AutoPrune: Automatic Network Pruning by Regularizing Auxiliary Parameters

Xia Xiao, Zigeng Wang, Sanguthevar Rajasekaran

Department of Computer Science and Engineering, University of Connecticut

Abstract

Reducing the model redundancy is an important task to deploy complex deep learning models to **resource-limited or time-sensitive devices**. Directly regularizing or modifying weight values makes pruning procedure less robust. To build a easy-to-use pruning method, we propose **AutoPrune**, which prunes the network through optimizing a set of **trainable auxiliary parameters** instead of original weights. The instability and noise during training on auxiliary parameters will not directly affect weight values, which makes pruning process more **robust to noise and less sensitive to hyperparameters**. Moreover, we design gradient update rules for auxiliary parameters to keep them **consistent** with pruning tasks. Our method can automatically eliminate network redundancy with **recoverability**, relieving the complicated prior knowledge required to design thresholding functions.

Application domains: *wearable devices, automatic driving, IoT devices*

Motivation

Benefits of a Pruned Light-Weight Neural Network:

- **Smaller size**: Reduce the model size and easy to store
- **Faster inference**: Accelerate model prediction/inference speed
- **Faster training**: Shrink training time for real time deployment
- **Same accuracy**: Slim a complicated model without accuracy drop

Limitations of State-of-the-Art Models:

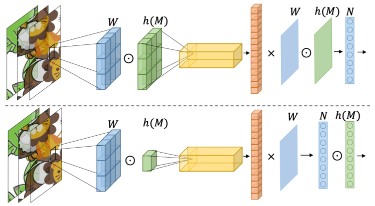
- **Necessary expert knowledge**: Prior knowledge is required to tune different hyperparameters for different models.
- **Unstable pruning process**: Pruning directly on weights/neurons makes pruning less robust and sensitive to hyperparameters.

Contribution

1) We offer a gradient **based automatic network pruning model**;
2) we propose **novel and weakly coupled update rule** for auxiliary parameters to stabilize pruning procedure;
3) we **reduce the sub-graph discrepancy** by iteratively evaluating recoverable sub-graph;
4) we evaluate different **smooth approximations** of the derivative of the rectifier;
5) we obtain the state-of-art results on both **structure and weight pruning** and our method is **scalable** on modern models and datasets.

Methods

Pruning framework:

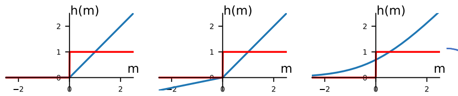


Problem formulation:

$$\min_{w} \mathcal{L}_1 = \min_{w} \sum_{i=1}^{N} \mathcal{L}(f(x_i, W \odot h(M)), y_i) + \lambda \mathcal{R}(W),\ x_i \in X_{train},$$

$$\min_{m} \mathcal{L}_2 = \min_{m} \sum_{i=1}^{N} \mathcal{L}(f(x_i, W \odot h(M)), y_i) + \mu \mathcal{R}(h(M)),\ x_i \in X_{val},$$

Different substitute gradient:



Update auxiliary parameters:

$$m_{ij} := m_{ij} - \eta\left(\frac{\partial \mathcal{L}_{acc}}{\partial t_{ij}} sgn(w_{ij}) \frac{\partial h(m_{ij})}{\partial m_{ij}}\right) - \mu \frac{\partial h(m_{ij})}{\partial m_{ij}}$$

- **Sensitivity Consistency:** Smaller the weight, more sensitive the corresponding auxiliary parameter. $\frac{\partial \mathcal{L}_{acc}}{\partial m_{ij}} \propto \frac{1}{f(|w_{ij}|)}$
- **Correlation Consistency**: The gradient of an arbitrary auxiliary parameter is the same as the direction of the gradient of its corresponding weight.
$$sgn(\frac{\partial \mathcal{L}_2}{\partial m_{ij}}) = sgn(\frac{\partial \mathcal{L}_1}{\partial |w_{ij}|})$$
- **Direction Consistency:** The inner product between the expected coarse and population gradients is greater than zero.

Recoverable pruning:

- Follow the idea of Dynamic Network Surgery.
- Incorrectly pruned weights will be recovered to compensate for the increase of loss.
- Reduce discrepancy between parent graph and child graph.

Experiments

Neuron Pruning:

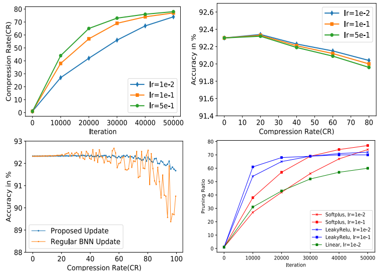
Table 1: Comparison of Different Neuron Pruning Techniques

Model	Methods	Base Error	Error	Epochs	Neurons per Layer	NCR	FLOPs
LeNet-300-100 784-300-100	Louizos *et al.* [2017]	1.60%	1.80%	-	278-98-13	3.04	11%
	Louizos *et al.* [2018]	-	**1.40%**	200	219-214-100	2.22	26%
	Louizos *et al.* [2018]	-	1.80%	200	266-88-33	3.06	10%
	Our method	1.60%	1.82%	**100**	244-85-37	**3.23**	**9%**
LeNet5 (MNIST) 20-50-800-500	Wen *et al.* [2016]	-	1.00%	-	3-12-800-500	1.04	25%
	Neklyudov *et al.* [2017]	-	0.86%	-	2-18-284-283	2.33	9%
	Louizos *et al.* [2017]	0.90%	1.00%	-	5-10-76-16	12.8	**7%**
	Louizos *et al.* [2018]	-	0.90%	200	20-25-45-462	2.48	50%
	Louizos *et al.* [2018]	-	1.00%	200	9-18-65-25	**11.71**	17%
	Our method	0.78%	**0.80%**	**100**	4-16-86-87	9.86	**7%**
VGG-like (CIFAR-10) 64x2-128x2-256x3-512x7	Li *et al.* [2017]	6.75%	**6.60%**	**40**	32-64-128-128-256-256-256-256-256-256-256-256-256-512	1.49	66%
	Neklyudov *et al.* [2017]	7.20%	7.50%	-	64-62-128-126-234-155-31-79-73-9-59-73-56-27	4.03	43%
	Neklyudov *et al.* [2017]	7.20%	9.00%	-	44-54-92-115-234-155-31-76-55-9-34-35-21-280	3.83	32%
	Our method	7.60%	8.50%	150	37-41-91-89-156-140-74-81-54-51-44-46-48-52	**4.72**	**23%**

Table 3: MobileNetV2(Top 1 Accuracy)

FLOPs	Methods	FLOPs	Accuracy
100M	Sandler *et al.* [2018]	97M	65.40%
	Yu *et al.* [2018]	97M	64.40%
	Yu and Huang [2019b]	97M	65.10%
	Our method	102M	**66.83%**
200M	Sandler *et al.* [2018]	209M	69.80%
	Tan *et al.* [2019]	216	71.5%
	Yu and Huang [2019b]	209M	69.60%
	Wu *et al.* [2019]	246M	73%
	Yu and Huang [2019a]	207M	73%
	Our method	209M	**73.32%**
300M	Sandler *et al.* [2018]	300M	69.80%
	Tan *et al.* [2019]	317M	74%
	Yu and Huang [2019a]	305M	**74.20%**
	Our method	305M	74.0%

Hyperparameter Sensitivity:



Weight Pruning:

Table 4: Comparison of Different Weight Pruning Techniques

Model	Methods	Error	CR
LeNet300-100 (MNIST)	Dong *et al.* [2017]	1.76%→2.43%	66.7
	Ullrich *et al.* [2017]	1.89%→1.94%	64
	Molchanov *et al.* [2017]	1.64%→1.92%	68
	Our method	1.72%→ **1.78%**	**80**
LeNet5 (MNIST)	Guo *et al.* [2016]	0.91%→0.91%	108
	Ullrich *et al.* [2017]	0.88%→0.97%	162
	Molchanov *et al.* [2017]	0.80%→ **0.75%**	280
	Li *et al.* [2018]	0.91%→0.91%	298
	Our method	0.78%→0.80%	260
	Our method	0.91%→0.91%	**310**
VGG-like (CIFAR-10)	Zhuang *et al.* [2018]	6.01%→**5.43%**	15.58
	Zhu *et al.* [2018]	6.42%→6.69%	8.5
	Molchanov *et al.* [2017]	7.55%→7.55%	65
	Our method	7.60%→7.82%	**75**
AlexNet (ILSVRC12)	Guo *et al.* [2016]	43.42%→43.09%	17.7
	Srinivas *et al.* [2017]	42.80%→**43.04%**	10.3
	Dong *et al.* [2017]	43.30%→50.04%	9.1
	Our method	43.26%→44.10%	**18.5**
ResNet50 (ILSVRC12)	Zhuang *et al.* [2018]	23.99%→**25.05%**	2.06
	Our method	25.10%→25.50%	**2.2**

References

1. Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In International Conference on Learning Representations, 2018.
2. Miguel A Carreira-Perpinán and Yerlan Idelbayev. “learning-compression” algorithms for neural net pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8532–8541, 2018.
3. Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In International Conference on Learning Representations, 2019.
4. Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In International Conference on Learning Representations, 2019.

Acknowledgement

This work has been supported in part by the following NSF grants: 1447711, 1514357, 1743418, and 1843025.