EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis

Chaoqi Wang, Roger Grosse, Sanja Fidler and Guodong Zhang

University of Toronto, Vector Institute

Jun 12, 2019

Structured Pruning

Structured Pruning:

- Prunes filters or neurons.
- GPU-friendly.

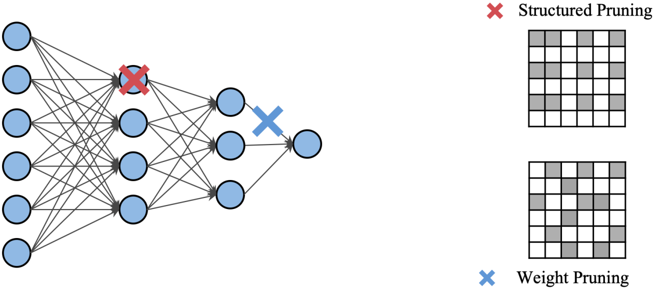


Figure 1: An illustration of structured pruning.

Background: Hessian-Based Pruning Methods

Hessian-based methods:

1. The pruning criteria is calibrated across layers,
2. Automatically determines the network structure,
3. Fewer hyper-parameters required. (Only the pruning ratio)

It relies on the Taylor expansion around the minimum $\boldsymbol{\theta}^*$, and directly approximates the effect on the loss when removing a given weight $\Delta\boldsymbol{\theta}$,

$$\Delta\mathcal{L} = \cancel{\underbrace{\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}^\top \Delta\boldsymbol{\theta}}_{\approx 0}} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top \mathbf{H}\Delta\boldsymbol{\theta} + \cancel{\mathcal{O}(\|\Delta\boldsymbol{\theta}\|^3)} \tag{1}$$

Background: Hessian-Based Pruning Methods

Two representative methods Optimal Brain Damage (OBD) and Optimal Brain Surgeon (OBS),:

- OBD uses a *diagonal Hessian* for fast computation, and computes the importance of each weight $\boldsymbol{\theta}_q^*$ by:

$$\Delta\mathcal{L}_{\mathrm{OBD}} = \frac{1}{2}(\boldsymbol{\theta}_q^*)^2\mathbf{H}_{qq} \tag{2}$$

- **OBS** uses the *full Hessian* for accounting the correlations, and computes the importance of each weight $\boldsymbol{\theta}_q^*$ by:

$$\Delta\mathcal{L}_{\mathrm{OBS}} = \frac{1}{2}\frac{(\boldsymbol{\theta}_q^*)}{[\mathbf{H}^{-1}]_{qq}} \tag{3}$$

Is OBS always better than OBD?

In the original paper, OBS is guaranteed to be better than OBD when pruning weights **one by one** (*i.e.* recompute the Hessian for each iteration).

But in practice, we will prune **multiple weights at a time**.

Is OBS always better than OBD?

We would like to ask:

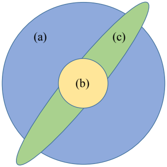
Is OBS always better than OBD
for pruning multiple weights at a time?

At the first glance.... Yes? Because OBS uses full Hessian, while OBD only uses diagonal Hessian.

Bayesian Interpretations

Surprisingly, **No**! Even if we can compute the exact Hessian.

Bayesian Interpretations of OBD and OBS:



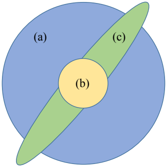
Both OBS and OBD are using a **factorial Gaussian** to approximate the highly coupled weight posterior (c) **under different objectives**:

-
-

Bayesian Interpretations

Surprisingly, **No**! Even if we can compute the exact Hessian.

Bayesian Interpretations of OBD and OBS:



Both OBS and OBD are using a **factorial Gaussian** to approximate the highly coupled weight posterior (c) **under different objectives**:

- OBD: Reverse KL divergence (b). (Too pessimistic)
- OBS: Forward KL divergence (a). (Too optimistic)

and neither of them will necessarily be better than the other.

More details in the Paper and Poster#22!

Method

OBD and OBS use *diagonal Hessian* and *diagonal Hessian inverse* for pruning. Both of them fail to capture the correlations when pruning multiple weights at a time.

Issues:

1.
2.

Method

OBD and OBS use *diagonal Hessian* and *diagonal Hessian inverse* for pruning. Both of them fail to capture the correlations when pruning multiple weights at a time.

Solution: Pruning in a new coordinate system (*i.e.* a new basis), in which the posterior is closer to factorial!

Issues:

1.
2.

Method

OBD and OBS use *diagonal Hessian* and *diagonal Hessian inverse* for pruning. Both of them fail to capture the correlations when pruning multiple weights at a time.

Solution: Pruning in a new coordinate system (*i.e.* a new basis), in which the posterior is closer to factorial!

The new basis ideally would be the eigenbasis of the Hessian! But..

Issues:

1. Exact Hessian is intractable for large neural networks.
2. The new basis will introduce extra parameters.

Approximating Hessian with K-FAC Fisher

1. **Fisher Information Matrix (FIM)** is commonly adopted for approximating the Hessian:

$$\mathbf{F} = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta})^\top] \tag{4}$$

2.

3.

Approximating Hessian with K-FAC Fisher

1. **Fisher Information Matrix (FIM)** is commonly adopted for approximating the Hessian:

$$\mathbf{F} = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta})^\top] \tag{7}$$

2. **K-FAC** decomposes the FIM of a neural network into the Kronecker product of two small matrices under the independence assumption:

$$\mathbf{F} = \mathbb{E}[\mathcal{D}\mathbf{s}\mathcal{D}\mathbf{s}^\top \otimes \mathbf{a}\mathbf{a}^\top] \approx \mathbb{E}[\mathcal{D}\mathbf{s}\mathcal{D}\mathbf{s}^\top] \otimes \mathbb{E}[\mathbf{a}\mathbf{a}^\top] = \mathbf{S} \otimes \mathbf{A} \tag{8}$$

3.

Approximating Hessian with K-FAC Fisher

1. **Fisher Information Matrix (FIM)** is commonly adopted for approximating the Hessian:

$$\mathbf{F} = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(y|\mathbf{x};\boldsymbol{\theta})^\top] \tag{10}$$

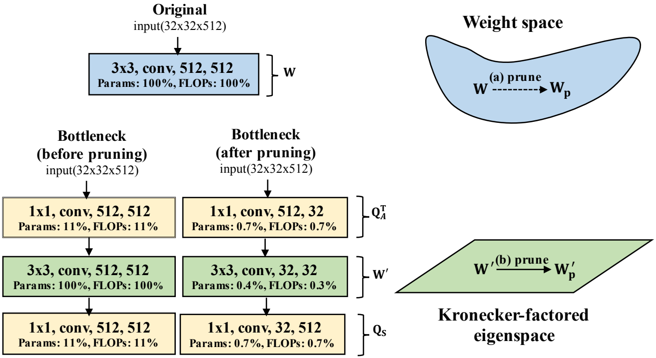
2. **K-FAC** decomposes the FIM of a neural network into the Kronecker product of two small matrices under the independence assumption:

$$\mathbf{F} = \mathbb{E}[\mathcal{D}\mathbf{s}\mathcal{D}\mathbf{s}^\top \otimes \mathbf{a}\mathbf{a}^\top] \approx \mathbb{E}[\mathcal{D}\mathbf{s}\mathcal{D}\mathbf{s}^\top] \otimes \mathbb{E}[\mathbf{a}\mathbf{a}^\top] = \mathbf{S} \otimes \mathbf{A} \tag{11}$$

3. **Kronecker-Factored Eigenbasis** (KFE):

$$\begin{aligned} \mathbf{F} &\approx (\mathbf{Q_S}\Lambda_\mathbf{S}\mathbf{Q_S}^\top) \otimes (\mathbf{Q_A}\Lambda_\mathbf{A}\mathbf{Q_A}^\top) \\ &= \underbrace{(\mathbf{Q_S} \otimes \mathbf{Q_A})}_{\text{KFE}}(\Lambda_\mathbf{S} \otimes \Lambda_\mathbf{A})(\mathbf{Q_S} \otimes \mathbf{Q_A})^\top \end{aligned} \tag{12}$$

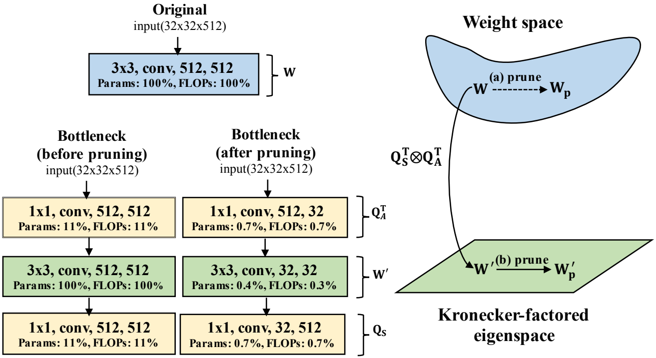
EigenDamage: Structured Pruning in the KFE



Rotate the weights to the KFE by:

$$\mathrm{vec}\{\mathbf{W}\} = (\mathbf{Q_S} \otimes \mathbf{Q_A})^\top \mathrm{vec}(\mathbf{W}') = \mathrm{vec}\{\mathbf{Q_A}^\top \mathbf{W}' \mathbf{Q_S}\} \tag{13}$$

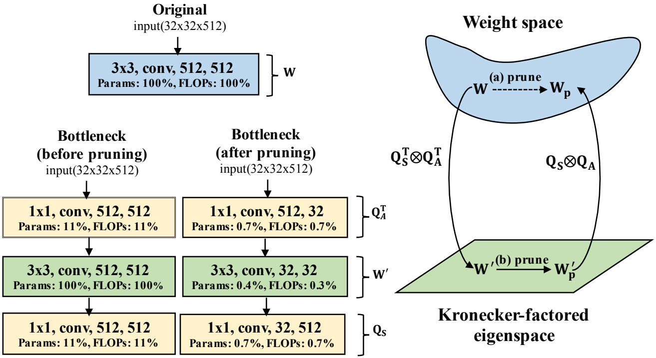
EigenDamage: Structured Pruning in the KFE



Rotate the weights to the KFE by:

$$\mathrm{vec}\{\mathbf{W}\} = (\mathbf{Q_S} \otimes \mathbf{Q_A})^\top \mathrm{vec}(\mathbf{W}') = \mathrm{vec}\{\mathbf{Q}_\mathbf{A}^\top \mathbf{W}' \mathbf{Q_S}\} \qquad (14)$$

EigenDamage: Structured Pruning in the KFE



Rotate the weights to the KFE by:

$$\mathrm{vec}\{\mathbf{W}\} = (\mathbf{Q_S} \otimes \mathbf{Q_A})^\top \mathrm{vec}(\mathbf{W}') = \mathrm{vec}\{\mathbf{Q}_\mathbf{A}^\top \mathbf{W}' \mathbf{Q_S}\} \qquad (15)$$

EigenDamage: Structured Pruning in the KFE

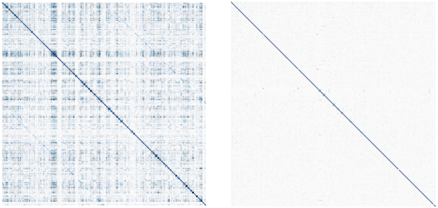
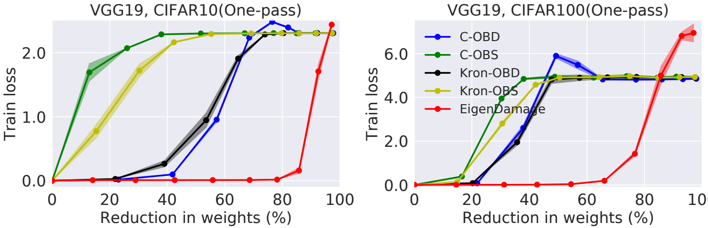


Figure 2: The Fisher matrices in the original weight coordinate and in the KFE.

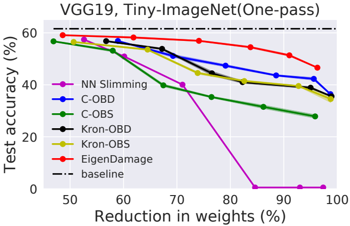
Compared to the Fisher in the original coordinate, the Fisher in KFE is approximately diagonal, and thus the weights are closer to be independent to each other.

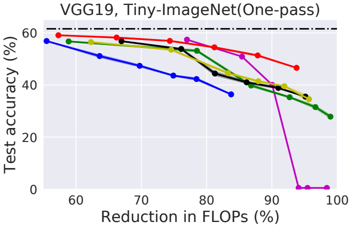
Pruning in the KFE is More Accurate



The network pruned by EigenDamage achieves significantly lower training error than others. (without fine-tuning!)

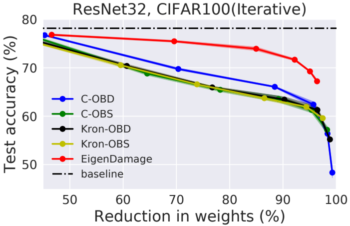
One-pass Pruning Results

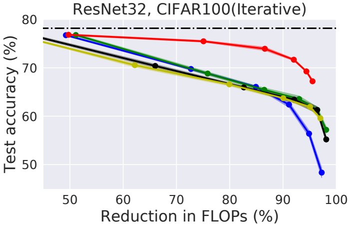




EigenDamage outperforms other methods by a significant margin, especially for high pruning ratios, *e.g.* $\geq 90\%$.

Iterative Pruning Results





Iterative pruning can yield more accurate pruning results. EigenDamage again outperforms other methods by an even more significant margin.

Poster Session

Poster Session:
Today 06:30 – 09:00 PM
Pacific Ballroom #22

Code available at:
https://github.com/alecwangcq/EigenDamage-Pytorch