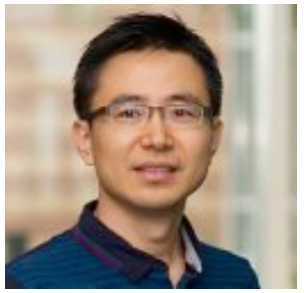
Neural Pruning via Growing Regularization

ICLR 2021 (Poster, Paper ID: 555)


Huan Wang


Can Qin


Yulun Zhang


Yun Fu

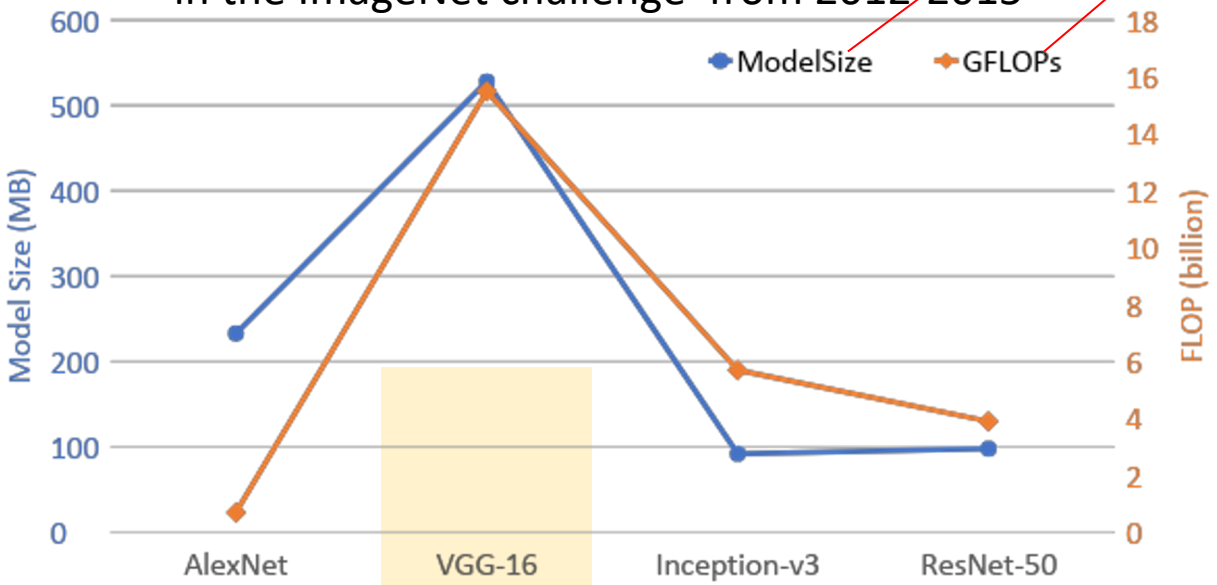
Northeastern University, Boston, MA, USA

Introduction: DNNs Effective, but not efficient (motivation)

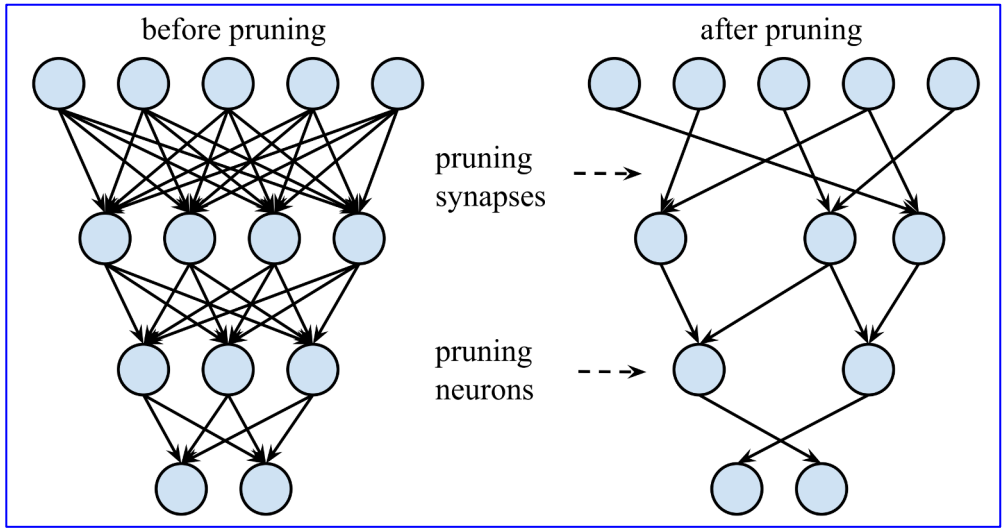
❑ Large DNNs are hard to deploy in practice.

- high storage (space complexity) – ***too big***
- massive computation (time complexity) – ***too slow***

Storage and computation of winners in the ImageNet challenge from 2012-2015



1.7 s/img (224x224 RGB image, on Intel Xeon(R) CPU E5-2620)



[Han et al., 2015 NeurIPS]

Pruning: reduce the parameters while maintain performance

[Han et al., 2015 NIPS] Han, Song, et al. "Learning both weights and connections for efficient neural networks." NeurIPS (2015)

How to do pruning: A typical way and its problem

typical 3-step pruning pipeline

Pruning criterion: Prune the weights whose absence leads to the *least* loss increase.

Solutions:

(1) Trial and error? Too many weights (millions and billions params!) , we cannot ablate them one by one.

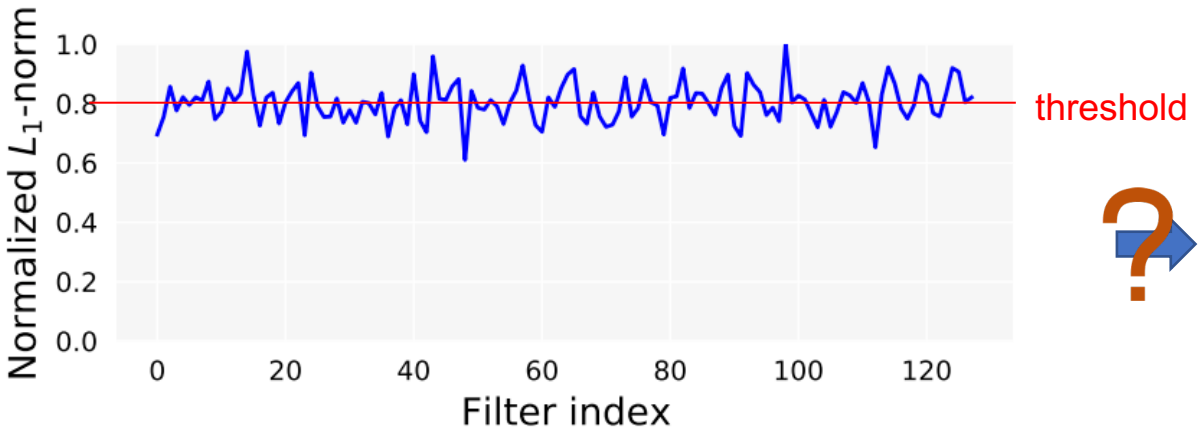
(2) Use some analytical formula ⇒ Taylor expansion ⇒ approximate the Hessian (very hard for DNNs).

$$\Delta\mathcal{L} = \underbrace{\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}^{\top}\Delta\boldsymbol{\theta}}_{\approx 0} + \frac{1}{2}\Delta\boldsymbol{\theta}^{\top}\mathbf{H}\Delta\boldsymbol{\theta} + \mathcal{O}(||\Delta\boldsymbol{\theta}||^3)$$

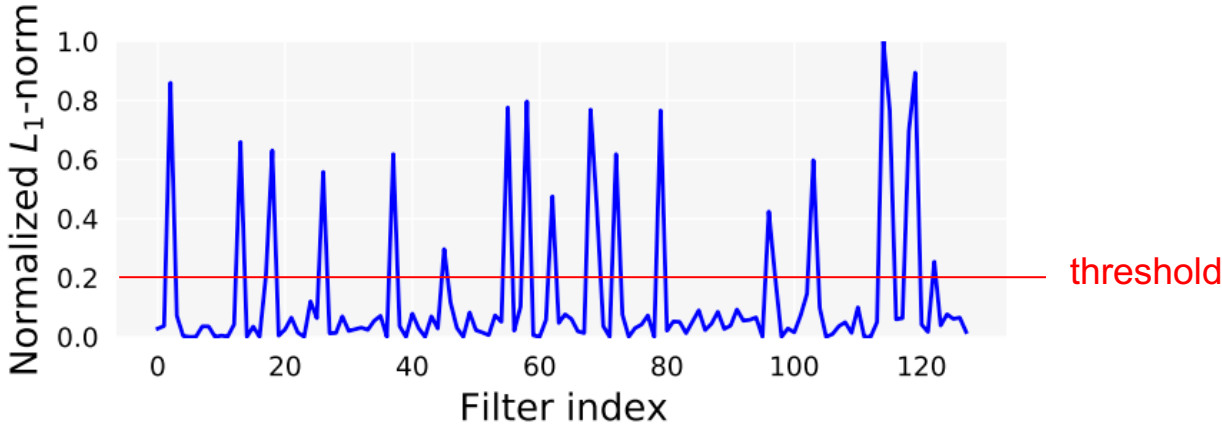
Can we leverage the Hessian *without* knowing their specific values?

Leverage Hessian without knowing Hessian

- How: Hessian can play its role by affecting weight magnitude (L1-norm).
- Magnitude is known to be *not accurate* for pruning? Not necessarily. It *is* accurate as long as the gap is significant enough.



Misjudgments probably happen;
L1-norm is not accurate.



Misjudgments rarely happen;
L1-norm is accurate.

How regularization can help

Parameter update in SGD:

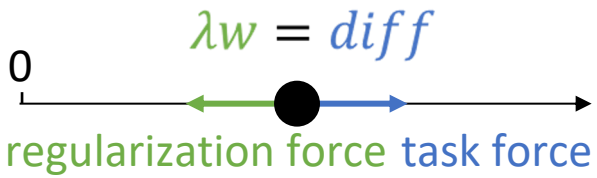
$$w = w - \alpha(diff - \lambda w)$$

α: learning rate, diff: gradient

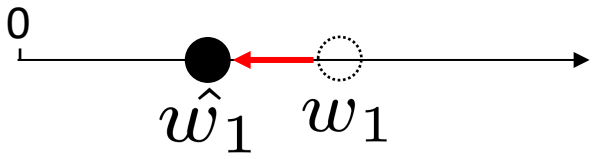
when a weight stops update

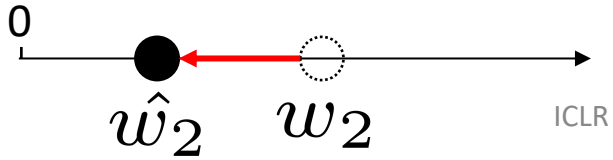
$\lambda w = diff$, where λ is the factor of weight decay.

(Ideally) The final position of a weight will be determined by both the task supervision and the regularization (task-agnostic prior), where λ can hold the balance.



Increase λ a little. See what happens





$$r_1 = \frac{\hat{w}_1}{w_1}, r_2 = \frac{\hat{w}_2}{w_2}$$

How r_1 and r_2 are related to the underlying Hessian?

Theoretical Analysis

$$\hat{\mathbf{w}}^* = (\mathbf{H} + \delta\lambda\,\mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*, \quad \text{(local quadratic approximation)}$$

Case 1: Hessian is diagonal

$$\hat{w}_i^* = \frac{h_{ii}}{h_{ii} + \delta\lambda} w_i^*, \Rightarrow r_i = \frac{\hat{w}_i^*}{w_i^*} = \frac{1}{\delta\lambda/h_{ii} + 1},$$

new converged weight

increased L2 penalty factor

previous converged weight

Case 2: Hessian is not diagonal (2d analysis)

$$\begin{Bmatrix} \hat{w}_1^* \\ \hat{w}_2^* \end{Bmatrix} = \frac{1}{|\hat{\mathbf{H}}|} \begin{Bmatrix} (h_{11}h_{22} + h_{11}\delta\lambda - h_{12}^2)w_1^* + \delta\lambda h_{12} w_2^* \\ (h_{11}h_{22} + h_{22}\delta\lambda - h_{12}^2)w_2^* + \delta\lambda h_{12} w_1^* \end{Bmatrix} \approx \frac{1}{|\hat{\mathbf{H}}|} \begin{Bmatrix} (h_{11}h_{22} + h_{11}\delta\lambda - h_{12}^2)w_1^* \\ (h_{11}h_{22} + h_{22}\delta\lambda - h_{12}^2)w_2^* \end{Bmatrix},$$

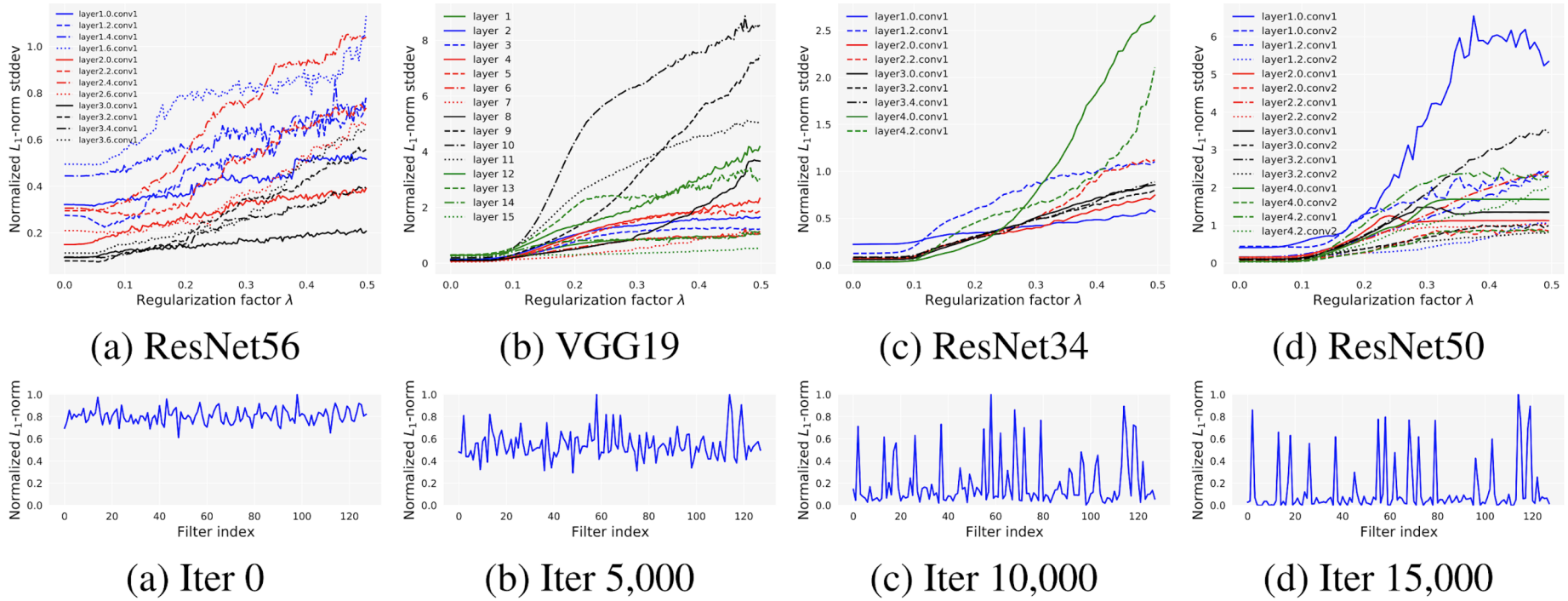
$$\Rightarrow r_1 = \frac{1}{|\hat{\mathbf{H}}|}(h_{11}h_{22} + h_{11}\delta\lambda - h_{12}^2),\ r_2 = \frac{1}{|\hat{\mathbf{H}}|}(h_{11}h_{22} + h_{22}\delta\lambda - h_{12}^2).$$

Conclusion: $h_{11} > h_{22} \Rightarrow r_1 > r_2$

A weight lying on a *sharper* local minimum will be pushed *less* towards zero.

Empirical Validation of the Theoretical Analysis

- The weight magnitude gap will be *larger and larger*.
- Eventually, the simple L1-norm will suffice to make a faithful criterion.



(a) ResNet56 (b) VGG19 (c) ResNet34 (d) ResNet50

(a) Iter 0 (b) Iter 5,000 (c) Iter 10,000 (d) Iter 15,000

Figure 1: Row 1: Illustration of weight separation as L_2 penalty grows. Row 2: Normalized filter L_1-norm over iterations for ResNet50 layer2.3.conv1 (please see the Appendix for VGG19 plots).

Regularization can also help in pruning schedule

$$\mathcal{E}(\mathbf{w}, \mathcal{D}) = \mathcal{L}(\mathbf{w}, \mathcal{D}) + \frac{1}{2}\lambda \|\mathbf{w}\|_2^2,$$
$$\lambda_j = \lambda_j + \delta\lambda, \text{ for all } j.$$

GReg-2

$$\mathcal{E}(\mathbf{w}, \mathcal{D}) = \mathcal{L}(\mathbf{w}, \mathcal{D}) + \frac{1}{2}\lambda \|\mathbf{w}\|_2^2,$$
$$\lambda_j = \lambda_j + \delta\lambda, j \in \{j \mid M[j] = 0\}$$

GReg-1

- Same pruning criterion as L1-norm pruning [1]: sort all the filters by their L1-norm, select those with the least L1-norms to prune (i.e., mask = 0).
- Two pruning schedules: (1) one-shot: remove the unimportant weights immediately; (2) pushing them towards zero first by the proposed growing regularization, then remove them.

[1] Li, Hao, et al. “Pruning Filters for Efficient ConvNets.” ICLR (2017).

Regularization can also help in pruning schedule

Table 1: Comparison between pruning schedules: one-shot pruning vs. our proposed GReg-1. Each setting is randomly run for 3 times, mean and std accuracies reported.

ResNet56 + CIFAR10: Baseline accuracy 93.36%, #Params: 0.85M, FLOPs: 0.25G					
Pruning ratio r (%)	50	70	90	92.5	95
Sparsity (%) / Speedup	49.82/1.99×	70.57/3.59×	90.39/11.41×	93.43/14.76×	95.19/19.31×
Acc. (%, L_1+one-shot)	$92.97_{\pm 0.15}$	$91.88_{\pm 0.09}$	$87.34_{\pm 0.21}$	$87.31_{\pm 0.28}$	$82.79_{\pm 0.22}$
Acc. (%, GReg-1, ours)	$\mathbf{93.06}_{\pm \mathbf{0.09}}$	$\mathbf{92.23}_{\pm \mathbf{0.21}}$	$\mathbf{89.49}_{\pm \mathbf{0.23}}$	$\mathbf{88.39}_{\pm \mathbf{0.15}}$	$\mathbf{85.97}_{\pm \mathbf{0.16}}$
Acc. gain (%)	0.09	0.35	2.15	1.08	3.18
VGG19 + CIFAR100: Baseline accuracy 74.02%, #Params: 20.08M, FLOPs: 0.80G					
Pruning ratio r (%)	50	60	70	80	90
Sparsity (%) / Speedup	74.87/3.60×	84.00/5.41×	90.98/8.84×	95.95/17.30×	98.96/44.22×
Acc. (%, L_1+one-shot)	$71.49_{\pm 0.14}$	$70.27_{\pm 0.12}$	$66.05_{\pm 0.04}$	$61.59_{\pm 0.03}$	$51.36_{\pm 0.11}$
Acc. (%, GReg-1, ours)	$\mathbf{71.50}_{\pm \mathbf{0.12}}$	$\mathbf{70.33}_{\pm \mathbf{0.12}}$	$\mathbf{67.35}_{\pm \mathbf{0.15}}$	$\mathbf{63.55}_{\pm \mathbf{0.29}}$	$\mathbf{57.09}_{\pm \mathbf{0.03}}$
Acc. gain (%)	0.01	0.06	1.30	1.96	5.73

Given *exactly the same* weights to prune, different pruning schedules can lead to starkly different performances.

- GReg-1 > One-shot.
- Larger pruning ratio, the advantage of GReg-1 is more pronounced.

Experimental Results (ImageNet, Filter Pruning)

Table 3: Acceleration comparison on ImageNet. FLOPs: ResNet34: 3.66G, ResNet50: 4.09G.

Method	Network	Base top-1 (%)	Pruned top-1 (%)	Top-1 drop	Speedup
L_1 (pruned-B) Li et al. (2017)	ResNet34	73.23	72.17	1.06	**1.32**×
Taylor-FO Molchanov et al. (2019)		73.31	72.83	0.48	1.29×
GReg-1 (**ours**)		73.31	73.54	-0.23	**1.32**×
GReg-2 (**ours**)		73.31	**73.61**	**-0.30**	**1.32**×
ProvableFP Liebenwein et al. (2020)	ResNet50	76.13	75.21	0.92	1.43×
GReg-1 (**ours**)		76.13	**76.27**	**-0.14**	**1.49**×
AOFP Ding et al. (2019b)	ResNet50	75.34	75.63	-0.29	**1.49**×
GReg-2 (**ours**)*		75.40	**76.13**	**-0.73**	**1.49**×
IncReg Wang et al. (2019b)	ResNet50	75.60	72.47	3.13	2.00×
SFP He et al. (2018a)		76.15	74.61	1.54	1.72×
HRank Lin et al. (2020a)		76.15	74.98	1.17	1.78×
Taylor-FO Molchanov et al. (2019)		76.18	74.50	1.68	1.82×
Factorized Li et al. (2019)		76.15	74.55	1.60	**2.33**×
DCP Zhuang et al. (2018)		76.01	74.95	1.06	2.25×
CCP-AC Peng et al. (2019)		76.15	75.32	0.83	2.18×
GReg-1 (**ours**)		76.13	75.16	0.97	2.31×
GReg-2 (**ours**)		76.13	**75.36**	**0.77**	2.31×
C-SGD-50 Ding et al. (2019a)	ResNet50	75.34	74.54	0.80	2.26×
AOFP Ding et al. (2019b)		75.34	75.11	0.23	**2.31**×
GReg-2 (**ours**)*		75.40	**75.22**	**0.18**	**2.31**×
LFPC He et al. (2020)	ResNet50	76.15	74.46	1.69	2.55×
GReg-1 (**ours**)		76.13	74.85	1.28	**2.56**×
GReg-2 (**ours**)		76.13	**74.93**	**1.20**	**2.56**×
IncReg Wang et al. (2019b)	ResNet50	75.60	71.07	4.53	3.00×
Taylor-FO Molchanov et al. (2019)		76.18	71.69	4.49	3.05×
GReg-1 (**ours**)		76.13	73.75	2.38	**3.06**×
GReg-2 (**ours**)		76.13	**73.90**	**2.23**	**3.06**×

* Since the base models of C-SGD and AOFP have a much lower accuracy than ours, for fair comparison, we train our own base models with similar accuracy.

Experimental Results (ImageNet, Unstructured Pruning)

Table 4: Compression comparison on ImageNet with ResNet50. #Parameters: 25.56M.

Method	Base top-1 (%)	Pruned top-1 (%)	Top-1 drop	Sparsity (%)
GSM Ding et al. (2019c)	75.72	74.30	1.42	80.00
Variational Dropout Molchanov et al. (2017a)	76.69	75.28	1.41	80.00
DPF Lin et al. (2020b)	75.95	74.55	1.40	82.60
WoodFisher Singh & Alistarh (2020)	75.98	75.20	0.78	**82.70**
GReg-1 (**ours**)	76.13	**75.45**	**0.68**	**82.70**
GReg-2 (**ours**)	76.13	75.27	0.86	**82.70**

The proposed methods can work *seamlessly* from filter pruning to unstructured pruning.

- Filter pruning: weight group is a filter.
- Unstructured pruning: weight group is a single weight element.

Conclusion

- we present two algorithms that exploit regularization in a new fashion that the penalty factor is uniformly raised to a large amount.

- The two algorithms show that:
  - GReg-1: Pruning schedule is another important axis in pruning, which may deserve more research attention.
  - GReg-2: Without any Hessian approximation, we can still tap into its power for pruning with the help of growing L2 regularization.

- Empirically, both algorithms achieve very promising results compared to many recent methods.

Our code and trained models are released at:
https://github.com/MingSun-Tse/Regularization-Pruning

Great thanks for your attention!

Acknowledgements: This work is supported by the National Science Foundation Award ECCS-1916839 and the U.S. Army Research Office Award W911NF-17-1-0367.