week 2

EPruner

Network Pruning using Adaptive Exemplar Filters (IEEE TNNLS 2021)

Mingbao Lin, Rongrong Ji, Senior Member, IEEE, Shaojie Li, Yan Wang,
Yongjian Wu, Feiyue Huang, Qixiang Ye, Senior Member, IEEE

2022.09.23

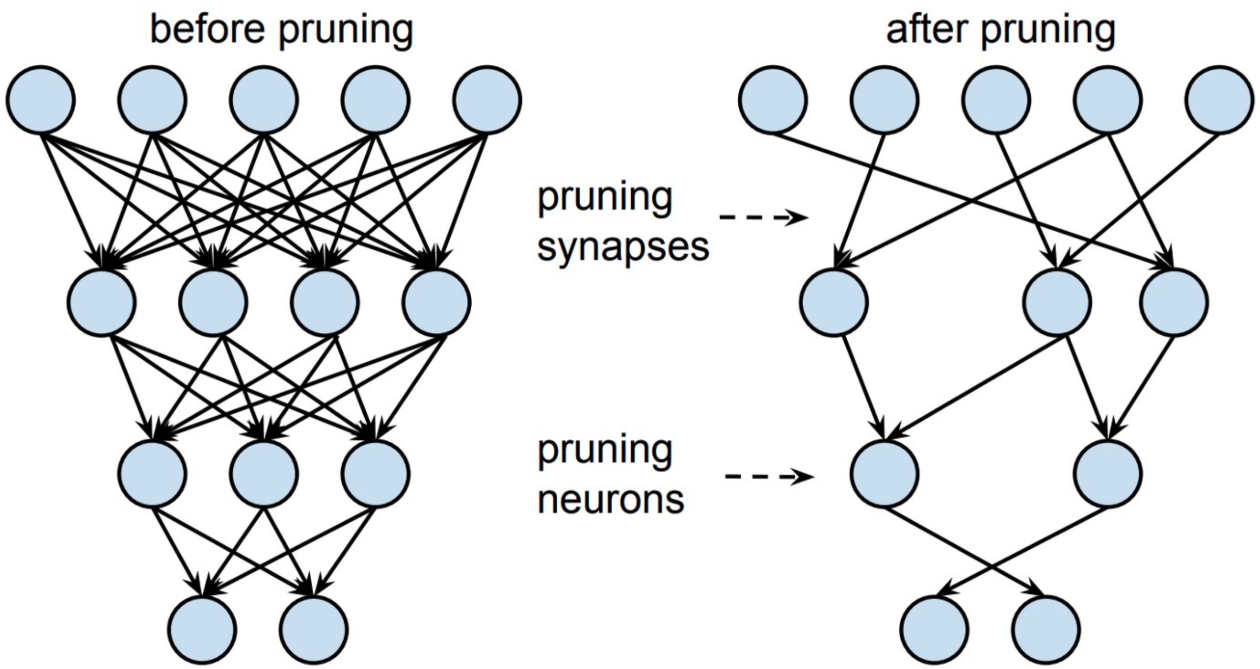
곽민지

Introduction

(1) Pruning

- **Pruning**

  : Model의 weight들 중 중요도가 낮은 weight의 연결을 제거하여 모델의 parameter를 줄이는 방법



https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9

Introduction

(1) Pruning

Most high-performing CNNs are designed to execute on high-end GPUs with substantial memory and computational power, which hinders their practical applications in resource-constrained environments, such as mobile and embedded devices.

⬇

Model compression techniques

(low-rank decomposition, parameter quantization and network pruning)

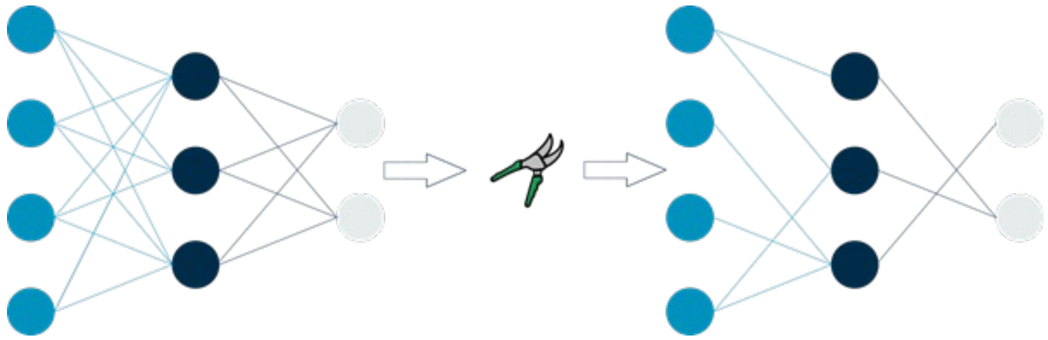
⬇

- Popular network pruning algorithms reduce redundant information by optimizing hand-crafted models and may cause suboptimal performance and long time in selecting filters.

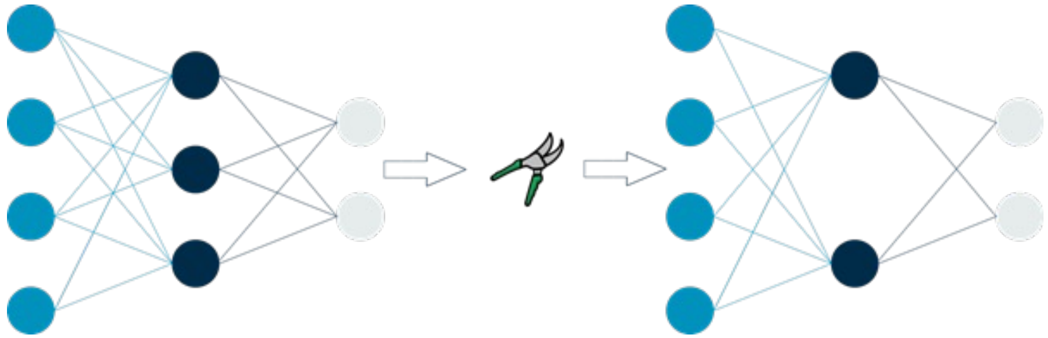
Introduction

(1) Pruning

(1) Unstructured weight pruning



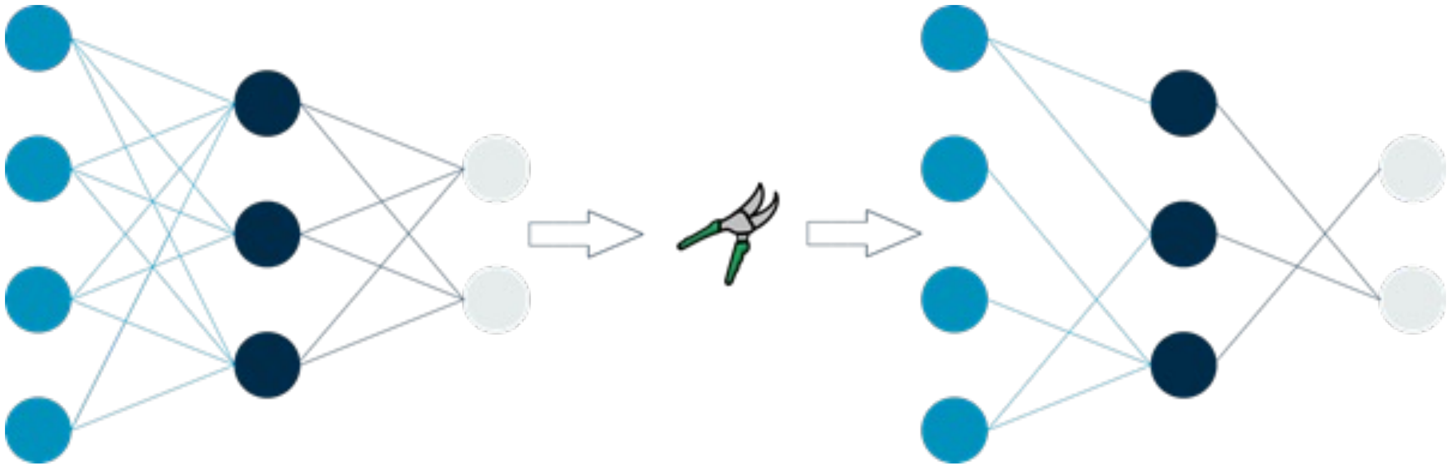
(2) Structured filter pruning



Introduction

(1) Pruning

(1) Unstructured weight pruning



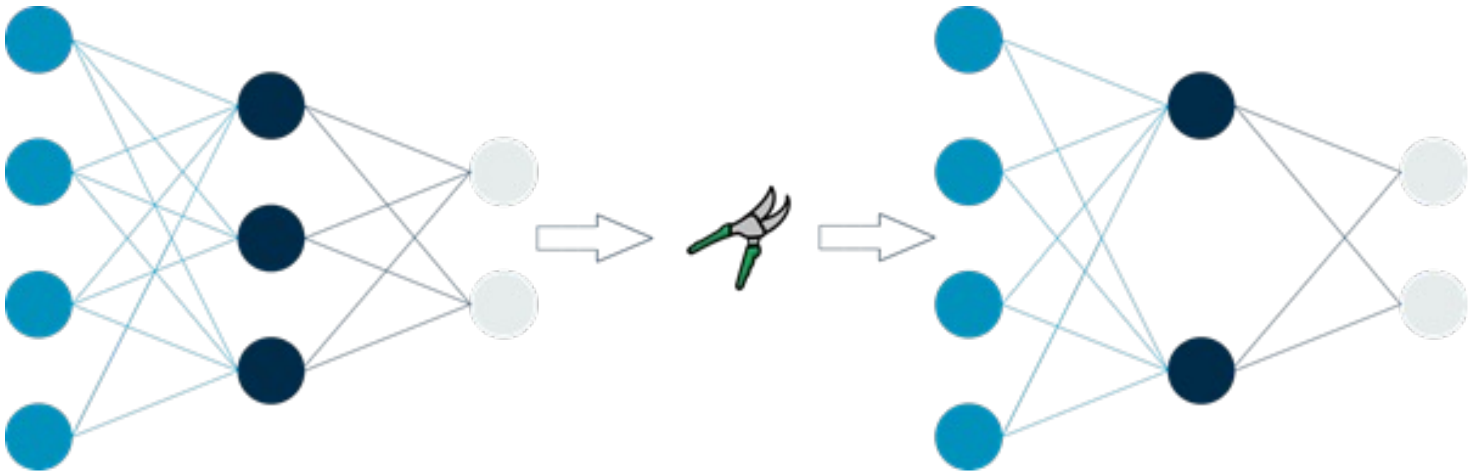
- Remove the individual neurons in the filter or the connection between fully-connected layers.
- Require specialized hardware or software to support the practical speedup.

→ weight를 0으로 만드는 구조이기 때문에, 실질적으로 matrix 연산이 필요 (inference 속도 개선이 힘들다.)

Introduction

(1) Pruning

(2) Structured weight pruning



- Remove the entire filters and corresponding channels.
- Require no extra requirements for the inference platforms.

→ It can be easily deployed.

Introduction

(1) Pruning

Filter pruning

- (1) Rule-based pruning
- (2) Regularization-driven pruning
- (3) Architecture-search-driven pruning

Introduction

(1) Pruning

Filter pruning

(1) Rule-based pruning

- Decide the pruned network architecture by hand-crafted designation.
- Inherit the most important filter weights either measured by an intrinsic property of the pretrained model.
- Typically, the designated architecture is sub-optimal.
- It usually performs layer-wise fine-tuning/optimization to recover the accuracy, which is computationally intensive.

Introduction

(1) Pruning

Filter pruning

(1) Rule-based pruning

ex) L1-norm based Filter Pruning

➔ 모델을 학습시킨 후 각 필터마다 L1 norm (필터에 속한 가중치 값들의 절대값의 합)을 구하고, 결과값이 작은 순서대로 pruning

➔ 절대값이 큰 필터일수록 중요하다고 가정하는 것

Introduction

(1) Pruning

Filter pruning

(2) Regularization-driven pruning

- Performs model complexity reduction by retraining the network with hand-crafted constraints.

- Retraining the model is expensive.

- The introduced hyper-parameters also require manual analysis.

Introduction

(1) Pruning

Filter pruning

(3) Architecture-search-driven pruning

- Focus on searching for a better architecture, typically through heuristic-based policies, such as evolutionary algorithm or the bee colony algorithm.

- The architecture search is data-dependent thus also computationally intensive.

- The search results are not deterministic.

Introduction

(1) Pruning

Filter pruning

- (1) Rule-based pruning
- (2) Regularization-driven pruning
- (3) Architecture-search-driven pruning

Existing methods still heavily depend on hand-crafted rules from humans and involve time-consuming retraining/search.

Introduction

(2) EPruner

- EPruner is able to figure out an adaptive number of exemplars and identify more informative exemplars in a unified framework.

- Higher reduction of model complexity, less time consumption in important filter selection and deterministic pruning results.

  → EPruner lessens the involvement of human labor in the pruning.

- Extensive experiments on VGGNet, GoogLeNet and ResNets.

  → We can demonstrate the efficacy of EPruner in reducing the model complexity and better performance in comparison with several state of the arts.

Method

(1) framework

- We use a graph message-passing algorithm Affinity propagation to make our approach adaptive to the pretrained CNN in both finding the number of exemplars and identifying more informative exemplars.
  → We treat each filter as a data point and connect them with weighted edges.
  → Affinity Propagation could select the representative exemplars from all the filters.

- Affinity propagation is able to obtain an adaptive number of exemplars, with a non-learning hyper-parameter controlling the compression strength.

- We found the proposed approach works surprisingly well even with the naïve negative Euclidean distance, over-performing state-of-the-art pruning algorithms with less time consumption in determining the important filters.

Method

(1) framework

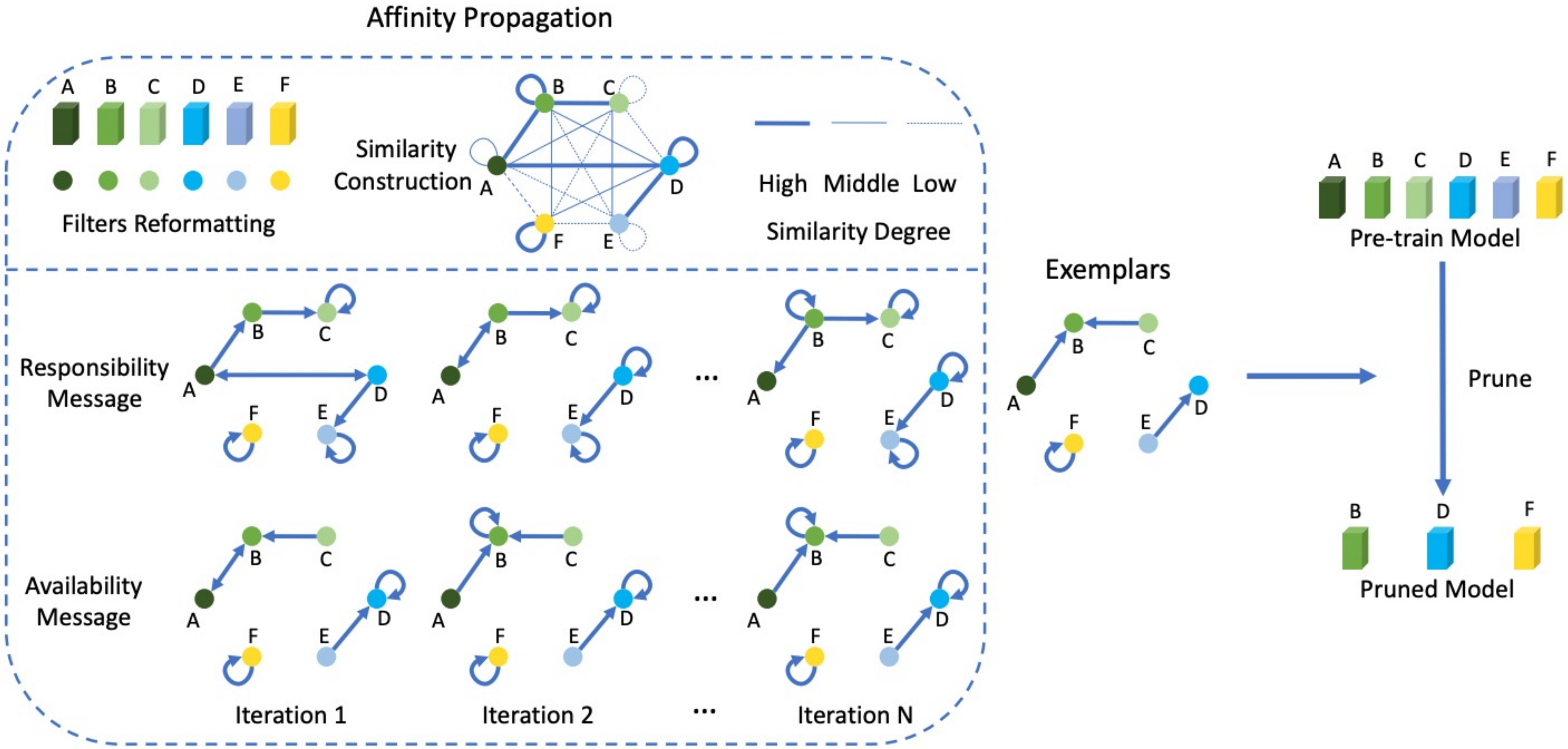
- Another counter-intuitive finding is related with the observation that randomly initialized weights can perform better than inheriting the most important filters.

- We find that proper filter weight initialization based on the exemplars could significantly outperform randomly initialized filter weights.

  → 이전에는 pruning을 통해 얻는 것은 남겨진 '구조'이고, '구조의 내용물'은 중요하지 않다는 의견으로 random initialization을 사용하기도 했음

Method

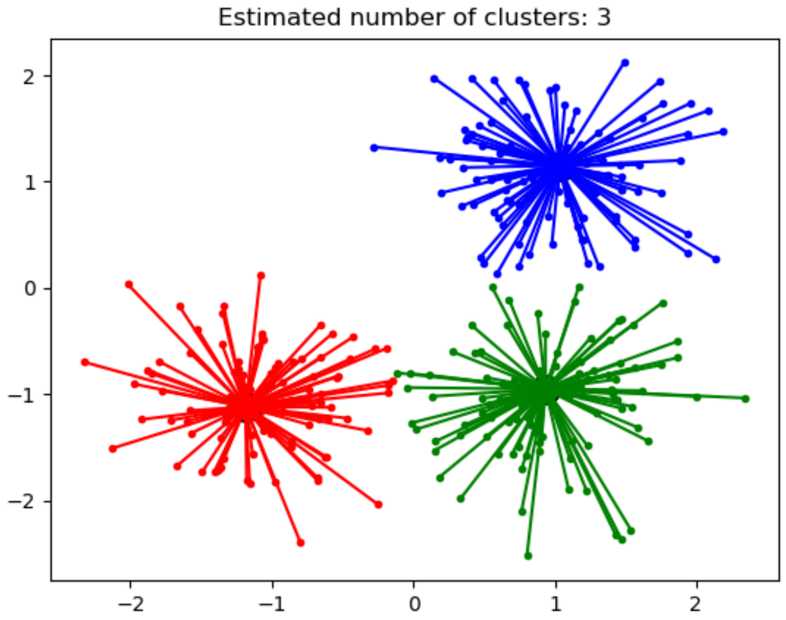
(1) framework



Method

(2) Affinity propagation

- Clustering method
- Affinity propagation does not require you to specify the number of clusters.
- 모든 데이터가 특정한 기준에 따라 자신을 대표할 대표 데이터를 선택 (스스로가 자기 자신을 대표할 수도 있음.)
- Similarity matrix를 계산하고, 이를 기반으로 Responsibility matrix와 Availability matrix를 번갈아 가면서 반복적으로 계산



Method

(2) Affinity propagation

- similarity $s_k(i, j)$

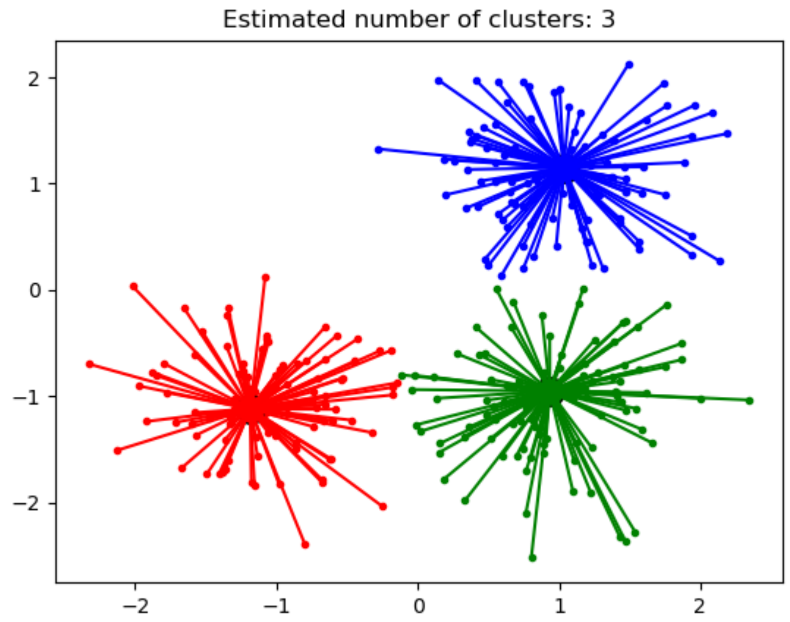
  : i번째 데이터와 j번째 데이터의 유사도

- responsibility $r(i, j)$

  : j번째 데이터가 i번째 데이터의 대표가 되어야 한다는 근거

- availability $a(i, j)$

  : i번째 데이터가 j번째 데이터를 대표로 선택해야 한다는 근거



Method

(2) Affinity propagation

- Similarity

$$s_k(i,j) = -\|\mathbf{w}_{ki} - \mathbf{w}_{kj}\|^2 \quad s.t.\ \ 1 \leq i,j \leq c_k,\ i \neq j.$$

$$s_k(i,i) = \beta * \mathrm{median}(\mathbf{w}_{ki}) \qquad s.t.\ \ 0 < \beta \leq 1$$

- Responsibility

$$r(i,j) \leftarrow s(i,j) - \max_{j' s.t. j' \neq j} \left(a(i,j') + s(i,j')\right)$$
$$s.t.\ \ 1 \leq i,j \leq c_k,\ i \neq j,$$

$$r(i,i) \leftarrow s(i,i) - \max_{i' s.t. i' \neq i} s(i,i')$$

Notation	Meaning
$\mathbf{w}_k$	Filters in the k-th layer
$\mathbf{w}_{ki}$	The i-th filter in the k-th layer
$\bar{\mathbf{w}}_k$	The exemplar filters in the k-th layer
$\bar{\mathbf{w}}_{ki}$	The i-th exemplar filter in the k-th layer
$s_k(i,j)$	How well the filter $\mathbf{w}_{kj}$ is suited to be the exemplar of the filter $\mathbf{w}_{ki}$
$r_k(i,j)$	The "responsibility" message in the Affinity Propagation
$a_k(i,j)$	The "availability" message in the Affinity Propagation

Method

(2) Affinity propagation

- Availability

$$a(i,j) \leftarrow \min\{0, r(j,j) + \sum_{i' \ s.t. \ i' \notin \{i,j\}} \max\left(0, r(i',j)\right)\}$$

$$s.t. \ 1 \leq i,j \leq c_k, \ i \neq j.$$

$$a(i,i) \leftarrow \sum_{i' \ s.t., i' \neq i} \max\left(0, r(i',i)\right)$$

Notation	Meaning
$\mathbf{w}_k$	Filters in the k-th layer
$\mathbf{w}_{ki}$	The i-th filter in the k-th layer
$\bar{\mathbf{w}}_k$	The exemplar filters in the k-th layer
$\bar{\mathbf{w}}_{ki}$	The i-th exemplar filter in the k-th layer
$s_k(i,j)$	How well the filter $\mathbf{w}_{kj}$ is suited to be the exemplar of the filter $\mathbf{w}_{ki}$
$r_k(i,j)$	The "responsibility" message in the Affinity Propagation
$a_k(i,j)$	The "availability" message in the Affinity Propagation

Method

(2) Affinity propagation

- responsibility와 availability가 더이상 변화하지 않고 수렴하면 계산이 종료, iteration 정하기도 함.

  (In paper, weighted factor = 0.5, iteration = 200)

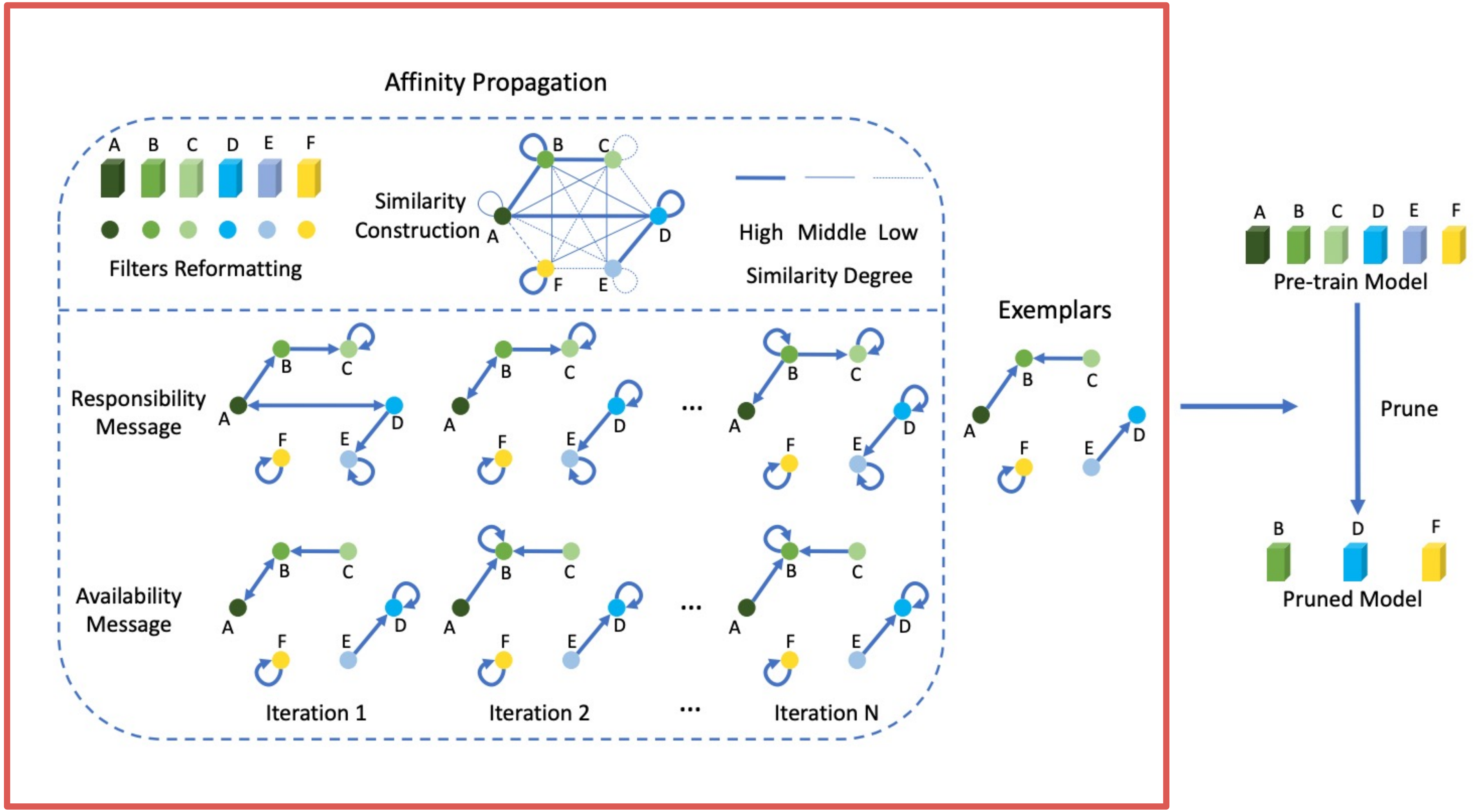
$$r^t(i,j) = \lambda * r^{(t-1)}(i,j) + (1-\lambda) * r^t(i,j),$$

$$a^t(i,j) = \lambda * a^{(t-1)}(i,j) + (1-\lambda) * a^t(i,j),$$

$$\arg\max_j \; r(i,j) + a(i,j) \qquad s.t. \;\; 1 \leq j \leq c_k.$$

Method

(2) Affinity propagation



Method

(3) Weight Initialization

The fine-tuning is required to recover the accuracy of pruned network such that
it would keep a better or at least comparable performance against the pre-trained model.

➔ *How to feed a good weight initialization to the pruned network architecture for the follow-up fine-tuning.*

⬇

Exemplar weights

: The weights of exemplar filters are regarded as the initial weights.

Method

(3) Weight Initialization

	Exemplar Weights (%)	Random Projection (%)	ℓ_1-norm weights (%)	Random Initialization (%)
VGGNet-16	**93.08**	92.95	92.98	92.61
GoogLeNet	**94.99**	94.49	94.41	94.19
ResNet-56	**93.18**	92.44	93.03	91.45
ResNet-110	**93.62**	93.02	92.99	92.44
ResNet-18	**67.31(87.42)**	66.68(87.45)	67.01(87.42)	66.46(87.13)
ResNet-34	**70.95(89.97)**	70.79(89.91)	70.76(89.93)	70.71(89.78)
ResNet-50	**74.26(91.88)**	73.80(91.83)	73.99(91.82)	73.54(91.55)
ResNet-101	**75.45(92.70)**	75.31(92.51)	75.40(92.58)	75.12(92.25)
ResNet-152	**76.51(93.22)**	76.43(93.14)	76.46(93.20)	76.15(92.97)

Accuracy

Experiments

(1) Training

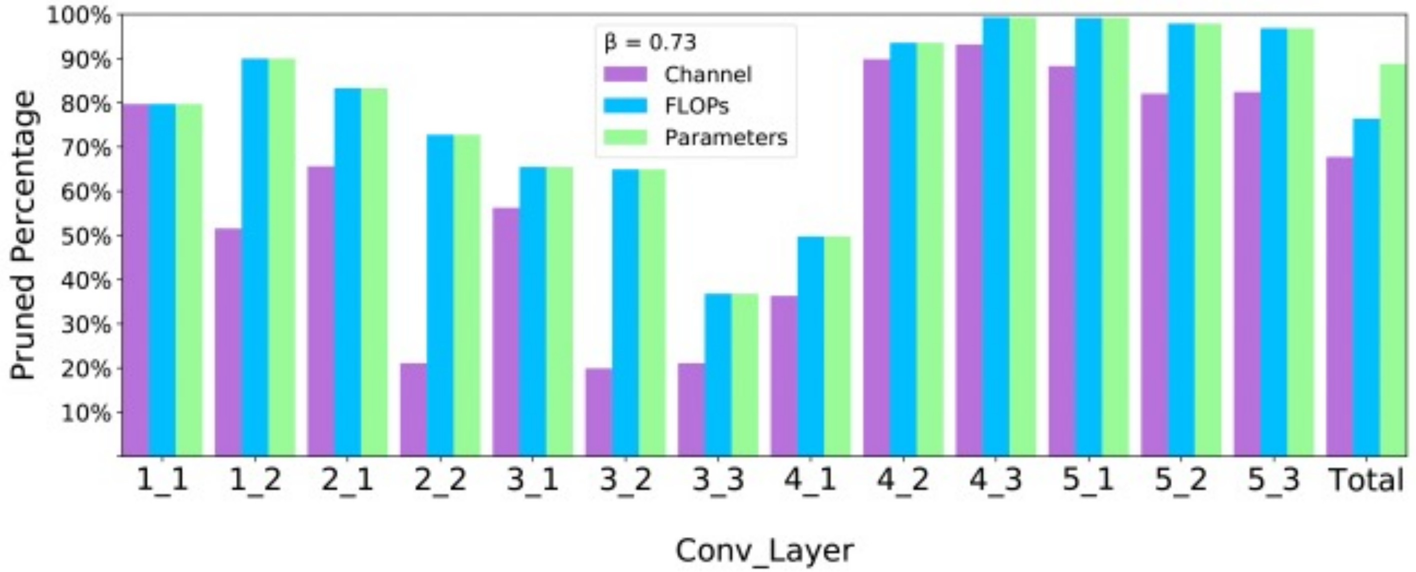
- CIFAR-10
- VGGNet, GoogLeNet, ResNet
- FLOPs: Floating point Operation Per Second (절대적인 연산량의 횟수를 지칭)

Model	Top1-acc	↑↓	Channels	Pruning Rate	FLOPs	Pruning Rate	Parameters	Pruning Rate
VGGNet-16	93.02%	0.00%	4224	0.00%	314.59M	0.00%	14.73M	0.00%
VGGNet-16 0.5×	92.68%	0.34%↓	2112	50.00%	79.20M	74.97%	3.69M	74.79%
EPruner-0.73	**93.08%**	**0.06%↑**	**1363**	**67.73%**	**74.42M**	**76.34%**	**1.65M**	**88.80%**
GoogLeNet	95.05%	0.00%	7904	0.00%	1534.55M	0.00%	6.17M	0.00%
GoogLeNet 0.25×	94.38%	0.67%↓	6236	21.10%	587.20M	61.61%	2.61M	57.72%
EPruner-0.65	**94.99%**	**0.06%↓**	**6110**	**22.70%**	**500.87M**	**67.36%**	**2.22M**	**64.20%**
ResNet-56	93.26%	0.00%	2032	0.00%	127.62M	0.00%	0.85M	0.00%
ResNet-56 0.5×	91.90%	1.36%↓	1528	24.80%	63.80M	49.61%	0.43M	49.82%
EPruner-0.76	**93.18%**	**0.08%↓**	**1450**	**28.64%**	**49.35M**	**61.33%**	**0.39M**	**54.20%**
ResNet-110	93.50%	0.00%	4048	0.00%	257.09M	0.00%	1.73M	0.00%
ResNet-110 0.4×	92.69%	0.81%↓	2806	30.68%	97.90M	61.13%	0.67M	61.62%
EPruner-0.60	**93.62%**	**0.12%↑**	**2580**	**36.26%**	**87.65M**	**65.91%**	**0.41M**	**76.30%**

Experiments

(2) Adaptive pruned architecture

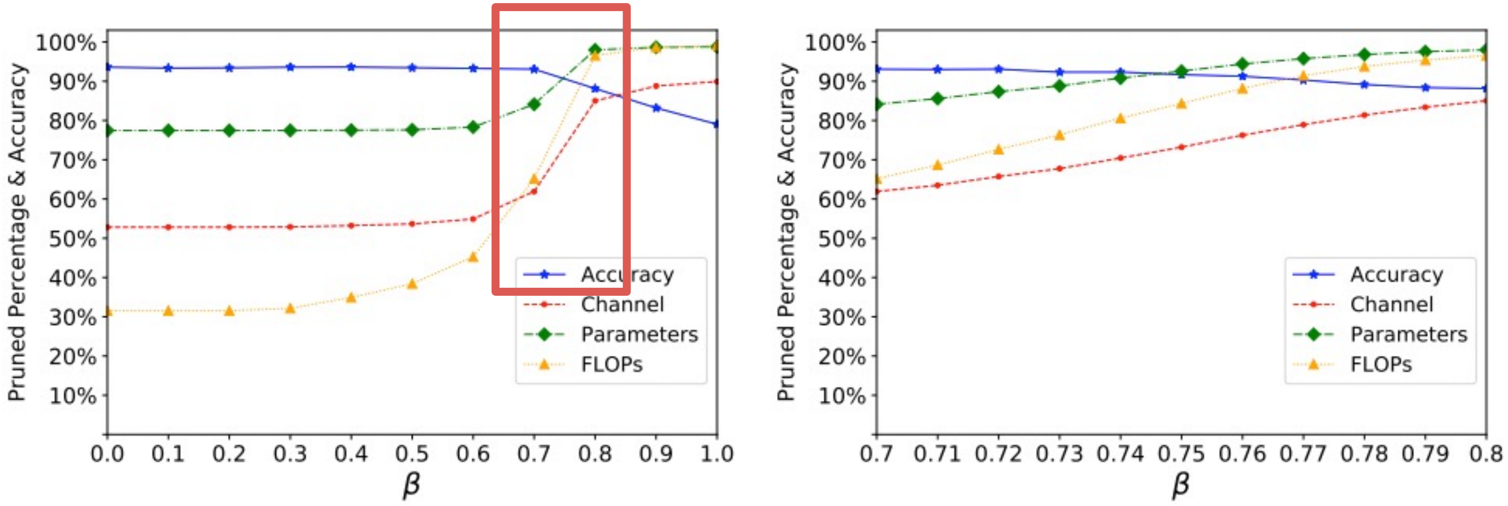
- More filters are preserved in the middle layers.
- EPruner self-adapts to the filter property and derives the deterministic optimal pruned architecture without human involvement.



Experiments

(3) Influence of β

- The β is used to make the reduction of model complexity more adjustable.
  → smaller β leads to more exemplars (fewer reductions of model complexity, but better accuracy)



(a) β ranges from 0 to 1. (b) β ranges from 0.7 to 0.8.

Conclusion

- Select exemplars among filters.
- The affinity propagation is applied to generate high-quality exemplars.
- The optimal architecture with EPruner can be efficiently implemented within a few seconds simply on the CPUs.
- The weights of exemplars can serve as a better warm-up for fine-tuning the network, which justifies the correctness of inheriting the most important filter weights.

Reference

- https://arxiv.org/abs/2101.07985
- https://towardsdatascience.com/unsupervised-machine-learning-affinity-propagation-algorithm-explained-d1fef85f22c8
- https://jaehc.github.io/ml/affinity-propagation/
- https://datascienceschool.net/03%20machine%20learning/16.05%20Affinity%20Propagation.html
- https://simpling.tistory.com/50
- https://do-my-best.tistory.com/entry/Network-pruning