Catalyst: Structured Pruning with Robust Bifurcation Dynamics

Jaeheun Jung WODSOS@KOREA.AC.KR

Donghun Lee* HOLY@KOREA.AC.KR

Korea University, Seoul, Republic of Korea

Abstract

Structured pruning reduces the computational cost of neural networks by removing filters, but conventional regularizers such as L1 or Group Lasso exhibit strong magnitude bias and unstable decision boundaries, suggesting suboptimal pruning dynamics. In this work, we revisit pruning through the lens of optimization, geometry and learning dynamics. We first characterize the precise algebraic conditions under which pruning preserves model outputs, then use this insight to design Catalyst, a novel regularizer defined in an extended parameter space with auxiliary variables. Catalyst reshapes the loss landscape to promote emergent bifurcation dynamics between filters to be pruned or preserved, ensuring magnitude-invariant, fair, and robust pruning decisions. Our formulation highlights how high-dimensional learning dynamics can be achieved via a well-founded regularizer for pruning. Empirically, the Catalyst pruning algorithm consistently outperforms standard approaches, demonstrating both its theoretical soundness and practical effectiveness.

1. Introduction

Structure and sparsity in overparametrized neural networks pose fundamental challenge in understanding dynamics pertaining to deep learning. Structured pruning [3], a model compression technique that removes entire filters or channels, impacts both structure and sparsity of the model while maintaining the model performance. Regularization techniques such as L1 and Group Lasso[16, 19] are commonly employed in structured pruning to encourage sparsity by shrinking filter weights toward zero, thereby mitigating the performance degradation caused by removing filters to be pruned. Despite practical effectiveness, this norm-minimization strategy reduces the potential of structural pruning. In particular, it creates magnitude bias in pruning, favoring the pruning of small-norm filters regardless of their importance.

In this work, we propose **Catalyst**, a novel regularization scheme to enable lossless structured pruning, and analyze the underlying dynamics of regularized pruning of overparametrized neural network unveiled by Catalyst. Catalyst regularization for structured pruning is represented by reshaping the dynamics of learning which filters to prune by augmenting the neural network parameter space with auxiliary "catalyst" variables. We show that Catalyst regularizer is geometrically aligned with the pruning-invariant set for lossless pruning, thereby eliminating magnitude bias and inducing a robust bifurcation: the filters naturally diverge into two groups – prune or preserve –following the gradient flow along trajectories with a wide margin robust to perturbations. This bifurcation behavior reveals a new perspective on pruning as a structured dynamic competition among filters, achieved by analytically well-founded regularizer. We demonstrate this robust dynamics theoretically and empirically, showing that Catalyst enhances both pruning stability and compression performance.

* Corresponding Author

© J. Jung & D. Lee.

2. Catalyst pruning: dynamics for lossless pruning

2.1. Equation of lossless pruning

Given weight matrix W and filters $F_i = W_{i,:}$ for $i = 1, \cdots, C$ where C is number of channels, the structured pruning sets F_i to zero for some i, to eliminate the ith channel. While existing regularization methods attempt to minimize the damage due to pruning by sparsifying the filter parameters, they fall short in one crucial aspect: the vicinity of the global minima of these regularizers does not geometrically align with what we term the pruning-invariant set. We define this pruning-invariant set, X^{tgt} , as a finite union of linear subspaces.

$$X_{tgt} := \bigcup_{i=1}^{N_W} \{W | W_{i,:} = F_i = 0\}. \tag{1}$$

If W is in X_{tgt}, then ith filter F_i is zero for some i and thus we can prune it without any damage.

2.2. Catalyst regularization

Geometry of X_{tgt}, the union of linear spaces is may not be an easy target for the optimization, since the defining equation is complicated. Instead, we consider the scope from the higher dimension. If we introduce auxiliary parameter and extend the weight space, following theorem simplifies the problem.

Theorem 1 *Let $\mathbb{D} = \{D = diag(\delta) | \delta \in \mathbb{R}^{N_W}\}$ be space of diagonal matrices and consider projection map*

$$\begin{aligned} p : \mathbb{R}^{N_W \times N_I} \times \mathbb{D} &\longrightarrow \mathbb{R}^{N_W \times N_I} \\ (W, D) &\longmapsto W \end{aligned} \tag{2}$$

then

(1) $X_{tgt} = p(\{(W, D) | DW = 0 \text{ and } D \neq 0\})$.

(2) Let $B(X, \epsilon)$ be the ϵ-neighborhood of X with the L_2 norm. If $k > 0$ is a positive real number, then

$$B(X_{tgt}, \epsilon) = p(\{(W, D) | \|DW\|_{2,1} < k\epsilon \text{ and } \|D\|_1 > k\}) \tag{3}$$

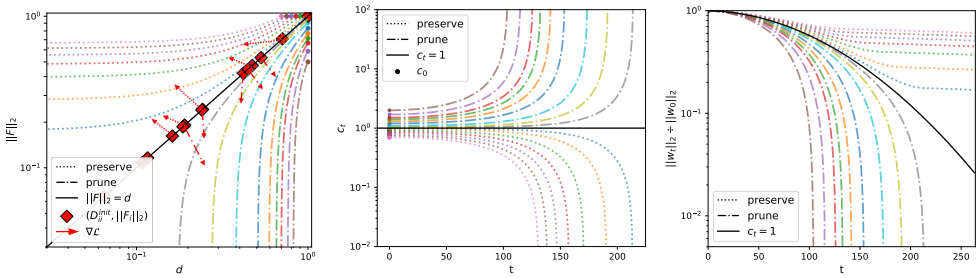
Proof [Proof of Theorem 1] We can easily prove (1) by the fact that $dw = 0$ implies $d = 0$ or $w = 0$. For (2), we directly find D. The detailed proof is in Appendix C. ■

Sending weight matrix W to X_{tgt}, is equivalent to send (W, D) to $V(DW)$, the zero locus of matrix equation $DW = 0$. Hence, we can achieve lossless pruning by minimization of $\|DW\|$. In particular, we choose to minimize $\|DW\|_{2,1} = \sum_{i=1}^{C} \|D_{ii} F_i\|_2$ to invoke robust bifurcation.

In implementation, we exploit Bypass pipeline[4] with slight modification. Instead of the original form of learnable activation $\psi_D(x) = Dx + \sigma(x)$ with $D = 0$ initialization, we introduce a modified version:

$$\psi_{D,\overline{D}} = Dx - \overline{D}x + \sigma(x) \tag{4}$$

with initialization $D = \overline{D} = D^{init}$. Interestingly, our regularization target $DW = 0$ implies $ADW = 0$, the comeback constraint of [4], and thus allow to remove D together without damage.



(a) (d, F) dynamics (b) bifurcation on c_t (c) bifurcation on W_t

Figure 1: Simulation on $\|DW\|_{2,1}$ minimization,

2.3. Dynamics of Catalyst regularization

The minimization of $\|DW\|_{2,1}$ is affected strongly by the initialization D^{init}. To ensure the fair chance of prunability and reduce the magnitude bias, we propose to use $D^{init} = diag(\|F_1\|_2, \cdots, \|F_C\|_2)$. This initialization places the filters (F_i, D_{ii}) on the pruning decision boundary $\frac{D_{ii}}{\|F_i\|} = 1$ described in the theorem below.

Theorem 2

Let d be the scalar parameter and M be N-dimensional vector. M_t and d_t be the trajectory of gradient descent movement of $\|dM\|_2$ at timestep t positive learning rate λ_t and weight decay term $\alpha \ll 1$. Let $c_t = \frac{|d_t|}{\|M_t\|_2}$ and assume that

$$0 < \lambda_* < \lambda_t \ll \min\left(\frac{(1-\alpha)}{c_0}, (1-\alpha)c_0\right) \tag{5}$$

where $\lambda_* = \inf_t \lambda_t$ is the infimum λ_t.

(1) If $c_0 = 1$, then $c_t = 1$ for all timestep t.

(2) If $c_0 < 1$, then c_t exponentially shrinks to $\frac{\lambda_t}{1-\alpha}$. i.e, If T be the smallest integer satisfying $c_T \leq \frac{\lambda_T}{1-\alpha}$ then there exists $k \in (0,1)$ such that $c_{t+1} \leq kc_t$ for all $t < T-1$.

(3) If $c_0 > 1$, then c_t exponentially grows to $\frac{(1-\alpha)}{\lambda_t}$. i.e, if T is the smallest integer satisfying $c_T \geq \frac{1-\alpha}{\lambda_T}$ then there exists $k > 1$ such that $c_{t+1} \geq kc_t$ for all $t < T-1$

Proof We can find recurrence relation between c_t and c_{t+1}. Detailed proofs can be found in Appendix E. ■

During training, due to performance loss $\mathcal{L}$ (red arrows in Fig. 1(a)), the (D_{ii}, F_i) escapes this decision boundary almost surely, and then move according to its position whether the ratio $c_0^{(i)} := \frac{D_{ii}}{\|F_i\|_2}$ is larger than 1 or not. This movement is illustrated in Fig. 1 by dashed lines.

Precisely, according to Theorem 2, if $c_0^{(i)} > 1$ then ith filter parameters F_i the parameters (D_{ii}, F_i) move toward hyperplane $F = 0$ of (d, F)-space and thus ith filter would be pruned. Otherwise if $c_0 < 1$, then the destination would be $d = 0$ and the ith filter would be preserved. This

Table 1: Performance Comparison of Various Filter Pruning Methods

	Method	Baseline ACC(%)	Pruned ACC(%)	Δ ACC(%)	speedup
Resnet56+ CIFAR10	Slimming[9, 21]	93.80	93.27	-0.53	1.92x
	Polar[21]	93.80	93.83	+0.03	1.89x
	SCP [5]	93.69	93.23	-0.46	2.06x
	ResRep[1]	93.71	93.71	0.00	2.12x
	Depgraph[2]	93.53	93.77	+0.24	2.13x
	ATO[17]	93.50	93.74	+0.24	2.22x
	Ours-BN	93.53	94.00	+0.47	2.06x
VGG19+ CIFAR100	SCP[5]	72.56	72.15	-0.41	2.63x
	Greg-1 [15]	74.02	71.30	-2.72	2.96x
	DepGraph[2]	73.50	72.46	-1.04	3.00x
	Ours-BN	73.51	73.37	-0.14	3.00x
	DepGraph[2]	73.50	70.39	-3.11	8.92x
	Greg-2 [15]	74.02	67.75	-6.27	8.84x
	EigenD[14]	73.34	65.18	-8.16	8.80x
	Ours-BN	73.51	70.08	-3.43	8.96x
	DepGraph[2]	73.50	66.20	-7.30	12.00x
	Ours-BN	73.51	68.13	-5.38	11.84x
	Ours-BN	73.51	66.44	-7.07	13.83x
Resnet50+Imagenet	PFP[8]	76.13	75.21	-0.91	1.49x
	Greg-1[15]	76.13	76.27	+0.14	1.49x
	Ours-BN	76.15	76.40	+0.36	1.49x
	ThiNet70[10]	72.88	72.04	-0.84	1.69x
	Whitebox[20]	76.15	75.32	-0.83	1.85x
	Ours-BN	76.15	76.04	-0.11	1.82x
	Slimming[9, 21]	76.15	74.88	-1.27	2.13x
	Polar[21]	76.15	75.63	-0.52	2.17x
	Depgraph[2]	76.15	75.83	-0.32	2.08x
	OICSR[7]	76.31	75.95	-0.37	2.00x
	Ours-group	76.15	75.96	-0.19	1.96x

gives the natural policy of pruning decision, that we can just prune filters with $c_t > 1$ after t steps of regularization. Since the pruning decision is made on c, filters with smaller initial magnitudes are not any more preferred to be pruned than those with larger magnitudes and thus completely resolved the magnitude bias.

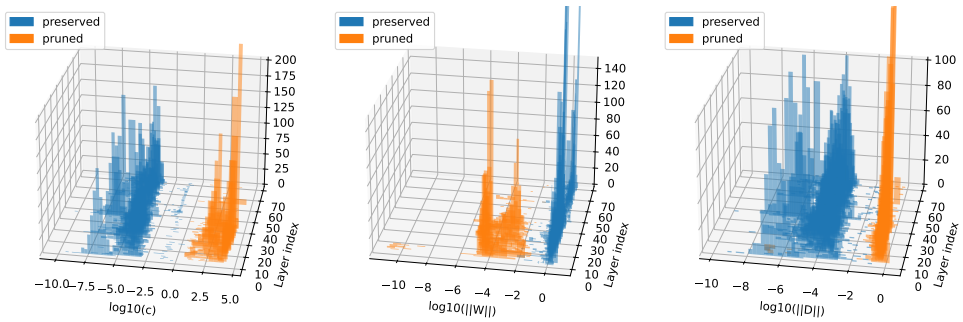
3. Experiments

3.1. Experiment Settings

We use Resnet-56 [18] on CIFAR10 [6], VGG-19 [13] on CIFAR100 and Resnet50 on Imagenet [12] for empirical verification. The detailed experimental settings including augmentation and hyperparameters such as learning rate and number of epochs can be found in Appendix F.

3.2. Performance Analysis

In Table 1, we show the performance of Catalyst, by comparing the test accuracy before and after the overall regularization and pruning process. The speedup is measured by the ratio $\frac{FLOPs \text{ of pretrained}}{FLOPs \text{ of pruned}}$.



(a) Bifurcation on c (b) Bifurcation on W (c) Bifurcation on D

Figure 2: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of Resnet50-1.8G model trained on Imagenet. The z-axis represents the frequency.

The Catalyst shows the superior performance compared to the benchmark methods [1, 2, 5, 7, 15, 17, 20] which smaller accuracy degradation driven from the pruning process.

3.3. Robust Bifurcation

We present a typically observed bifurcation behavior catalyst regularization in Fig. 2. Fig. 2(a) shows robust bifurcation of $c = \frac{D_{ii}}{\|F_i\|}$ centered at $c = 1$, by $\|DW\|_{2,1}$ minimization, as theoretically expected in Theorem 2. The ratio of c between pruned filters and preserved filters are extremely large, around 10^8, since the c value changes exponentially during the minimization.

As shown in Fig. 2(b) and Fig. 2(c), bifurcation on C derives the bifurcation on W and D inherently. Also, we can achieve lossless pruning since the filter norms of pruned filters are pushed to have extremely small value, while the filter norms the rest are preserved.

4. Effect of $prune$ operation

The $prune$ operation, which is composition of pruning and $proj$ for D removal, does not harm the model when $DW = 0$, but numerically the norm $\|DW\|$ is inevitably nonzero hence the $prune$ does change the model's functionality.

Table 2: Effect of $prune$ operation in Catalyst pruning.

Task	phase	avg Δ acc (%p)	avg $\Delta\mathcal{L}$ (%)	MACs drop(%)
R56_2.06x CIFAR10	opt_1	-0.001	0.0032	9.062
	opt_2	0.004	-0.0025	46.562
V19_8.96x CIFAR100	opt_1	-0.019	0.0001	85.075
	opt_2	0.009	0.0021	25.179
R50_1.82x Imagenet	opt_1	0.002	-0.0001	19.233
	opt_2	-0.037	0.0008	31.811

In Table 2 we measure the difference of accuracy and loss value, before and after the $prune$ for each layer and report the average. Due to successful regularization, the $prune$ shows minimal effect on model performance and shows the success of regularization.

5. Discussion

The loss landscape of deep neural networks is notoriously complex, shaped by overparameterization and diverse architectural choices, making it difficult to derive general intuitions or common behaviors. However, by intentionally extending the parameter space with auxiliary variables, it is possible to reshape this landscape and induce desirable training dynamics. Prior work such as Bypassing [4] has leveraged this idea to escape unfavorable stationary points. In our work, we apply a similar principle to structured pruning, using an extended space to guide sparsification and promote robust filter bifurcation.

The ratio c has geometric meaning in the extended space, when we define projective space on it, that it is in fact a cotangent of angular distance between representation of zero-filter and current parameter. Future directions include a deeper theoretical exploration of the projective geometry underlying Catalyst's behavior, as well as extending this framework to analyze other forms of modularity and sparsity in neural networks. Also, carefully design of dynamics in augmented parameter spaces may offer new ways to address longstanding challenges in optimization, generalization, and structure learning in high-dimensional neural networks.

6. Conclusion

We introduce Catalyst, a novel regularizer for structured pruning, grounded in an algebraic characterization of lossless pruning. Catalyst induces a controlled deformation of the network during training, enabling magnitude-independent pruning decisions and robust bifurcation dynamics with a wide decision margin. The resulting Catalyst pruning algorithm achieves strong empirical performance across benchmarks, aligning with its theoretical guarantees and outperforming conventional regularizers for structured pruning. Our work highlights how carefully designed regularization can offer insight into emergent dynamics that links function approximation, model compression, and learning behavior of overparametrized neural networks. Structural modifications through controlled parameter extension opens up a promising direction for future research on high-dimensional neural networks.

References

[1] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021.

[2] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023.

[3] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2023. doi: 10.1109/TPAMI.2023.3334614.

[4] Jaeheun Jung and Donghun Lee. Bypassing stationary points in training deep learning models. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2024. doi: 10.1109/TNNLS.2024.3411020.

[5] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International conference on machine learning*, pages 5122–5131. PMLR, 2020.

[6] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[7] Jiashi Li, Qi Qi, Jingyu Wang, Ce Ge, Yujian Li, Zhangzhang Yue, and Haifeng Sun. Oicsr: Out-in-channel sparsity regularization for compact deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7046–7055, 2019.

[8] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxkOlSYDH.

[9] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[10] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[11] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pages 6566–6575. PMLR, 2019.

[15] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243*, 2020.

[16] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[17] Xidong Wu, Shangqian Gao, Zeyu Zhang, Zhenzhen Li, Runxue Bao, Yanfu Zhang, Xiaoqian Wang, and Heng Huang. Auto-train-once: Controller network guided automatic network pruning from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16163–16173, 2024.

[18] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[19] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006.

[20] Yuxin Zhang, Mingbao Lin, Chia-Wen Lin, Jie Chen, Yongjian Wu, Yonghong Tian, and Rongrong Ji. Carrying out cnn channel pruning in a white box. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7946–7955, 2023. doi: 10.1109/TNNLS.2022.3147269.

[21] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. *Advances in neural information processing systems*, 33:9865–9877, 2020.

Appendix A. Detailed components of Bypassing for Catalyst

The algebraic constraint for the pruning, $DW = 0$ is defined on extended parameter space with diagonal matrix D. To exploit this constraint with external parameter D, we modify bypass algorithm[4] to manage the model extension and regularization for the implementation.

Referring [4], Given model $\varphi_1 : \mathcal{D}_1 \to \mathcal{F}$ with $\varphi_1(\theta) = f_\theta$, the Bypass pipeline is comprised of the following three core components:

1. $\varphi_2 : \mathcal{D}_2 \to \mathcal{F}$: extension of model φ_1 satisfying $Im\varphi_1 \subseteq Im\varphi_2$
2. $embed$: extends the model parameter $\theta \in \mathcal{D}_1$ into extended space by

$$embed : \mathcal{D}_1 \to \mathcal{D}_2 \text{ s.t. } \varphi_1 = \varphi_2 \circ embed \tag{A1}$$

3. $proj$: contracts the model parameter $\omega \in \varphi_2^{-1}(Im\varphi_1)$ into original space by

$$\begin{aligned} proj : \varphi_2^{-1}(Im\varphi_1) \to \mathcal{D}_1 \text{ s.t.} \\ \varphi_2 = \varphi_1 \circ proj \text{ on } \varphi_2^{-1}(Im\varphi_1) \end{aligned} \tag{A2}$$

In this section, we describe how the three core components of Bypass pipeline would be constructed for structured pruning.

A.1. Model extension

Unlike the original bypass algorithm [4], we propose to use learnable activation which employs two parameters D and $\overline{D}$ for extended model φ_2, to give nontrivial initialization on D, as follows:

Definition 3 *Let $\overline{\theta} = (W, b_W, A, b_A)$ and rewrite*

$$\overline{\varphi_1}(\overline{\theta}) = \overline{\varphi_1}(W, b_W, A, b_A) = NN(W, b_W, A, b_A, \sigma). \tag{A3}$$

1. *We first define learnable activation $\psi_{D,\overline{D}}$ with additional parameter D and $\overline{D}$:*

$$\psi_{D,\overline{D}} : x \mapsto Dx - \overline{D}x + \sigma(x) \tag{A4}$$

2. *We define $\overline{\varphi_2}$ by*

$$\begin{aligned} \overline{\varphi_2}(\overline{\theta}, D, \overline{D}) &= \overline{\varphi_2}(W, b_W, A, b_A, D, \overline{D}) \\ &= NN(W, b_W, A, b_A, \psi_{D,\overline{D}}) \end{aligned} \tag{A5}$$

3. *The $\varphi_2(\theta, D, \overline{D})$ is defined by replacing $\overline{\varphi_1}(\overline{\theta})$ to $\overline{\varphi_2}(\overline{\theta}, D, \overline{D})$ from original model $\varphi_1(\theta)$.*

A.2. The *embed* function

Now we set the $embed$ map as follows:

Definition 4 *Given D^{init}, we define $embed(D^{init})$ as function-preserving operator on extended parameter space, as follows:*

$$embed(D^{init}) : \overline{\theta} \mapsto (\overline{\theta}, D^{init}, D^{init}) \tag{A6}$$

Remark 5 *The map $embed$ defined in Theorem 4 has a function-preserving property on φ_2, that $\varphi_2 = \varphi_2 \circ embed(D^{init})$ because $\psi_{D^{init},D^{init}} = \sigma$.*

For the implementation, we need to choose D^{init} to complete the model extension. In this paper, we proposed to use $D^{init} = diag(\|F_1\|_2, \cdots, \|F_{N_W}\|_2)$, where N is the number of entries in each filter. With the proposed initialization, the fair pruning chance would be ensured and the $\|DW\|_{2,1}$ minimization would show bifurcation behavior. The detailed explanation and mathematical backgrounds which supports this proposed initialization are included in Section 2.3 with simulations.

A.3. *prune*: combination of pruning and *proj*

Now we propose the last core component of Catalyst pruning by defining contraction map $prune$, which replaces $proj$ of bypass pipeline.

Definition 6 *Let P be set of filter indices to be pruned and P^c be its complement.*
We define $prune$ by

$$\begin{aligned} &prune(P) : (W, b_W, A, b_A, D, \overline{D}) \mapsto \\ &(W[P^c], b_W[P^c], (A^T[P^c])^T, b'_A, -\overline{D}[P^c], 0) \end{aligned} \tag{A7}$$

where $b'_A = b_A + ADb_W + (A^T[P])^T(\psi_{-\overline{D},0}(b_W))[P]$

Theorem 7 *Let $prune$ be the mappings defined in Theorem 6. If $DW = 0$ and $P = \{i|D_{ii} \neq 0\} = \{i|\forall j W_{ij} = 0\} \in 2^{N_W}$, then $prune(P)$ becomes function-preserving. That is,*

$$\overline{\varphi_2} = \overline{\varphi_2} \circ prune(P). \tag{A8}$$

Proof [Proof of Theorem 7] For arbitrary input, if $DW = 0$ then some filters would provide constant features. We pass those constant filters to next layer and add them to bias vector of next layer. The detailed proof is presented in Appendix D. ■

Combining φ_2, $embed(D^{init})$ and $prune(P)$ we can build the training pipeline for the structured pruning, with constrained optimization algorithm, opt_1 and opt_2. Starting with $\varphi_1(\theta)$, we extend the model to $\varphi_2(\theta, D, \overline{D})$ by $embed(D^{init})$, train with constrained optimization opt_1 and algebraic constraints $DW = 0$, and prune the model by $prune(P)$ to get pruned but still extended model $\varphi_2(\theta[P], D[P], 0)$. Again, with constrained optimization opt_2 and $\|DW\|_{2,1} = 0$, we get final pruned model by second $prune$. As stated in Theorem 5 and Theorem 7, the $embed(D^{init})$ and $prune(P)$ are function-preserving operations and thus there would be no pruning-caused damage if $\|DW\|_{2,1} = 0$.

Appendix B. Pseudocode of Catalysis pruining

We present our implementation of Catalyst pruning in Appendix B which is adaptation of Bypass pipeline [4] with catalyst regularization and other theoretical aspects described in Section 2. The algorithm can be summarized as: extend the model φ_1 by introducing additional parameters D and $\overline{D}$, train in extended space with constrained optimization, and contract back to the original model.

The proposed algorithm starts with $embed(D^{init})$, where D^{init} is set to be $D^{init} = c \times diag(\|F_1\|_2, \cdots, \|F_{N_W}\|_2)$ with $c = 1$, as proposed in Section 2.3. The $c = 1$ is proposed to place the pair of (D_{ii}, F_i) on the pruning decision boundary, but the practitioners may set this value to $c > 1$ to prune more, or $c < 1$ to prune less.

After initialization, the proposed algorithm repeats regularize-and-prune loop twice, to remove D and $\overline{D}$ with pruning operation, respectively. During the first loop, namely opt_1 (line 3-6 in Appendix B), we minimize $\mathcal{L}_{\in}(\overline{\theta}, \mathcal{D}, \overline{\mathcal{D}}) + \gamma_t(\|DW\|_{2,1})$ with SGD optimizer until the training budget T. The $Optimizer_{\lambda}(\cdot)$ represents the single SGD update with hyperparameter λ and γ_t is the parameter which controls the weight of regularization as in [4].

We can control the pressure of sparsification during opt_1, by changing α_θ and α_D which are the weight decay terms of θ and D each. Those are considered to be same in Section 2.3, but we may set $\alpha_\theta > \alpha_D$ to promote larger pruning ratio. In case of $\alpha_\theta > \alpha_D$, larger γ_t would induce larger pressure on sparsification since the influence of $\nabla\mathcal{L}_2$ is weakened, compared to the deterministic movement of $\nabla\|DW\|_{2,1}$.

If $\|DW\|_{2,1}$ decreases to small positive value ϵ (line 6 of Appendix B) or all c_t are bifurcated enough to satisfy $|log(c_t)| > \kappa$, the regularization loop of opt_1 may be stopped early. For experiments,we used $\kappa = 1$ for Imagenet and $\kappa = \infty$ for CIFAR, since early stopping was not necessary. After regularization stage, we choose the pruning indices by threshold $c = 1$, which is equivalent to $P = \{i | D_{ii} > \|F_i\|_2\}$, and prune the selected filters by $embed(P)$ defined in Appendix A to obtain intermediate pruning results with extra (but pruned) parameter D. Applying similar loop in line 9-14 of Appendix B, but with $\overline{D} = 0$, we can prune the model again and obtain pruned model with original architecture.

Algorithm 1: Regularization with catalyst

1: **Input** $\overline{\theta} = (W, b_W, A, b_A), \lambda, \lambda', \mathcal{L}_2, \epsilon, \epsilon', \gamma_t, \gamma'_t, T, T', c = 1$
2: **Initialize** $\overline{\theta}, D, \overline{D} \leftarrow embed(D^{init})(\overline{\theta}, 0, 0)$ where $D^{init} = c \cdot diag(\|F_1\|_2, \cdots, \|F_{N_W}\|_2)$
3: **repeat**
4: $\quad \overline{\theta}, D, \overline{D} \leftarrow \text{Optimizer}_{\lambda}(\mathcal{L}_2(\overline{\theta}, D, \overline{D}) + \gamma_t\|DW\|_{2,1})$
5: $\quad t \leftarrow t + 1$
6: **until** $\|DW\|_{2,1} < \epsilon$ or $log(c_t) > \kappa$ or $t > T$
7: $P \leftarrow \{i | D_{ii} > \|F_i\|_2\}$
8: $\overline{\theta}, D, 0 \leftarrow prune(P)(\overline{\theta}, D, \overline{D})$
9: **repeat**
10: $\quad \overline{\theta}, D, 0 \leftarrow \text{Optimizer}_{\lambda'}(\mathcal{L}_2(\overline{\theta}, D, 0) + \gamma'_t\|DW\|_{2,1})$
11: $\quad t \leftarrow t + 1$
12: **until** $\|DW\|_{2,1} < \epsilon'$ or $log(c_t) > 1$or $t > T'$
13: $P \leftarrow \{i | D_{ii} > \|F_i\|_2\}$
14: $\overline{\theta}, 0, 0 \leftarrow prune(P)(\overline{\theta}, D, 0)$
15: **return** the pruned parameter $\overline{\theta}$ (and continue finetune.)

Appendix C. Proof of Theorem 1

Theorem 1 *Let $\mathbb{D} = \{D = diag(\delta) | \delta \in \mathbb{R}^{N_W}\}$ be space of diagonal matrices and consider projection map*

$$\begin{aligned} p : \mathbb{R}^{N_W \times N_I} \times \mathbb{D} &\longrightarrow \mathbb{R}^{N_W \times N_I} \\ (W, D) &\longmapsto W \end{aligned} \tag{2}$$

then

(1) $X_{tgt} = p(\{(W, D) | DW = 0$ *and* $D \neq 0\})$.

(2) Let $B(X, \epsilon)$ be the ϵ-neighborhood of X with the L_2 norm. If $k > 0$ is a positive real number, then

$$B(X_{tgt}, \epsilon) = p(\{(W, D) | \|DW\|_{2,1} < k\epsilon \text{ and } \|D\|_1 > k\}) \tag{3}$$

Proof [Proof of Theorem 1 (1)] We first prove that $X_{tgt} \subseteq p(\{(W, D) | DW = 0 \text{ and } D \neq 0\})$.

if $\overline{W} \in X_{tgt}$, then there exists $\bar{i}$ such that $\overline{W_{\bar{i}}} = 0$. Without loss of generality, let $\bar{i} = 1$ and consider

$$\overline{D} = diag(1, 0, \cdots, 0). \tag{C1}$$

Then $\overline{DW} = 0$ and $\overline{D} \neq 0$. Therefore, we have

$$(\overline{W}, \overline{D}) \in \{(W, D) | DW = 0 \text{ and } D \neq 0\} \tag{C2}$$

and hence

$$\overline{W} = p(\overline{W}, \overline{D}) \in p(\{(W, D) | DW = 0 \text{ and } D \neq 0\}). \tag{C3}$$

For the opposite inclusion, let $\tilde{W} \in p(\{(W, D) | DW = 0 \text{ and } D \neq 0\})$. Then, there exist $\tilde{D} \neq 0$ that satisfy $\tilde{D}\tilde{W} = 0$.

Without loss of generality, we have $\tilde{D}_{11} \neq 0$ and thus $\tilde{W}_1 = 0$ since $(\tilde{D}\tilde{W})_1 = 0$. Therefore, $\tilde{W}$ in X_{tgt} and

$$X = p(\{(W, D) | DW = 0 \text{ and } D \neq 0\}). \tag{C4}$$

■

Proof [Proof of Theorem 1 (2)]

Suppose $W \in B(X_{tgt}, \epsilon)$ and let $\bar{i} = argmin_{i \in [N_W]} \|W_{i,:}\|_2$. WLOG, let $\bar{i} = 1$ then we have $\|W_{1,:}\|_2 < \epsilon$ and thus we can choose $k' \in (k, \frac{\epsilon}{\|W_{1,:}\|_2} k)$.

Let $\overline{D} = diag(k', 0, \cdots, 0)$. Then we have $\|D\|_1 = k' > k$ and

$$\|DW\|_{2,1} = k' \|W_{1,:}\|_2 + \sum_{j>1} 0 \cdot \|W_{j,:}\|_2 < k\epsilon. \tag{C5}$$

Therefore, $W \in \{(W, D) | \|DW\|_{2,1} < k\epsilon \text{ and } \|D\|_1 > k\}$.

For the opposite direction, let $\|\tilde{D}\tilde{W}\|_{2,1} < k\epsilon$ and $\|\tilde{D}\|_1 > k$. Let $\tilde{i} = argmin_{i \in [N_W]} \|\tilde{W}_{i,:}\|_2$ Then we have

$$\|\tilde{W}_{\tilde{i},:}\|_2 = \sum_{i=1}^{N_W} \frac{|\tilde{D}_{ii}|}{\|\tilde{D}\|_1} \|\tilde{W}_{\tilde{i},:}\|_2 \leq \sum_{i=1}^{N_W} \frac{|\tilde{D}_{ii}|}{\|\tilde{D}\|_1} \|\tilde{W}_{i,:}\|_2 < \frac{1}{k} \sum_{i=1}^{N_W} |D_{ii}| \|\tilde{W}_{i,:}\|_2 = \frac{1}{k} \|\tilde{D}\tilde{W}\|_{2,1} < \epsilon \tag{C6}$$

Therefore, we have $\|\tilde{W}_{\tilde{i},:}\|_2 < \epsilon$ and thus $W \in B(X_{tgt}, \epsilon)$.

■

Appendix D. Proof of Theorem 7

Theorem 7 *Let $prune$ be the mappings defined in Theorem 6. If $DW = 0$ and $P = \{i | D_{ii} \neq 0\} = \{i | \forall j W_{ij} = 0\} \in 2^{N_W}$, then $prune(P)$ becomes function-preserving. That is,*

$$\overline{\varphi_2} = \overline{\varphi_2} \circ prune(P). \tag{A8}$$

Proof [Proof of Theorem 7] Recall that

$$prune(P)(W, b_W, A, b_A, D, \overline{D}) = (W[P^c], b_W[P^c], (A^T[P^c])^T, b'_A, 0, \overline{D}[P^c]) \tag{D1}$$

where $b'_A = b_A + ADb_W + (A^T[P])^T(\psi_{-\overline{D},0}(b_W))[P]$.

Let $|P| = n$ and WLOG let $P = \{1, \cdots, n\}$. Then we can write each parameters can be written by block matrix representations:

$$A = \left[\begin{array}{c|c} A^T[P]^T & A^T[P^c]^T \end{array}\right], W = \begin{bmatrix} 0 \\ W[P^c] \end{bmatrix}, b_W = \begin{bmatrix} b_W[P] \\ b_W[P^c] \end{bmatrix} \tag{D2}$$

Let X_{tgt} be arbitrary input tensor of $\overline{\varphi_2}(\overline{\theta}, D, \overline{D})$ then the output would be given by

$$\begin{aligned} \overline{\varphi_2}(\overline{\theta}, D, \overline{D})(x) &= b_A + A\psi_{D,\overline{D}}(b_W + Wx) \\ &= b_A + ADb_W + \cancelto{0}{ADWx} - A\overline{D}b_W - A\overline{D}Wx + A\sigma(b_W + Wx) \\ &= b_A + ADb_W + A\psi_{0,\overline{D}}(b_W + Wx) \end{aligned} \tag{D3}$$

Since $\psi_{0,\overline{D}}$ is channel-wise operation, $\psi_{-\overline{D},0}(b_W)[P] = \psi_{0,\overline{D}[P]}(b_W[P])$. Considering the block matrix representation Eq. (D2), $A\psi_{0,\overline{D}}(b_W + Wx)$ becomes

$$\begin{aligned} &A\psi_{0,\overline{D}}(b_W + Wx) \\ &= \left[\begin{array}{c|c} A^T[P]^T & A^T[P^c]^T \end{array}\right] \psi_{0,\overline{D}}\left(\begin{bmatrix} b_W[P] \\ b_W[P^c] + W[P^c]x \end{bmatrix}\right) \\ &= \left[\begin{array}{c|c} A^T[P]^T & A^T[P^c]^T \end{array}\right] \begin{bmatrix} \psi_{0,\overline{D}[P]}(b_W[P]) \\ \psi_{0,\overline{D}[P^c]}(b_W[P^c] + W[P^c]x) \end{bmatrix} \\ &= A^T[P]^T\psi_{0,\overline{D}[P]}(b_W[P]) \\ &\qquad + A^T[P^c]^T\psi_{0,\overline{D}[P^c]}(b_W[P^c] + W[P^c]x) \end{aligned} \tag{D4}$$

Hence, letting $b'_A = b_A + ADb_W + (A^T[P])^T(\psi_{-\overline{D},0}(b_W))[P]$ we finish the proof by

$$\begin{aligned} \overline{\varphi_2}(\overline{\theta}, D, \overline{D})(x) &= b'_A + A^T[P^c]^T\psi_{0,\overline{D}[P^c]}(b_W[P^c] + W[P^c]x) \\ &= \overline{\varphi_2}(b_W[P^c], W[P^c], A^T[P^c]^T, b'_A, 0, \overline{D}[P^c]) \\ &= (\overline{\varphi_2} \circ prune)(\overline{\theta}, D, \overline{D})(x) \end{aligned} \tag{D5}$$

■

Appendix E. Proof of Theorem 2

Theorem 2

Let d be the scalar parameter and M be N-dimensional vector. M_t and d_t be the trajectory of gradient descent movement of $\|dM\|_2$ at timestep t positive learning rate λ_t and weight decay term $\alpha \ll 1$. Let $c_t = \frac{|d_t|}{\|M_t\|_2}$ and assume that

$$0 < \lambda_* < \lambda_t \ll \min\left(\frac{(1-\alpha)}{c_0}, (1-\alpha)c_0\right) \tag{5}$$

where $\lambda_* = \inf_t \lambda_t$ is the infimum λ_t.

(1) If $c_0 = 1$, then $c_t = 1$ for all timestep t.

(2) If $c_0 < 1$, then c_t exponentially shrinks to $\frac{\lambda_t}{1-\alpha}$. i.e, If T be the smallest integer satisfying $c_T \leq \frac{\lambda_T}{1-\alpha}$ then there exists $k \in (0,1)$ such that $c_{t+1} \leq kc_t$ for all $t < T-1$.

(3) If $c_0 > 1$, then c_t exponentially grows to $\frac{(1-\alpha)}{\lambda_t}$. i.e, if T is the smallest integer satisfying $c_T \geq \frac{1-\alpha}{\lambda_T}$ then there exists $k > 1$ such that $c_{t+1} \geq kc_t$ for all $t < T-1$

Proof [Proof of Theorem 2] First consider the gradient descent movement of d_t and $M_t^{(i)}$, the ith entry of M_t, as follows:

$$\begin{aligned} d_{t+1} &= d_t - \alpha d_t - sgn(d_t)\lambda_t\|M_t\|_2 = (1-\alpha-\frac{\lambda_t}{c_t})d_t \\ M_{t+1}^{(i)} &= M_t^{(i)} - \alpha M_t^{(i)} - \lambda_t|d_t| \cdot \frac{M_t^{(i)}}{\|M_t\|_2} = (1-\alpha-\lambda_t c_t)M_t^{(i)}. \end{aligned} \tag{E1}$$

Note that the second inequalities of each are induced from the definition of $c_t = \frac{|d_t|}{\|M_t\|_2}$.

From second equation of Eq. (E1) we can induce following vector-formed updates:

$$M_{t+1} = (1-\alpha-\lambda_t c_t)M_t. \tag{E2}$$

Therefore, if

$$\frac{\lambda_t}{1-\alpha} \leq c_t \leq \frac{(1-\alpha)}{\lambda_t} \tag{E3}$$

then we have

$$\|M_{t+1}\|_2 = (1-\alpha-\lambda_t c_t)\|M_t\|_2. \tag{E4}$$

and

$$d_{t+1} = (1-\alpha-\frac{\lambda_t}{c_t})d_t. \tag{E5}$$

Therefore, we get

$$c_{t+1} = \frac{1-\alpha-\frac{\lambda_t}{c_t}}{1-\alpha-\lambda_t c_t}c_t = f(c_t, \lambda_t)c_t \tag{E6}$$

where $f(x,y) = \frac{1-\alpha-\frac{y}{x}}{1-\alpha-xy}$.

(1) Suppose $c_0 = 1$ then simple induction shows that $c_t = 1$ for all time t since Eq. (E3) holds by assumption in Eq. (5).

(2) Suppose $c_0 < 1$ and let T be the smallest integer which satisfies $\frac{\lambda_T}{1-\alpha} > c_T$.

If $t < T$ and $c_t < 1$, then $f(c_t, \lambda_t) < 1$ by Theorem 8(1) and thus $c_{t+1} < c_t < 1$, which means that $\{c_t\}$ is a decreasing sequence.

Also, by Theorem 8(1) we have $f(c_t, \lambda_t) < f(c_0, \lambda_*) < 1$ since $c_t < c_0$ because $\{c_t\}$ is a decreasing sequence, and $\lambda_t > \lambda_*$ due to the assumption in Eq. (5). Therefore, we have

$$c_t < f(c_0, \lambda_*)c_{t-1} < \cdots < f(c_0, \lambda_*)^t c_0 \tag{E7}$$

which finishes the proof of (2).

(3) Suppose $c_0 > 1$ and let T be the smallest integer which satisfies $\frac{\lambda_T}{1-\alpha} < c_T$.

If $t < T$ and $c_t > 1$, then $f(c_t, \lambda_t) > 1$ by Theorem 8(2) and thus $c_{t+1} > c_t > 1$, which means that $\{c_t\}$ is a increasing sequence.

Also, by Theorem 8(2) we have $f(c_t, \lambda_t) > f(c_0, \lambda_*) > 1$ since $c_t > c_0$ because $\{c_t\}$ is a increasing sequence, and $\lambda_t > \lambda_*$ due to the assumption in Eq. (5). Therefore, we have

$$c_t > f(c_0, \lambda_*)c_{t-1} > \cdots > f(c_0, \lambda_*)^t c_0 \tag{E8}$$

which completes the proof of (3).

■

Lemma 8 *Let* $f(x, y) = \frac{1-\alpha-\frac{y}{x}}{1-\alpha-xy}$ *and* $y \in (0, 1-\alpha)$.

(1) If $x \in (0, 1)$ *then* $f(x, y) < 1$, $\frac{\partial f}{\partial x} > 0$ *and* $\frac{\partial f}{\partial y} < 0$

(2) If $x \in (1, \infty)$ *then* $f(x, y) > 1$, $\frac{\partial f}{\partial x} > 0$ *and* $\frac{\partial f}{\partial y} > 0$

Proof *We first compute the partial derivatives:*

$$\frac{\partial f}{\partial x} = \frac{y(1-\alpha)}{x^2(1-\alpha-xy)^2}\{(x - \frac{y}{1-\alpha})^2 - \frac{y^2}{(1-\alpha)^2} + 1\} \tag{E9}$$

$$\frac{\partial f}{\partial y} = \frac{(1-\alpha)(x^2-1)}{x(1-\alpha-1xy)^2} \tag{E10}$$

Since $y \in (0, 1-\alpha)$, *the* $-\frac{y^2}{(1-\alpha)^2} + 1$ *term in RHS of Eq. (E9) becomes positive. Therefore,* $\frac{\partial f}{\partial x} > 0$ *for all* X_{tgt}.

Now assume that $x \in (0, 1)$. *Then* $f(x, y) < 1$ *since* $x < \frac{1}{x}$ *and* $\frac{\partial f}{\partial y} < 0$ *because* $x^2 - 1 < 0$.

Similarly, if $x \in (1, \infty)$ *then* $f(x, y) > 1$ *and* $\frac{\partial f}{\partial y} > 0$. ■

Appendix F. Details on experiments

In this section, we list the details on expeirment designs.

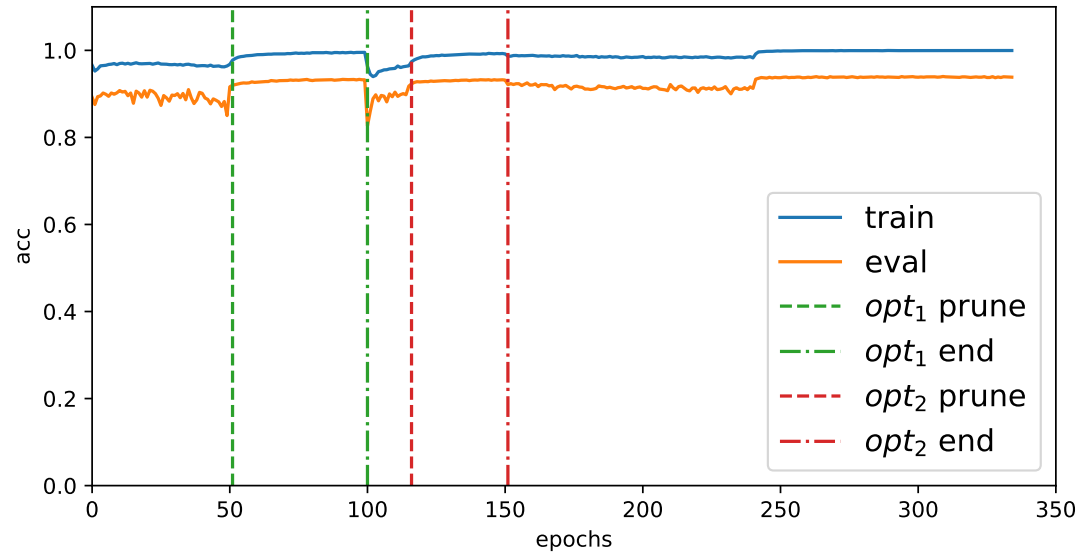
For Resnet-56 and VGG-19, we utilize the implementation and pre-trained models from [2], which achieve 93.53% and 73.50% top-1 accuracy, respectively. For Resnet-50, we use official torchvision [11] base model with pre-trained weights, which has 76.15% top-1 accuracy. Standard data augmentation including random cropping and flipping were applied.

All experiments in this paper were conducted with Linux (Ubuntu 20.04) computer equipped with single RTX 4090 GPU with 24GB VRAM. For imagenet experiment, we use gradient accumulation to run with single GPU.

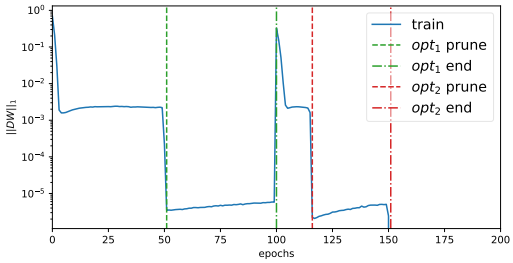
	Resnet56+CIFAR10	VGG19+CIFAR100	Resnet50+Imagenet
batch size	128	128	64
Gradient accumulation	1	1	2
optimizer	SGD(momentum=0.9)	SGD(momentum=0.9)	SGD(momentum=0.9)
Weight decay $(\alpha_\theta, \alpha_D)$	(5e-4,5e-5)	(5e-4,0)	(1e-4,0)
c	1	1	1
γ_t	0.007(1+0.25t),	γ(1+0.25t), γ=[2e-3,9e-3,12e-3]	3e-4(1+0.25t)
ϵ (opt1,opt2)	1e-6,1e-6	2e-6,1e-6	3e-6,3e-6
Stage finish epochs (opt1,opt2,finetune)	100,200,300	200,300,400	20,40,170
LR (opt1,opt2,finetune)	1e-2,1e-2	5e-3,5e-3, 1e-3 for 3x 0.01, 0.01, 0.01	5e-3,5e-3,(5e-3,1e-5)
LR decay epoch (opt1,opt2,finetune)	[50],[150],[240,270]	[],[],[390] for 3x [75,750],[250],[340,370] for rest	[10,15],[30,35] ,[70,100,120,160]
LR decay ratio (opt1,opt2,finetune)	0.1, 0.1, 0.1	NA,NA,0.1 for 3x 0.2,0.2,0.1 for rest	0.1,0.1,0.1
training runtime	2 hours	2 hours	60.42hours

Appendix G. Loss curves during the training

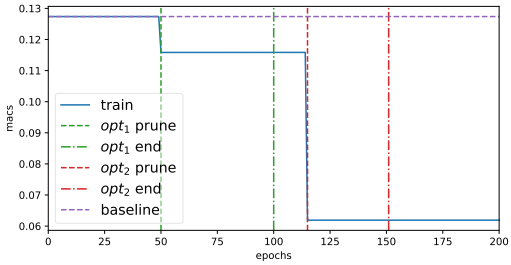
In this section, we plot the training logs of loss, accuracy, $\|DW\|$ and MACs, in CIFAR10 experiment. The target layers of the model were pruned early in epoch 51 and 116.



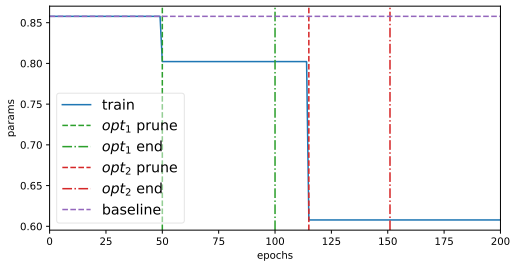
(a) Train and Test Accuracy



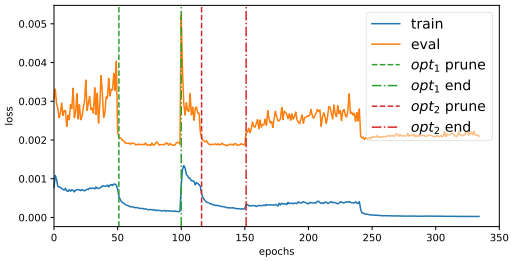
(b) total $\|DW\|_{2,1}$



(c) MACs reduction



(d) number of parameters



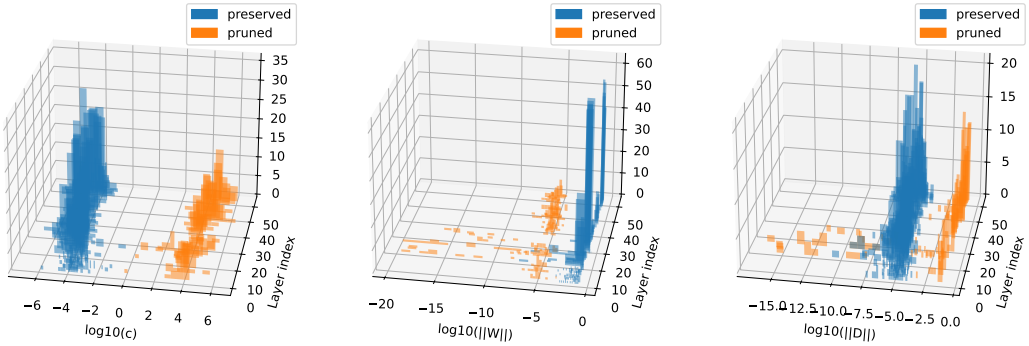
(e) Loss $\mathcal{L}_2$

Figure 3: Learning curves of CIFAR10 experiment with Resnet56

Appendix H. Full evaluation on bifurcation behavior

In this section, we provide histograms of filter norm, $\{D_{ii}\}_{i\in[N_W]}$ and $C = \frac{D}{\|W\|_{2,1}}$ for every layer of our pruned VGG19 model (in Section 3.2) and Resnet50 model(speedup 2.00x), to show that the bifurcation behavior claimed in Section 2.3 always happens.

H.1. Resnet56+CIFAR10

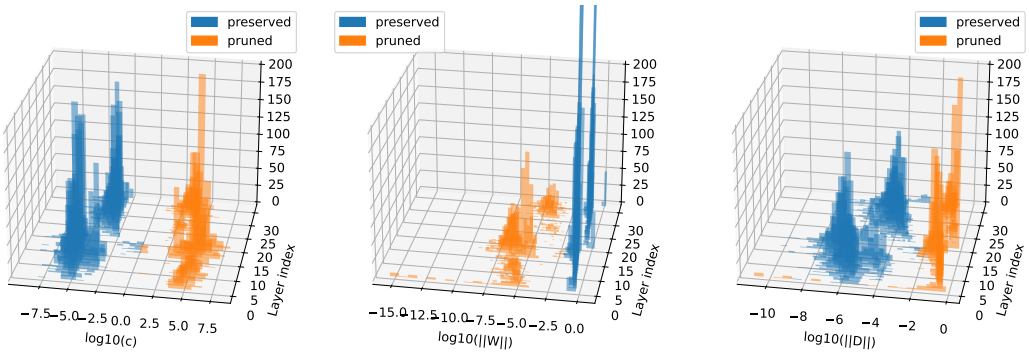


(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 4: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of Resnet56 model trained on CIFAR10. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

.

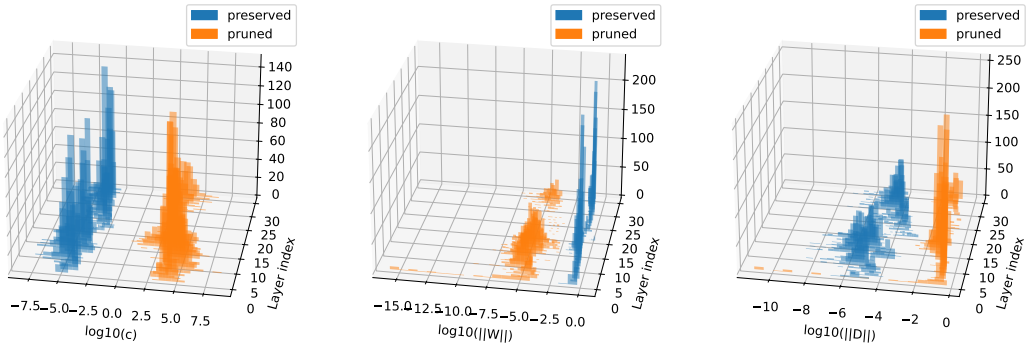
H.2. VGG19+CIFAR100



(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 5: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of VGG19 model (3x speedup) trained on CIFAR100. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

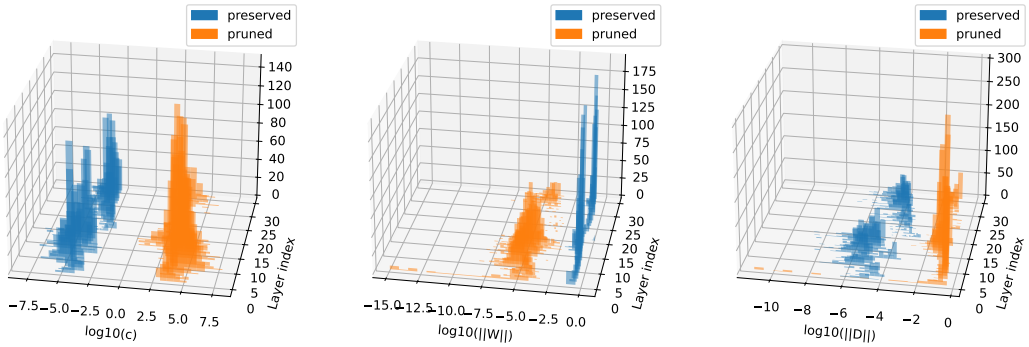
.



(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 6: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of VGG19 model (8.96x speedup) trained on CIFAR100. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

.

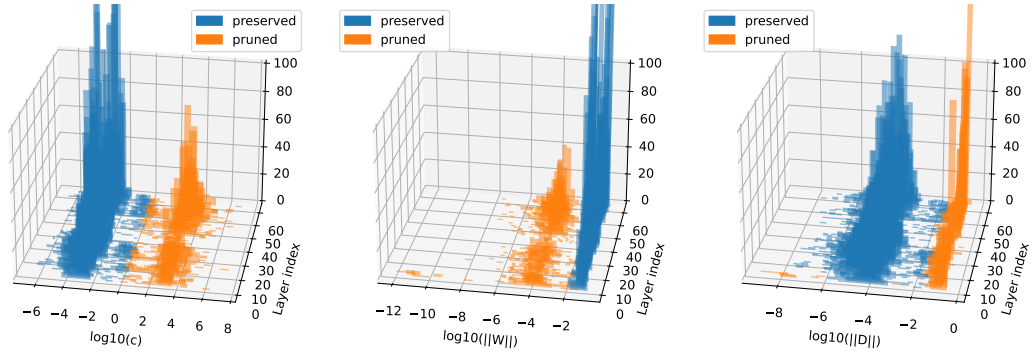


(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 7: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of VGG19 model (11.84x speedup) trained on CIFAR100. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

.

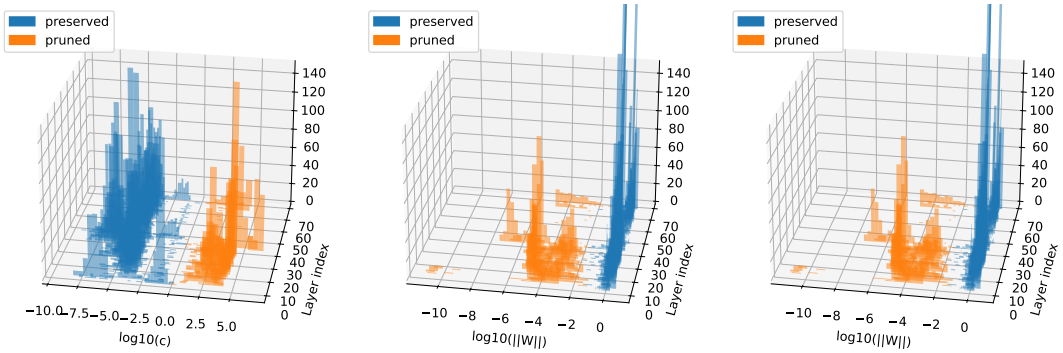
H.3. Resnet50+Imagenet



(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 8: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of Resnet50 model (1.49x speedup) trained on Imagenet. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

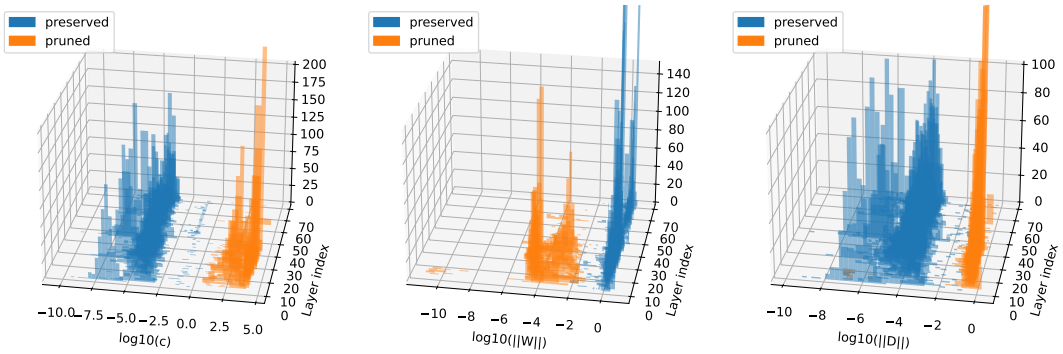
.



(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 9: The histograms of ratio $c = \frac{D_{ii}}{\|F_i\|}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of Resnet50 model (1.96x speedup) trained on Imagenet. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

.



(a) Bifurcation on c_is (b) Bifurcation on F_is (c) Bifurcation on D_{ii}s

Figure 10: The histograms of ratio $c = \frac{D_{ii}}{||F_i||}$, filter vector F_i of weight tensor W, and D_{ii}'s for each layers of Resnet50 model (2.33x speedup) trained on Imagenet. The z-axis represents the frequency. The layer indices of results in $opt2$ are shifted to start from the last layer.

.

Appendix I. Visualizations on fair pruning chance

The L1 and Group Lasso regularizer is claimed to prefer the filters with small initial magnitude. In this section, we regularize VGG19 model on CIFAR100, prune filters according to magnitude with pruning ratio which is same to our pruned VGG19 model (speedup 8.96x in Table 1) and visualize the initial magnitude of pruned filters on every layers.

We also visualize the histogram of filter norms after the regularization too. The bifurcation behaviors can be found on some layers, but still there is preference of small initial magnitude.

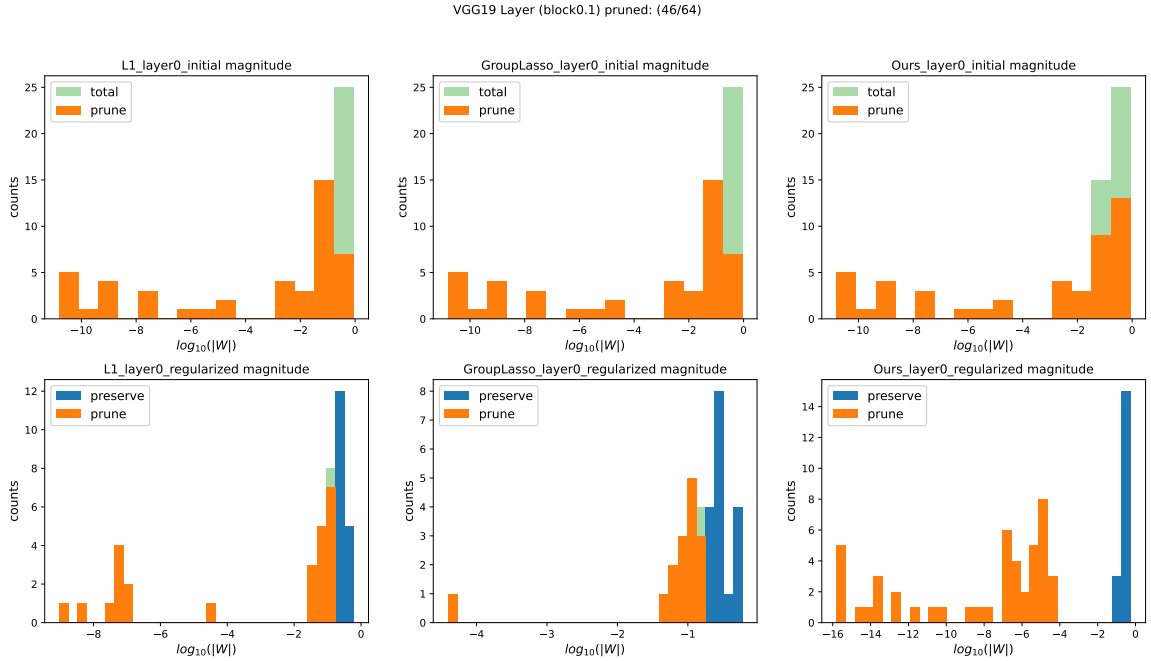


Figure 11: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 1st layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

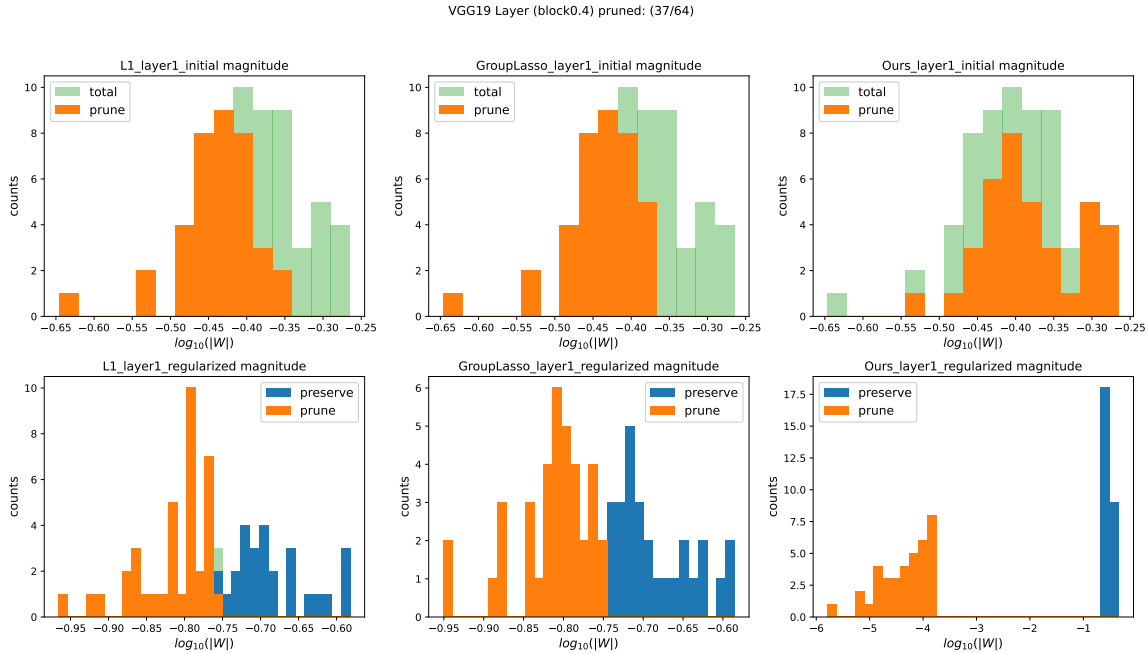


Figure 12: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 2nd layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

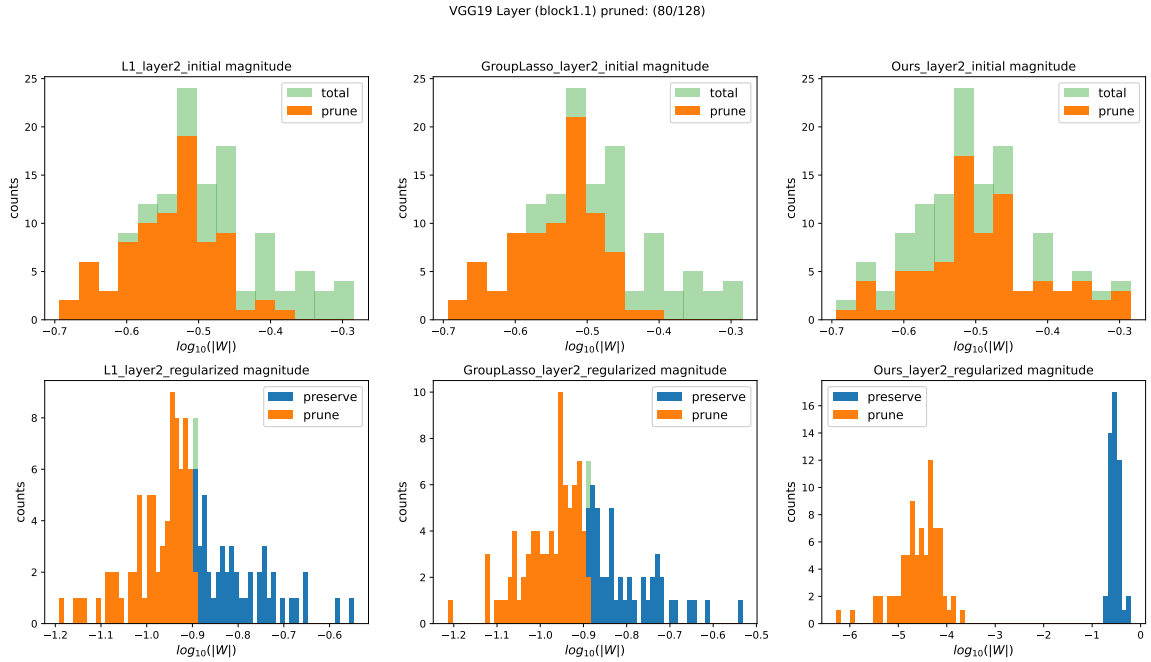


Figure 13: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 3rd layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

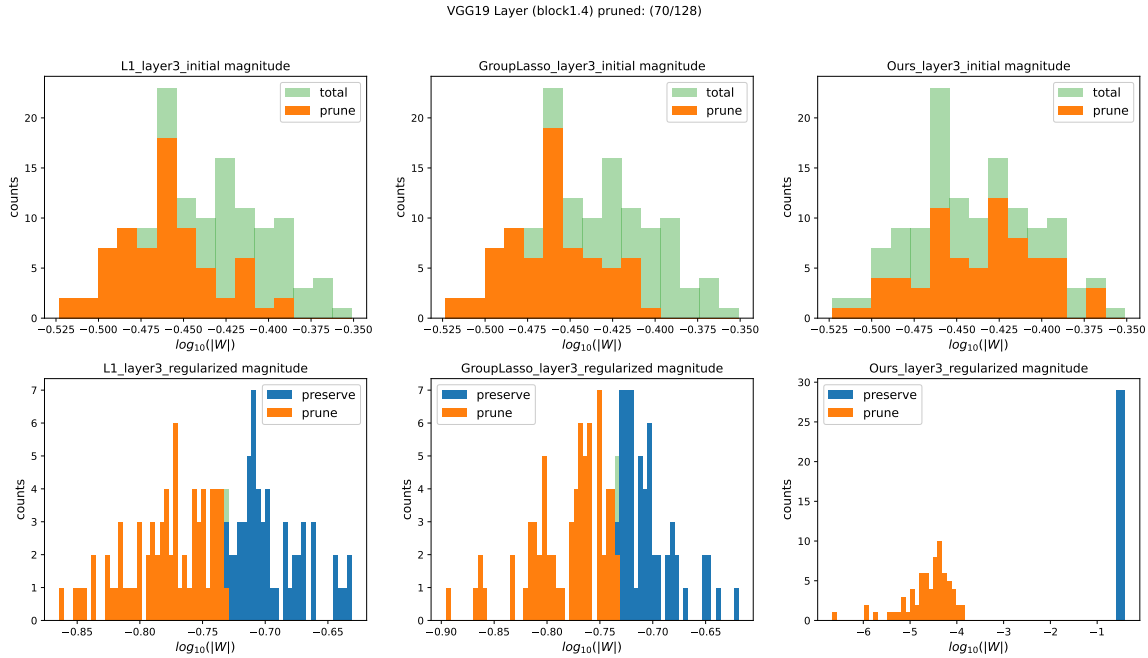


Figure 14: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 4th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

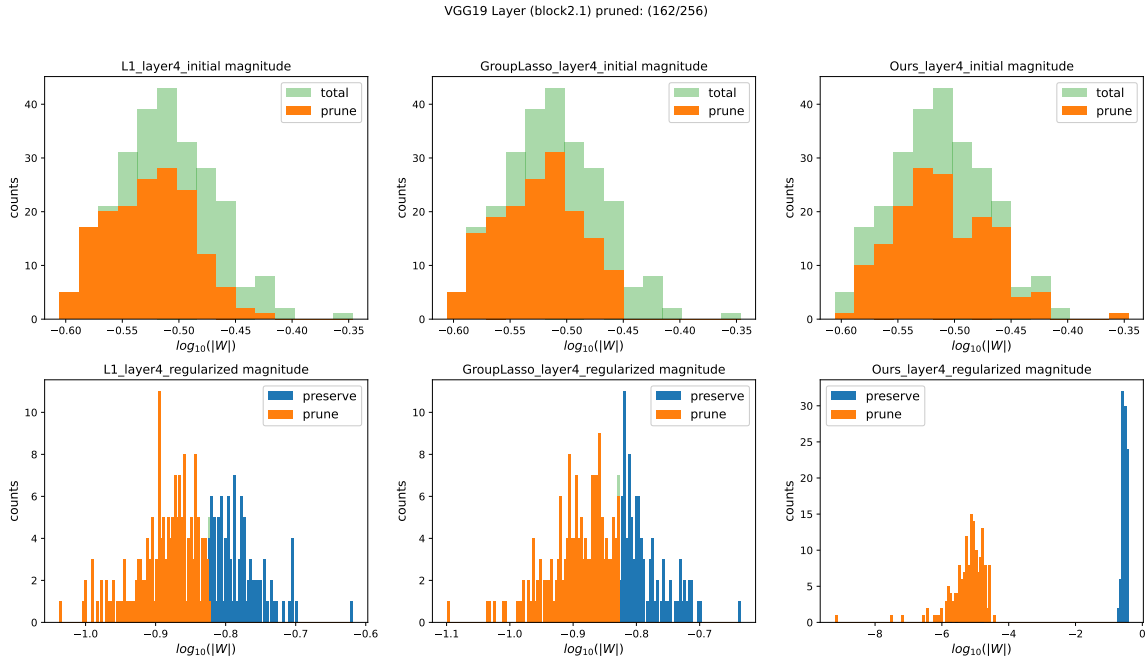


Figure 15: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 5th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

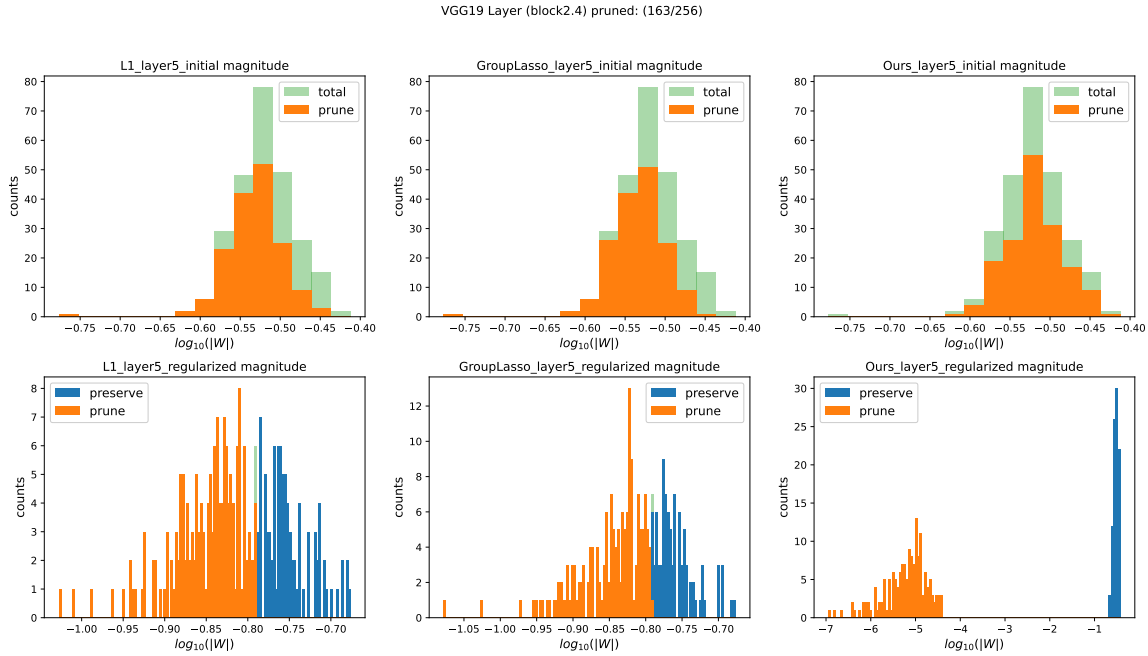


Figure 16: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 6th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

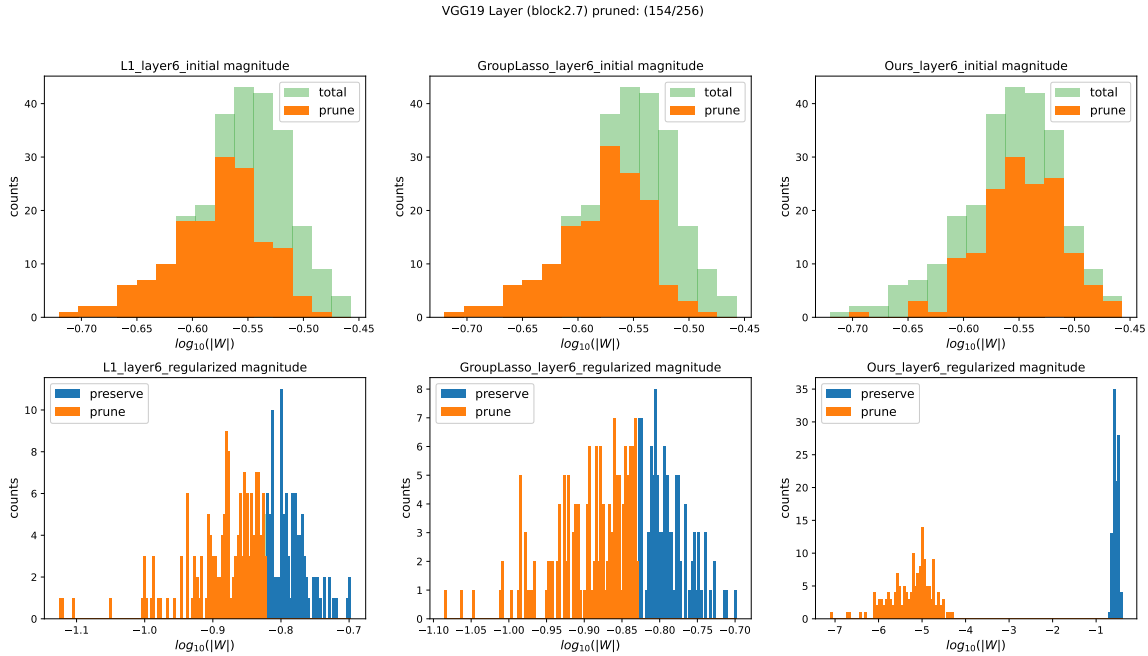


Figure 17: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 7th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

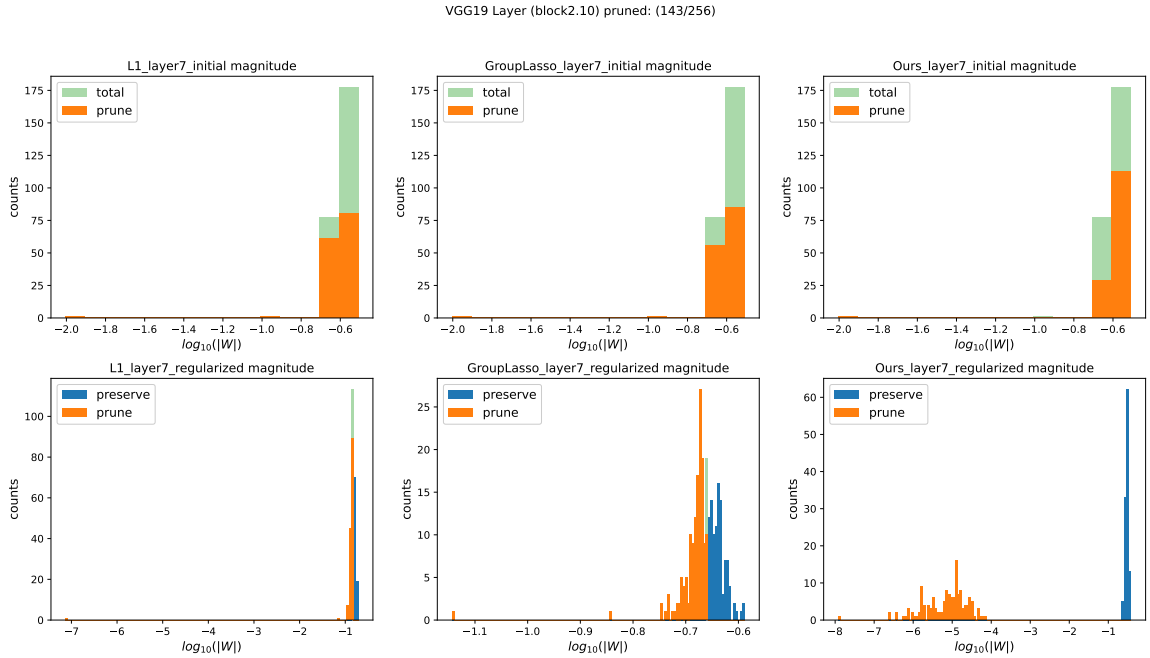


Figure 18: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 8th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

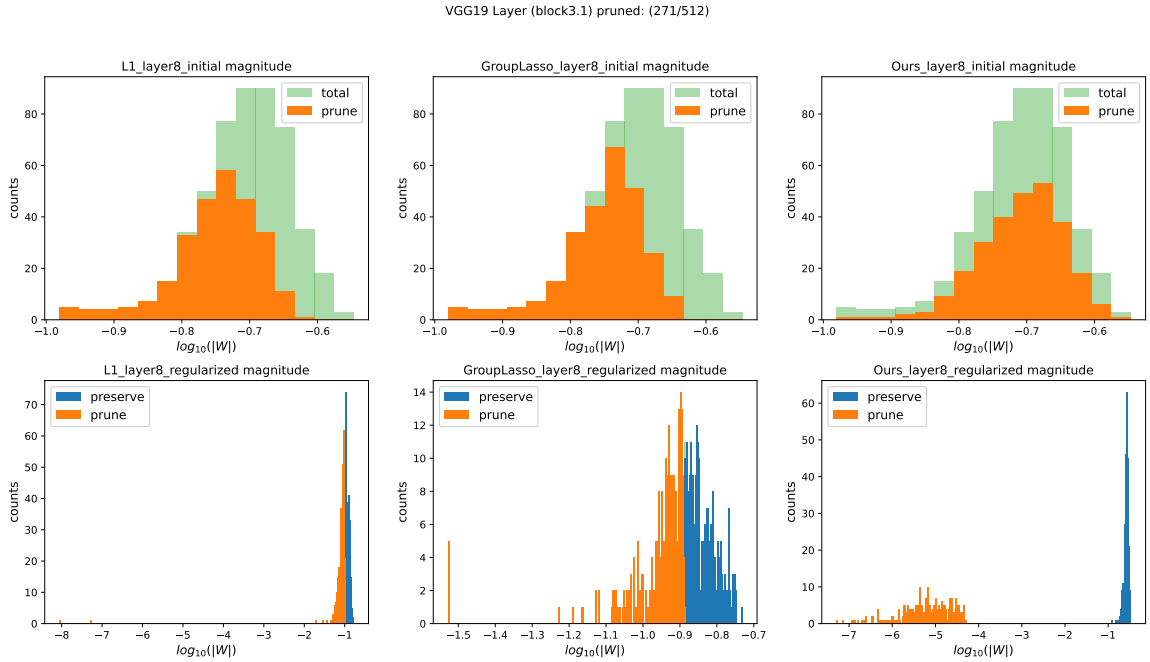


Figure 19: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 9th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

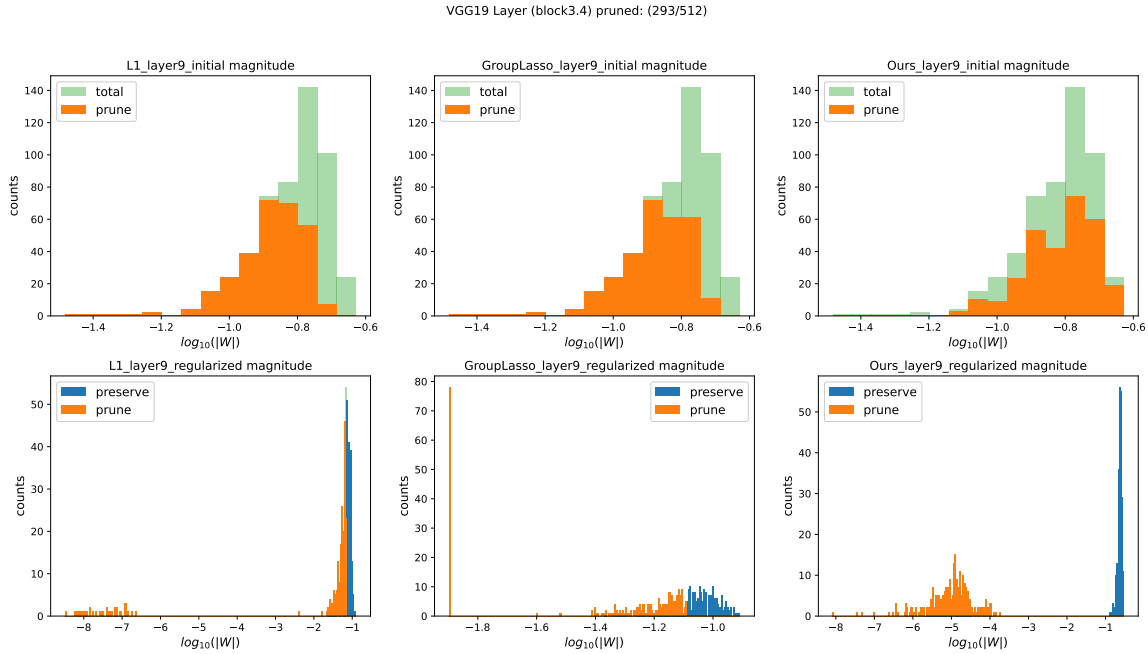


Figure 20: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 10th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

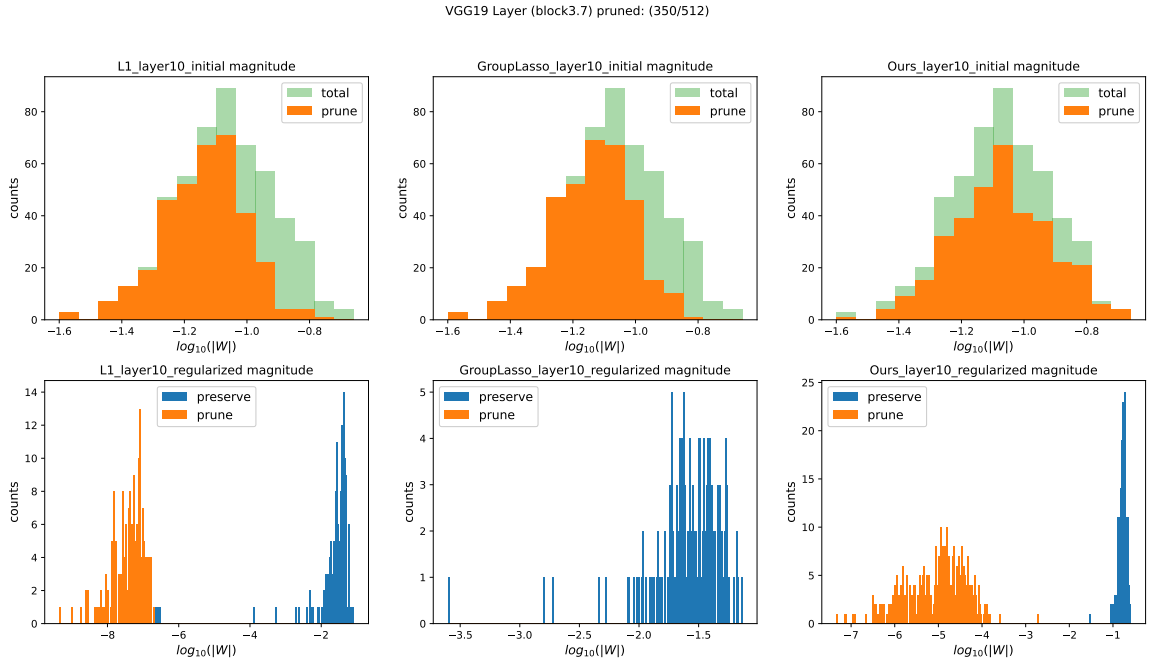


Figure 21: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 11th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

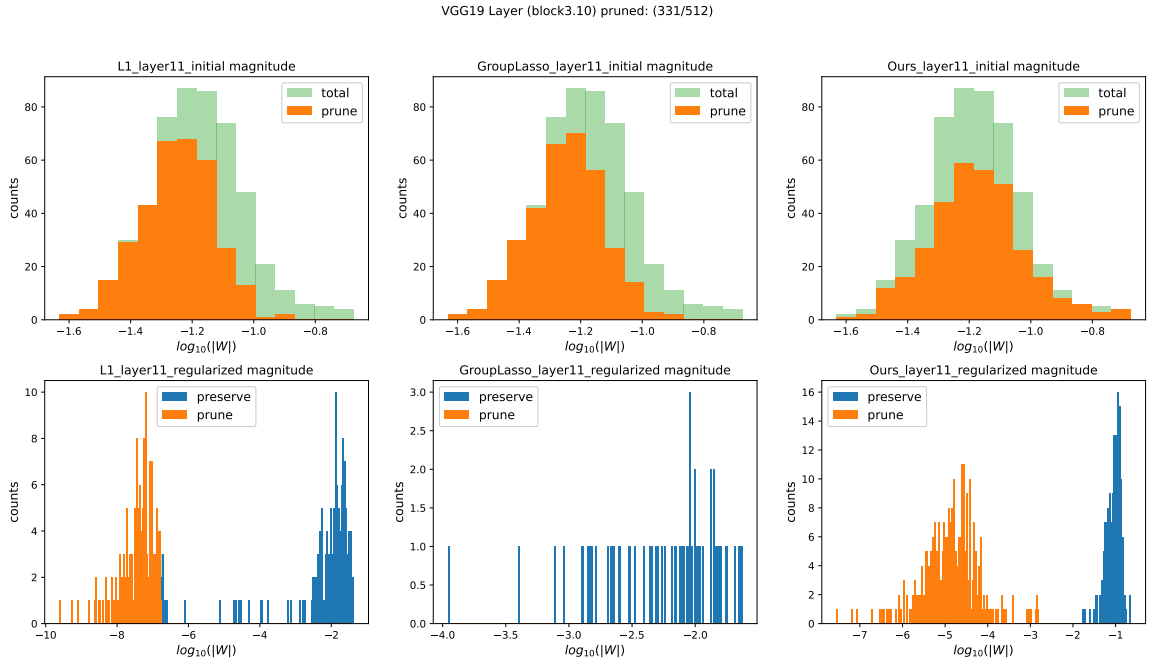


Figure 22: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 12th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

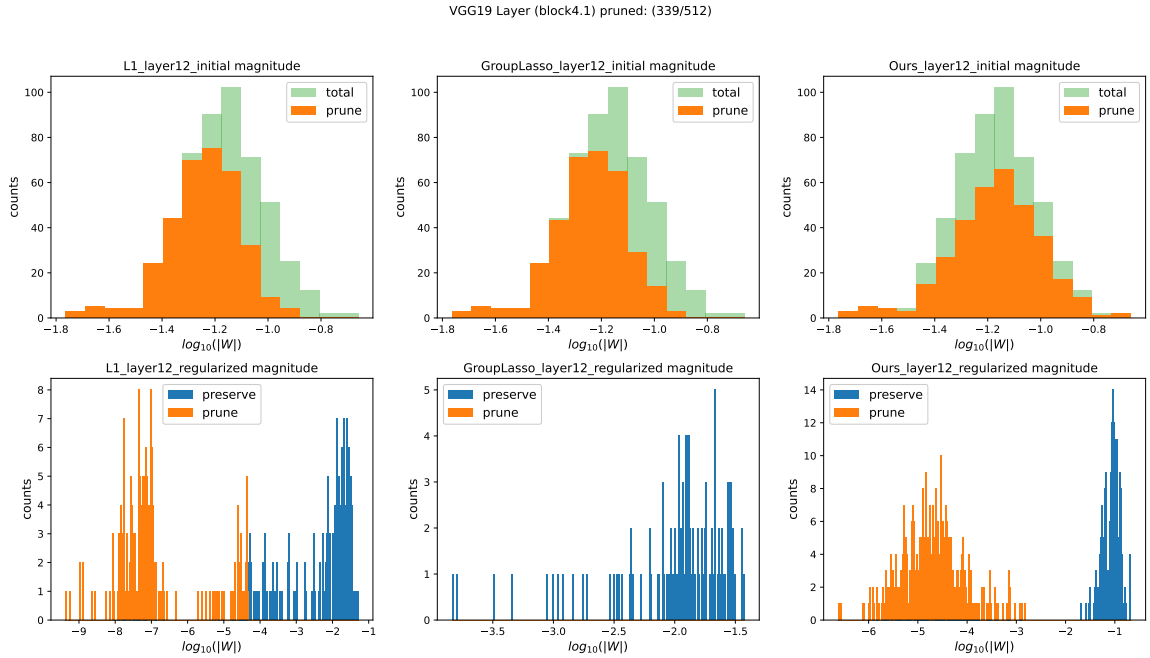


Figure 23: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 13rd layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

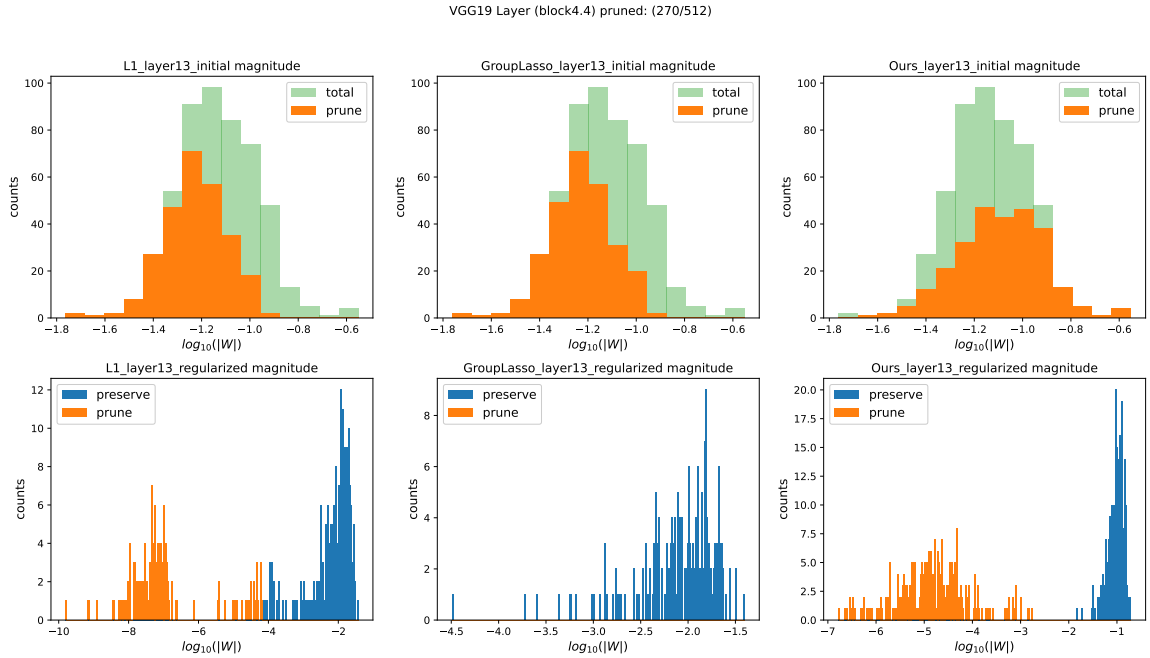


Figure 24: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 14th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

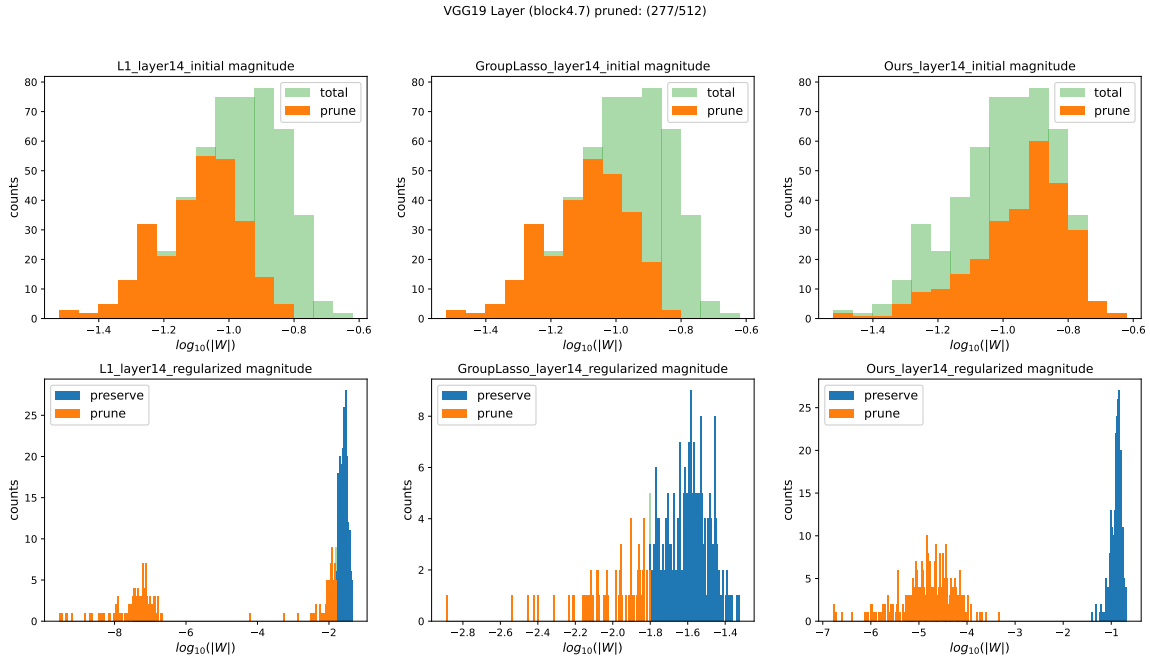


Figure 25: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 15th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.

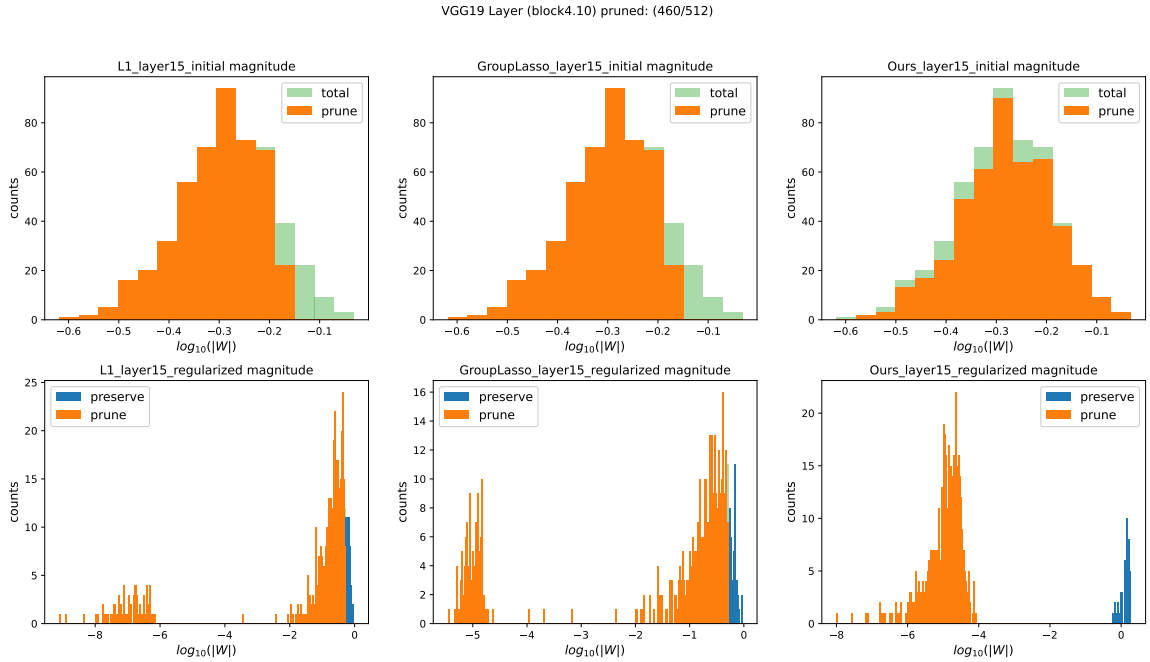


Figure 26: Initial magnitude and regularized magnitude of L1, Group Lass and our regularizer on 16th layer of VGG19 model. Each columns correspond to L1, Group Lasso, $\|DW\|_{2,1}$ respectively. First row is histogram of initial filter norms and second is historgram of regularized filter norms.