SNIP: Single-shot Network Pruning based on Connection Sensitivity

Namhoon Lee, Thalaiyasingam Ajanthan, Philip H. S. Torr
University of Oxford

UNIVERSITY OF OXFORD

Network Pruning: Background and Motivation

▶ **Issues with overparameterization in large neural networks**
- ▷ memory and time complexity
- ▷ energy consumption
- ▷ generalization capability

▶ **Drawbacks of existing methods**
- ▷ hyperparameters with heuristics
- ▷ architecture specific requirements
- ▷ optimization difficulties
- ▷ pretraining

▶ **What we want**
- ▷ No hyperparameters
- ▷ No pretraining
- ▷ No iterative prune–retrain cycle
- ▷ not require the whole training set

Single-shot pruning prior to training

Single-shot Network Pruning based on Connection Sensitivity

▶ **Neural Network Pruning**

Write pruning as constrained optimization:

$$\min_{\mathbf{w}} L(\mathbf{w};\mathcal{D}) = \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)) \,, \qquad (1)$$
$$\text{s.t.} \quad \mathbf{w}\in\mathbb{R}^m, \quad \|\mathbf{w}\|_0 \le \kappa \,.$$

Here, $\mathcal{D} = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^n$ is a dataset, κ is a desired sparsity level, $\ell(\cdot)$ is the loss function, $\mathbf{w}$ is the set of parameters of the network.

▶ **Connection Sensitivity: Architectural Perspective**

▷ Introduce auxiliary indicator variables $\mathbf{c}\in\{0,1\}^m$:

$$\min_{\mathbf{c},\mathbf{w}} L(\mathbf{c}\odot\mathbf{w};\mathcal{D}) = \min_{\mathbf{c},\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{c}\odot\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)) \,, \qquad (2)$$
$$\text{s.t.} \quad \mathbf{w}\in\mathbb{R}^m \,, \quad \mathbf{c}\in\{0,1\}^m, \quad \|\mathbf{c}\|_0 \le \kappa \,.$$

The idea is to measure the effect of removing a parameter on the loss by separating $\mathbf{w}$ from $\mathbf{c}$:

▷ The effect of removing paramter j:

$$\Delta L_j(\mathbf{w};\mathcal{D}) = L(\mathbf{1}\odot\mathbf{w};\mathcal{D}) - L((\mathbf{1}-\mathbf{e}_j)\odot\mathbf{w};\mathcal{D}) \,, \qquad (3)$$

where $\mathbf{e}_j$ is the indicator vector of element j.

▷ Infinitesimal version of ΔL_j:

$$\Delta L_j(\mathbf{w};\mathcal{D}) \approx \left.\frac{\partial L(\mathbf{c}\odot\mathbf{w};\mathcal{D})}{\partial c_j}\right|_{\mathbf{c}=\mathbf{1}} = \lim_{\delta\to 0} \left.\frac{L(\mathbf{c}\odot\mathbf{w};\mathcal{D}) - L((\mathbf{c}-\delta\,\mathbf{e}_j)\odot\mathbf{w};\mathcal{D})}{\delta}\right|_{\mathbf{c}=\mathbf{1}} , \qquad (4)$$

which, denoted as $g_j(\mathbf{w};\mathcal{D})$, measures the rate of change of L with respect to an infinitesimal change in c_j from $1\to 1-\delta$.

▷ Define connection sensitivity as the saliency criterion by taking the normalized magnitude of g_j :

$$s_j = \frac{|g_j(\mathbf{w};\mathcal{D})|}{\sum_{k=1}^m |g_k(\mathbf{w};\mathcal{D})|} \,. \qquad (5)$$

▶ **Single-shot Pruning at Initialization**

▷ Use variance scaling methods to initialize weights so that the impact of weights on s_j is minimized while making it robust to architecture variations.

▷ Using a reasonable number of training examples in one mini-batch can lead to effective pruning.

Algorithm 1 SNIP

Require: Loss function L, training dataset $\mathcal{D}$, sparsity level κ
Ensure: $\|\mathbf{w}^*\|_0 \le \kappa$

1: $\mathbf{w} \leftarrow$ VarianceScalingInitialization
2: $\mathcal{D}^b = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^b \sim \mathcal{D}$ ▷ Sample a mini-batch of training data
3: $s_j \leftarrow \frac{|g_j(\mathbf{w};\mathcal{D}^b)|}{\sum_{k=1}^m |g_k(\mathbf{w};\mathcal{D}^b)|}\,, \quad \forall j\in\{1\dots m\}$
4: $c_j \leftarrow \mathbb{1}[s_j - \tilde{s}_\kappa \ge 0] \,, \quad \forall j\in\{1\dots m\}$ ▷ Pruning: choose top-κ connections
5: $\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w}\in\mathbb{R}^m} L(\mathbf{c}\odot\mathbf{w};\mathcal{D})$ ▷ Regular training
6: $\mathbf{w}^* \leftarrow \mathbf{c}\odot\mathbf{w}^*$

Results

▶ LeNets

Method	Criterion	LeNet-300-100 $\bar{\kappa}$ (%)	LeNet-300-100 err. (%)	LeNet-5-Caffe $\bar{\kappa}$ (%)	LeNet-5-Caffe err. (%)	Pretrain	# Prune	Additional hyperparam.	Augment objective	Arch. constraints
Ref.	–	–	1.7	–	0.9	–	–	–	–	–
LWC	Magnitude	91.7	**1.6**	91.7	**0.8**	✓	many	✓	✗	✓
DNS	Magnitude	98.2	2.0	99.1	0.9	✓	many	✓	✗	✓
LC	Magnitude	99.0	3.2	99.0	1.1	✓	many	✓	✓	✗
SWS	Bayesian	95.6	1.9	99.5	1.0	✓	soft	✓	✓	✗
SVD	Bayesian	98.5	1.9	99.6	**0.8**	✓	soft	✓	✓	✗
OBD	Hessian	92.0	2.0	92.0	2.7	✓	many	✓	✗	✗
L-OBS	Hessian	98.5	2.0	99.0	2.1	✓	many	✓	✗	✓
SNIP (ours)	Connection sensitivity	95.0	**1.6**	98.0	**0.8**	✗	**1**	✗	✗	✗
		98.0	2.4	99.0	1.1					

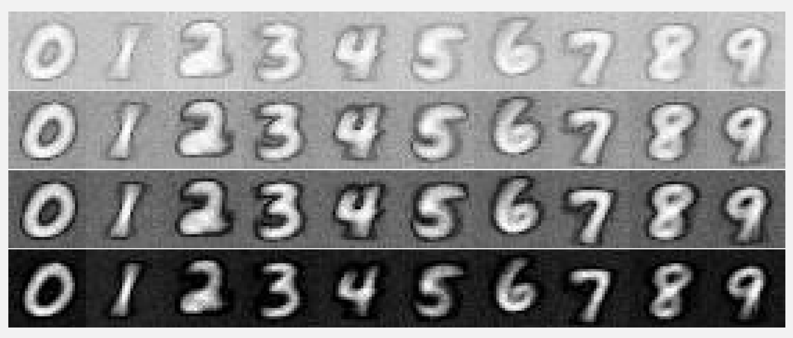
SNIP is capable of pruning extreme sparsity levels (e.g., 99% for LeNet-5-Caffe), while being significantly simpler than other approaches.

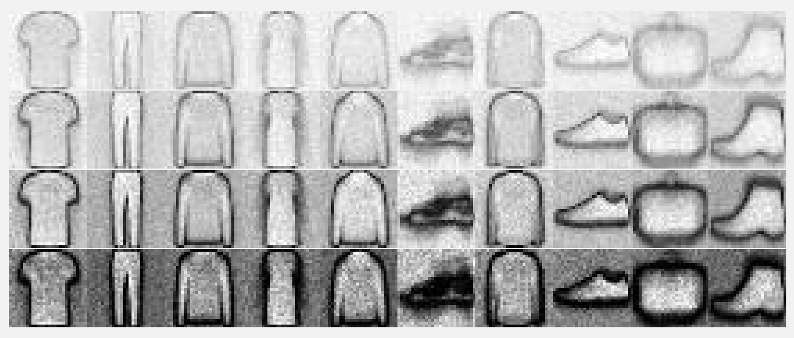
▶ Various modern architectures

Architecture	Model	Sparsity (%)	# Parameters	Error (%)	Δ
Convolutional	AlexNet-s	90.0	5.1m → 507k	14.12 → 14.99	+0.87
	AlexNet-b	90.0	8.5m → 849k	13.92 → 14.50	+0.58
	VGG-C	95.0	10.5m → 526k	6.82 → 7.27	+0.45
	VGG-D	95.0	15.2m → 762k	6.76 → 7.09	+0.33
	VGG-like	97.0	15.0m → 449k	8.26 → 8.00	**−0.26**
Residual	WRN-16-8	95.0	10.0m → 548k	6.21 → 6.63	+0.42
	WRN-16-10	95.0	17.1m → 856k	5.91 → 6.43	+0.52
	WRN-22-8	95.0	17.2m → 858k	6.14 → 5.85	**−0.29**
Recurrent	LSTM-s	95.0	137k → 6.8k	1.88 → 1.57	**−0.31**
	LSTM-b	95.0	535k → 26.8k	1.15 → 1.35	+0.20
	GRU-s	95.0	104k → 5.2k	1.87 → 2.41	+0.54
	GRU-b	95.0	404k → 20.2k	1.71 → 1.52	**−0.19**

SNIP is generally applicable to various architectures and models and reduces a significant amount of parameters with minimal loss in performance.

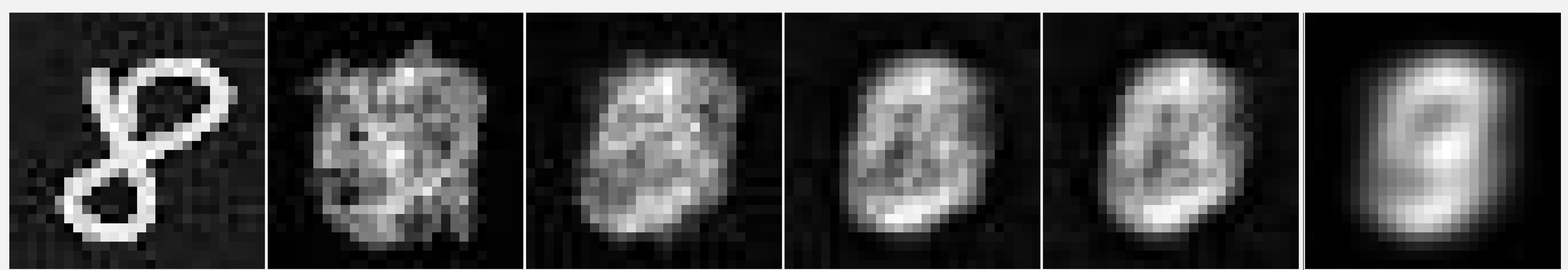
▶ Visualizing pruned/retained parameters


(a) MNIST


(b) Fashion-MNIST

The parameters connected to the discriminative part of image survive and the irrelevant parts gets pruned.

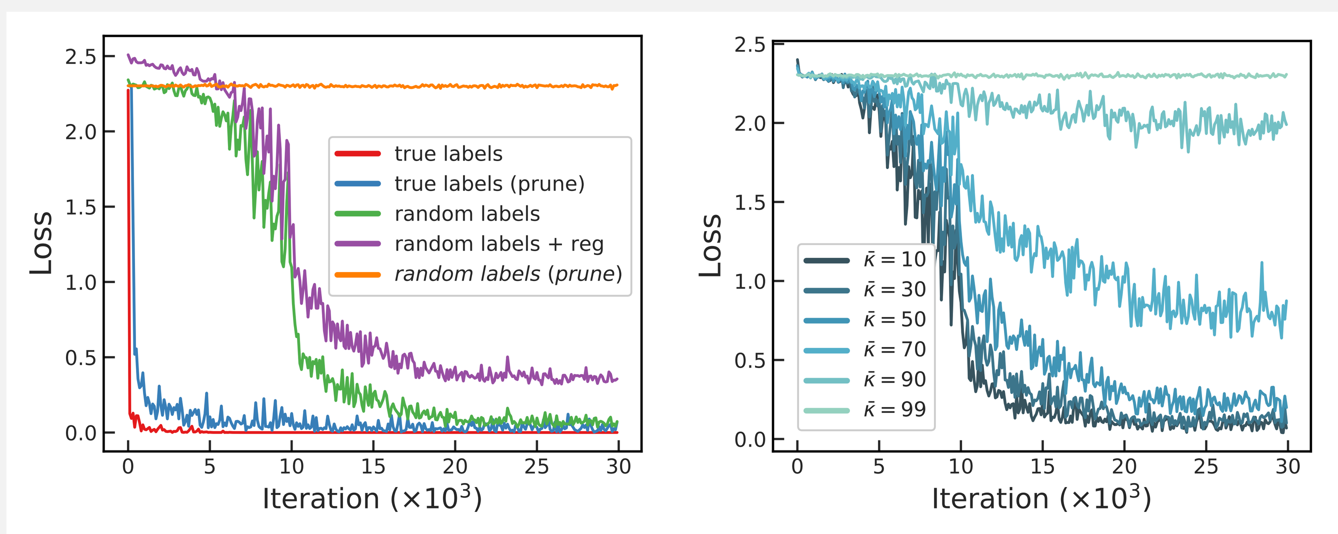
▶ Survived parameters and resulting performance for different batch sizes



$\lvert\mathcal{D}^b\rvert = 1$	$\lvert\mathcal{D}^b\rvert = 10$	$\lvert\mathcal{D}^b\rvert = 100$	$\lvert\mathcal{D}^b\rvert = 1000$	$\lvert\mathcal{D}^b\rvert = 10000$	train set
(1.94%)	(1.72%)	(1.64%)	(1.56%)	(1.40%)	–

For $|\mathcal{D}^b| = 1$, the sample was 8; SNIP precisely retains valid connections. As $|\mathcal{D}^b|$ increases, connections get close to the train average, and the error decreases.

▶ Fitting random labels



(left) The SNIP-pruned model does not fit the random labels. (right) The effect of varying sparsity ($\bar{\kappa}$). This indicates that the pruned network does not have sufficient capacity to fit the random labels, but is capable of performing the task.

Conclusion

▶ SNIP: a new pruning algorithm that is simple, versatile and interpretable
▶ Pruning at single-shot prior to training
▶ Applicable to a variety of neural network models without modifications

{namhoon,ajanthan,phst}@robots.ox.ac.uk
https://github.com/namhoonlee/snip-public