Efficient Neural Network Compression

Namhoon Lee

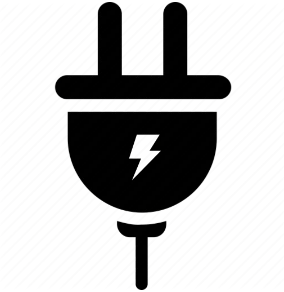
University of Oxford

3 May 2019

A Challenge in Deep Learning: *Overparameterization*

Large neural networks require:

memory & computations

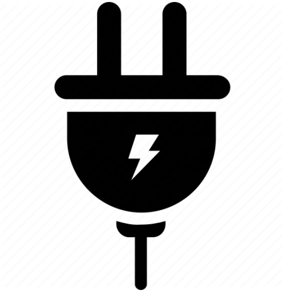


power consumption

A Challenge in Deep Learning: *Overparameterization*

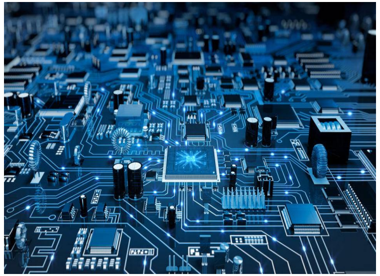
Large neural networks require:

memory & computations



power consumption

Critical to resource constrained environments

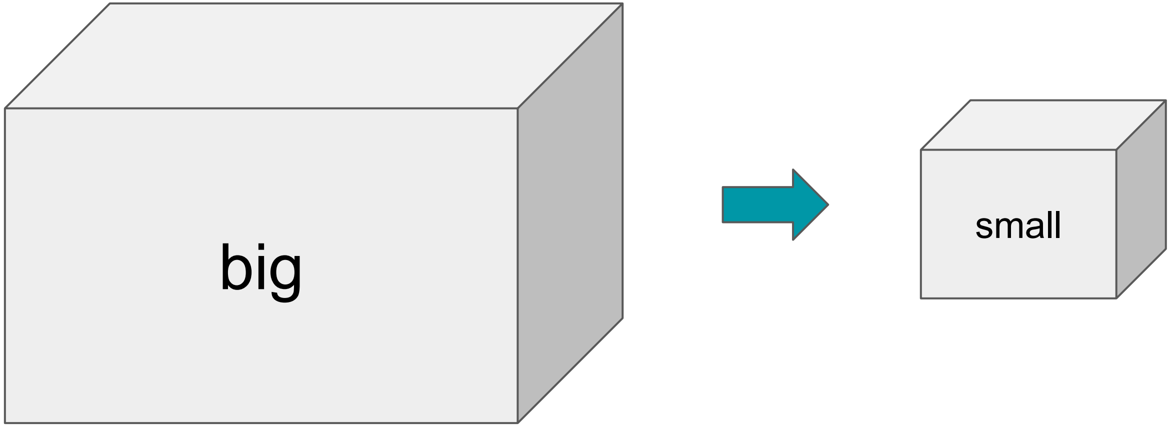


embedded systems
e.g., mobile devices

real-time tasks
e.g., autonomous car

Network compression

The goal is to reduce the ***size*** of neural network without compromising accuracy.



~ same accuracy

Approaches

- Network pruning

  : reduce the number of parameters

Approaches

- Network pruning
  : reduce the number of parameters

- Network quantization
  : reduce the precision of parameters

Approaches

- Network pruning
  : reduce the number of parameters

- Network quantization
  : reduce the precision of parameters

Others: knowledge distillation, conditional computation, etc.

Approaches

- Network pruning
  : reduce the number of parameters
- Network quantization
  : reduce the precision of parameters

Others: knowledge distillation, conditional computation, etc.

Network pruning

Different forms

- Parameters (weights, biases)
- Activations (neurons)

can be done structured way
(e.g., channel, filter, layer)

Network pruning

Different forms

- Parameters (weights, biases)
- Activations (neurons)

can be done structured way
(e.g., channel, filter, layer)

Different principles

- Magnitude based
- Hessian based
- Bayesian

Network pruning

Different forms

- Parameters (weights, biases)
- Activations (neurons)

can be done structured way
(e.g., channel, filter, layer)

Different principles

- Magnitude based
- Hessian based
- Bayesian

$\boxed{\Rightarrow \text{ remove} > 90\% \text{ parameters}}$

Drawbacks in existing approaches

- Hyperparameters with weakly grounded heuristics

  (*e.g.*, layer-wise threshold [5], stochastic pruning rule [2])

References
[1] Learning both weights and connections for efficient neural network, Han et al. NIPS'15
[2] Dynamic network surgery for efficient dnns, Guo et al. NIPS'16.
[3] Learning-compression algorithms for neural net pruning, Carreira-Perpinan & Idelbayev. CVPR'18.
[4] Variational dropout sparsifies deep neural networks, Molchanov et al. ICML'17.
[5] Learning to prune deep neural networks via layer-wise optimal brain surgeon, Dong et al. NIPS'17.
[6] Learning Sparse Neural Networks through L0 Regularization, Louizos et al. ICLR'18

Drawbacks in existing approaches

- Hyperparameters with weakly grounded heuristics

  (*e.g.,* layer-wise threshold [5], stochastic pruning rule [2])

- Architecture specific requirements

  (*e.g.,* conv/fc separate prune in [1])

References
[1] Learning both weights and connections for efficient neural network, Han et al. NIPS'15
[2] Dynamic network surgery for efficient dnns, Guo et al. NIPS'16.
[3] Learning-compression algorithms for neural net pruning, Carreira-Perpinan & Idelbayev. CVPR'18.
[4] Variational dropout sparsifies deep neural networks, Molchanov et al. ICML'17.
[5] Learning to prune deep neural networks via layer-wise optimal brain surgeon, Dong et al. NIPS'17.
[6] Learning Sparse Neural Networks through L0 Regularization, Louizos et al. ICLR'18

Drawbacks in existing approaches

- Hyperparameters with weakly grounded heuristics

  (*e.g.,* layer-wise threshold [5], stochastic pruning rule [2])
- Architecture specific requirements

  (*e.g.,* conv/fc separate prune in [1])
- Optimization difficulty

  (*e.g.,* convergence in [3, 6])

References
[1] Learning both weights and connections for efficient neural network, Han et al. NIPS'15
[2] Dynamic network surgery for efficient dnns, Guo et al. NIPS'16.
[3] Learning-compression algorithms for neural net pruning, Carreira-Perpinan & Idelbayev. CVPR'18.
[4] Variational dropout sparsifies deep neural networks, Molchanov et al. ICML'17.
[5] Learning to prune deep neural networks via layer-wise optimal brain surgeon, Dong et al. NIPS'17.
[6] Learning Sparse Neural Networks through L0 Regularization, Louizos et al. ICLR'18

Drawbacks in existing approaches

- Hyperparameters with weakly grounded heuristics

  (*e.g.,* layer-wise threshold [5], stochastic pruning rule [2])
- Architecture specific requirements

  (*e.g.,* conv/fc separate prune in [1])
- Optimization difficulty

  (*e.g.,* convergence in [3, 6])
- Pretraining step

  ([1,2,3,4,5,6]; almost all)

References

[1] Learning both weights and connections for efficient neural network, Han et al. NIPS'15
[2] Dynamic network surgery for efficient dnns, Guo et al. NIPS'16.
[3] Learning-compression algorithms for neural net pruning, Carreira-Perpinan & Idelbayev. CVPR'18.
[4] Variational dropout sparsifies deep neural networks, Molchanov et al. ICML'17.
[5] Learning to prune deep neural networks via layer-wise optimal brain surgeon, Dong et al. NIPS'17.
[6] Learning Sparse Neural Networks through L0 Regularization, Louizos et al. ICLR'18

Drawbacks in existing approaches

- Hyperparameters with weakly grounded heuristics

  (*e.g.*, layer-wise threshold [5], stochastic pruning rule [2])
- Architecture specific requirements

  (*e.g.*, conv/fc separate prune in [1])
- Optimization difficulty

  (*e.g.*, convergence in [3, 6])
- Pretraining step

  ([1,2,3,4,5,6]; almost all)

poor scalability & utility

References
[1] Learning both weights and connections for efficient neural network, Han et al. NIPS'15
[2] Dynamic network surgery for efficient dnns, Guo et al. NIPS'16.
[3] Learning-compression algorithms for neural net pruning, Carreira-Perpinan & Idelbayev. CVPR'18.
[4] Variational dropout sparsifies deep neural networks, Molchanov et al. ICML'17.
[5] Learning to prune deep neural networks via layer-wise optimal brain surgeon, Dong et al. NIPS'17.
[6] Learning Sparse Neural Networks through L0 Regularization, Louizos et al. ICLR'18

We want ..

No hyperparameters

No iterative prune -- retrain cycle

No pretraining

No large data

We want ..

No hyperparameters

No iterative prune -- retrain cycle

No pretraining

No large data

Single-shot pruning prior to training

SNIP: Single-shot Network Pruning based on Connection Sensitivity

N. Lee, T. Ajanthan, P. Torr

International Conference on Learning Representations (ICLR) 2019

Objective

- Identify important parameters in the network and remove unimportant ones

$$\min_{\mathbf{w}} L(\mathbf{w}; \mathcal{D}) = \min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}; (\mathbf{x}_i, \mathbf{y}_i)) \,,$$

$$\text{s.t.} \quad \mathbf{w} \in \mathbb{R}^m, \quad \|\mathbf{w}\|_0 \leq \kappa \,.$$

Objective

- Identify important parameters in the network and remove unimportant ones

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

Idea

- Measure the effect of removing each parameter on the loss

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

Idea

- Measure the effect of removing each parameter on the loss

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

Idea

- Measure the effect of removing each parameter on the loss

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

- The greedy way is prohibitively expensive to perform: $O(m!)$

SNIP

The effect on the loss can be approximated by

1. auxiliary variables representing the connectivity of parameters
2. derivative of the loss w.r.t. these indicator variables

SNIP

1. Introduce c

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

SNIP

1. Introduce c

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

SNIP

1. Introduce c

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

2. Derivative w.r.t. c

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

SNIP

1. Introduce c

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

2. Derivative w.r.t. c

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

- ∂L/∂cj is an infinitesimal version of ∆Lj
- measures the rate of change of L w.r.t. infinitesimal change in cj from 1 → 1 − δ
- computed efficiently in one forward-backward pass using auto differentiation, for all j at once

Reference: Understanding black-box predictions via influence functions, Koh & Liang. ICML’17

SNIP

1. Introduce c

$$\min_{\mathbf{c},\mathbf{w}} L(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) = \min_{\mathbf{c},\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{c} \odot \mathbf{w}; (\mathbf{x}_i, \mathbf{y}_i)) \, ,$$

$$\text{s.t.} \quad \mathbf{w} \in \mathbb{R}^m \, ,$$

$$\mathbf{c} \in \{0,1\}^m, \quad \|\mathbf{c}\|_0 \leq \kappa \, ,$$

2. Derivative w.r.t. c

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

3. Connection sensitivity

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

Prune at initialization

- Measure CS on untrained networks prior to training

  → Or zero gradients at pretrained

- Sample weights from a dist. with architecture aware variance

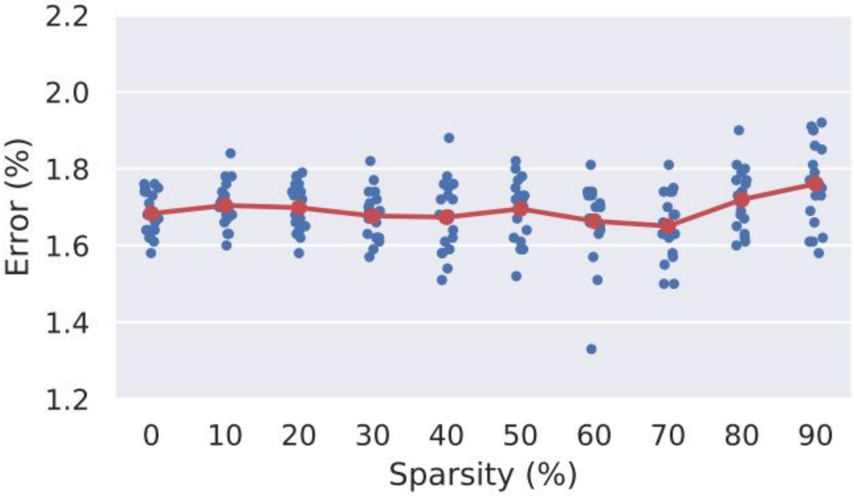
  → Ensure the variance of weights to remain throughout the network ([1])

- Alleviate the dependency on the weights in computing CS

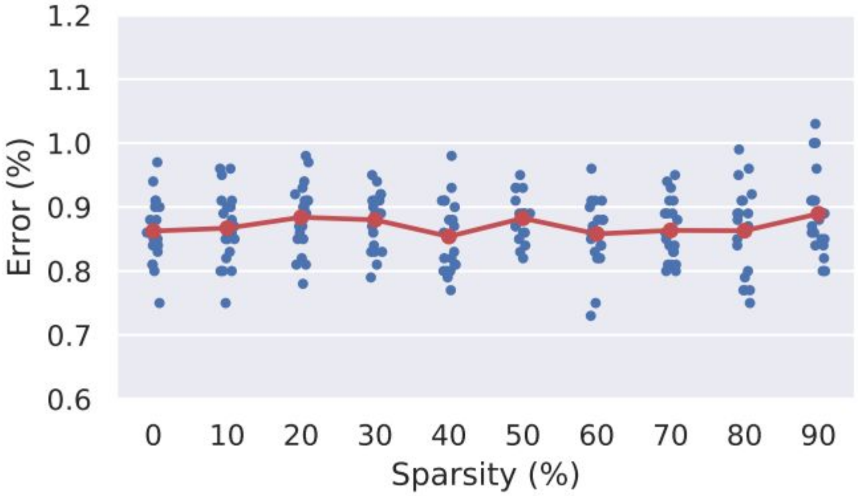
  → Remove the pretraining requirement, architecture dep. hyperparameters

[1] Understanding the difficulty of training deep feedforward neural networks, Glorot & Bengio, AISTATS 2010

LeNets

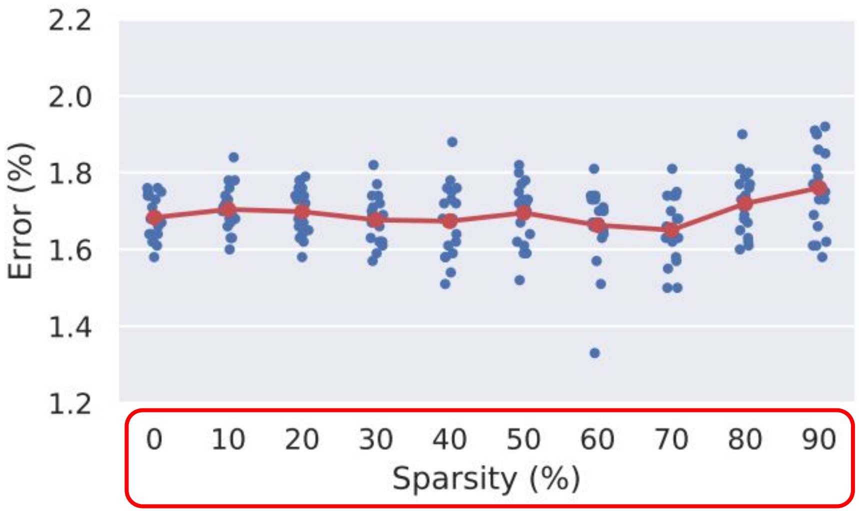


(a) LeNet-300-100

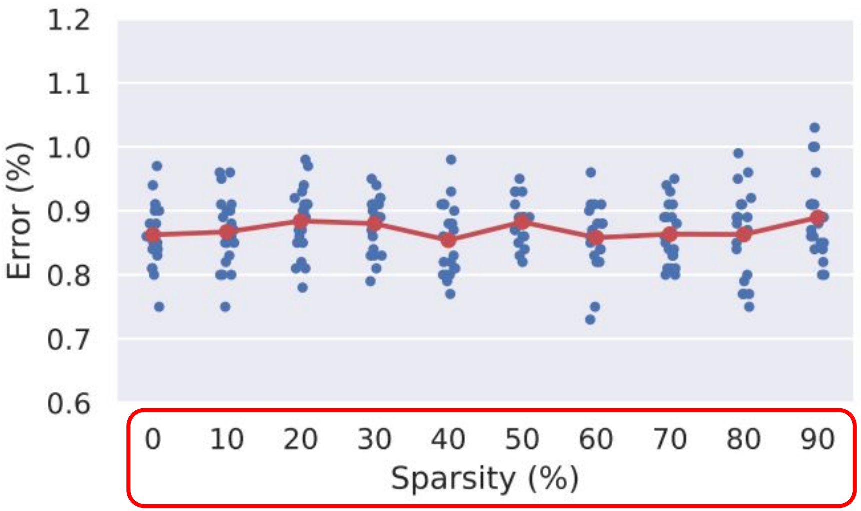


(b) LeNet-5-Caffe

LeNets



(a) LeNet-300-100



(b) LeNet-5-Caffe

LeNets: comparison to SOTA

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 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

LeNets: comparison to SOTA

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 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

LeNets: comparison to SOTA

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 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

LeNets: comparison to SOTA

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 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

LeNets: comparison to SOTA

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 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

Various architectures & models

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**

Various architectures & models

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

Various architectures & models

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

Various architectures & models

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

Which parameters are pruned?

Visualize c in the first fc layer for varying data

1. curate a mini-batch
2. compute the connection sensitivity
3. create the pruning mask
4. visualize the first layer (fully connected)

Which parameters are pruned?

Visualize c in the first fc layer for varying data

1. curate a mini-batch
2. compute the connection sensitivity
3. create the pruning mask
4. visualize the first layer (fully connected)

The input was digit 8.

Which parameters are pruned?

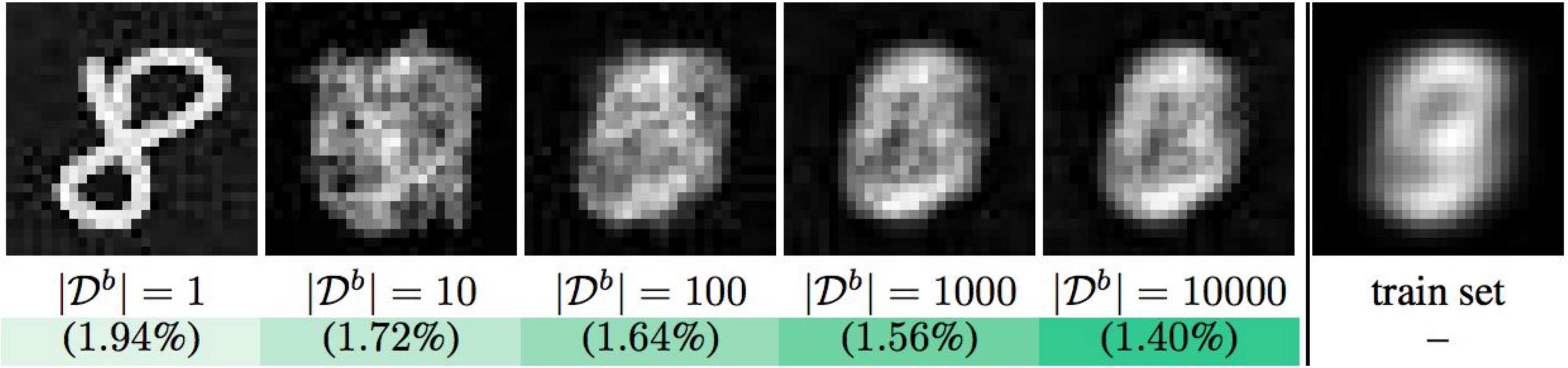
Visualize c in the first fc layer for varying data

1. curate a mini-batch
2. compute the connection sensitivity
3. create the pruning mask
4. visualize the first layer (fully connected)

The input was digit 8.

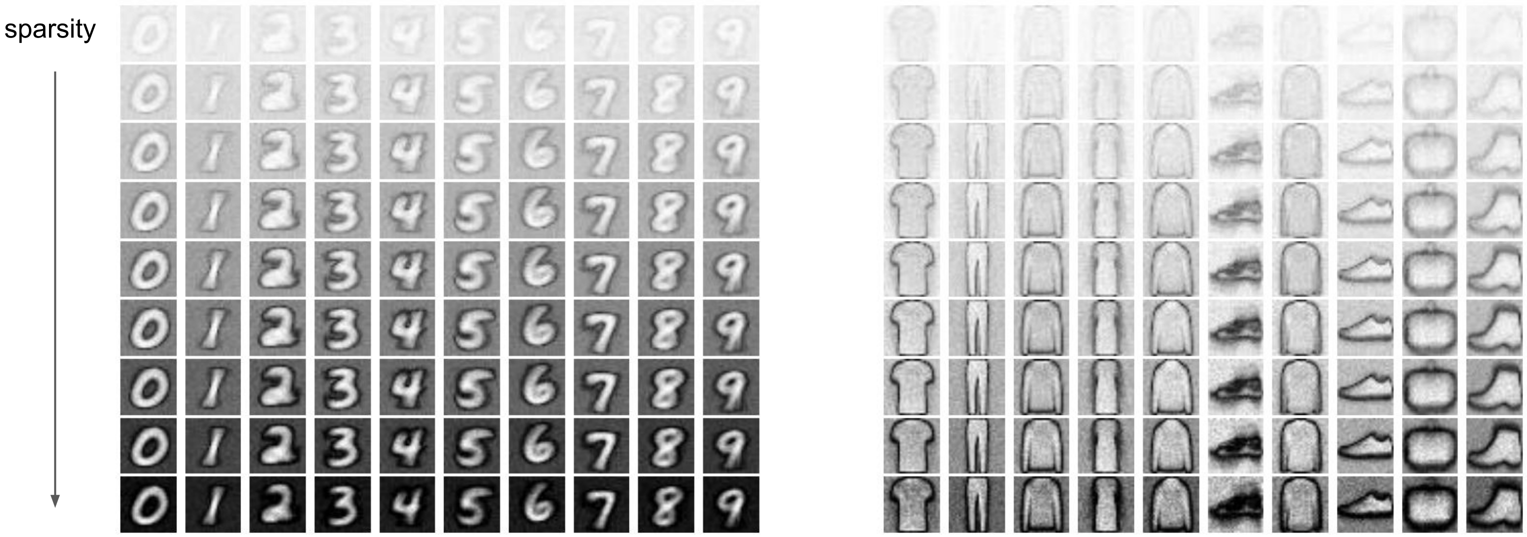
Carrying out such inspection is not straightforward with other methods.

Which parameters are pruned?



Carrying out such inspection is not straightforward with other methods.

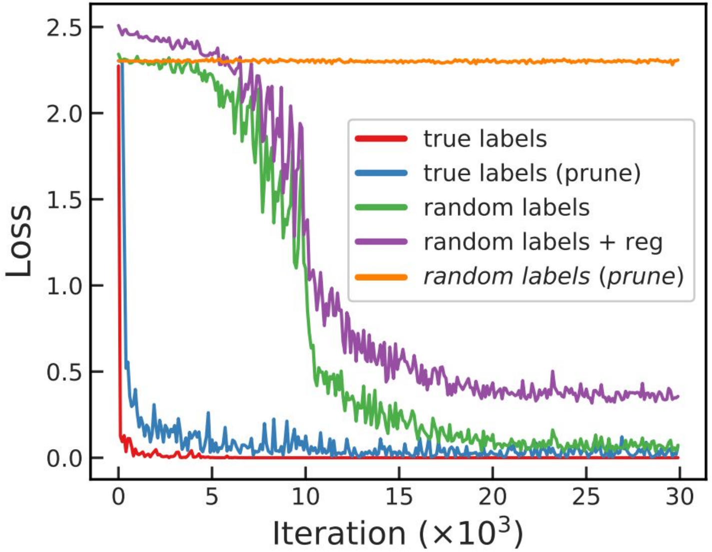
Which parameters are pruned?



(a) MNIST (b) Fashion-MNIST

<u>The parameters connected to the discriminative part of image are retained.</u>

Prevent memorization



[Fitting random labels]
Understanding deep learning requires rethinking generalization, Zhang et al. ICLR'17

<u>The pruned network does not have sufficient capacity to fit the random labels, but is capable of performing the task.</u>

SNIP

Simple
Versatile
Interpretable

Paper:

https://arxiv.org/abs/1810.02340

Code:

https://github.com/namhoonlee/snip-public

Contact:

http://www.robots.ox.ac.uk/~namhoon/