AutoPrune: Automatic Network Pruning by Regularizing Auxiliary Parameters

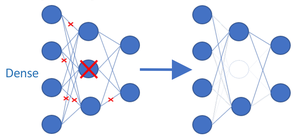
Xia Xiao, Zigeng Wang, Sanguthevar Rajasekaran

Department of Computer Science and Engineering, University of Connecticut, CT, USA

Introduction

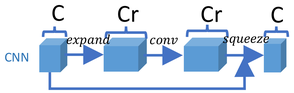
➢ **Problem: Find a sub-architecture that maximizes sparsity and accuracy.**

1) Find a proper substructure that has the same validation/test performance.
2) Choose the proper size of channels and expansion/squeeze ratio automatically.
3) Make the winning ticket as sparse as possible.



➢ **Contributions:**

1) Achieve robust and stable pruning.
2) Achieve hyperparameter insensitive pruning.
3) Automatic global network pruning.
4) Support train from scratch.
5) Naturally support both weight and neuron pruning.



➢ **Proposal: Use nested auxiliary network (parameters) to sparsify original network.**

$$\underset{w,m}{\operatorname{argmin}} \frac{1}{N}\left(\sum_{i=1}^{N}\mathcal{L}(f(x_i, W\odot h(M)), y_i)\right) + \lambda\mathcal{R}(W) + \mu\mathcal{R}(h(M))$$

1) Propose gradient based bi-level optimization framework.
2) Propose decoupled training for auxiliary network.
3) Propose non-zero smooth gradient approximations for recoverable pruning.

Benefits and Applications

➢ **Benefits of a Pruned Light-Weight Neural Network:**

- **Smaller size**: Reduce the model size and enables easy storage.
- **Faster inference**: Accelerate model prediction/inference speed .
- **Faster training**: Shrink training time for real time deployment.
- **High accuracy**: Simplify a complex model with no accuracy drop.
- **Winning tickets**: Find a sparse winning lottery ticket.

➢ **Applications:**

Wearable devices, Mobile phones, Vehicular networking, Automatic cars, Surveillance equipment, robots, IoTs/Resource limited devices.

Related Works:

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.

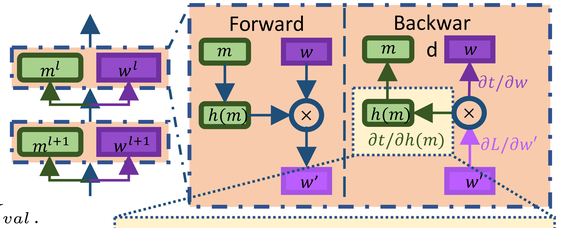
Methods

➢ **Bi-level optimization:**

Optimize two sets of parameters with bi-level optimization:

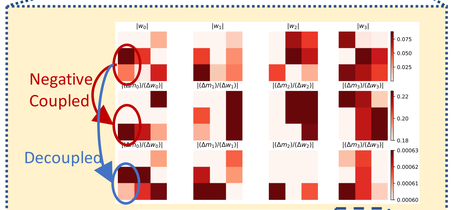
Weight parameters(w)

Auxiliary parameters(m)



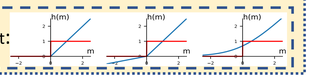
➢ **Decoupled Training:**

1) Sample batch $x_i \in X_{val}$.
2) Compute architecture loss $\sum_{i=1}^{N}\mathcal{L}(f(x_i, W\odot h(M)), y_i)$
3) Update m using update rule.
4) Sample batch $x_i \in X_{train}$.
5) Compute weight loss and update weights w.



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

Coarse gradient:



➢ **Regularization:**

For Weight parameters (w) $||w||_1, or ||w||_2$

For Auxiliary parameters (m): $\mathcal{R}(h(M)) = \sum_{i,j}|h(m_{ij})| = count(h(M))$

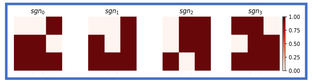
➢ **Sensitivity Consistency:** Sensitivity of auxiliary parameters are negatively proportional to weight magnitude.

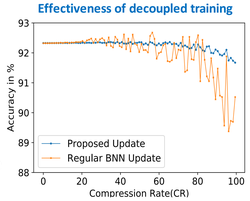
$$\frac{\partial\mathcal{L}_{acc}}{\partial m_{ij}} \propto \frac{1}{f(|w_{ij}|)}$$

➢ **Correlation Consistency**: The gradient direction of an arbitrary auxiliary parameter is the same as the direction of the gradient of its corresponding weight.

$$sgn\left(\frac{\partial\mathcal{L}_2}{\partial m_{ij}}\right) = sgn\left(\frac{\partial\mathcal{L}_1}{\partial|w_{ij}|}\right)$$

➢ **Direction Consistency:** The inner product between the expected coarse and population gradients is non-negative.



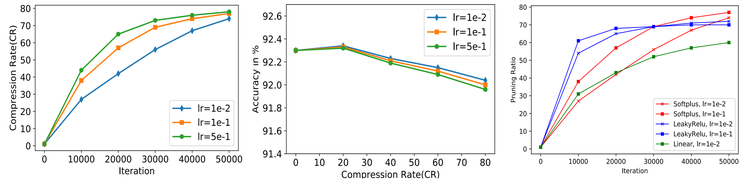


➢ **Recoverability**:

1) Approximate gradient with strictly monotone functions.
2) Incorrectly pruned weights will be recovered to compensate for the increase of loss.
3) Reduce the discrepancy between the parent graph and the child graphs.

Experiments

➢ We show that the pruning sparsity and accuracy are not sensitive to learning rate and the choice of STEs (taking weight pruning with VGG-like on CIFAR-10 as an example).



(a) CR vs. Iter (b) Accuracy vs. CR (c) Different STEs

Figure 1. Illustration of Hyperparameter Sensitivity

Table 1. Neuron Pruning on LeNet and VGG

Model	Methods	Base Error	Error	Epochs	Neurons per Layer	NCR	FLOPs
LeNet-300-100	Louizos *et al.* [2017]	1.60%	1.80%	-	278-98-13	3.04	11%
784-300-100	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	Wen *et al.* [2016]	-	1.00%	-	3-12-800-500	1.04	25%
(MNIST)	Neklyudov *et al.* [2017]	-	0.86%	-	2-18-284-283	2.33	9%
20-50-	Louizos *et al.* [2017]	0.90%	1.00%	-	5-10-76-16	12.8	**7%**
800-500	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	Li *et al.* [2017]	6.75%	**6.60%**	**40**	32-64-128-128-256-256-256-256-256-256-256-256-512	1.49	66%
(CIFAR-10)	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%
64x2-128x2-	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%
256x3-512x7	Our method	7.60%	8.50%	150	37-41-91-89-156-140-74-81-54-51-44-46-48-52	**4.72**	**23%**

➢ Our method performs well in terms of accuracy, pruning speed and FLOPs.

Table 2. Neuron Pruning on MobileNet V2

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%

➢ Performs better in extreme cases.

Table 3. Weight Pruning Comparison

Model	Methods	Error	CR
LeNet300-100	Dong *et al.* [2017]	1.76%→2.43%	66.7
(MNIST)	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	Guo *et al.* [2016]	0.91%→0.91%	108
(MNIST)	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	Zhuang *et al.* [2018]	6.01%→**5.43%**	15.58
(CIFAR-10)	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	Guo *et al.* [2016]	43.42%→43.09%	17.7
(ILSVRC12)	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	Zhuang *et al.* [2018]	23.99%→**25.05%**	2.06
(ILSVRC12)	Our method	25.10%→25.50%	**2.2**