Progressive Skeletonization: Trimming more fat from a network at initialization

Pau de Jorge, Amartya Sanyal, Harkirat Behl,
Philip Torr, Grégory Rogez and Puneet Dokania

Code: https://github.com/naver/force

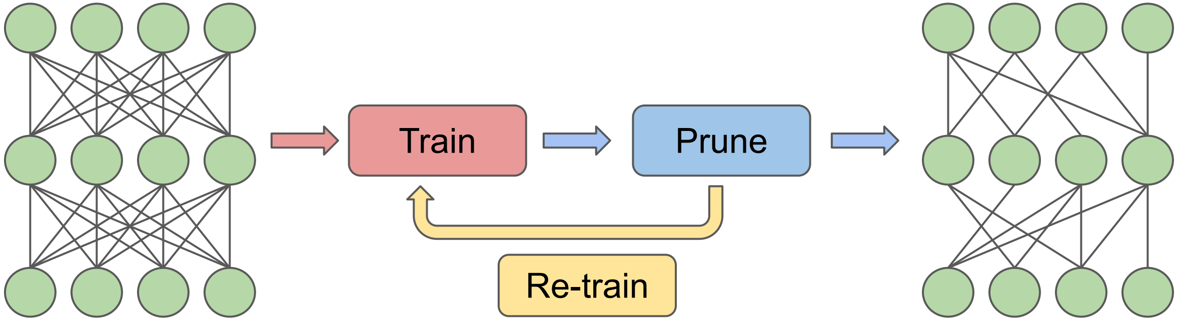
ICLR 2021

FIVE

UNIVERSITY OF OXFORD

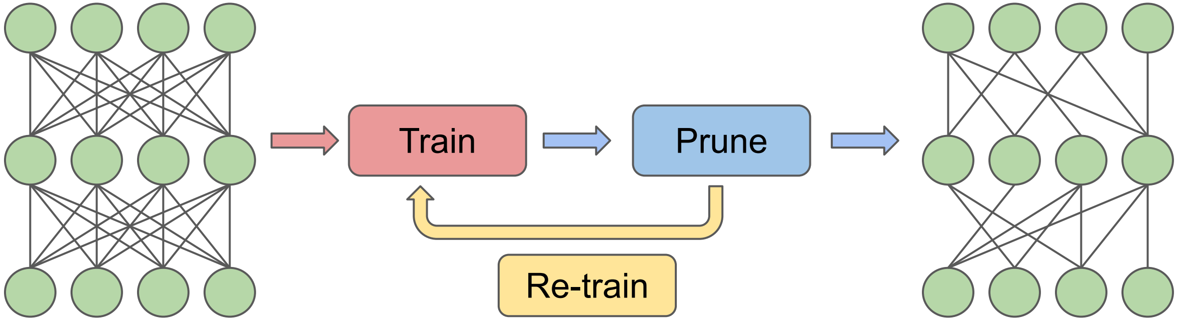
NAVER LABS Europe

Network Pruning

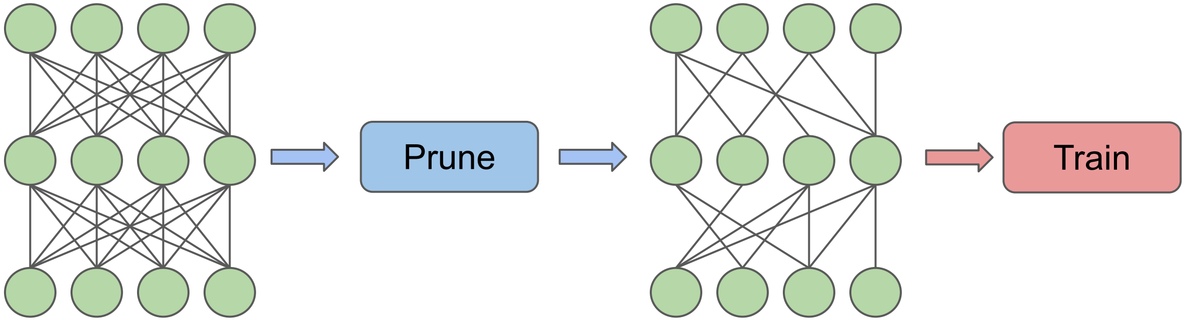


Han *et al.* 2015
Renda *et al.* 2020

Network Pruning



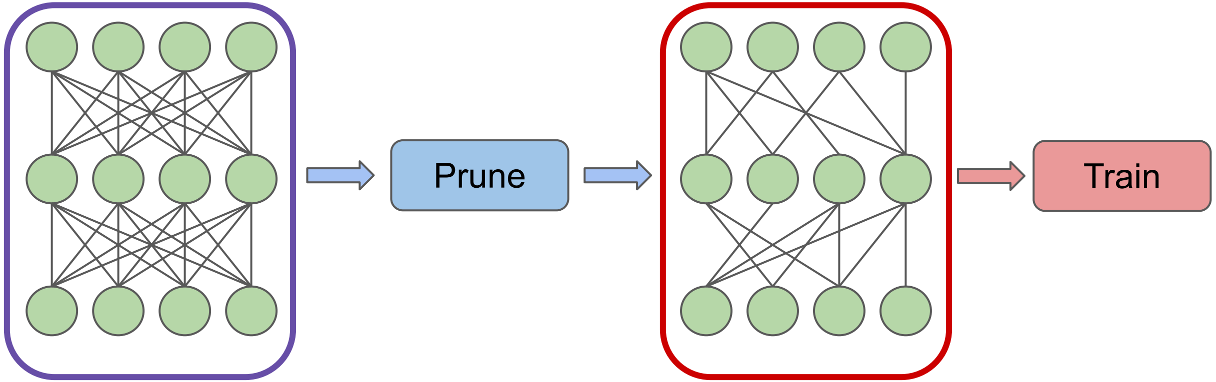
Han *et al.* 2015
Renda *et al.* 2020



Lee *et al.* 2019
Wang *et al.* 2020

Pruning at initialization

Problem: Given a randomly initialized network and sparsity level, we want to find a sub-network such that, after training, we obtain maximum accuracy.



Lee *et al.* 2019
Wang *et al.* 2020

Solving this optimization problem would require training all possible sub-networks!

Related work (SNIP)

SNIP (Lee *et al.* 2019) use the *Connection Sensitivity*:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

- Intuitively, it measures how sensitive is the loss to perturbing each connection.
- **However**, they measure this effect *prior* to pruning!

Related work (SNIP)

SNIP (Lee *et al.* 2019) use the *Connection Sensitivity*:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boxed{\boldsymbol{c}})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \;\; \|\boldsymbol{c}\|_0 = k.$$

- Intuitively, it measures how sensitive is the loss to perturbing each connection.
- **However**, they measure this effect *prior* to pruning!

Related work (SNIP)

SNIP (Lee *et al.* 2019) use the *Connection Sensitivity*:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \boxed{\sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta})_i|} \text{ s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

- Intuitively, it measures how sensitive is the loss to perturbing each connection.
- **However**, they measure this effect *prior* to pruning!

Related work (GRASP)

GRASP (Wang *et al.* 2020) use the Gradient Norm:

$$\boxed{\Delta\mathcal{L}(\boldsymbol{\theta}) = \nabla\mathcal{L}(\boldsymbol{\theta})^T\nabla\mathcal{L}(\boldsymbol{\theta})}$$

$$S(\boldsymbol{\delta}) = \Delta\mathcal{L}(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \underbrace{\Delta\mathcal{L}(\boldsymbol{\theta}_0)}_{\text{Const}} = 2\boldsymbol{\delta}^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_0)\nabla\mathcal{L}(\boldsymbol{\theta}_0) + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$$

$$= 2\boldsymbol{\delta}^\top\mathbf{H}\mathbf{g} + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$$

- GRASP suggest pruning weights by maximizing the gradient norm.
- **However**, they assume pruning is a *perturbation* of the weights!

Related work (GRASP)

GRASP (Wang *et al.* 2020) use the Gradient Norm:

$$\Delta\mathcal{L}(\boldsymbol{\theta}) = \nabla\mathcal{L}(\boldsymbol{\theta})^T\nabla\mathcal{L}(\boldsymbol{\theta})$$

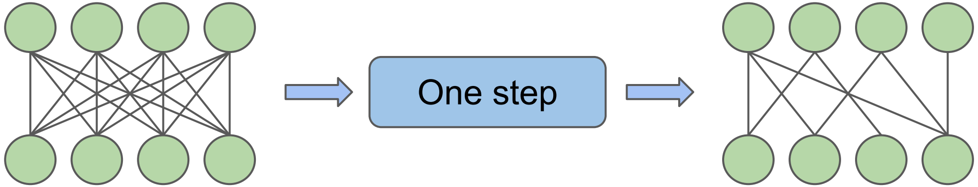
$$S(\boldsymbol{\delta}) = \Delta\mathcal{L}(\boldsymbol{\theta}_0 \boxed{+\, \boldsymbol{\delta})} - \underbrace{\Delta\mathcal{L}(\boldsymbol{\theta}_0)}_{\text{Const}} = 2\boldsymbol{\delta}^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}_0)\nabla\mathcal{L}(\boldsymbol{\theta}_0) + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$$

$$= 2\boldsymbol{\delta}^\top\mathbf{H}\mathbf{g} + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$$

- GRASP suggest pruning weights by maximizing the gradient norm.
- **However**, they assume pruning is a *perturbation* of the weights!

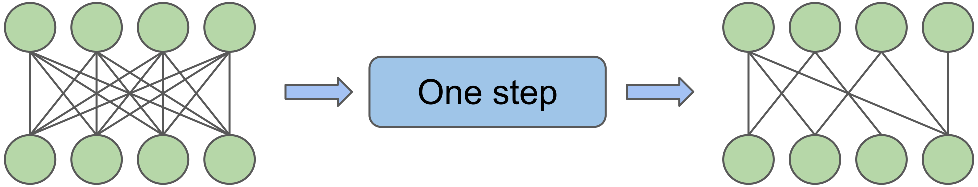
Motivation

SNIP and GRASP assume removing one weight is **independent** of other weights.

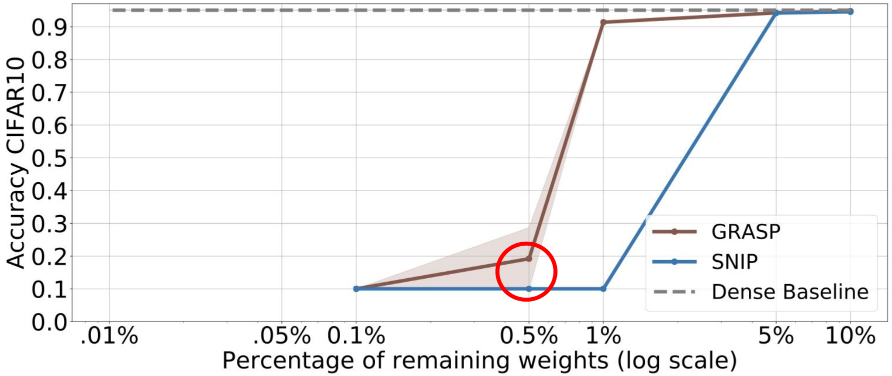


Motivation

SNIP and GRASP assume removing one weight is **independent** of other weights.

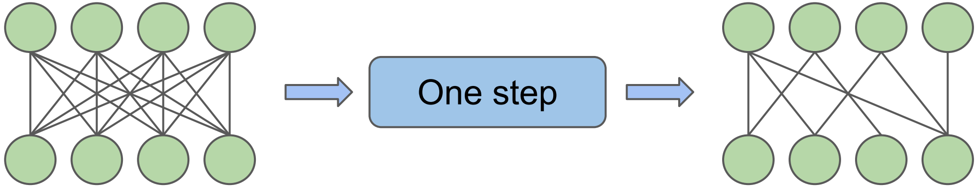


We argue this leads to collapse in performance for high sparsities

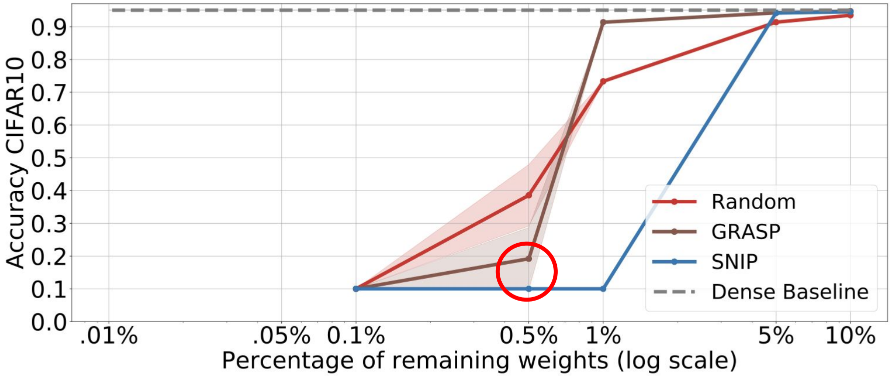


Motivation

SNIP and GRASP assume removing one weight is **independent** of other weights.

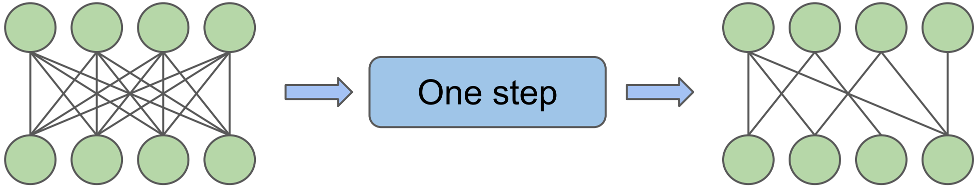


We argue this leads to collapse in performance for high sparsities

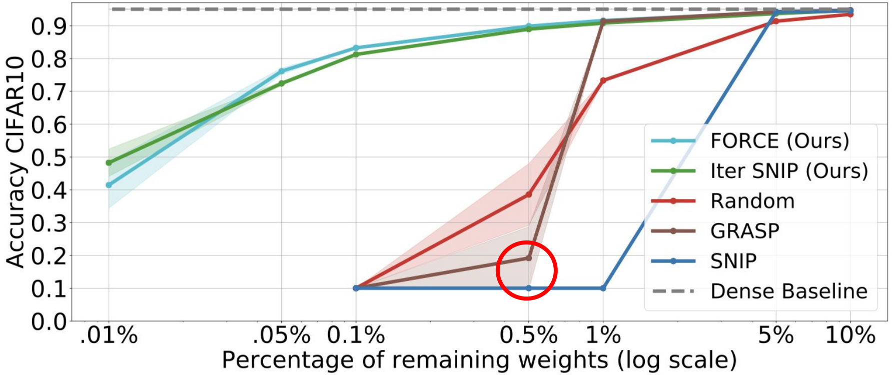


Motivation

SNIP and GRASP assume removing one weight is **independent** of other weights.



We argue this leads to collapse in performance for high sparsities



Method

Foresight Connection Sensitivity

FOResight Connection sEnsitivity (FORCE) evaluates connections *after* pruning.

Connection Sensitivity:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

Foresight Connection Sensitivity (FORCE)

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \boldsymbol{c}, \quad \boldsymbol{g}(\bar{\boldsymbol{\theta}}) := \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \left.\frac{\partial \bar{\boldsymbol{\theta}}}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \boldsymbol{\theta}$$

Foresight Connection Sensitivity

FOResight Connection sEnsitivity (FORCE) evaluates connections *after* pruning.

Connection Sensitivity:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boxed{\boldsymbol{c}=\mathbf{1}}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

Foresight Connection Sensitivity (FORCE)

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \boldsymbol{c}, \quad \boldsymbol{g}(\bar{\boldsymbol{\theta}}) := \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{c}}\right|_{\boxed{\boldsymbol{c}=\hat{\boldsymbol{c}}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \left.\frac{\partial \bar{\boldsymbol{\theta}}}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \boldsymbol{\theta}.$$

Foresight Connection Sensitivity

FOResight Connection sEnsitivity (FORCE) evaluates connections *after* pruning.

Connection Sensitivity:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \boxed{\sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta})_i|} \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

Foresight Connection Sensitivity (FORCE)

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \boldsymbol{c}, \quad \boldsymbol{g}(\bar{\boldsymbol{\theta}}) := \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \left.\frac{\partial \bar{\boldsymbol{\theta}}}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \boxed{\sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i|} \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

Foresight Connection Sensitivity

FOResight Connection sEnsitivity (FORCE) evaluates connections *after* pruning.

Connection Sensitivity:

$$\boldsymbol{g}(\boldsymbol{\theta}) := \left.\frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

Foresight Connection Sensitivity (FORCE)

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \boldsymbol{c}, \quad \boldsymbol{g}(\bar{\boldsymbol{\theta}}) := \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \left.\frac{\partial \bar{\boldsymbol{\theta}}}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} = \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}=\hat{\boldsymbol{c}}} \odot \boldsymbol{\theta}$$

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \boxed{\sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i|} \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \boxed{\sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i|} \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \ \|\boldsymbol{c}\|_0 = k.$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

$$\boxed{\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}}}$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \ \|\boldsymbol{c}\|_0 = k.$$

Iterative SNIP

$$\boxed{\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1}, \ \boldsymbol{c} \odot \boldsymbol{c}_t = \boldsymbol{c}$$

FORCE (algorithm)

$$\boxed{\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1},$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \ \|\boldsymbol{c}\|_0 = k.$$

Iterative SNIP

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boxed{\boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c})} \ \text{ s.t. } \ \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1}, \ \boldsymbol{c} \odot \boldsymbol{c}_t = \boldsymbol{c}$$

FORCE (algorithm)

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boxed{\boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c})} \ \text{ s.t. } \ \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1},$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \operatorname{supp}(\boldsymbol{c})} |\theta_i \; \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \; \|\boldsymbol{c}\|_0 = k.$$

Iterative SNIP

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \; S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1}, \boxed{\boldsymbol{c} \odot \boldsymbol{c}_t = \boldsymbol{c}}$$

FORCE (algorithm)

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \; S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1},$$

Progressive skeletonization

In order to optimize FORCE, we compute the pruning mask iteratively.

FORCE (Saliency)

$$\max_{\boldsymbol{c}} S(\boldsymbol{\theta}, \boldsymbol{c}) := \sum_{i \in \mathbf{supp}(\boldsymbol{c})} |\theta_i \, \nabla \mathcal{L}(\boldsymbol{\theta} \odot \boldsymbol{c})_i| \quad \text{s.t. } \boldsymbol{c} \in \{0,1\}^m, \ \|\boldsymbol{c}\|_0 = k.$$

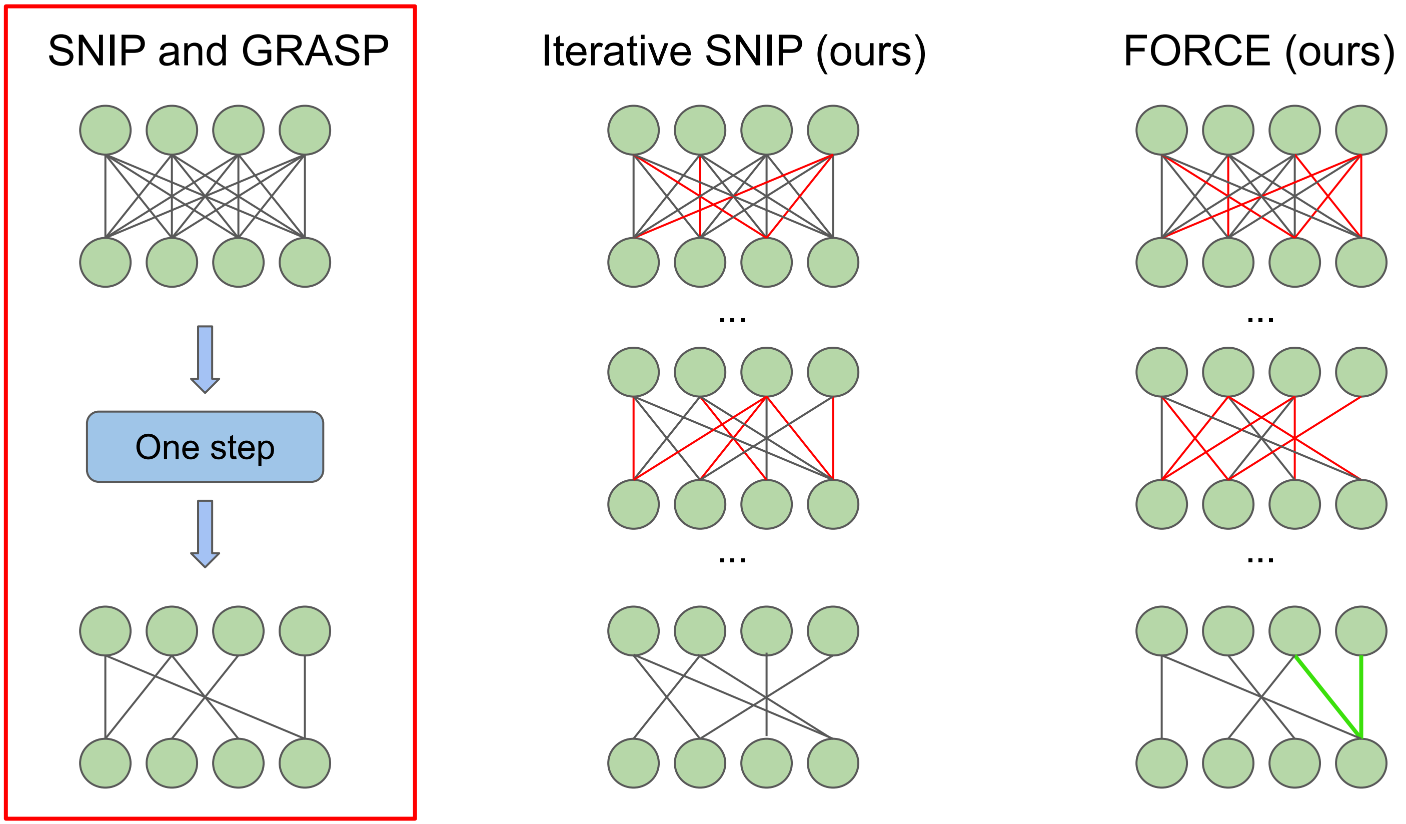
Iterative SNIP

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1}, \boxed{\boldsymbol{c} \odot \boldsymbol{c}_t = \boldsymbol{c}}$$

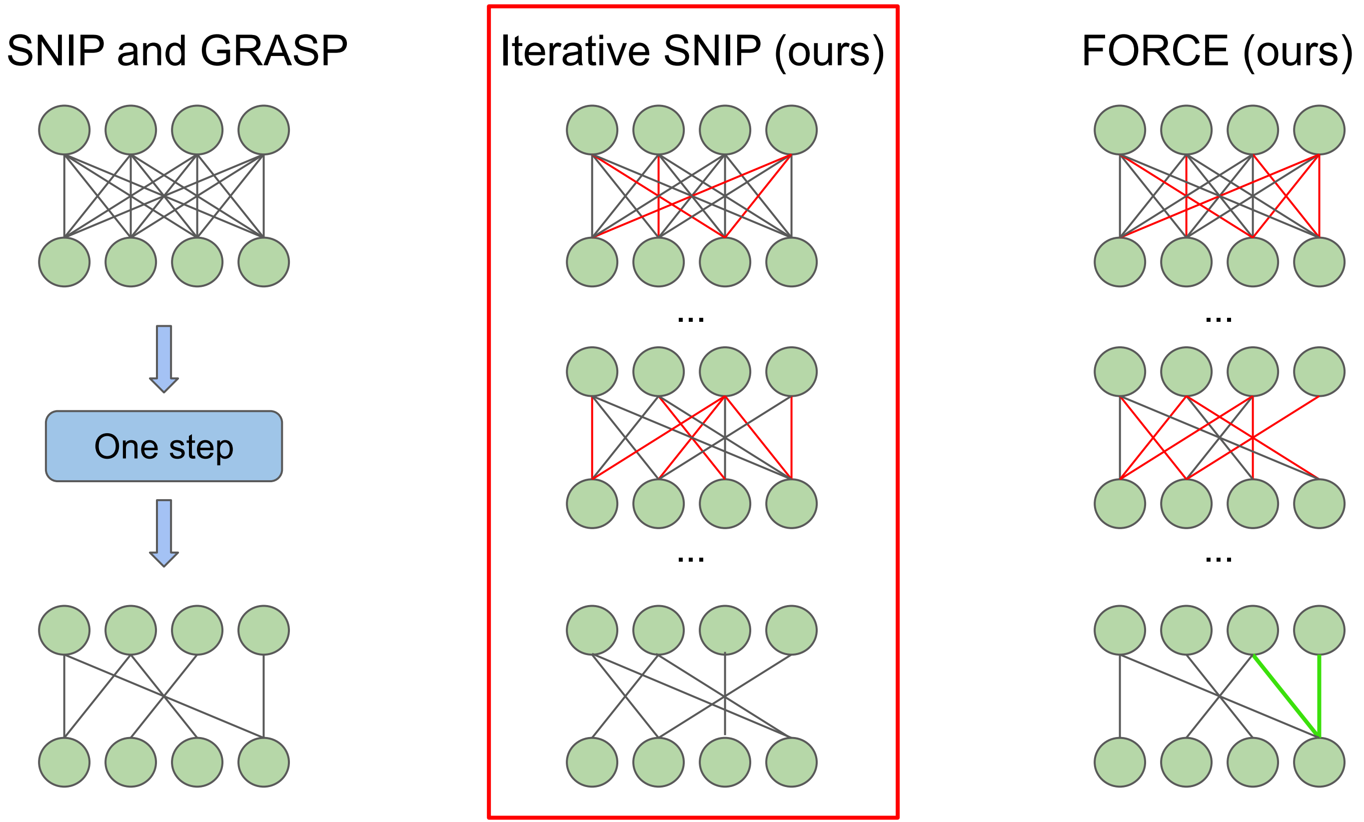
FORCE (algorithm)

$$\left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_t} \approx \left.\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \bar{\boldsymbol{\theta}}}\right|_{\boldsymbol{c}_{t+1}} \qquad \boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c}} \ S(\bar{\boldsymbol{\theta}}, \boldsymbol{c}) \quad \text{s.t.} \quad \boldsymbol{c} \in \{0,1\}^m, \quad \|\boldsymbol{c}\|_0 = k_{t+1},$$

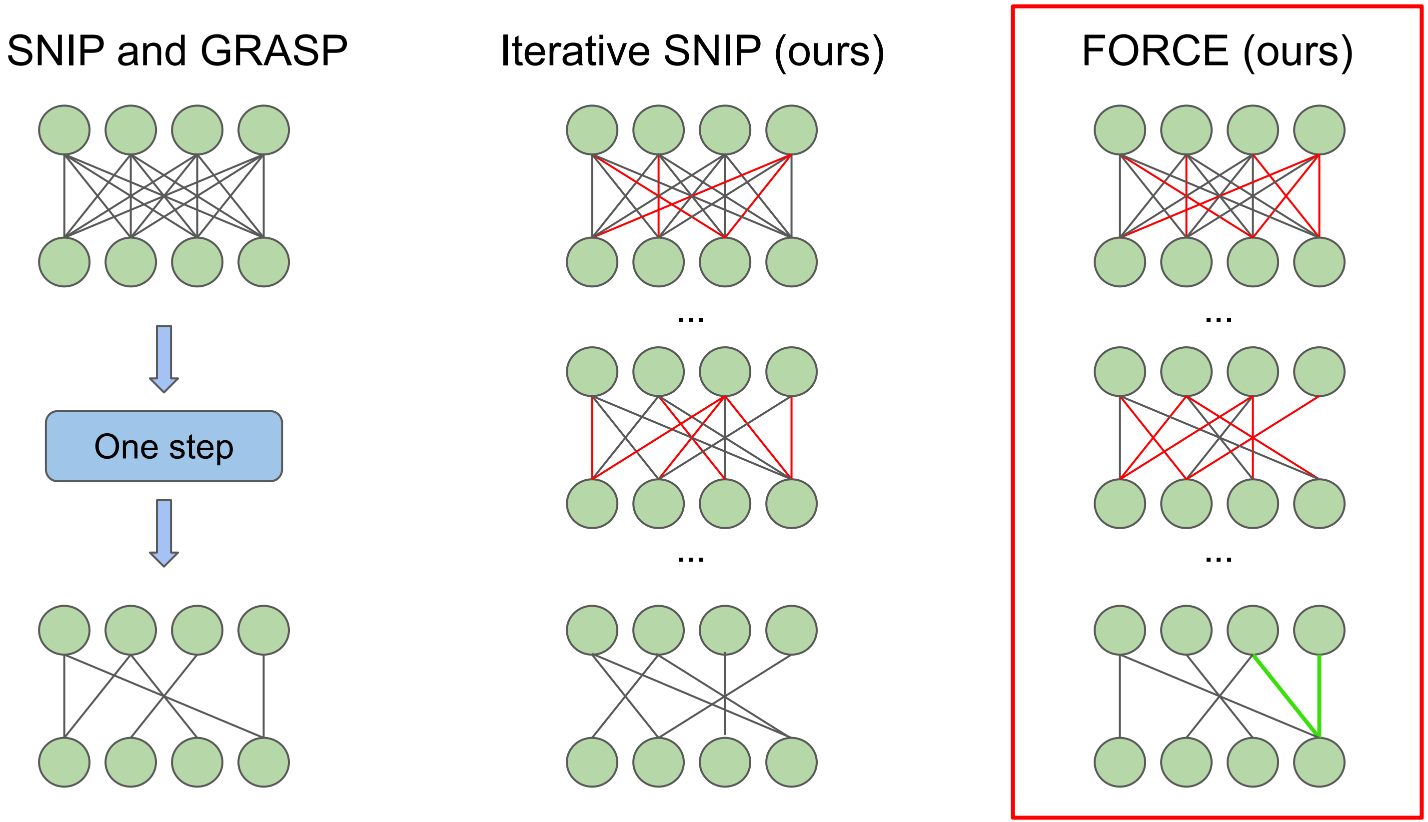
Overview



Overview

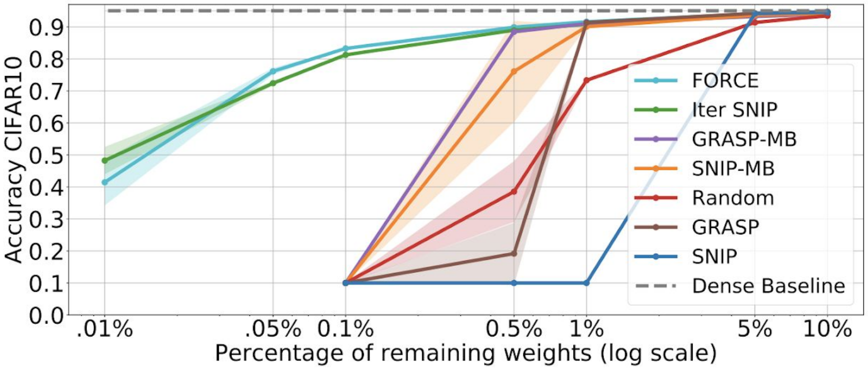


Overview



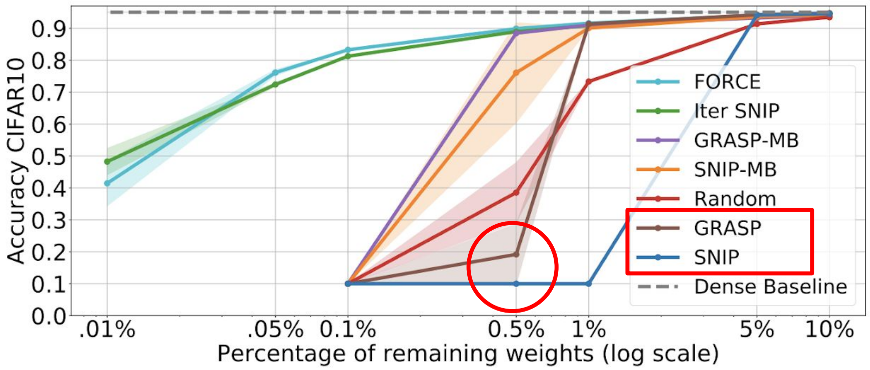
Results

Accuracy of pruned models



(a) Resnet50

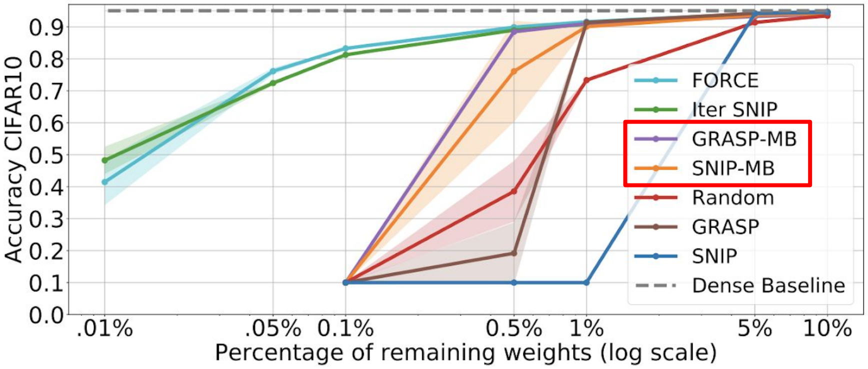
Accuracy of pruned models



(a) Resnet50

- SNIP and GRASP perform worse than random pruning at high sparsity.

Accuracy of pruned models

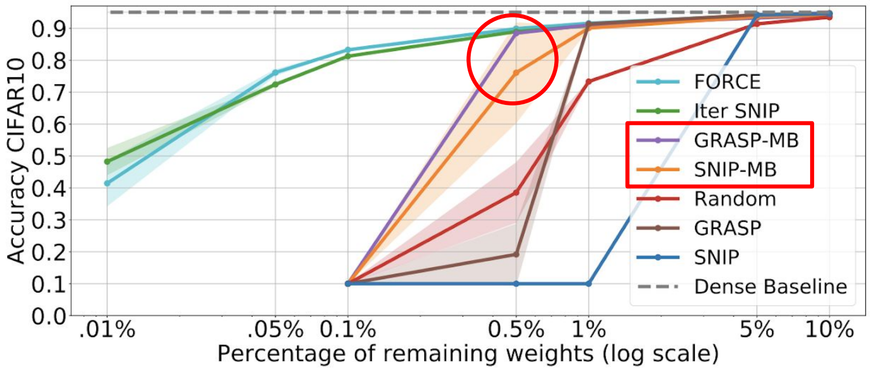


(a) Resnet50

- SNIP and GRASP perform worse than random pruning at high sparsity.

- SNIP-MB and GRASP-MB use as many batches as FORCE and Iter-SNIP.

Accuracy of pruned models



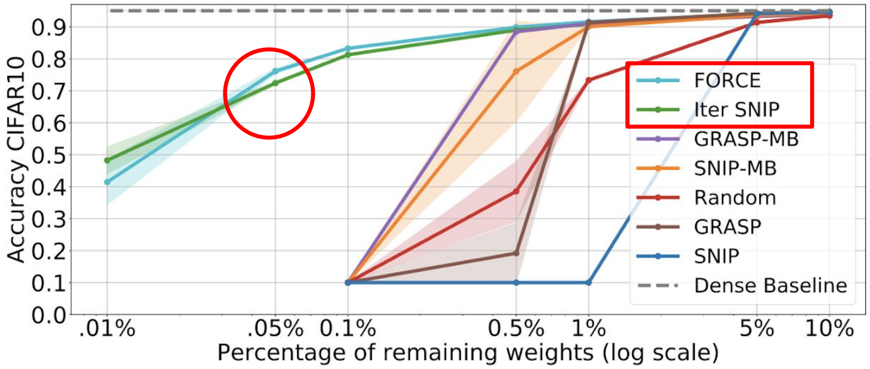
(a) Resnet50

- SNIP and GRASP perform worse than random pruning at high sparsity.

- SNIP-MB and GRASP-MB use as many batches as FORCE and Iter-SNIP.

- Using multiple batches to compute the saliency improves SNIP and GRASP dramatically.

Accuracy of pruned models



(a) Resnet50

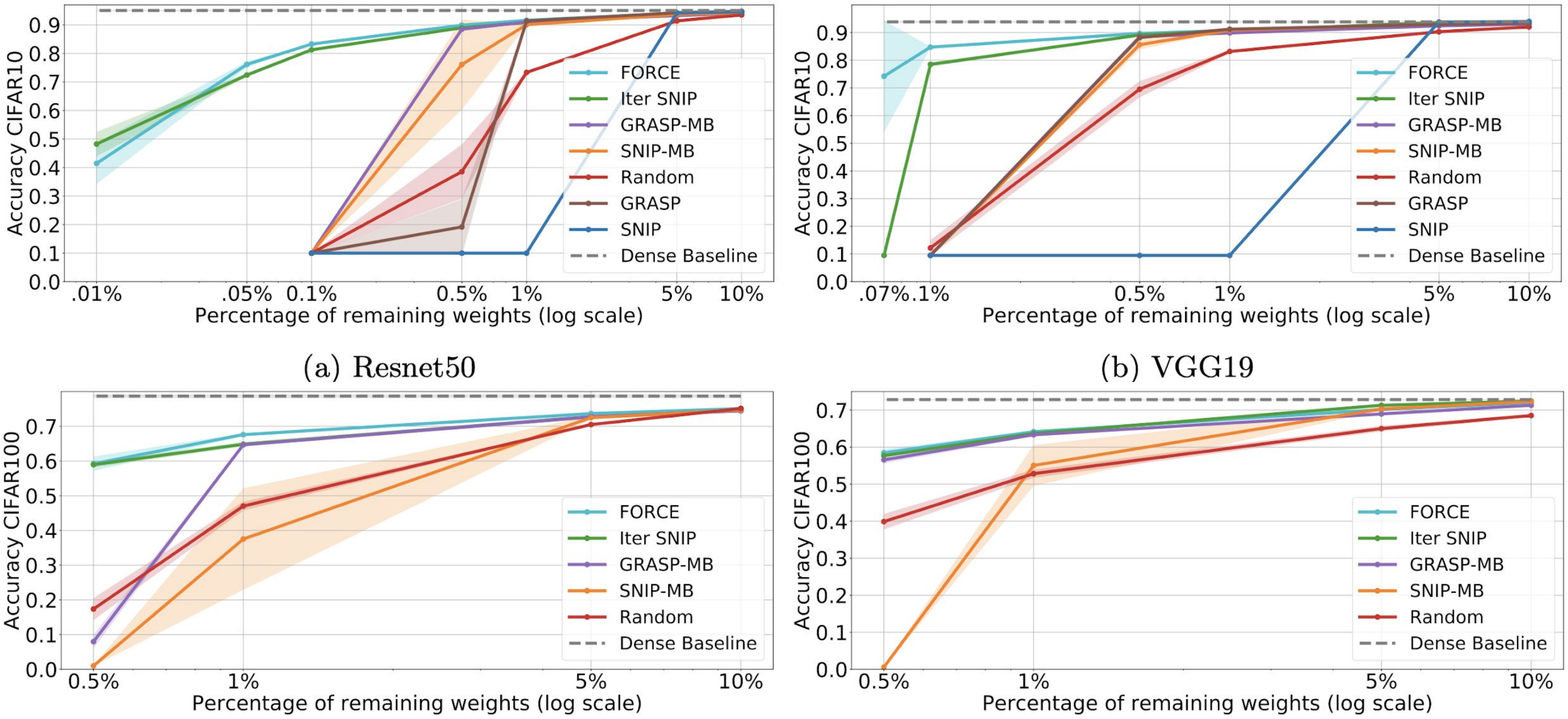
- SNIP and GRASP perform worse than random pruning at high sparsity.

- SNIP-MB and GRASP-MB use as many batches as FORCE and Iter-SNIP.

- Using multiple batches to compute the saliency improves SNIP and GRASP dramatically.

- Our “Progressive Skeletonization” methods find trainable models at much higher sparsity levels.

Accuracy of pruned models



(a) Resnet50

(b) VGG19

Future directions

- We observe that for Imagenet, the difference between pruning methods and random pruning is much larger for VGG than for Resnet.
- Understanding the trade-off between the complexity of the task and network capacity
- Improving the pruning baselines at moderate sparsities.
- Taking into account bias and batch norm layers for pruning.

Future directions

- We observe that for Imagenet, all evaluated methods work much better for VGG than for Resnet compared to the random baseline.
- Understanding the trade-off between the complexity of the task and network capacity
- Improving the pruning baselines at moderate sparsities.
- Taking into account bias and batch norm layers for pruning.

Future directions

- We observe that for Imagenet, all evaluated methods work much better for VGG than for Resnet compared to the random baseline.
- Understanding the trade-off between the complexity of the task and network capacity
- Improving the pruning baselines at moderate sparsities.
- Taking into account bias and batch norm layers for pruning.

Future directions

- We observe that for Imagenet, all evaluated methods work much better for VGG than for Resnet compared to the random baseline.
- Understanding the trade-off between the complexity of the task and network capacity
- Improving the pruning baselines at moderate sparsities.
- Taking into account bias and batch norm layers for pruning.

Summary

- SNIP and GRASP collapse at high sparsities.
- We argue interactions between weights should be taken into account.
- We suggest two methods to optimize FORCE: With and without recovery of weights.
- We validate empirically iterative pruning is crucial at high sparsities.

Code: https://github.com/naver/force