Progressive Gradient Pruning for Classification, Detection and Domain Adaptation

ICPR 2020

Le Thanh Nguyen-Meidine, Eric Granger, Madhu Kiran, Marco Pedersoli, Louis-Antoine Blais-Morin

LIVIA LABORATOIRE D'IMAGERIE, DE VISION ET D'INTELLIGENCE ARTFICIELLE

Genetec

ÉTS Le génie pour l'industrie

1) Introduction

Channel pruning methods can be seen as:

- **as a post processing step (L1, Entropy):** train a model -> prune -> fine-tune
- **as an iterative process on trained model (Taylor[2]):** train a model -> iteratively prune and fine-tune
- **as a reconstruction error (ThiNet, DCP):** train a model -> minimize the error between the pruned model and the original model
- **from scratch using constraints (DCP, Slimming):** add pruning constraints to original model -> optimize the model and the pruning constraints
- **as progressive pruning (PSFP, ours):** pruning while training the model from scratch without any constraints

1) Introduction

Current challenges with SOA methods:

- Existing methods for pruning from scratch is difficult to optimize
- Existing progressive pruning method does not take advantage of pruning during training
- It does not handle back-propagation pruning properly

1) Introduction

Contributions:

- A pruning technique and criterion suitable for training.
- Momentum pruning for backward pass
- The technique can be easily adapted to other tasks such as object detection or unsupervised domain adaptation

2) Progressive Gradient Pruning

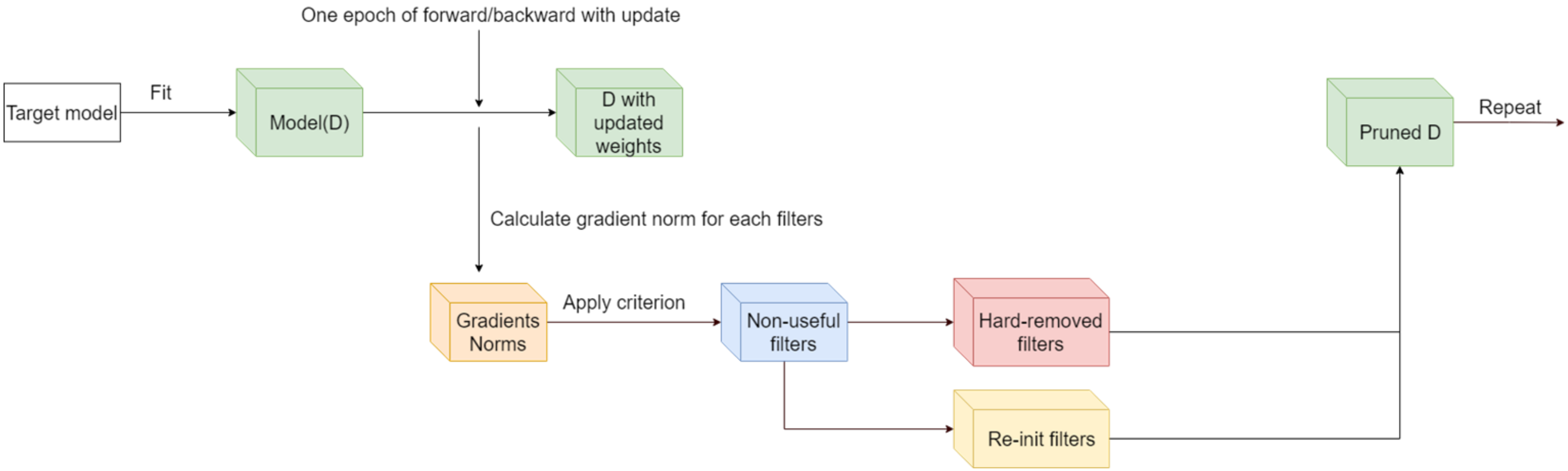


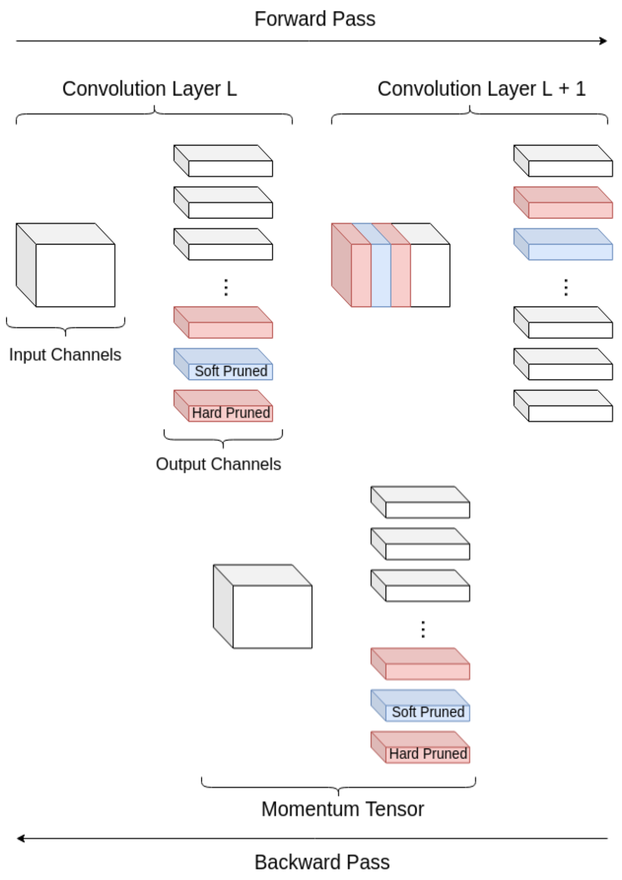
Illustration of the overall working of our method for pruning during training

2) Progressive Gradient Pruning

Forward Backward Pruning:

- Pruning of forward pass follows the same procedure as traditional pruning procedure

- Pruning of momentum tensor for the backward pass use the same indexes as forward pass



2) Progressive Gradient Pruning

Gradient Norm criterion:

- Inspired by Taylor(Molchanov)[2] equation:

$$\left|\Delta\mathcal{L}(\mathbf{H}_i)\right| = \left|\mathcal{L}(\mathcal{D}|\mathbf{H}_i = 0) - \mathcal{L}(\mathcal{D}|\mathbf{H}_i)\right| \approx \left|\frac{\partial\mathcal{L}}{\partial\mathbf{H}_i}\mathbf{H}_i\right|$$

- With L any loss function, D a labeled dataset, H_i the feature map of layer i
- Experimentally, we found that, by changing the formulation to Weight (W_i) instead of feature map yield better result:

$$\mathrm{TW} = \left|\mathcal{L}(\mathcal{D}|\mathbf{W}_i = 0) - \mathcal{L}(\mathcal{D}|\mathbf{W}_i)\right| \approx \left|\frac{\partial\mathcal{L}}{\partial\mathbf{W}_i}\mathbf{W}_i\right|$$

3) Experimental Results

Tasks And Datasets:

- Cifar, Mnist for classification
- PascalVOC for object detection
- Office31 for domain adaptation

Backbones:

- For classification:
  - LeNet and ResNet20 for MNIST
  - VGG19 and ResNet56 for CIFAR
- For object detection:
  - Faster R-CNN with VGG16 Backbone
- For domain adaptation:
  - MMD based domain adaptation with VGG16

3) Experimental Results

Baselines:

- For classification and object detection:
  - L1 pruning
  - Taylor
  - DCP
  - PSFP
- For domain adaptation:
  - TCP

Our Techniques:

- RPGP: Pruning is done at each training iteration
- PGP: Pruning is done at the end of each epoch

3) Experimental Results

Results of our algorithm(red) on MNIST(left) and CIFAR(right) on Resnet

Methods	t_{pruned}	Params	FLOPS	Error % ($\pm$ gap)
Baseline Resnet20	0%	272K	41M	0.74 (0)
L1 [8]	30%	137K	22M	0.75 (+0.01)
	50%	68K	10M	1.09 (+0.35)
	70%	27K	4.2M	2.02 (+1.28)
Taylor [10]	30%	149K	17.7M	0.87 (+0.13)
	50%	87K	7.8M	0.95 (+0.21)
	70%	36K	2.6M	1.04 (+0.30)
DCP [15]	30%	193K	30.3M	1.11 (+0.37)
	50%	138K	21.1M	0.62 (-0.12)
	70%	87.7K	13.5M	1.19 (+0.45)
PSFP [16]	30%	137K	22M	0.5 (-0.24)
	50%	68K	10M	0.61 (-0.13)
	70%	27K	4.2M	0.72 (-0.02)
PGP_GN$_G$ (ours)	30%	137K	22M	0.4 (-0.34)
	50%	68K	10M	0.51 (-0.23)
	70%	27K	4.2M	0.57 (-0.17)
RPGP_GN$_S$ (ours)	30%	137K	22M	0.4 (-0.34)
	50%	68K	10M	0.48 (-0.29)
	70%	27K	4.2M	0.5 (-0.24)

Methods	t_{pruned}	Params	FLOPS	Error % ($\pm$ gap)
Baseline Resnet56	0%	855K	128M	6.02 (0)
L1 [8]	30%	431K	67M	13.34 (+7.32)
	50%	215K	32M	15.57 (+9.55)
	70%	84K	13M	17.89 (+11.87)
Taylor [10]	40%	491K	51M	13.90 (+7.88)
	50%	268K	23M	15.34 (+9.32)
	70%	100k	8M	22.10 (+16.08)
DCP [15]	30%	600K	90M	5.67 (-0.35)
	50%	430K	65M	6.43 (+0.41)
	70%	270K	41M	7.18 (+1.16)
PSFP [16]	30%	431K	67M	8.94 (+2.92)
	50%	215K	32M	10.93 (+4.91)
	70%	84K	13M	14.18 (+8.16)
PGP_GN$_G$ (ours)	30%	431K	67M	8.95 (+2.93)
	50%	215K	32M	10.59 (+4.57)
	70%	84K	13M	13.02 (+7)
RPGP_GN$_S$ (ours))	30%	431K	67M	9.37 (+3.35)
	50%	215K	32M	10.46 (+4.44)
	70%	84K	13M	14.16 (+8.14)

- Our method can outperform baselines in some cases while having competitives results with state-of-the-art

3) Experimental Results

Results of our algorithm(red) for object detection with Faster R-CNN VGG16 on PascalVOC

Methods	No. Params	FLOPS	mAP	Training Time
Baseline VGG16	137M	250G	69.6%	428 min
L1 [8]	125M	174G	62.3%	(428) + 31 min
PSFP [16]	125M	174G	63.5%	428 min
PGP_GN$_G$ (ours)	125M	174G	65.5%	769 min
RPGP_GN$_S$ (ours)	125M	174G	**66.0%**	**281 min**

- Our method outperforms baselines in both accuracy and training time

3) Experimental Results

Results of our algorithm(red) for domain adaptation with VGG16 on Office31

Scenario	Source-only	Baseline VGG	TCP	**RPGP_GN$_S$ (ours)**
Reduction (% FLOPS)	0%	0%	26%	**35%**
$A \rightarrow W$	68.5	74.0	76.1	**78.2**
$A \rightarrow D$	61.1	72.3	76.2	**77.7**
$W \rightarrow A$	41.6	55.2	51.2	**51.6**
$W \rightarrow D$	94.3	97.5	**99.8**	99.4
$D \rightarrow W$	94.5	94.0	96.1	**96.5**
$D \rightarrow A$	50.3	54.1	47.9	**48.0**
Average	68.3	74.5	74.5	**75.2**

- Our method outperforms baselines both in compression and accuracy.

4) Conclusion

- A new progressive pruning method that's suitable for pruning during training
- The proposed approach can work on classification as well as object detection and unsupervised domain adaptation
- It provides faster training and pruning time compared to state-of-the-art

Thank you for listening