcal{L}_i = \frac{1}{2}\mathbf{S}_{ii}\boldsymbol{\theta}_i^{*\top}\mathbf{A}\boldsymbol{\theta}_i^* \quad \text{and} \quad \Delta\boldsymbol{\theta}_i = -\boldsymbol{\theta}_i^* \tag{20}$$

Derivation of Kron-OBS. Under the assumption of Kron-OBS that different filters are correlated to each other, $\mathbf{S}$ is no longer diagonal. Then, similar to eqn. (20), the corresponding structured version of eqn.(9) becomes:

$$\min_i\left\{\min_{\Delta\mathbf{W}}\frac{1}{2}\mathrm{Tr}\left(\Delta\mathbf{W}^\top\mathbf{A}\Delta\mathbf{W}\mathbf{S}\right)\right\} \quad \text{s.t.} \quad \Delta\mathbf{W}\mathbf{e}_i + \boldsymbol{\theta}_i^* = \mathbf{0} \tag{21}$$

We can solve the above constrained optimization problem with Lagrange multiplier:

$$\min_i\left\{\min_{\Delta\mathbf{W}}\frac{1}{2}\mathrm{Tr}\left(\Delta\mathbf{W}^\top\mathbf{A}\Delta\mathbf{W}\mathbf{S}\right) - \boldsymbol{\lambda}^\top\left(\Delta\mathbf{W}\mathbf{e}_i + \boldsymbol{\theta}_i^*\right)\right\} \tag{22}$$

Taking the derivatives w.r.t to $\Delta\mathbf{W}$ and set it to $\mathbf{0}$, we get:

$$\Delta\mathbf{W} = \mathbf{A}^{-1}\boldsymbol{\lambda}\mathbf{e}_i^\top\mathbf{S}^{-1} \tag{23}$$

Substitute it back to the constrain to solve the equation, we get:

$$\boldsymbol{\lambda} = \frac{-\mathbf{A}\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}} \tag{24}$$

Then substitute eqn. (24) back to eqn. (23), we can finally get the optimal change in weights if we remove filter $\mathcal{F}_i$:

$$\Delta\mathbf{W} = -\frac{\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}}\mathbf{e}_i^\top\mathbf{S}^{-1} \quad \text{and} \quad \Delta\boldsymbol{\theta} = -\frac{\mathbf{S}^{-1}\mathbf{e}_i \otimes \boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}} \tag{25}$$

In order to evaluating the importance of each filter, we can substitute eqn. (25) back to eqn. (21):

$$\begin{aligned}\Delta\mathcal{L}_i &= \frac{1}{2}\mathrm{Tr}\left(\mathbf{S}^{-1}\mathbf{e}_i\frac{\boldsymbol{\theta}_i^{*\top}}{[\mathbf{S}^{-1}]_{ii}}\mathbf{A}\frac{\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}}\mathbf{e}_i^\top\mathbf{S}^{-1}\mathbf{S}\right) = \frac{1}{2}\mathrm{Tr}\left(\frac{\boldsymbol{\theta}_i^{*\top}\mathbf{A}\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}^2}\mathbf{S}^{-1}\mathbf{e}_i\mathbf{e}_i^\top\right) \\ &= \frac{1}{2}\mathrm{Tr}\left(\frac{\boldsymbol{\theta}_i^{*\top}\mathbf{A}\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}^2}[\mathbf{S}^{-1}]_{ii}\right) = \frac{1}{2}\frac{\boldsymbol{\theta}_i^{*\top}\mathbf{A}\boldsymbol{\theta}_i^*}{[\mathbf{S}^{-1}]_{ii}}\end{aligned} \tag{26}$$

C. Algorithm for Solving eqn. (18)

In this section, we will introduce the algorithm for solving the optimization problem in eqn. (18).

Khatri-Rao product. The Khatri-Rao product $\odot$ of two matrices $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{n\times r}$ is the column-wise Kronecker product, that is:

$$\mathbf{A}\odot\mathbf{B} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1r}b_{1r} \\ a_{11}b_{21} & a_{12}b_{22} & \cdots & a_{1r}b_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{n1} & a_{m2}b_{n2} & \cdots & a_{mr}b_{nr} \end{pmatrix} \in \mathbb{R}^{mn\times r} \tag{27}$$

Kruskal tensor notation. Suppose $\mathbf{T} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ has low-rank Canonical Polyadic (CP) structure. Following (Bader & Kolda, 2007), we refer to it as a *Kruskal tensor*. Normally, it can be defined by a collection of factor matrices, $\mathbf{A}_k \in \mathbb{R}^{n_k\times r}$ for $k = 1, ..., d$, such that:

$$\mathbf{T}(i_1, i_2, \cdots, i_d) = \sum_{j=1}^{r}\mathbf{A}_1(i_1, j)\mathbf{A}_2(i_2, j)\cdots\mathbf{A}_d(i_d, j) \quad \text{for all } (i_1, i_2, \cdots, i_d) \in \mathcal{I} \tag{28}$$

where $\mathcal{I} \equiv \{1, \cdots, n_1\}\otimes\{1, \cdots n_2\}\otimes\cdots\otimes\{1, \cdots, n_d\}$. Denote $\mathbf{T}_{(k)} \in \mathbb{R}^{n_k\times(n_d\cdots n_{k-1}n_{k+1}\cdots n_1)}$ is the mode-k unfolding of a Kruskal tensor, which has the following form that depends on the Khatri-Rao products of the factor matrices:

$$\mathbf{T}_{(k)} = \mathbf{A}_k\mathbf{Z}_k^\top \quad \text{where } \mathbf{Z}_k \equiv \mathbf{A}_d\odot\cdots\odot\mathbf{A}_{k+1}\odot\mathbf{A}_{k-1}\odot\cdots\odot\mathbf{A}_1 \tag{29}$$

Alternating Least Squares (ALS). We can use ALS to solve problems similar to eqn. (18). Suppose we are approximating $\mathbf{T}$ using $\mathbf{A}_1, \cdots, \mathbf{A}_d$. Specifically, for fixed $\mathbf{A}_1, \cdots, \mathbf{A}_{k-1}, \mathbf{A}_{k+1}, \cdots, \mathbf{A}_d$, there is a closed form solution for $\mathbf{A}_k$. Specifically, we can update update $\mathbf{A}_1, \cdots, \mathbf{A}_d$ by the following update rule:

$$\mathbf{A}_k^\top = \mathbf{Z}_k^\dagger\mathbf{T}_{(k)}^\top \quad \text{for } k = 1, \cdots, d \tag{30}$$

alternatively until converge or reach the maximum number of iterations. For the Mahalanobis norm case (with $\mathbf{F}$ as the metric tensor), if we take the derivative with respect to $\mathbf{A}_k$ to be $\mathbf{0}$,

$$\mathrm{unvec}(\mathbf{F}\mathrm{vec}(\mathbf{A}_k\mathbf{Z}_k^\top - \mathbf{T}_{(k)}))\mathbf{Z} = \mathbf{0} \tag{31}$$

we can get the corresponding update rule for $\mathbf{A}_k$:

$$\mathbf{A}_k^\top = \mathbf{Z}_k^\dagger\left(\mathbf{T}_{(k)} + \mathrm{unvec}(\mathbf{F}^{-1}\mathrm{vec}(\mathbf{P}))\right)^\top \tag{32}$$

where unvec and vec are inverse operators to each other, and in our case, unvec operation is to convert the vectorized matrix back to the original matrix form. $\mathbf{Z}^\dagger = (\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top$ and $\mathbf{P}$ has the same shape with $\mathbf{T}_k$, and for each column $\mathbf{P}_i \in \mathrm{Null}(\mathbf{Z}_k^\top)$.

D. Loss after Pruning and Fine-tuning Curves

We present the loss after pruning in Figure 6, and the finetuning curve of the pruned network in Figure 7. We observe the network pruned by EigenDamage can achieve much lower training loss than other methods without finetuning, and thus will need less epochs in the finetuning stage.

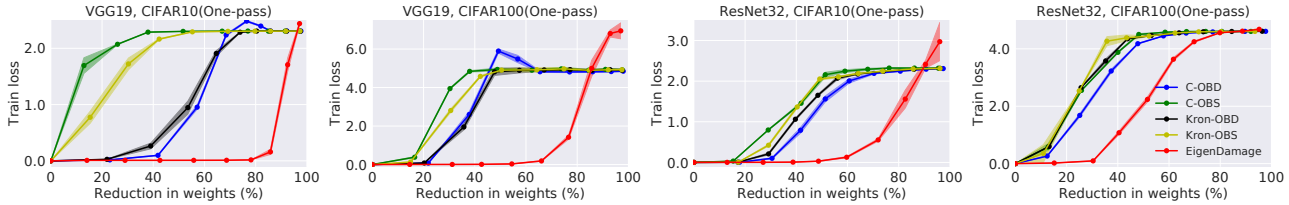

Figure 6. The above four figures show the training loss after one-pass pruning (without finetuning) vs. reduction in weights. The network pruned by EigenDamage achieves significant lower loss on the training set, which shows pruning in KFE is very accurate in reflecting the sensitivity of weight to loss.

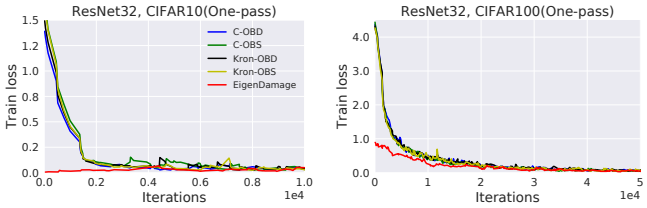

Figure 7. Loss when finetune the network after pruning (with a ratio of 0.5) with ResNet32 on CIFAR10 and CIFAR100 datasets.

E. Additional Results on Iterative Pruning

The following figure (Figure 8) shows the results of iterative pruning obtained by different methods. Almost all the methods can achieve close results on VGGNet, but EigenDamage outperforms others by a significant margin on ResNet32, which is a more complicated and compact network.

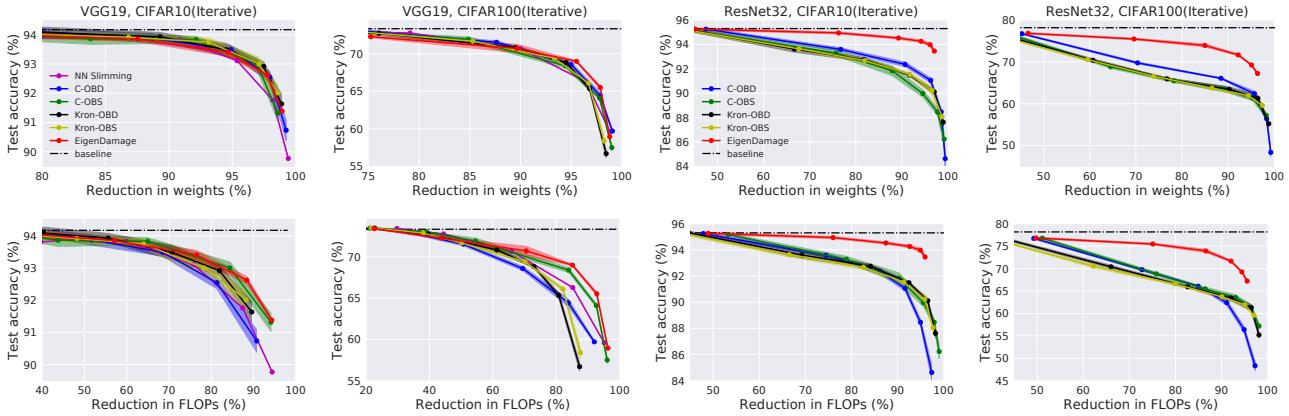

Figure 8. The results of iterative pruning. The first row are the curves of reduction in weights vs. test accuracy, and second row are the curves of pruned FLOPs vs. test accuracy of VGGNet and ResNet trained on CIFAR10 and CIFAR100 dataset. The shaded areas represent the variance over five runs.

F. Additional Results on One-pass Pruning

We present the additional results on one-pass pruning in the following tables. We also present the data in tables as trade-off curves in terms of acc vs. reduction in weight and acc vs. reduction in FLOPs for making it easy to tell the difference in performances of each method.

Table 3. One pass pruning on CIFAR-10 with VGG19

Prune Ratio (%)	50%			70%			80%		
Method	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)
VGG19(Baseline)	94.17	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	92.84 ± -	73.84 ± -	38.88 ± -	92.89 ± -	84.30 ± -	54.83 ± -	91.92 ± -	91.77 ± -	76.43 ± -
C-OBD	94.01 ± 0.15	76.84 ± 0.30	35.07 ± 0.38	94.04 ± 0.09	85.88 ± 0.10	41.17 ± 0.23	93.70 ± 0.07	92.17 ± 0.07	56.87 ± 0.33
C-OBS	**94.19 ± 0.10**	66.91 ± 0.08	26.12 ± 0.13	93.97 ± 0.16	84.97 ± 0.02	43.16 ± 0.20	93.77 ± 0.12	91.52 ± 0.09	63.64 ± 0.13
Kron-OBD	93.91 ± 0.16	73.93 ± 0.42	33.71 ± 0.69	93.95 ± 0.12	85.80 ± 0.09	43.78 ± 0.24	93.78 ± 0.17	92.04 ± 0.04	60.81 ± 0.26
Kron-OBS	94.03 ± 0.13	69.17 ± 0.20	28.02 ± 0.26	94.10 ± 0.15	85.83 ± 0.09	42.56 ± 0.14	**93.87 ± 0.14**	92.00 ± 0.04	60.19 ± 0.38
EigenDamage	94.15 ± 0.05	68.64 ± 0.19	28.09 ± 0.21	**94.15 ± 0.14**	85.78 ± 0.06	45.68 ± 0.31	93.68 ± 0.22	92.51 ± 0.05	66.98 ± 0.36
VGG19+L_1 (Baseline)	93.71	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	93.79 ± -	77.44 ± -	45.19 ± -	93.74 ± -	88.81 ± -	52.15 ± -	**93.48 ± -**	92.60 ± -	62.23 ± -
C-OBD	93.85 ± 0.03	76.83 ± 0.01	41.14 ± 0.05	93.88 ± 0.03	89.04 ± 0.03	52.73 ± 0.14	93.38 ± 0.04	93.29 ± 0.05	63.74 ± 0.12
C-OBS	**93.88 ± 0.04**	74.95 ± 0.03	37.56 ± 0.06	93.84 ± 0.04	88.53 ± 0.20	51.88 ± 0.00	93.27 ± 0.04	92.01 ± 0.02	63.96 ± 0.10
Kron-OBD	**93.88 ± 0.01**	83.43 ± 0.00	49.58 ± 0.00	**93.89 ± 0.03**	89.02 ± 0.01	53.40 ± 0.09	93.33 ± 0.05	93.55 ± 0.06	67.01 ± 0.30
Kron-OBS	93.85 ± 0.03	76.95 ± 0.01	42.04 ± 0.10	93.88 ± 0.04	88.69 ± 0.02	52.38 ± 0.08	93.44 ± 0.07	92.66 ± 0.05	63.77 ± 0.27
EigenDamage	93.84 ± 0.04	78.14 ± 0.11	39.02 ± 0.30	93.85 ± 0.04	85.71 ± 0.01	46.56 ± 0.03	93.40 ± 0.07	91.48 ± 0.06	62.18 ± 0.29

Table 4. One pass pruning on CIFAR-100 with VGG19

Prune Ratio (%)	50%			70%			80%		
Method	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)
VGG19(Baseline)	73.34	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	72.77 ± -	66.50 ± -	30.61 ± -	69.98 ± -	85.56 ± -	54.51 ± -	66.09 ± -	92.33 ± -	76.76 ± -
C-OBD	72.82 ± 0.15	65.47 ± 0.13	24.24 ± 0.10	71.10 ± 0.22	86.06 ± 0.04	41.18 ± 0.04	67.46 ± 0.26	93.31 ± 0.06	60.39 ± 0.25
C-OBS	72.73 ± 0.17	62.31 ± 0.05	25.50 ± 0.06	71.25 ± 0.21	84.49 ± 0.04	49.25 ± 0.52	67.47 ± 0.13	91.04 ± 0.06	68.38 ± 0.26
Kron-OBD	72.88 ± 0.12	67.11 ± 0.21	28.57 ± 0.19	71.16 ± 0.11	85.83 ± 0.10	47.19 ± 0.35	67.70 ± 0.32	92.86 ± 0.05	65.26 ± 0.26
Kron-OBS	72.89 ± 0.12	67.26 ± 0.08	25.80 ± 0.16	71.36 ± 0.17	84.75 ± 0.02	45.74 ± 0.17	68.17 ± 0.34	92.16 ± 0.03	63.95 ± 0.19
EigenDamage	**73.39 ± 0.12**	66.05 ± 0.11	28.55 ± 0.11	**71.62 ± 0.14**	85.69 ± 0.05	54.83 ± 0.79	**69.50 ± 0.22**	92.92 ± 0.03	74.55 ± 0.33
VGG19+L_1 (Baseline)	73.08	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	73.24 ± -	72.68 ± -	35.37 ± -	71.55 ± -	84.38 ± -	51.59 ± -	66.55 ± -	92.48 ± -	76.54 ± -
C-OBD	73.39 ± 0.05	74.16 ± 0.01	35.13 ± 0.01	71.40 ± 0.09	86.13 ± 0.01	45.47 ± 0.10	67.56 ± 0.16	93.00 ± 0.01	63.42 ± 0.11
C-OBS	**73.44 ± 0.04**	71.17 ± 0.03	33.77 ± 0.67	71.30 ± 0.12	84.07 ± 0.01	56.74 ± 0.13	66.90 ± 0.23	91.20 ± 0.04	73.39 ± 0.31
Kron-OBD	73.24 ± 0.05	74.00 ± 0.03	36.56 ± 0.03	71.01 ± 0.13	86.66 ± 0.05	52.66 ± 0.21	67.24 ± 0.20	92.90 ± 0.05	68.62 ± 0.21
Kron-OBS	73.20 ± 0.12	72.27 ± 0.03	36.45 ± 0.66	**71.88 ± 0.11**	84.77 ± 0.01	50.53 ± 0.08	67.75 ± 0.14	92.08 ± 0.01	67.39 ± 0.17
EigenDamage	73.23 ± 0.08	66.80 ± 0.02	29.49 ± 0.03	71.81 ± 0.13	84.27 ± 0.04	52.75 ± 0.21	**69.83 ± 0.24**	92.36 ± 0.01	73.68 ± 0.13

Table 5. One pass pruning on CIFAR-10 with ResNet

Prune Ratio (%)	50%			70%			80%		
Method	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)
ResNet32(Baseline)	95.30	-	-	-	-	-	-	-	-
C-OBD	95.27 ± 0.10	60.67 ± 0.44	55.46 ± 0.35	**95.00 ± 0.17**	80.64 ± 0.40	76.78 ± 0.63	94.41 ± 0.10	90.73 ± 0.11	86.66 ± 0.34
C-OBS	95.30 ± 0.15	58.99 ± 0.02	65.54 ± 0.25	94.43 ± 0.17	76.27 ± 0.35	85.89 ± 0.23	93.45 ± 0.25	86.15 ± 0.68	92.77 ± 0.05
Kron-OBD	95.30 ± 0.09	56.05 ± 0.24	52.21 ± 0.36	94.94 ± 0.02	73.98 ± 0.46	74.97 ± 0.63	**94.60 ± 0.14**	85.96 ± 0.41	86.36 ± 0.38
Kron-OBS	**95.46 ± 0.08**	56.48 ± 0.26	50.93 ± 0.46	94.92 ± 0.11	73.77 ± 0.24	74.58 ± 0.44	94.44 ± 0.08	85.65 ± 0.46	86.05 ± 0.55
EigenDamage	95.28 ± 0.16	59.68 ± 0.28	58.32 ± 0.23	94.86 ± 0.11	82.57 ± 0.27	80.88 ± 0.36	94.23 ± 0.13	90.48 ± 0.35	88.86 ± 0.50
PreResNet29+L_1 (Baseline)	94.42	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	92.60 ± -	58.12 ± -	69.88 ± -	90.60 ± -	81.99 ± -	87.05 ± -	87.24 ± -	86.68 ± -	91.33 ± -
C-OBD	92.85 ± 0.14	79.67 ± 0.41	68.90 ± 0.49	88.74 ± 0.32	93.22 ± 0.08	86.37 ± 0.32	85.75 ± 0.66	96.14 ± 0.04	91.39 ± 0.16
C-OBS	92.43 ± 0.03	73.81 ± 0.14	76.33 ± 0.21	85.85 ± 0.36	91.74 ± 0.17	92.39 ± 0.07	78.97 ± 0.84	96.55 ± 0.02	96.60 ± 0.06
Kron-OBD	93.19 ± 0.06	59.72 ± 0.31	54.26 ± 0.34	87.99 ± 0.37	86.50 ± 0.05	78.96 ± 0.17	86.40 ± 0.28	95.02 ± 0.04	88.58 ± 0.04
Kron-OBS	92.88 ± 0.08	58.95 ± 0.14	57.61 ± 0.12	87.71 ± 0.22	85.38 ± 0.03	82.00 ± 0.10	85.67 ± 0.31	93.93 ± 0.07	90.74 ± 0.31
EigenDamage	**94.15 ± 0.07**	62.06 ± 0.15	54.40 ± 0.10	**93.33 ± 0.07**	77.71 ± 0.11	71.92 ± 0.15	**92.30 ± 0.15**	86.27 ± 0.04	81.59 ± 0.07

Table 6. One pass pruning on CIFAR-100 with ResNet

Prune Ratio (%)	50%			70%			80%		
Method	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)
ResNet32(Baseline)	78.17	-	-	-	-	-	-	-	-
C-OBD	76.59 ± 0.06	57.82 ± 0.48	56.88 ± 0.37	73.74 ± 0.42	77.08 ± 0.32	78.64 ± 0.32	68.86 ± 0.40	89.67 ± 0.19	88.45 ± 0.19
C-OBS	76.26 ± 0.25	58.47 ± 0.22	63.81 ± 0.26	73.06 ± 0.23	74.33 ± 0.15	86.06 ± 0.03	63.15 ± 0.41	80.61 ± 0.15	91.10 ± 0.06
Kron-OBD	76.41 ± 0.29	53.08 ± 0.35	52.06 ± 0.27	72.82 ± 0.28	73.40 ± 0.11	75.25 ± 0.15	69.62 ± 0.38	84.50 ± 0.16	88.04 ± 0.10
Kron-OBS	**76.74 ± 0.32**	52.21 ± 0.20	49.40 ± 0.13	73.00 ± 0.19	71.60 ± 0.15	73.33 ± 0.48	70.42 ± 0.20	80.34 ± 0.25	86.96 ± 0.27
EigenDamage	76.12 ± 0.12	61.73 ± 0.12	60.87 ± 0.38	**73.82 ± 0.07**	79.85 ± 0.07	81.03 ± 0.11	**70.78 ± 0.08**	88.68 ± 0.08	88.68 ± 0.06
PreResNet29+L_1 (Baseline)	75.70	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	71.93 ± -	47.55 ± -	73.44 ± -	63.47 ± -	78.41 ± -	89.92 ± -	61.64 ± -	85.47 ± -	92.38 ± -
C-OBD	67.18 ± 0.10	84.72 ± 0.05	75.48 ± 0.11	55.02 ± 0.50	93.42 ± 0.01	88.12 ± 0.06	47.87 ± 0.57	96.57 ± 0.01	93.48 ± 0.02
C-OBS	71.17 ± 0.11	65.67 ± 0.12	80.77 ± 0.05	51.45 ± 0.74	91.88 ± 0.06	95.00 ± 0.03	41.55 ± 0.91	95.64 ± 0.02	97.20 ± 0.01
Kron-OBD	69.64 ± 0.19	56.63 ± 0.20	58.24 ± 0.05	46.46 ± 0.72	89.50 ± 0.07	82.44 ± 0.05	41.64 ± 0.86	95.83 ± 0.01	89.63 ± 0.06
Kron-OBS	69.87 ± 0.17	49.09 ± 0.13	62.59 ± 0.05	49.00 ± 0.68	87.03 ± 0.05	87.82 ± 0.10	40.51 ± 1.04	94.72 ± 0.01	93.86 ± 0.01
EigenDamage	**74.50 ± 0.13**	57.98 ± 0.15	53.87 ± 0.14	**72.41 ± 0.16**	75.79 ± 0.04	71.92 ± 0.12	**70.09 ± 0.11**	85.34 ± 0.02	81.61 ± 0.06

Table 7. One pass pruning on Tiny-ImageNet with VGG19. N/A denotes the network failed to converge, and achieves random guess performance on the test set.

Prune Ratio (%)	40%			60%			70%			80%		
Method	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)	Test acc (%)	Reduction in weights (%)	Reduction in FLOPs (%)
VGG19(Baseline)	61.56	-	-	-	-	-	-	-	-	-	-	-
NN Slimming (Liu et al., 2017)	57.40 ± -	52.61 ± -	76.99 ± -	40.05 ± -	71.04 ± -	90.04 ± -	N/A ± -	84.63 ± -	94.09 ± -	N/A ± -	93.02 ± -	95.45 ± -
C-OBD	56.87 ± 0.13	58.95 ± 0.06	55.49 ± 0.50	47.36 ± 0.47	79.10 ± 0.32	69.74 ± 0.38	43.61 ± 0.31	88.57 ± 0.14	74.90 ± 0.31	42.29 ± 0.53	95.62 ± 0.13	78.45 ± 0.46
C-OBS	56.72 ± 0.21	46.90 ± 0.18	68.87 ± 0.14	39.80 ± 0.53	67.46 ± 0.34	86.81 ± 0.25	35.30 ± 0.33	76.47 ± 0.07	92.71 ± 0.07	31.52 ± 0.66	86.19 ± 0.07	96.66 ± 0.04
Kron-OBD	56.81 ± 0.22	56.75 ± 0.27	66.96 ± 0.65	44.41 ± 0.82	76.55 ± 0.15	81.27 ± 0.16	41.03 ± 0.50	85.28 ± 0.11	86.12 ± 0.05	38.88 ± 0.43	95.02 ± 0.33	90.98 ± 0.34
Kron-OBS	56.47 ± 0.20	50.67 ± 0.11	62.28 ± 0.66	44.54 ± 0.43	73.88 ± 0.10	83.27 ± 0.13	41.44 ± 0.41	82.61 ± 0.41	87.91 ± 0.22	39.54 ± 0.20	92.77 ± 0.19	91.91 ± 0.33
EigenDamage	**59.09 ± 0.05**	48.62 ± 0.06	57.30 ± 0.12	**56.92 ± 0.23**	74.12 ± 0.15	74.37 ± 0.13	**54.46 ± 0.32**	83.77 ± 0.02	81.19 ± 0.16	**51.34 ± 0.37**	91.05 ± 0.06	87.82 ± 0.16

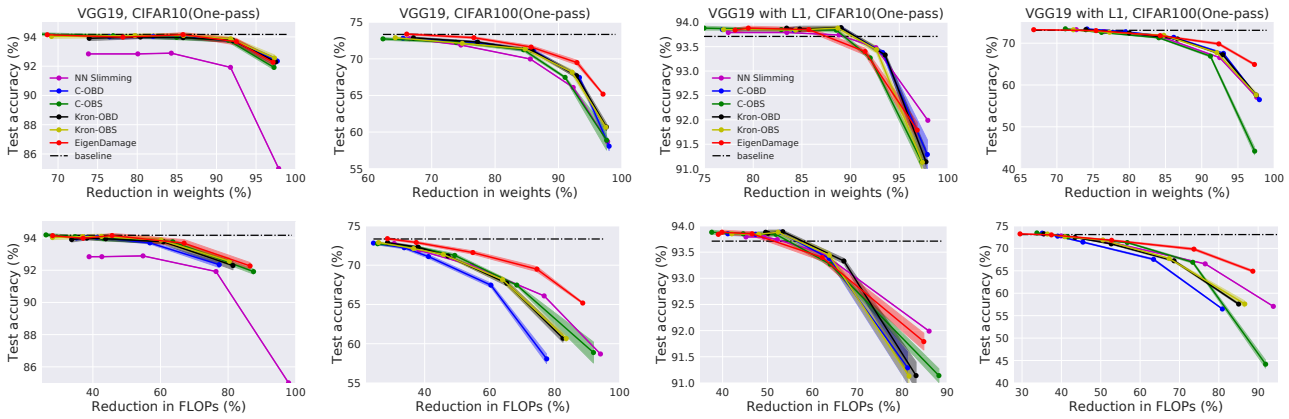

Figure 9. The results of one pass pruning, which are plotted based on the results in Tables. The first row are the curves of reduction in weights vs. test accuracy, and second row are the curves of pruned FLOPs vs. test accuracy of VGGNet trained on CIFAR10 and CIFAR100 dataset under the settings of with and without L_1 sparsity on BatchNorm. The shaded areas represent the standard variance over five runs.

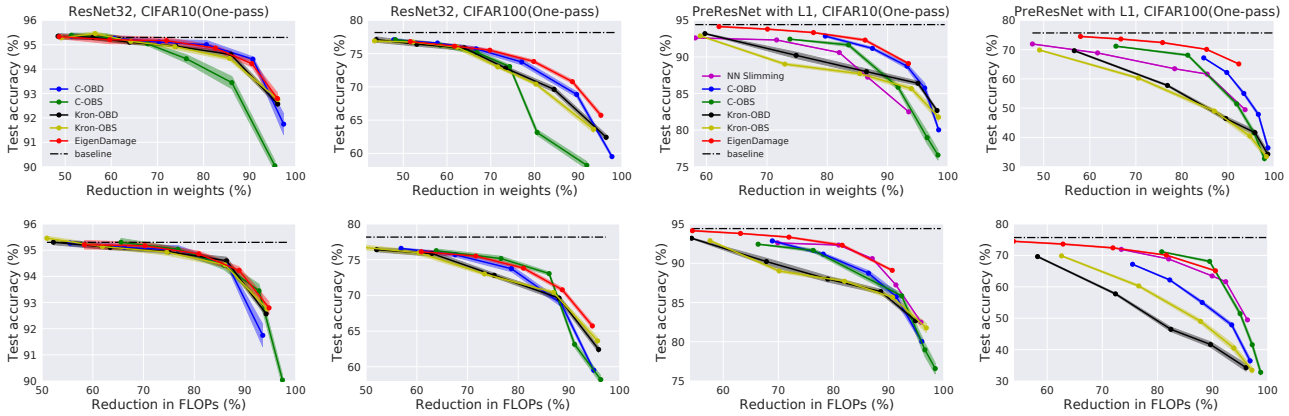

Figure 10. The results of one pass pruning, which are plotted based on the results in Tables. The first row are the curves of reduction in weights vs. test accuracy, and second row are the curves of pruned FLOPs vs. test accuracy of (Pre)ResNet trained on CIFAR10 and CIFAR100 dataset under the settings of with and without L_1 sparsity on BatchNorm. The shaded areas represent the standard variance over five runs.

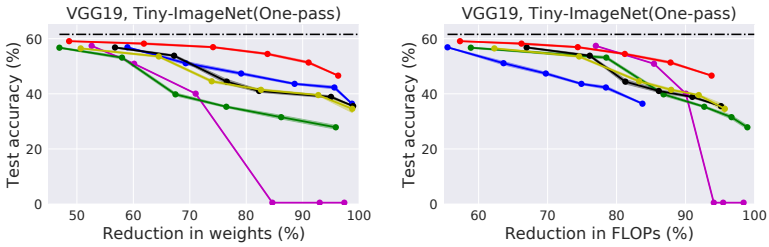

Figure 11. The results of one pass pruning, which are plotted based on the results in Tables. The base network for NN Slimming is pre-trained with L_1 sparsity on BatchNorm as required, and the others are normally pre-trained.