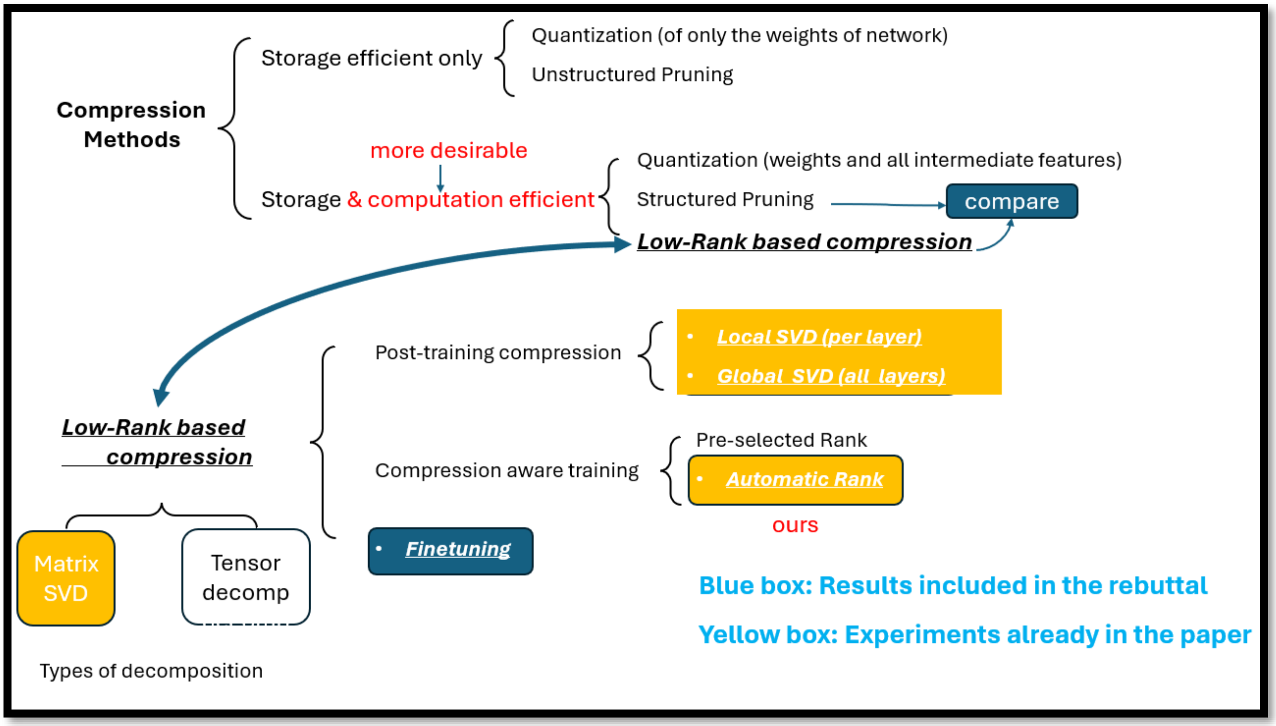
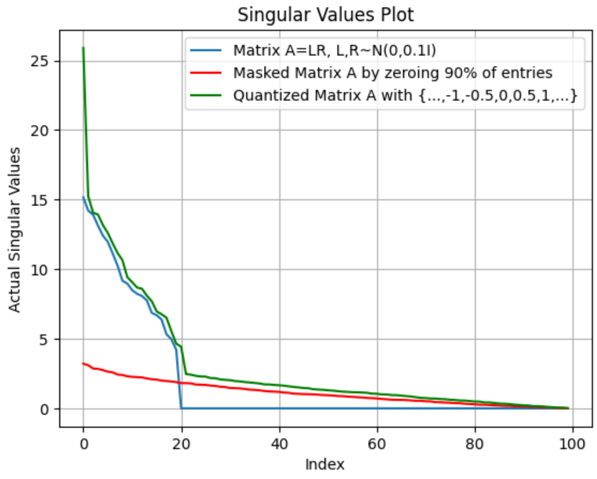
Structure-Preserving Network Compression Via Low-Rank Induced Training Through Linear Layers Composition

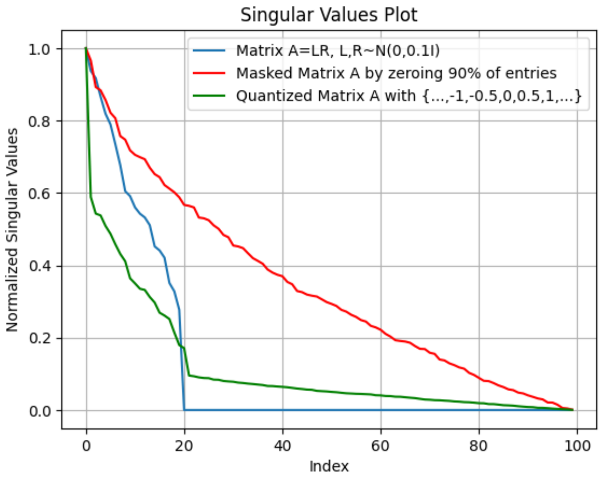
Response Supplementary

Diagram for Comparison with Related Work:


Compression Methods
Storage efficient only
Quantization (of only the weights of network)
Unstructured Pruning
more desirable
Storage & computation efficient
Quantization (weights and all intermediate features)
Structured Pruning
compare
Low-Rank based compression
Post-training compression
• Local SVD (per layer)
• Global SVD (all layers)
Low-Rank based compression
Compression aware training
Pre-selected Rank
• Automatic Rank
ours
• Finetuning
Matrix SVD
Tensor decomp
Types of decomposition
Blue box: Results included in the rebuttal
Yellow box: Experiments already in the paper

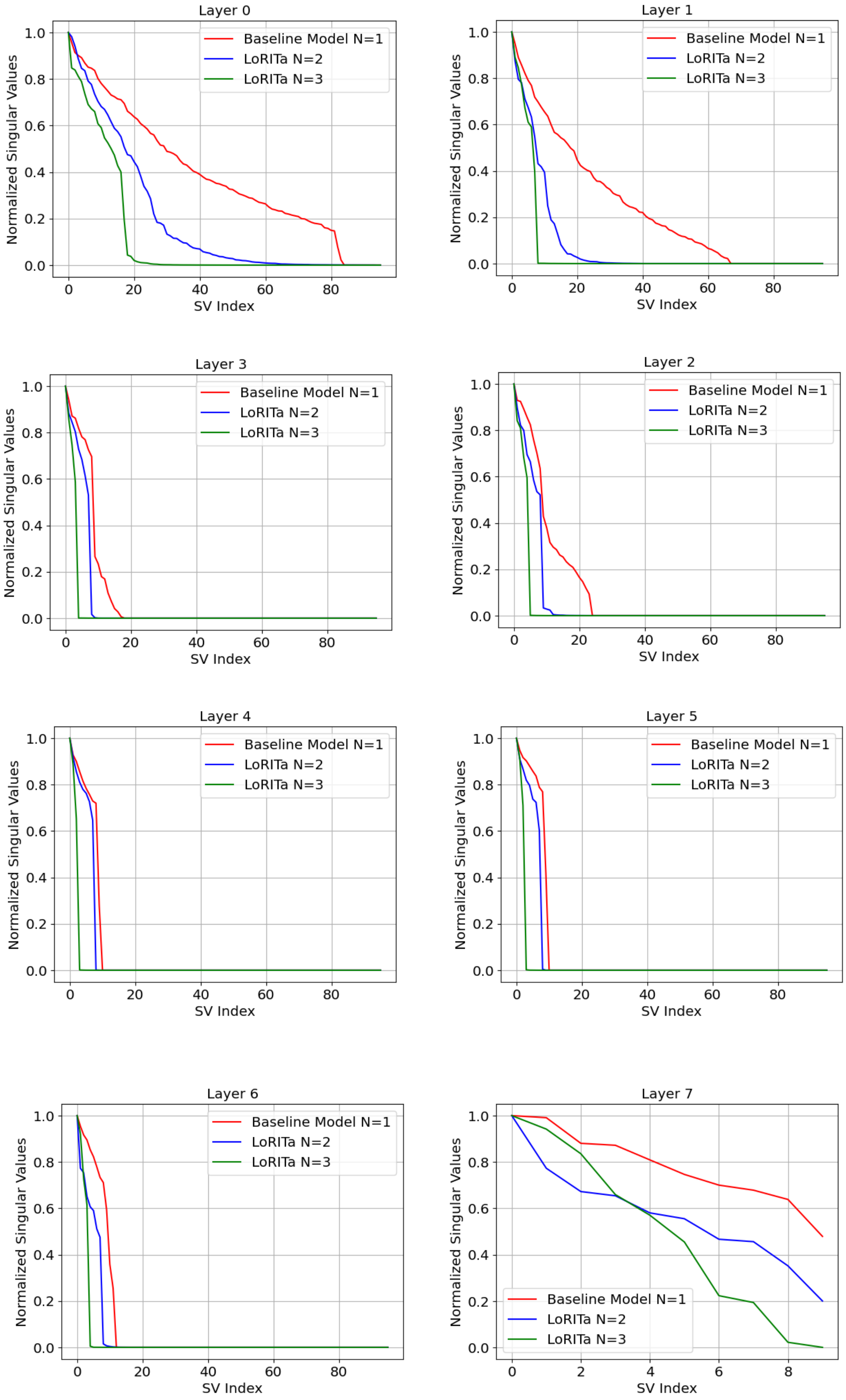
Experiment 1: (Reviewer LSTL Comment 5) The impact of quantization and pruning on the rank using SVs.





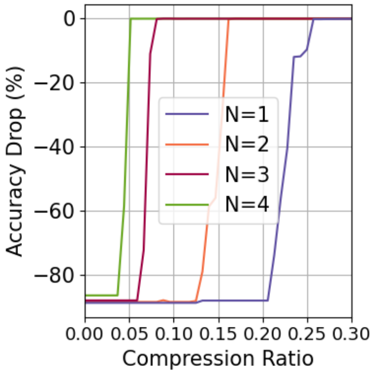
Experiment 2: (Reviewer 1gem Comment 5)

Empirically showing the faster decay of singular values of standard Model ($N = 1$) vs. LoRITa model ($N = 2$ and $N = 3$) using FCN8 from Table 1.

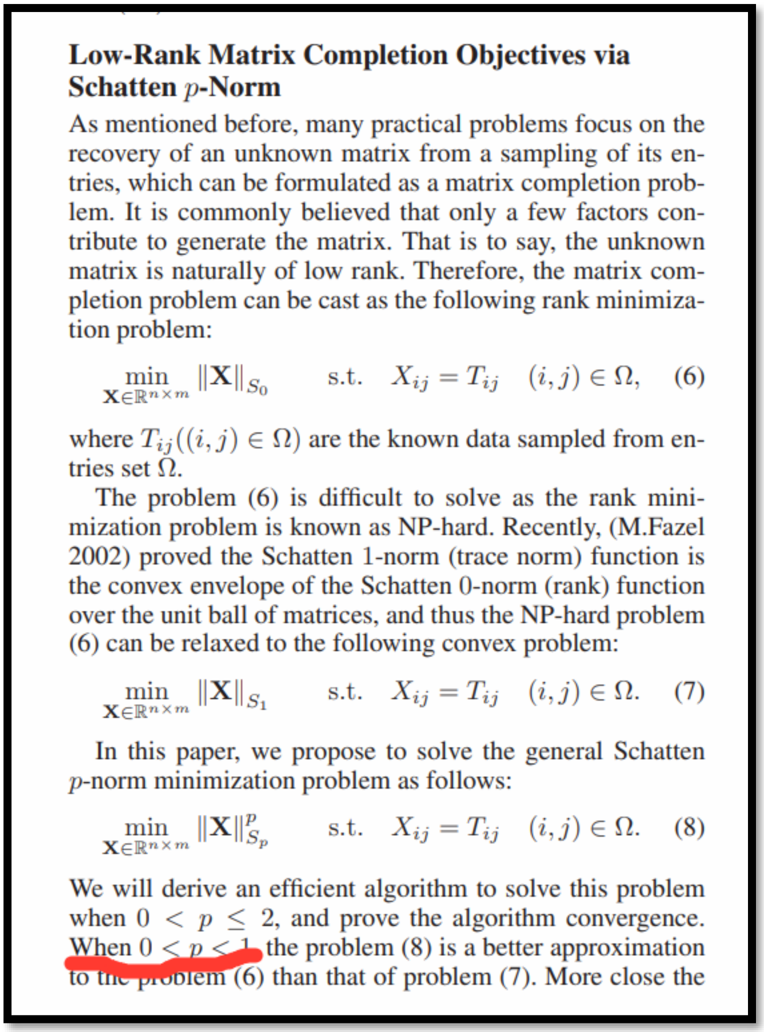


Experiment 3: (Reviewer HnzA Comment 2, Reviewer 1gem Comment 2, Reviewer EECw Comment 1)

Including results with $N = 4$.



Citation 1: (Reviewer EECw Question 3) Low-Rank Matrix Recovery via Efficient Schatten p-Norm Minimization, AAAI 2021



Low-Rank Matrix Completion Objectives via Schatten p-Norm

As mentioned before, many practical problems focus on the recovery of an unknown matrix from a sampling of its entries, which can be formulated as a matrix completion problem. It is commonly believed that only a few factors contribute to generate the matrix. That is to say, the unknown matrix is naturally of low rank. Therefore, the matrix completion problem can be cast as the following rank minimization problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times m}} \|\mathbf{X}\|_{S_0} \quad \text{s.t.} \quad X_{ij} = T_{ij} \quad (i,j) \in \Omega, \qquad (6)$$

where $T_{ij}((i,j) \in \Omega)$ are the known data sampled from entries set Ω.

The problem (6) is difficult to solve as the rank minimization problem is known as NP-hard. Recently, (M.Fazel 2002) proved the Schatten 1-norm (trace norm) function is the convex envelope of the Schatten 0-norm (rank) function over the unit ball of matrices, and thus the NP-hard problem (6) can be relaxed to the following convex problem:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times m}} \|\mathbf{X}\|_{S_1} \quad \text{s.t.} \quad X_{ij} = T_{ij} \quad (i,j) \in \Omega. \qquad (7)$$

In this paper, we propose to solve the general Schatten p-norm minimization problem as follows:

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times m}} \|\mathbf{X}\|_{S_p}^{p} \quad \text{s.t.} \quad X_{ij} = T_{ij} \quad (i,j) \in \Omega. \qquad (8)$$

We will derive an efficient algorithm to solve this problem when $0 < p \leq 2$, and prove the algorithm convergence. When $0 < p < 1$ the problem (8) is a better approximation to the problem (6) than that of problem (7). More close the

Comparison with previous methods: Columns: 1) Baseline Accuracy 2) Accuracy After Pruning 3) Accuracy Drop (%) 4) Baseline FLOPs 5) FLOPs After Pruning 6) FLOPs Drop (%) 7) Baseline Parameters 8) Parameters After Pruning 9) Parameters Drop (%).

The following tables (where the results of previous methods are reported from [1], the most recent survey paper) are ranked according to the **FLOPs Drop** column (larger values are better) as Flop drops is the most desirable feature when compressing and/or pruning NNs. As long as the top 1 accuracy is not compromised by much (~2% according to [1]), larger Flops drop means better compression methods. Note: the last column **Parameters Drops** is also important as it is the amount of memory reduction.
[1] Structured pruning for deep convolutional neural networks: A survey (TPAMI Nov. 2023).

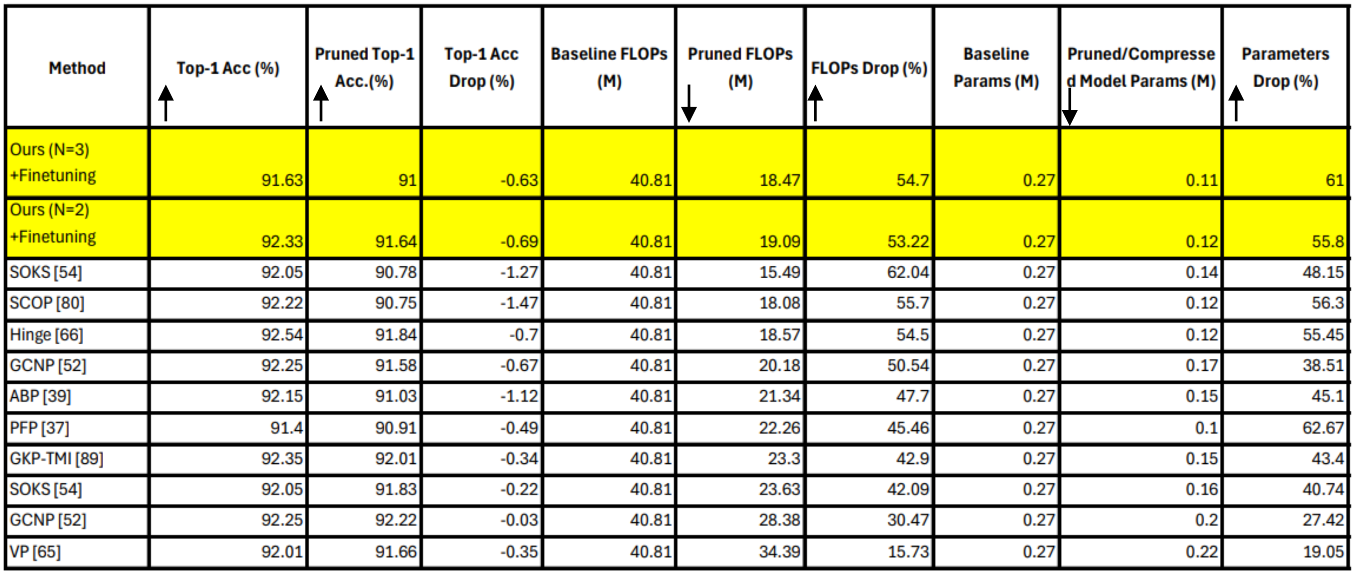
For VGG16, our result is the best in Flops Drop and has competitive Parameters Drop

For ResNet20, our result achieves nearly the best Parameters Drop. Our Flops Drop is 8% lower than [54], and similar to [80] [66]. However, among these three, [54] [66] perform very bad on Vgg16, with a Flops Drop of 54% and 39% versus our 84%. [80] didn’t report result for any Vgg networks.

We remark that for both cases, our N=3 model achieves better results than our N=2 model in terms of the FLOPs Drop and Parameters Drop.

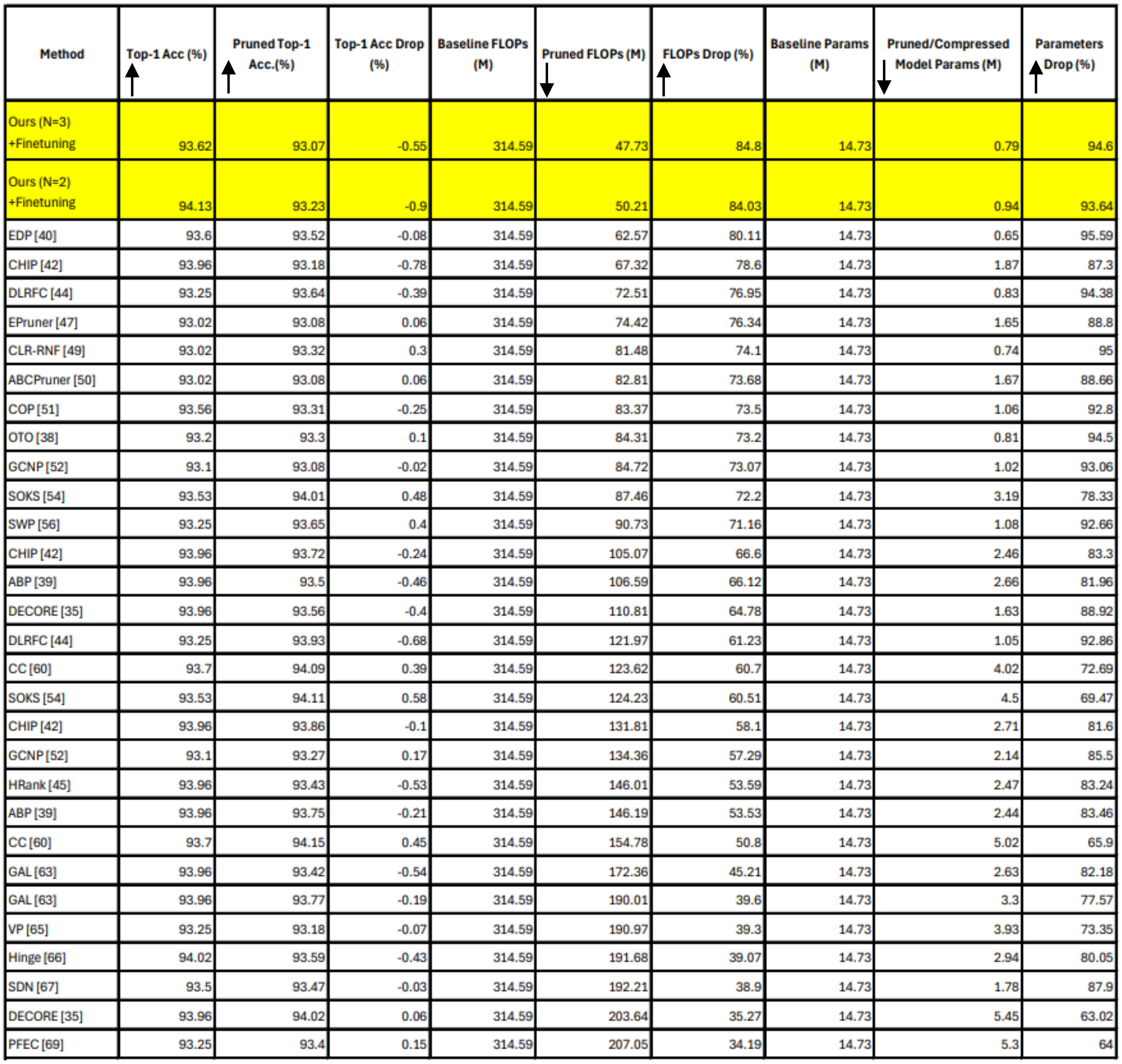
The FLOPs and parameters are computed using the following “ptflops” python package. https://pypi.org/project/ptflops/

ResNet20 Results



Method	Top-1 Acc (%) ↑	Pruned Top-1 Acc.(%) ↑	Top-1 Acc Drop (%)	Baseline FLOPs (M)	Pruned FLOPs (M) ↓	FLOPs Drop (%) ↑	Baseline Params (M)	Pruned/Compressed Model Params (M) ↓	Parameters Drop (%) ↑
Ours (N=3) +Finetuning	91.63	91	-0.63	40.81	18.47	54.7	0.27	0.11	61
Ours (N=2) +Finetuning	92.33	91.64	-0.69	40.81	19.09	53.22	0.27	0.12	55.8
SOKS [54]	92.05	90.78	-1.27	40.81	15.49	62.04	0.27	0.14	48.15
SCOP [80]	92.22	90.75	-1.47	40.81	18.08	55.7	0.27	0.12	56.3
Hinge [66]	92.54	91.84	-0.7	40.81	18.57	54.5	0.27	0.12	55.45
GCNP [52]	92.25	91.58	-0.67	40.81	20.18	50.54	0.27	0.17	38.51
ABP [39]	92.15	91.03	-1.12	40.81	21.34	47.7	0.27	0.15	45.1
PFP [37]	91.4	90.91	-0.49	40.81	22.26	45.46	0.27	0.1	62.67
GKP-TMI [89]	92.35	92.01	-0.34	40.81	23.3	42.9	0.27	0.15	43.4
SOKS [54]	92.05	91.83	-0.22	40.81	23.63	42.09	0.27	0.16	40.74
GCNP [52]	92.25	92.22	-0.03	40.81	28.38	30.47	0.27	0.2	27.42
VP [65]	92.01	91.66	-0.35	40.81	34.39	15.73	0.27	0.22	19.05

VGG16 Results



Method	Top-1 Acc (%) ↑	↑ Pruned Top-1 Acc.(%)	Top-1 Acc Drop (%)	Baseline FLOPs (M)	Pruned FLOPs (M) ↓	↑ FLOPs Drop (%)	Baseline Params (M)	↓ Pruned/Compressed Model Params (M)	↑ Parameters Drop (%)
Ours (N=3) +Finetuning	93.62	93.07	-0.55	314.59	47.73	84.8	14.73	0.79	94.6
Ours (N=2) +Finetuning	94.13	93.23	-0.9	314.59	50.21	84.03	14.73	0.94	93.64
EDP [40]	93.6	93.52	-0.08	314.59	62.57	80.11	14.73	0.65	95.59
CHIP [42]	93.96	93.18	-0.78	314.59	67.32	78.6	14.73	1.87	87.3
DLRFC [44]	93.25	93.64	-0.39	314.59	72.51	76.95	14.73	0.83	94.38
EPruner [47]	93.02	93.08	0.06	314.59	74.42	76.34	14.73	1.65	88.8
CLR-RNF [49]	93.02	93.32	0.3	314.59	81.48	74.1	14.73	0.74	95
ABCPruner [50]	93.02	93.08	0.06	314.59	82.81	73.68	14.73	1.67	88.66
COP [51]	93.56	93.31	-0.25	314.59	83.37	73.5	14.73	1.06	92.8
OTO [38]	93.2	93.3	0.1	314.59	84.31	73.2	14.73	0.81	94.5
GCNP [52]	93.1	93.08	-0.02	314.59	84.72	73.07	14.73	1.02	93.06
SOKS [54]	93.53	94.01	0.48	314.59	87.46	72.2	14.73	3.19	78.33
SWP [56]	93.25	93.65	0.4	314.59	90.73	71.16	14.73	1.08	92.66
CHIP [42]	93.96	93.72	-0.24	314.59	105.07	66.6	14.73	2.46	83.3
ABP [39]	93.96	93.5	-0.46	314.59	106.59	66.12	14.73	2.66	81.96
DECORE [35]	93.96	93.56	-0.4	314.59	110.81	64.78	14.73	1.63	88.92
DLRFC [44]	93.25	93.93	-0.68	314.59	121.97	61.23	14.73	1.05	92.86
CC [60]	93.7	94.09	0.39	314.59	123.62	60.7	14.73	4.02	72.69
SOKS [54]	93.53	94.11	0.58	314.59	124.23	60.51	14.73	4.5	69.47
CHIP [42]	93.96	93.86	-0.1	314.59	131.81	58.1	14.73	2.71	81.6
GCNP [52]	93.1	93.27	0.17	314.59	134.36	57.29	14.73	2.14	85.5
HRank [45]	93.96	93.43	-0.53	314.59	146.01	53.59	14.73	2.47	83.24
ABP [39]	93.96	93.75	-0.21	314.59	146.19	53.53	14.73	2.44	83.46
CC [60]	93.7	94.15	0.45	314.59	154.78	50.8	14.73	5.02	65.9
GAL [63]	93.96	93.42	-0.54	314.59	172.36	45.21	14.73	2.63	82.18
GAL [63]	93.96	93.77	-0.19	314.59	190.01	39.6	14.73	3.3	77.57
VP [65]	93.25	93.18	-0.07	314.59	190.97	39.3	14.73	3.93	73.35
Hinge [66]	94.02	93.59	-0.43	314.59	191.68	39.07	14.73	2.94	80.05
SDN [67]	93.5	93.47	-0.03	314.59	192.21	38.9	14.73	1.78	87.9
DECORE [35]	93.96	94.02	0.06	314.59	203.64	35.27	14.73	5.45	63.02
PFEC [69]	93.25	93.4	0.15	314.59	207.05	34.19	14.73	5.3	64