Pruning at Initialization

2020-06-19

hyounguk.shon@kaist.ac.kr

Contents

- Lottery Ticket Hypothesis (ICLR 2019)
- SNIP: Single-shot Network Pruning based on Connection Sensitivity (ICLR 2019)
- A Signal Propagation Perspective for Pruning Neural Networks at Initialization (ICLR 2020)

Overview of Network Pruning

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

Weight pruning is optimization with bound on L0-norm of weight.

- Weight pruning
    - Goal of pruning is to slim down overparametrized NNs.
    - The problem is to identify less significant weights.
- Saliency-based pruning (vs. regularization-based pruning)
    - Identify weights that cause least degradation of loss.
    - We define a saliency score s_j on each weight.
- Most pruning algorithms consist of prune-retrain cycles:
    - Prune a tiny portion of weights at each step.
    - Fine-tune weights to loss function.
    - Repeat until you reach the target pruning ratio.

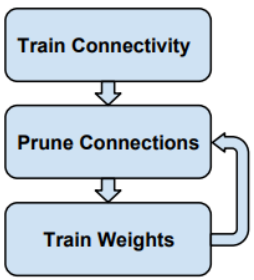


Figure 2: Three-Step Training Pipeline.

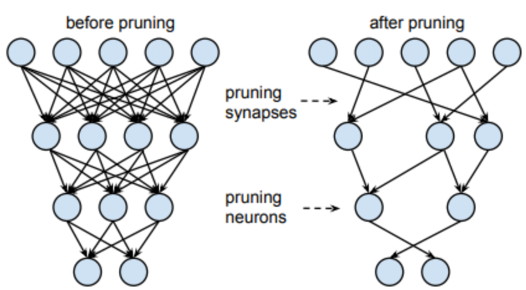


Figure 3: Synapses and neurons before and after pruning.

Overview of Network Pruning

- Hessian-based saliency score (LeCun 1990, Hassibi 1993)
  - Second-order approximation of $\mathcal{L}(\theta)$ because all gradients are zero at convergence. ($\theta = \theta^*$)
  - Hessian is approximated as a diagonal matrix because full Hessian is intractable.
  - EigenDamage (ICML 2019) presents a more complete Hessian-based approach.
- Magnitude-based saliency score (Han 2015, Guo 2016)
  - Set a threshold based on the norm of the weight.
  - Somewhat heuristic approximation of $\mathcal{L}(\theta)$.

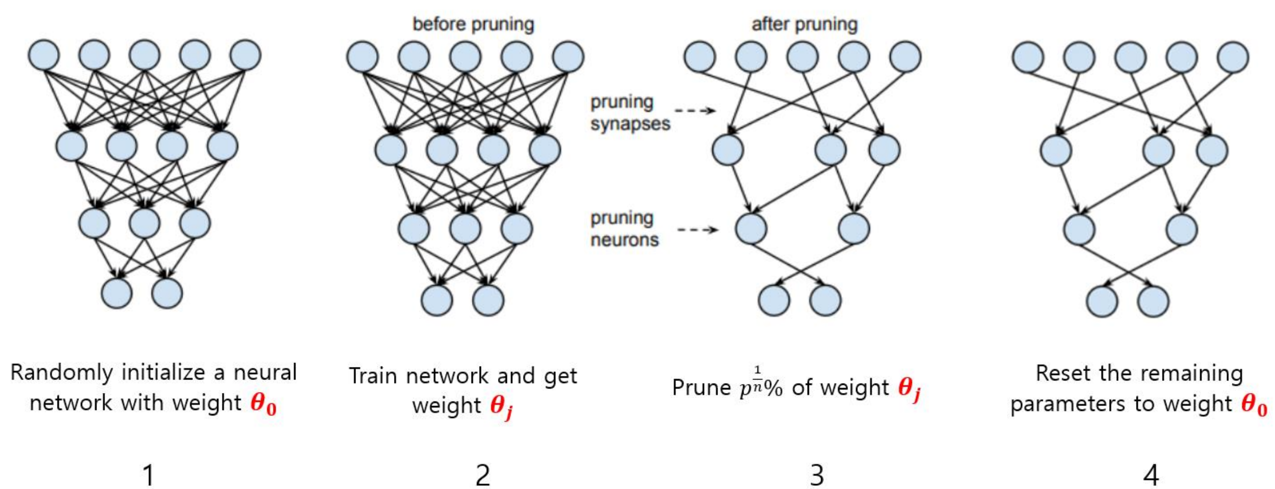
$$s_j = \begin{cases} |w_j|\,, & \text{for magnitude based} \\ \dfrac{w_j^2 H_{jj}}{2} \quad \text{or} \quad \dfrac{w_j^2}{2H_{jj}^{-1}} & \text{for Hessian based}\,. \end{cases}$$

- Gradient-based saliency score
  - $s_j = \left|\frac{\partial \mathcal{L}}{\partial w_j} w_j\right|$ (Gate Decorator, SNIP, …)

$$\mathcal{L}(\theta) = \mathcal{L}(\theta^*) + \left.\frac{\partial \mathcal{L}(\theta)}{\partial \theta}\right|_{\theta^*} (\theta - \theta^*) + \frac{1}{2}(\theta - \theta^*)^T H (\theta - \theta^*)$$

Lottery Ticket Hypothesis (ICLR 2019)

- Can you reinitialize the pruned network and train to the same accuracy?
    - Well, No. (Actually, yes.)
- (J Frankle et al. 2019) shows that it is possible (with the correct initialization):



- Perhaps with the correct method, we could perform *pruning before training*.
- This lead us to the Lottery Ticket Hypothesis:

> A randomly initialized network contains a sub-network (“a lottery ticket”) that is initialized such that – when trained in isolation – it can match the test accuracy of the original network after training for at most the same number of iterations.

Single-shot Network Pruning based on Connection Sensitivity (ICLR 2019)

- SNIP:
  - prune at initialization
  - single-shot (no prune-retrain cycles)
  - saliency metric: *connection sensitivity* (gradient of loss against connectivity)
- Think of connectivity as on-off switch $\boldsymbol{c}$ for each weight. ($\boldsymbol{w} \mapsto \boldsymbol{c} \odot \boldsymbol{w}_0$)
  Pruning an initial weight $\boldsymbol{w}_j$ is equivalent to setting $\boldsymbol{c}_j = 0$.
- We are interested in $\Delta\mathcal{L}_j$ (change in loss when $\boldsymbol{c}_j \longrightarrow 0$)

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

- To efficiently estimate $\Delta\mathcal{L}$, we take gradient of $\mathcal{L}$ at $\mathbf{c} = 1$. $\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}\right)$
  - In other words, linearly approximate $\mathcal{L}$ wrt $\mathbf{c}$.

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

Single-shot Network Pruning based on Connection Sensitivity (ICLR 2019)

- Now we define *connection sensitivity* as normalized ($\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_j}$ at $\boldsymbol{c} = 1$).
  - Notice that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_j}$ is equivalent to $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_j} \boldsymbol{w}_j$. (by chain rule)

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

- Overall algorithm

Algorithm 1 SNIP: Single-shot Network Pruning based on Connection Sensitivity

Require: Loss function L, training dataset $\mathcal{D}$, sparsity level κ ▷ Refer Equation 3
Ensure: $\|\mathbf{w}^*\|_0 \leq \kappa$
$\mathbf{w} \leftarrow$ VarianceScalingInitialization ▷ Refer Section 4.2
$\mathcal{D}^b = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{b} \sim \mathcal{D}$ ▷ Sample a mini-batch of training data
$s_j \leftarrow \frac{|g_j(\mathbf{w}; \mathcal{D}^b)|}{\sum_{k=1}^{m} |g_k(\mathbf{w}; \mathcal{D}^b)|}\,, \quad \forall j \in \{1 \dots m\}$ ▷ Connection sensitivity
$\tilde{\mathbf{s}} \leftarrow \text{SortDescending}(\mathbf{s})$
$c_j \leftarrow \mathbb{1}[s_j - \tilde{s}_\kappa \geq 0]\,, \quad \forall j \in \{1 \dots m\}$ ▷ Pruning: choose top-κ connections
$\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w} \in \mathbb{R}^m} L(\mathbf{c} \odot \mathbf{w}; \mathcal{D})$ ▷ Regular training
$\mathbf{w}^* \leftarrow \mathbf{c} \odot \mathbf{w}^*$

Performance

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**

Table 2: Pruning results of the proposed approach on various modern architectures (before $\rightarrow$ after). AlexNets, VGGs and WRNs are evaluated on CIFAR-10, and LSTMs and GRUs are evaluated on the sequential MNIST classification task. The approach is generally applicable regardless of architecture types and models and results in a significant amount of reduction in the number of parameters with minimal or no loss in performance.

Conclusion

- SNIP is the earliest attempt to find the "winning ticket" at initialization.
  - Pruning at initialization based on connection sensitivity.
- The experiments indicate that initialization scheme matters significantly:
  - we should use variance-scaled Gaussians.
  - tune the variance of weights so that $Var[\boldsymbol{h}]$ remains constant regardless of fan-ins.
  - LeCun / Xavier initialization (sigmoid, tanh): $\boldsymbol{w} \sim \mathcal{N}\left(0, \frac{1}{N_{in}}\right)$ or $\mathcal{N}\left(0, \frac{2}{N_{in}+N_{out}}\right)$
  - He initialization (ReLU): $\boldsymbol{w} \sim \mathcal{N}\left(0, \frac{2}{N_{in}}\right)$

- Question: Why does pruning at initialization work?
  - Does not seem to be clear in a lot of current literatures.
  - Some literatures hint to convergence analysis of NTK. (theory on infinitely wide networks)
    - GraSP (C Wang and R Grosse et al. 2020)
  - With infinitely wide network, the kernel Θ_t remains constant throughout training.

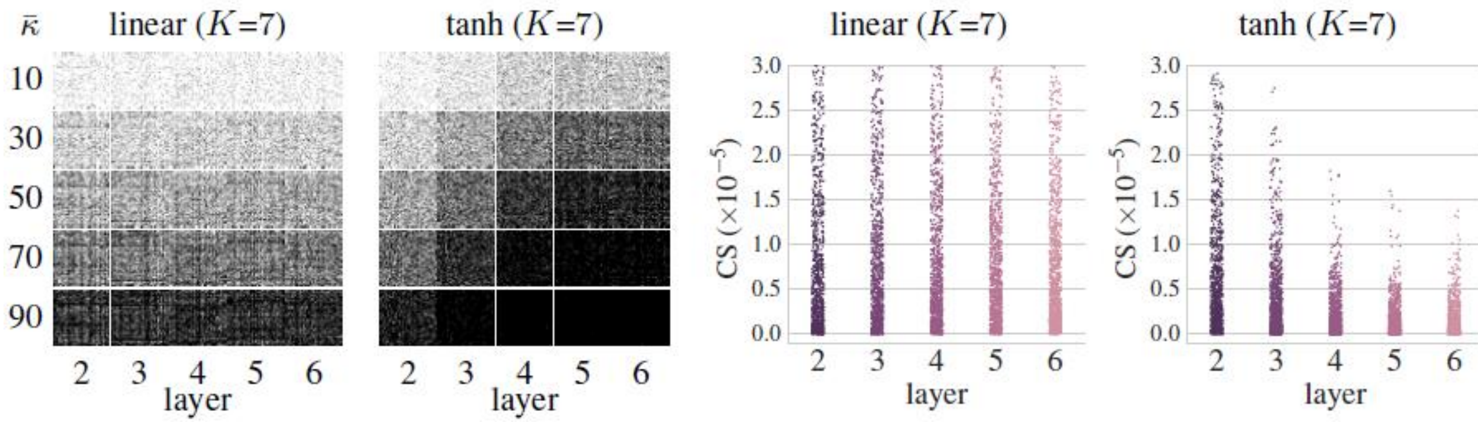
A signal propagation perspective for pruning neural networks at initialization (ICLR 2020)

- Signal propagation perspective of connection sensitivity:
  - connection sensitivity is the product of weight and gradient.
  - Connection sensitivity is a backpropagated signal.
  - We need “faithful” connection sensitivity.

“Faithful signals” – Signals propagating in a network isometrically with minimal amplification or attenuation. (Saxe et al. 2014)

- Poor initialization leads to poor signal propagation.
  - We demand that per-layer Jacobian satisfies Dynamical Isometry.
  - Isometric transformation: preserves the norm and angles between vectors. (reflection, rotation, translation)
  - Signal propagation can be characterized by singular values of Jacobian.

Initialization matters for pruning



Connectivity
(black dot indicates pruned weight)

Connection sensitivity

- The example demonstrates the effect of bad gradient propagation:
  - linear network vs. nonlinear (tanh) network.
  - Weights are initialized with variance-scaling Gaussian.
  - Sparsity is biased towards the later layers, lower connection sensitivity.
- The layer collapse problem
  - The bias gets stronger as you increase the variance of initializer.
  - Final layer retains only few weights → the network catastrophically loses ability to train.
  - Pruning a nonlinear network is unreliable when the initializer is not properly scaled.
- The unreliable pruning result of SNIP is due to bad gradient propagation.
  - The gradient signal need to be faithful.

Layerwise Dynamical Isometry (LDI)

- LDI define the condition for faithful gradient propagation.
  - LDI is a slight generalization of Dynamical Isometry (DI) condition.
  - DI has been used to train extremely deep networks (Training 10,000-Layer Vanilla CNN. Xiao et al., ICML 2018)
- Layerwise Dynamical Isometry
  - The layerwise Jacobian is decomposed into weights and derivative of activations:

$$\mathbf{J}^{l-1,l} = \frac{\partial \mathbf{x}^l}{\partial \mathbf{x}^{l-1}} = \mathbf{D}^l \mathbf{W}^l$$

  - A network satisfies layerwise dynamical isometry if the layerwise Jacobian singular values (JSV) are close to 1.
- Orthogonal initialization
  - We initialize $\mathbf{W}$ such that layerwise Jacobian is orthogonal matrix. ($\mathbf{J}^{\mathrm{T}}\mathbf{J} = (\mathbf{DW})^{\mathrm{T}}(\mathbf{DW}) = \mathbf{I}$)
  - In practice, we SVD a random matrix. (V^T is orthogonal matrix)
- Variance-scaled Gaussian vs. LDI
  - VS Gaussian: mean squared JSV are close to 1.
  - LDI: all JSV are close to 1.

Layerwise Dynamical Isometry (LDI)

- Question: Why does orthogonal initialization work even when there is nonlinear layer?
  - Even if weight W is orthogonal, the Jacobian is not LDI because of D. (slope of activation)

$$\mathbf{J}^{l-1,l} = \frac{\partial \mathbf{x}^l}{\partial \mathbf{x}^{l-1}} = \mathbf{D}^l \mathbf{W}^l$$

- Mean Field Approximation
  - Preactivations of wide, untrained networks follow a Gaussian distribution. (according to CLT)
  - The mass of preactivation values are focused around zero with bell-shaped curve.
  - Visualization: https://ai.googleblog.com/2020/03/fast-and-easy-infinitely-wide-networks.html
- The preactivations can be placed on linear region of activation function.
  - In this case, $D \approx I$. (or scalar multiple of I, depending on nonlinearity)
- Therefore, orthogonal initialization can satisfy LDI even with nonlinear layer.

Enforcing Approximate LDI

- Pruning breaks LDI property of the initial weights.
  - As we increase sparsity, JSV decreases. (weaker gradient propagation)
- Therefore, after pruning, we adjust weights to approximately satisfy LDI.

$$\min_{\mathbf{W}^l} \|(\mathbf{C}^l \odot \mathbf{W}^l)^T(\mathbf{C}^l \odot \mathbf{W}^l) - \mathbf{I}^l\|_F$$

- Approximate LDI
  - Better gradient propagation
  - quicker convergence during training.
  - Simple, data-free optimization that restores broken LDI condition.
  - We found that enforcing approximate isometry improves the training performance of the pruned network quite significantly.

Unsupervised Pruning

- Pruning with unlabeled data
  - Replace the target label as uniform distribution.
  - Tested on MNIST, Fashion-MNIST with VGG16 and ResNets.
  - Slightly worse, but competitive errors compared to supervised pruning.
- Transfer of pruned network to other dataset
  - Transfer of sparsity works. (kind of.)

Table 5: Transfer of sparsity experiment results for LeNet. We prune for $\bar{\kappa} = 97\%$ at orthogonal initialization, and report gen. errors (average over 10 runs).

Category	**Dataset** prune	train&test	**Error** sup. $\rightarrow$ unsup.	(Δ)	**Error** rand
Standard	MNIST	MNIST	2.42 $\rightarrow$ 2.94	+0.52	15.56
Transfer	F-MNIST	MNIST	2.66 $\rightarrow$ 2.80	**+0.14**	18.03
Standard	F-MNIST	F-MNIST	11.90 $\rightarrow$ 13.01	+1.11	24.72
Transfer	MNIST	F-MNIST	14.17 $\rightarrow$ 13.39	**-0.78**	24.89

Conclusion

- The authors identifies key challenge to pruning at initialization.
    - Main difficulty comes from degradation of signal propagation.
    - Correct initializer is critical for connection sensitivity measurement.
    - The trainability of pruned network is linked with the singular values of Jacobian.
- To tackle this challenge, the authors propose:
    - Layerwise dynamical isometry condition for layerwise Jacobians.
    - Apply orthogonal initialization when pruning at initialization.
- Additionally,
    - Enforcing approximate LDI improves signal propagation of pruned network.
    - Interesting idea of unsupervised pruning. (at initialization)

Key Challenges in recent works

- Pruning at initialization.
- Pruning at single-shot.
- Pruning without supervision.
    - A signal propagation perspective … (ICLR 2020)
    - SynFlow (arXiv 2020) (pruning without looking at the data)
- Still, there is performance gap between foresight pruning and traditional pruning.
    - On top of that, recent study (rewinding. ICLR 2020) shows that lottery tickets with late resetting trains to higher accuracy than original initialization.

Recent works in foresight pruning

- Synaptic saliency (arXiv 2020-06) is a general class of gradient-based saliency scores:

$$\mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta$$

- SNIP (ICLR 2019): $\mathcal{R}$ is the training loss $\mathcal{L}$

$$\mathcal{S}(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|$$

- GraSP (ICLR 2020):

$$\mathcal{S}(\theta) = \left| H \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|$$

- Data-free pruning:
    - SynFlow (Pruning neural networks without any data by iteratively conserving synaptic flow. arXiv 2020)

References

- Jonathan Frankle, Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations* (ICLR), 2019
- Namhoon Lee, Thalaiyasingam Ajanthan, Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations* (ICLR), 2019
- Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, Philip H. S. Torr. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. In *International Conference on Learning Representations* (ICLR), 2020
- Chaoqi Wang, Guodong Zhang, Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations* (ICLR), 2020
- Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. arXiv 9 Jun 2020