Neural Networks Model Compression

Nilanjan Ray (PI)
Sara Elkerdawy (PhD student)
Aaron Liu (MSc student)

Huawei advisor: Mostafa Elhoushi

Motivation

- Deep neural networks (DNN) achieved high performance in different areas.

- Because DNN models can be large, inference becomes computationally expensive. Deployment on devices with limited computational resources such as wearable devices and embedded boards is still challenging.

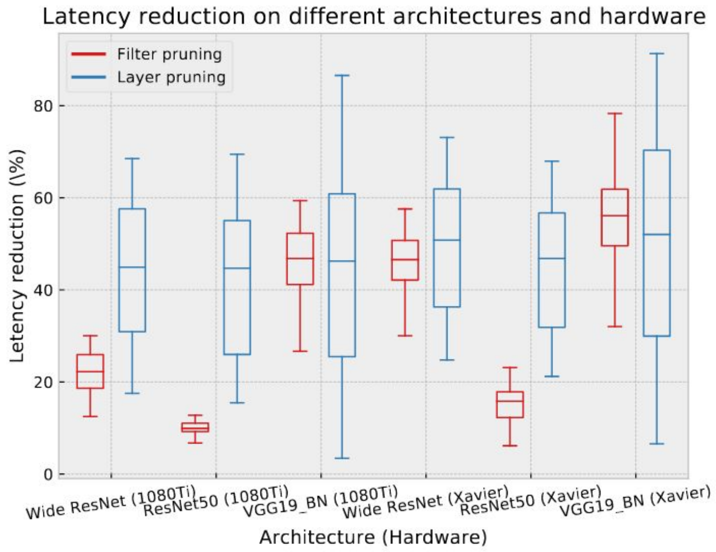
- Our emphasis in this research has been latency reduction

UNIVERSITY OF ALBERTA

Model Pruning

Filter Pruning Limitation

- In filter pruning, the range of attainable latency reduction is limited by the depth of the model.



How well do layer pruned models perform in terms of accuracy compared to filter pruned methods ?

Fig. 1: Example of 100 randomly pruned models per boxplot generated from different architectures. The plot shows layer pruned models have a wider range of attainable latency reduction consistently across architectures and different hardware platforms (1080Ti and Xavier). Latency is estimated using 224x224 input image and batch size=1.

Publications

- - S. Elkerdawy, M. Elhoushi, A. Singh, H. Zhang and N. Ray, "**To Filter Prune, or to Layer Prune, That Is The Question**," Asian Conference on Computer Vision (**ACCV**), Springer, **2020**.
- - S. Elkerdawy, M. Elhoushi, A. Singh, H. Zhang and N. Ray, "**One-Shot Layer-Wise Accuracy Approximation For Layer Pruning**," **2020** IEEE International Conference on Image Processing (**ICIP**).
- - S. Elkerdawy, H. Zhang and N. Ray, "**Lightweight Monocular Depth Estimation Model by Joint End-to-End Filter Pruning,**" 2019 IEEE International Conference on Image Processing (**ICIP**).

1) LayerPrune Framework

- Instead of iterative optimization adopted in filter pruning, LayerPrune is a **one-shot pruning paradigm.**
- Evaluation across **multiple statistical criteria**.
- Evaluation **across different inference setup** with respect to batch size (1, 8, 64) and hardware (1080Ti, Xavier embedded board)

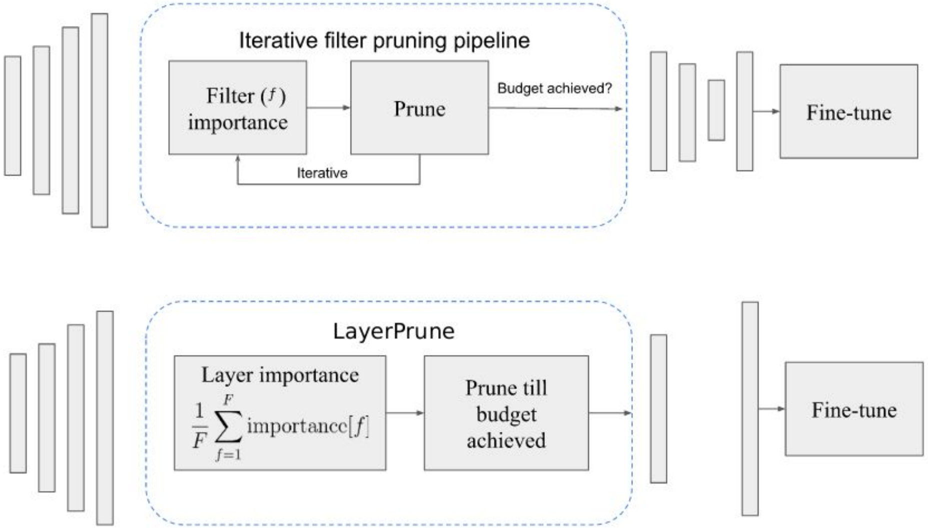
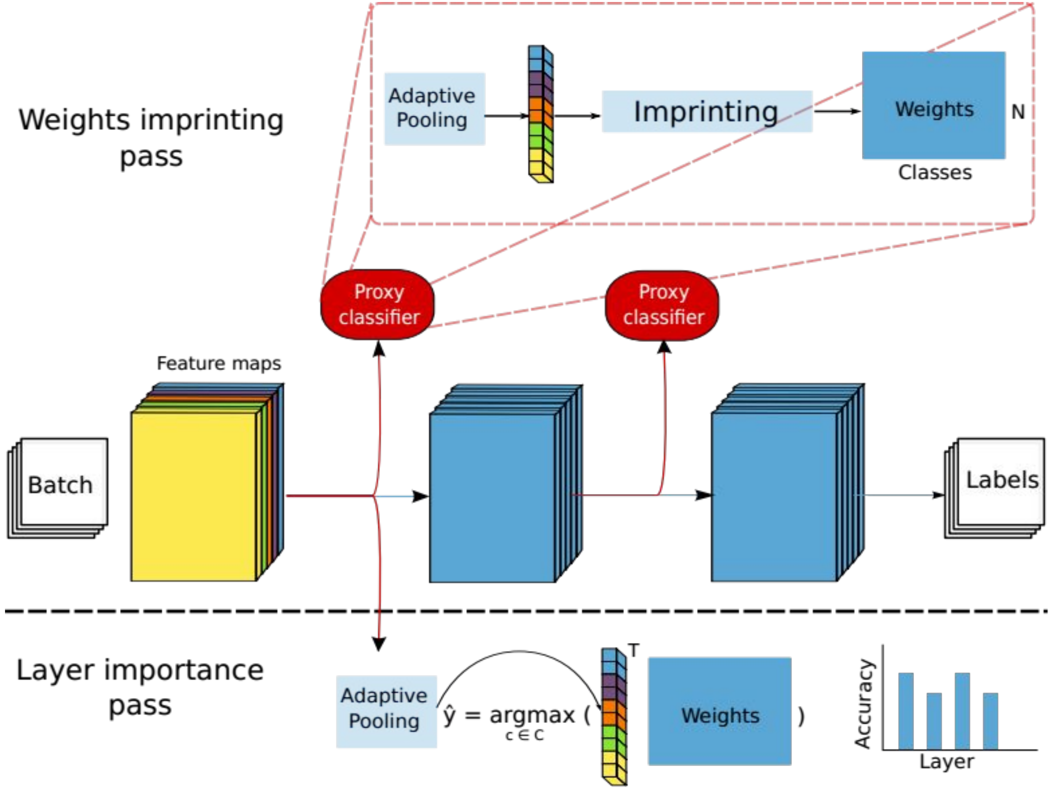


Fig. 3: Main pipeline illustrates the difference between typical iterative filter pruning and proposed LayerPrune framework. Filter pruning (top) produces thinner architecture in an iterative process while LayerPrune (bottom) prunes whole layers in one-shot. In LayerPrune, layer's importance is calculated as the average importance of each filter f in all filters F at that layer.

2) Accuracy Approximation by Imprinting

- We adopt weights imprinting motivated by few-shot learning work [1, 2] to approximate the classification accuracy after each layer.



[1] Qi, Hang, Matthew Brown, and David G. Lowe. "Low-shot learning with imprinted weights." CVPR 2018.

[2] M. Siam, B.O., Jagersand, M.: “Amp: Adaptive masked proxies for few-shot segmentation.” ICCV 2019.

Weights Imprinting

- Classification weights for the *i*th layer $\boldsymbol{W_i}$
-
- Weight for each class $\boldsymbol{c}$ can be represented as the average of embeddings for all samples belonging to that class, each sample with embedding $\boldsymbol{E_j}$
-
- The prediction for each sample $\boldsymbol{j}$ in the validation set is calculated by:

$$W_i[:, c] = \frac{1}{N_c} \sum_{j=1}^{N} I_{[c_j == c]} E_j$$

$$\hat{y}_j = \underset{c \in \{1, \ldots, C\}}{\operatorname{argmax}} W_i[:, c]^T E_j,$$

Evaluation & Analysis

Smoke test - Filter vs Layer pruning

- Layer pruning outperforms filter pruning in accuracy by **7.09%** on average and achieves **up to 60% latency reduction**
- With **same latency budget**, filter pruning shows **higher variance in accuracy**
- Latency constrained optimization with filter pruning is complex and requires careful per layer pruning ratio selection

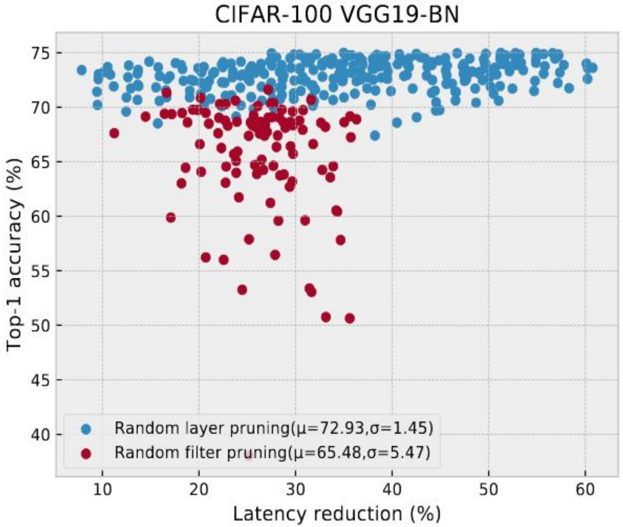
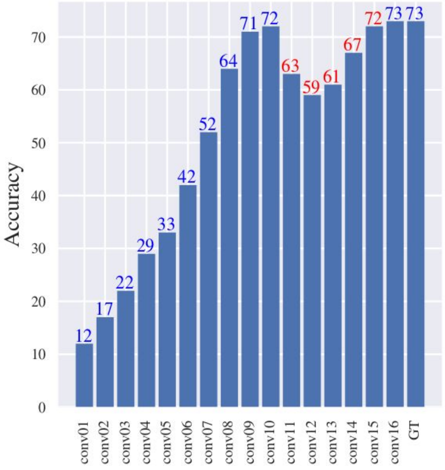


Fig. 4: Example of 100 random filter pruned and layer pruned models generated from VGG19-BN (Top-1=73.11%). Accuracy mean and standard deviation is shown in parentheses. Latency is calculated on 1080Ti with batch size 8.

VGG - CIFAR100

LR → Latency Reduction
bs → Batch size

- Layer pruned models achieve higher accuracy than **baseline** and **filter pruned models.**



Convolution Layers

VGG19 (73.11%)						
Method	Shallower?	Top1 Acc. (%)	1080Ti LR (%)		Xavier LR (%)	
			bs=8	bs=64	bs=8	bs=64
Chen et al. [38]	✓	73.25	56.01	52.86	58.06	49.86
LayerPrune$_8$-Imprint	✓	**74.36**	56.10	53.67	57.79	49.10
Weight norm [25]	✗	73.01	-2.044	-0.873	-4.256	-0.06
ECC [21]	✗	72.71	16.37	36.70	29.17	**36.69**
LayerPrune$_2$	✓	73.60	17.32	14.57	19.512	10.97
LayerPrune$_5$	✓	**74.80**	**39.84**	**37.85**	**41.86**	34.38
Slimming [31]	✗	72.32	16.84	**40.08**	40.55	**39.53**
LayerPrune$_2$	✓	73.60	17.34	13.86	18.85	10.90
LayerPrune$_5$	✓	**74.80**	**39.56**	37.30	**41.40**	34.35
Taylor [24]	✗	72.61	15.87	19.77	-4.89	17.45
LayerPrune$_2$	✓	73.60	17.12	13.54	18.81	10.89
LayerPrune$_5$	✓	**74.80**	**39.36**	**37.12**	**41.34**	**34.44**

Table 1: **Comparison of different pruning methods on VGG19-BN CIFAR-100.** The accuracy for baseline model is shown in parentheses. LR, bs stands for latency reduction and batch size respectively. x in LayerPrune$_x$ indicates number of layers removed. -ve LR indicates increase in latency. Shallower indicates whether a method prunes layers. Best is shown in **bold**.

ResNet50-ImageNet

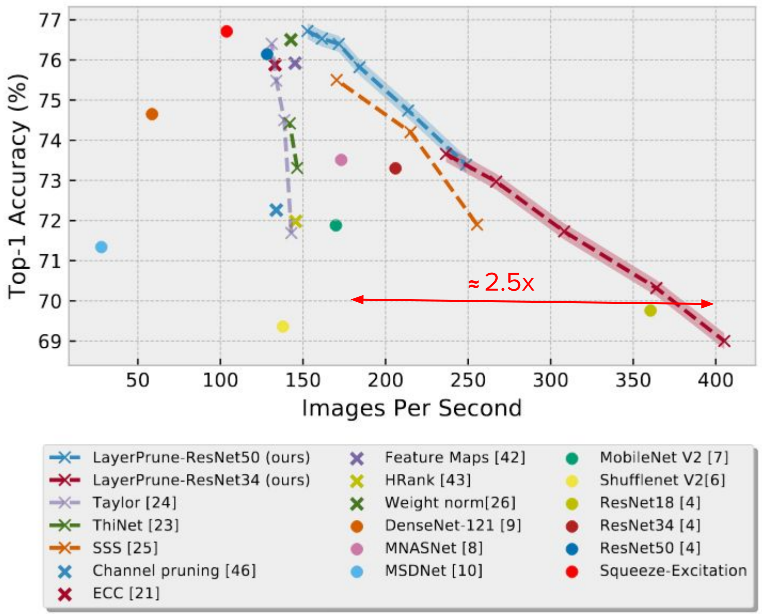


Fig. 2: Evaluation on ImageNet between our LayerPrune framework, handcrafted architectures (dots) and pruning methods on ResNet50 (crosses). Inference time is measured on 1080Ti GPU.

Conclusion

- We presented LayerPrune framework, main findings:

    - For a filter criterion, training a LayerPrune model based on this criterion **achieves the same, if not better, accuracy as the filter pruned model obtained by using the same criterion with higher latency reduction.**
    - Latency reduction of **filter pruned models depend on hardware targets and batch sizes unlike layer pruned models.**
    - We also showed the importance of incorporating **accuracy approximation in layer ranking by imprinting.**

Future work

- Large models are needed for those hard samples while we can most of the time perform fairly well using a light-weight model.
- Can we **dynamically** process different parts/sub-networks from the large model **conditioned** on the input?
    - Conditional or dynamica inference

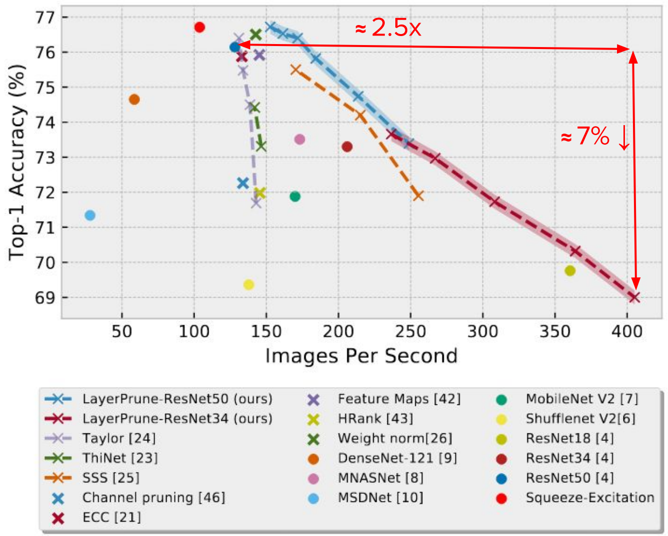


Fig. 2: Evaluation on ImageNet between our LayerPrune framework, handcrafted architectures (dots) and pruning methods on ResNet50 (crosses). Inference time is measured on 1080Ti GPU.

Thanks!

Questions?

ImageNet - ResNet50

LR → Latency Reduction
bs → Batch size

	ResNet50 baseline (76.14)						
	Method	Shallower?	Top1-accuracy (%)	LR(%) 1080Ti bs=1	LR (%) 1080Ti bs=64	LR (%) Xavier bs=1	LR (%) Xavier bs = 64
Weight norm	Weight norm [25]	✗	76.50	6.79	3.46	6.57	8.06
	ECC [21]	✗	75.88	13.52	1.59	-4.91**	3.09**
	LayerPrune$_1$	✓	**76.70**	15.95	4.81	21.38	6.01
	LayerPrune$_2$	✓	76.52	**20.32**	**13.23**	**26.14**	**13.20**
BN Scalars	Batch Normalization	✗	75.23	2.49	1.61	-2.79	4.13
	LayerPrune$_1$	✓	**76.70**	15.95	4.81	21.38	6.01
	LayerPrune$_2$	✓	76.52	**20.41**	**8.36**	**25.11**	**9.96**
Gradients + Weight Norm	Taylor [24]	✗	76.4	2.73	3.6	-1.97	6.60
	LayerPrune$_1$	✓	**76.48**	15.79	3.01	21.52	4.85
	LayerPrune$_2$	✓	75.61	**21.35**	**6.18**	**27.33**	**8.42**
Feature Maps	Feature maps [42]	✗	**75.92**	10.86	3.86	20.25	8.74
	Channel pruning* [48]	✗	72.26	3.54	6.13	2.70	7.42
	ThiNet* [23]	✗	72.05	10.76	**10.96**	15.52	**17.06**
	LayerPrune$_1$	✓	75.00	16.56	2.54	23.82	4.49
	LayerPrune$_2$	✓	71.90	**22.15**	5.73	**29.66**	8.03
Shallow models	SSS-ResNet41 [37]	✓	75.50	25.58	24.17	31.39	21.76
	LayerPrune$_3$-Imprint	✓	**76.40**	22.63	25.73	30.44	20.38
	LayerPrune$_4$-Imprint	✓	75.82	**30.75**	**27.64**	**33.93**	**25.43**
	SSS-ResNet32 [37]	✓	74.20	**41.16**	29.69	**42.05**	29.59
	LayerPrune$_6$-Imprint	✓	**74.74**	40.02	**36.59**	41.22	34.50
High pruning ratio	HRank-2.6x-FLOPs* [43]	✗	71.98	11.89	36.09	20.63	**40.09**
	LayerPrune$_7$-Imprint	✓	**74.31**	**44.26**	**41.01**	**41.01**	38.39

Table 3: **Comparison of different pruning methods on ResNet50 ImageNet.** * manual pre-defined signatures. ** same pruned model optimized for 1080Ti latency consumption model in ECC optimization

ImageNet - ResNet50

- Using the same criteria, LayerPrune models achieve on par accuracy with their counterpart filter pruned models with higher latency reduction.

ResNet50 baseline (76.14)						
Method	Shallower?	Top1-accuracy (%)	LR(%) 1080Ti bs=1	LR (%) 1080Ti bs=64	LR (%) Xavier bs=1	LR (%) Xavier bs = 64
Weight norm [25]	✗	76.50	6.79	3.46	6.57	8.06
ECC [21]	✗	75.88	13.52	1.59	-4.91**	3.09**
LayerPrune$_1$	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune$_2$	✓	76.52	**20.32**	**13.23**	**26.14**	**13.20**
Batch Normalization	✗	75.23	2.49	1.61	-2.79	4.13
LayerPrune$_1$	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune$_2$	✓	76.52	**20.41**	**8.36**	**25.11**	**9.96**
Taylor [24]	✗	76.4	2.73	3.6	-1.97	6.60
LayerPrune$_1$	✓	**76.48**	15.79	3.01	21.52	4.85
LayerPrune$_2$	✓	75.61	**21.35**	**6.18**	**27.33**	**8.42**
Feature maps [42]	✗	**75.92**	10.86	3.86	20.25	8.74
Channel pruning* [48]	✗	72.26	3.54	6.13	2.70	7.42
ThiNet* [23]	✗	72.05	10.76	**10.96**	15.52	**17.06**
LayerPrune$_1$	✓	75.00	16.56	2.54	23.82	4.49
LayerPrune$_2$	✓	71.90	**22.15**	5.73	**29.66**	8.03
SSS-ResNet41 [37]	✓	75.50	25.58	24.17	31.39	21.76
LayerPrune$_3$-Imprint	✓	**76.40**	22.63	25.73	30.44	20.38
LayerPrune$_4$-Imprint	✓	75.82	**30.75**	**27.64**	**33.93**	**25.43**
SSS-ResNet32 [37]	✓	74.20	**41.16**	29.69	**42.05**	29.59
LayerPrune$_6$-Imprint	✓	**74.74**	40.02	**36.59**	41.22	34.50
HRank-2.6x-FLOPs* [43]	✗	71.98	11.89	36.09	20.63	**40.09**
LayerPrune$_7$-Imprint	✓	**74.31**	**44.26**	**41.01**	**41.01**	38.39

Table 3: **Comparison of different pruning methods on ResNet50 ImageNet.** * manual pre-defined signatures. ** same pruned model optimized for 1080Ti latency consumption model in ECC optimization

ImageNet - ResNet50

- On similar latency reduction as SSS (a pruning method that allows shallower models in case of ResNet only), LayerPrune with imprinting achieves higher accuracy.

ResNet50 baseline (76.14)						
Method	Shallower?	Top1-accuracy (%)	LR(%) 1080Ti bs=1	LR (%) 1080Ti bs=64	LR (%) Xavier bs=1	LR (%) Xavier bs = 64
Weight norm [25]	✗	76.50	6.79	3.46	6.57	8.06
ECC [21]	✗	75.88	13.52	1.59	-4.91**	3.09**
LayerPrune_1	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune_2	✓	76.52	**20.32**	**13.23**	**26.14**	**13.20**
Batch Normalization	✗	75.23	2.49	1.61	-2.79	4.13
LayerPrune_1	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune_2	✓	76.52	**20.41**	**8.36**	**25.11**	**9.96**
Taylor [24]	✗	76.4	2.73	3.6	-1.97	6.60
LayerPrune_1	✓	**76.48**	15.79	3.01	21.52	4.85
LayerPrune_2	✓	75.61	**21.35**	**6.18**	**27.33**	**8.42**
Feature maps [42]	✗	**75.92**	10.86	3.86	20.25	8.74
Channel pruning* [48]	✗	72.26	3.54	6.13	2.70	7.42
ThiNet* [23]	✗	72.05	10.76	**10.96**	15.52	**17.06**
LayerPrune_1	✓	75.00	16.56	2.54	23.82	4.49
LayerPrune_2	✓	71.90	**22.15**	5.73	**29.66**	8.03
SSS-ResNet41 [37]	✓	75.50	25.58	24.17	31.39	21.76
LayerPrune_3-Imprint	✓	**76.40**	22.63	25.73	30.44	20.38
LayerPrune_4-Imprint	✓	75.82	**30.75**	**27.64**	**33.93**	**25.43**
SSS-ResNet32 [37]	✓	74.20	**41.16**	29.69	**42.05**	29.59
LayerPrune_6-Imprint	✓	**74.74**	40.02	**36.59**	41.22	34.50
HRank-2.6x-FLOPs* [43]	✗	71.98	11.89	36.09	20.63	**40.09**
LayerPrune_7-Imprint	✓	**74.31**	**44.26**	**41.01**	**41.01**	38.39

Table 3: **Comparison of different pruning methods on ResNet50 ImageNet.** * manual pre-defined signatures. ** same pruned model optimized for 1080Ti latency consumption model in ECC optimization

ImageNet - ResNet50

- Even with aggressive filter pruning (2.6x FLOPs pruning ratio), speed up is noticeable with large batch (36:40%) size but shows small speed gain with small batch size (11:20%).

ResNet50 baseline (76.14)						
Method	Shallower?	Top1-accuracy (%)	LR(%) 1080Ti bs=1	LR (%) 1080Ti bs=64	LR (%) Xavier bs=1	LR (%) Xavier bs = 64
Weight norm [25]	✗	76.50	6.79	3.46	6.57	8.06
ECC [21]	✗	75.88	13.52	1.59	-4.91**	3.09**
LayerPrune_1	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune_2	✓	76.52	**20.32**	**13.23**	**26.14**	**13.20**
Batch Normalization	✗	75.23	2.49	1.61	-2.79	4.13
LayerPrune_1	✓	**76.70**	15.95	4.81	21.38	6.01
LayerPrune_2	✓	76.52	**20.41**	**8.36**	**25.11**	**9.96**
Taylor [24]	✗	76.4	2.73	3.6	-1.97	6.60
LayerPrune_1	✓	**76.48**	15.79	3.01	21.52	4.85
LayerPrune_2	✓	75.61	**21.35**	**6.18**	**27.33**	**8.42**
Feature maps [42]	✗	**75.92**	10.86	3.86	20.25	8.74
Channel pruning* [48]	✗	72.26	3.54	6.13	2.70	7.42
ThiNet* [23]	✗	72.05	10.76	**10.96**	15.52	**17.06**
LayerPrune_1	✓	75.00	16.56	2.54	23.82	4.49
LayerPrune_2	✓	71.90	**22.15**	5.73	**29.66**	8.03
SSS-ResNet41 [37]	✓	75.50	25.58	24.17	31.39	21.76
LayerPrune_3-Imprint	✓	**76.40**	22.63	25.73	30.44	20.38
LayerPrune_4-Imprint	✓	75.82	**30.75**	**27.64**	**33.93**	**25.43**
SSS-ResNet32 [37]	✓	74.20	**41.16**	29.69	**42.05**	29.59
LayerPrune_6-Imprint	✓	**74.74**	40.02	**36.59**	41.22	34.50
HRank-2.6x-FLOPs* [43]	✗	71.98	11.89	36.09	20.63	**40.09**
LayerPrune_7-Imprint	✓	**74.31**	**44.26**	**41.01**	**41.01**	38.39

Table 3: **Comparison of different pruning methods on ResNet50 ImageNet.** * manual pre-defined signatures. ** same pruned model optimized for 1080Ti latency consumption model in ECC optimization