UNIVERSITY OF ALBERTA

Session: Sparse and Low Rank Models

Abu Dhabi ICIP 2020

IEEE International Conference on Image Processing
25-28 October 2020, United Arab Emirates

IEEE Signal Processing Society

One-Shot Layer-Wise Accuracy Approximation for Layer Pruning

Sara Elkerdawy[1], Mostafa Elhoushi[2], Abhineet Singh[1], Hong Zhang[1], Nilanjan Ray[1]

(1) Department of Computing Science, University of Alberta
(2) Toronto Heterogeneous Compilers Lab, Huawei

Motivation

- Deep neural networks (DNN) are one of the state-of-the-art methods for a variety of prediction and supervised learning tasks.

- Because DNN models can be large, inference becomes computationally expensive. Embedded and mobile devices that are resource constrained may not be able to effectively use DNNs trained for powerful high-end GPU environment.

Model Pruning

Related work

- Weights pruning [1,2,3]
    - Speedup requires special backend library
- Hardware-agnostic filter pruning [4,5,6]
    - The number of parameters or FLOPs do not correlate strongly with latency
- Hardware-aware filter pruning [8,9,10]

[1] Han, Song, et al. "Learning both weights and connections for efficient neural network." NeurIPS 2015.
[2] Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through *L0* Regularization." ICLR 2018.
[3] Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." ICLR 2019.
[4] Li, Hao, et al. "Pruning filters for efficient convnets." ICLR (2017).
[5] Liu, Zhuang, et al. "Learning efficient convolutional networks through network slimming." ICCV (2017)
[6] Molchanov, Pavlo, et al. "Importance estimation for neural network pruning." CVPR (2019)
[7] van Werkhoven, Ben. "Kernel Tuner: A search-optimizing GPU code auto-tuner." Future Generation Computer Systems 90 (2019)
[8] Yang, Tien-Ju, et al. "Netadapt: Platform-aware neural network adaptation for mobile applications." ECCV (2018).
[9] He, Yihui, et al. "Amc: Automl for model compression and acceleration on mobile devices." ECCV (2018)
[10] Yang, Haichuan, Yuhao Zhu, and Ji Liu. "Ecc: Platform-independent energy-constrained deep neural network compression via a bilinear regression model." CVPR (2019).

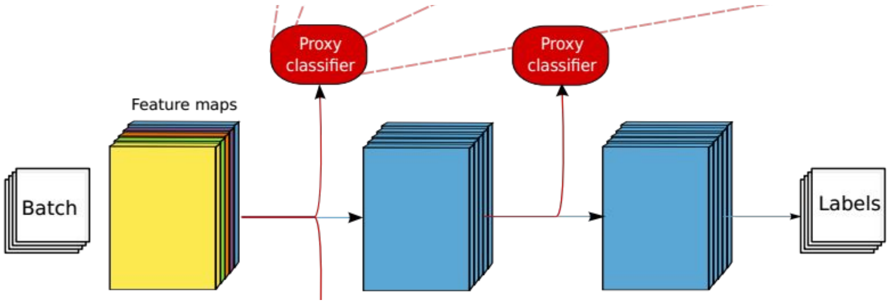
Filter Pruning Limitations

- The range of attainable latency reduction is limited by the depth of the model.
  - Filter pruning, in general, is able to achieve slimmer models
- Resource consumption (e.g latency) modeling can take days to generate data measurements per hardware and architecture specially on low-end hardware platforms.

One-Shot Layer-Wise Accuracy Approximation for Layer Pruning

Layer-wise Accuracy Approximation

- Proxy classifiers after each layer ➔ Accuracy up to this layer.
- How to calculate layer-wise accuracy efficiently without the need for re-training?
  - We adopt weights imprinting
  - Motivated by few-shot learning work [11, 12]



[11] Qi, Hang, Matthew Brown, and David G. Lowe. "Low-shot learning with imprinted weights." CVPR 2018.
[12] 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 $\boldsymbol{E_j}$

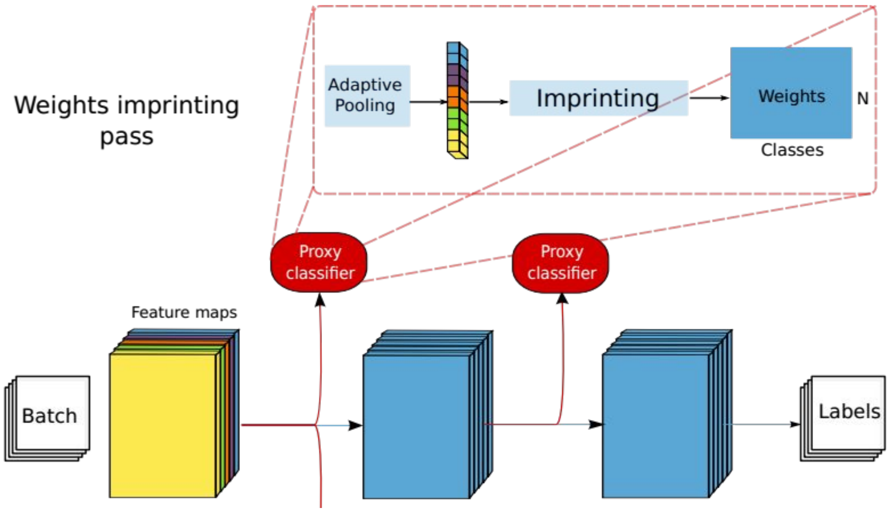
$$W_i[:, c] = \frac{1}{N_c} \sum_{j=1}^{N} I_{[c_j == c]} E_j$$

- The prediction for each sample $\boldsymbol{j}$ in the validation set is calculated by:

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

Layer-wise Accuracy Approximation

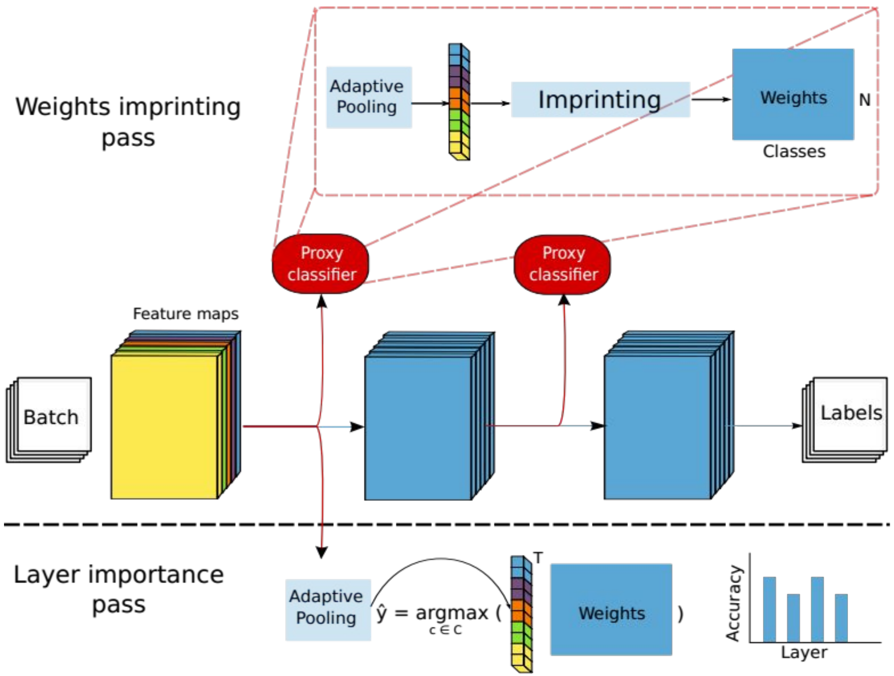
- Proxy classifiers after each layer → Accuracy up to this layer.
- How to calculate layer-wise accuracy efficiently without the need for re-training?
  - We adopt weights imprinting
  - Motivated by few-shot learning work [11, 12]
- One-shot layer importance by imprinting



[11] Qi, Hang, Matthew Brown, and David G. Lowe. "Low-shot learning with imprinted weights." CVPR 2018.
[12] M. Siam, B.O., Jagersand, M.: “Amp: Adaptive masked proxies for few-shot segmentation.” ICCV 2019.

Layer-wise Accuracy Approximation

- Proxy classifiers after each layer ➔ Accuracy up to this layer.
- How to calculate layer-wise accuracy efficiently without the need for re-training?
    - We adopt weights imprinting
    - Motivated by few-shot learning work [11, 12]
- One-shot layer importance by imprinting
- Rank layers based on their accuracy relative to previous pruning candidate

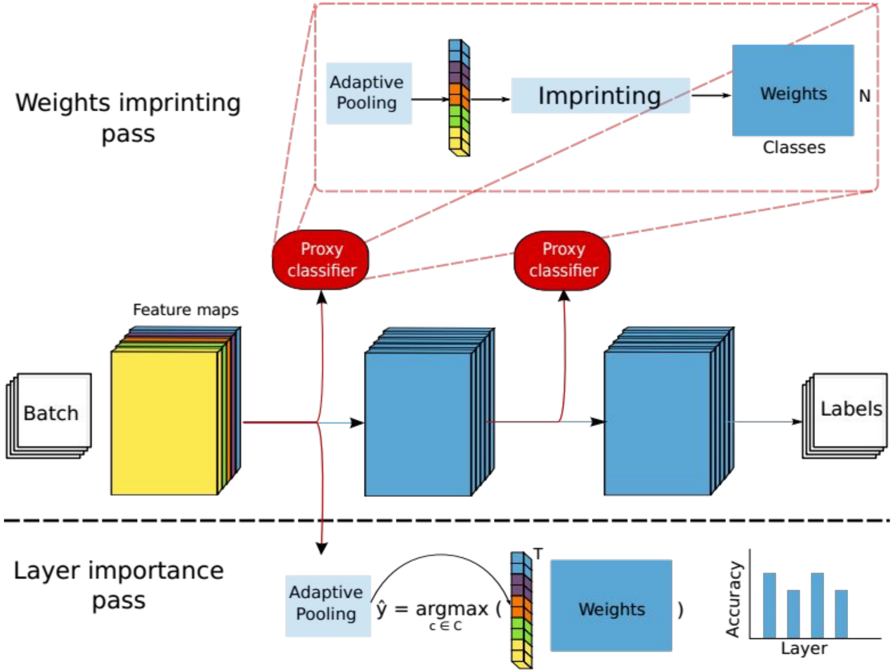


[11] Qi, Hang, Matthew Brown, and David G. Lowe. "Low-shot learning with imprinted weights." CVPR 2018.
[12] M. Siam, B.O., Jagersand, M.: "Amp: Adaptive masked proxies for few-shot segmentation." ICCV 2019.

Layer-wise Accuracy Approximation

- Classification weights for the *i*th layer $\boldsymbol{W_i}$
- Weight for each class **c** can be represented as the average of embeddings for all samples belonging to that class, each sample with embedding $\boldsymbol{E_j}$

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



Evaluation & Analysis

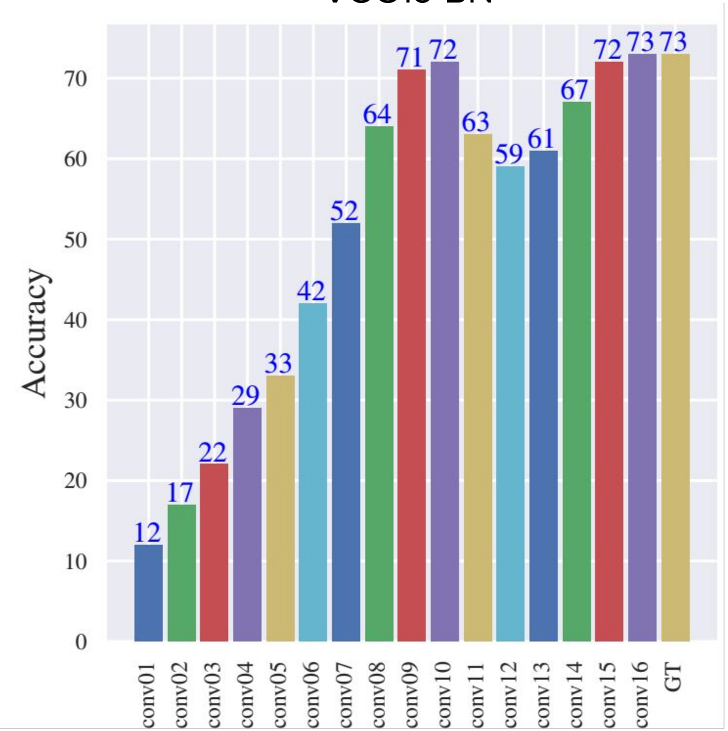
Experiments

- Datasets
  - CIFAR-100
  - ImageNet
- Architecture
  - VGG
  - ResNet-50

VGG - CIFAR100

- Drop in accuracy followed by an increasing trend from conv10 to conv14.
  - This is likely because the number of features is the same from conv10 to conv12.
  - We start to observe an accuracy increase only at conv13 that follows a max pooling layer and has twice as many features.

VGG19 BN



VGG19 Convolution Layers

VGG - CIFAR100

- Drop in accuracy followed by an increasing trend from conv10 to conv14.
- Our method improves on the previously reported accuracy [Masking in 22] by **1.18%** while achieving a **43.70%** latency reduction over VGG19 vs. the previous state-of-the-art at **26.75%** (Masking).
- In terms of accuracy, we outperform the average of **10 randomly** layer-pruned models of similar latency reduction as ours (≈ 40%) by **5.43%**

Method	Accuracy	N layers	Params ($1e^6$)	Latency reduction (%)
VGG19 baseline	73.11	16	20.09	0
Random layer pruning	68.95	-	-	40.00
Layer-wise proxy (ours)	**74.38**	**12**	9.28	**43.70**
Slimming [9]	72.32	16	5.00	25.26
Masking [22]	**73.2**	16	**4.20**	**26.75**
Taylor [8]	72.61	16	**4.79**	23.24
ECC [15]	72.71	16	7.86	25.17

Table 1: Pruning results on CIFAR100 showing **best** and **second best** in each criterion. Latency reduction is measured on 1080Ti GPU across 1000 runs.

ResNet50 - ImageNet

- On bar with the state-of-the art filter pruning method in accuracy.
- Minimal model that can be achieved by filter pruning methods such as ECC achieves **14.45%** latency reduction.

Method	Accuracy	N layers	Params ($1e^6$)	Latency reduction (%)
ResNet-50 baseline	76.14	53	25.5	0
Layer-wise proxy - 1 block (ours)	**76.72**	**50**	25.4	**16.06**
Layer-wise proxy - 1 block + 3 layers (ours)	75.0	**44**	24.1	**24.02**
ThinNet [9]	72.04	53	**16.94**	10.52
Taylor [7]	**76.43**	53	**22.6**	2.73
ECC [14]	74.88	53	23.5	1.93
ECC minimal model	16.3	53	6.14	11.56

Table 2: Pruning results on ImageNet showing **best** and **second best** in each criterion. Latency reduction is measured on 1080Ti GPU across 1000 runs with batch size=1.

* Minimal model is the one with the same depth as the dense model but with one filter per each prunable layer.

ResNet50 - ImageNet

- We further compare imprinting layer pruning on similar latency budget with smaller ResNet variants such as ResNet34 and ResNet41
- We outperform ResNet41 by **0.9%** and ResNet34 by **1.44%**.

Method	Accuracy	N layers	Params ($1e^6$)	Latency reduction (%)
ResNet-50 baseline	76.14	53	25.5	0
Layer-wise proxy - 4 blocks (ours)	**76.40**	**41**	**24.8**	25
ResNet-41 [24]	75.50	44	25.3	25
Layer-wise proxy - 6 blocks (ours)	**74.74**	**35**	23.4	39
ResNet-34 [19]	73.30	37	**21.7**	39

Conclusion

- We proposed a one-shot layer pruning method that incorporates a layer-wise accuracy approximation through imprinting.
- Our method achieves higher latency reduction compared to filter pruning methods and manually crafted variants.
- Our method is not limited by model architecture design.

Thanks!

Code: https://github.com/selkerdawy/one-shot-layer-pruning