Progressive Gradient Pruning for Classification, Detection and Domain Adaptation

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

Le Thanh Nguyen-Meidine [1] Eric Granger [2] Madhu Kiran [1] Louis-Antoine Blais-Morin [2] Marco Pedersoli [1]

[1]ETS Montreal [2]Genetec Inc.

ÉTS Le génie pour l'industrie

Introduction

Convolutional Neural Networks have proven to be able to achieve considerable performance in many task in computer vision. However, CNNs still have a lot of shortcomings and challenges:

- Over-parameterization
- Computationally expensive at training and inference time
- Unsuitable for real-time applications

In this paper we propose a new technique that improves the trade-off between accuracy and complexity by:

- Improve existing progressive pruning technique by combining hard and soft pruning
- Handle the pruning of momentum tensor during back-propagation
- Introduce a criterion that's more suited for progressive pruning

Compression and Acceleration of CNNs

Currently, in literature, we've identified some ways of compressing and accelerating CNNs:

- **Quantization:** By reducing the precision of the weights of network, i.e. Floating point to integer, a network can be accelerated and compressed
- **Low rank decomposition:** : By decomposing a tensor multiplication into operations of lower rank tensor, a CNN can be accelerated
- **Knowledge distillation:** Taking advantage of a teach-student model where the student learns to mimic the output of the teacher, a student can be considerable smaller and faster
- **Compact network design:** An alternative way to have faster deep neural network, light model can be achieved by design, for example MobileNet
- **Pruning:** A family of techniques that removes non-useful parameters from a network. This can be done on individual weights or entire channels at a time

In this paper we are focusing on the pruning channels in deep neural network, we summarized families of these techniques below:

- **Prune Once:** A type of technique that prunes a trained model once and then re-finetune the network
- **Prune Iteratively** : An extension of first type that prune and re-finetune in an iterative way until reaching a goal
- **Prune by minimizing reconstruction error:** A type of technique that prunes by minimizing the difference between the output of the original network and the pruned network
- **Pruning by regularization:** By adding regularization to the original network, a network can be pruned using an optimization process
- **Progressive Pruning:** An extension of the first and the second type, progressive pruning can prune a model from scratch during the training phrase

Progressive Gradient Pruning - Pruning Procedure

An exponential decay function is used to calculate the pruning ratio at each epoch:

$$p_t = \exp\left(\frac{\log(1 - t_{\text{prune}})}{T} t\right), \quad (1)$$

where t_{prune} is a hyper-parameter that defines the ratio of output channels to be pruned, and $t \in \{1, 2, \ldots, T\}$ is the epoch. Since we progressively prune layer by layer and epoch by epoch, we calculate the the number of weak channels or the number of remaining channels at each layer, n_{WC}. Given ratio p_t at epoch t, the number of weak output channels for any layer is defined as:

$$n_{\text{WC}} = n_c(1 - p_t), \quad (2)$$

where n_c can be the original number of output channels of any layers. Using the the number of weak channels n_{WC} we can prune that amount channels at each layer and each epochs.

Progressive Gradient Pruning - Forward Pass and Backward Pass

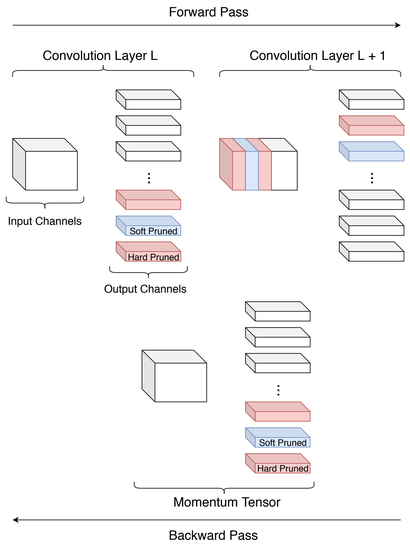

Figure 1: Illustration of the pruning procedure of the proposed PGP approach between two convolutional layers.

Progressive Gradient Pruning - Momentum Pruning for training

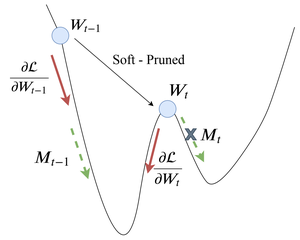

Figure 2: Illustration of the importance of pruning momentum during training

Progressive Gradient Pruning - Pruning Criterion

From the criterion proposed by Molchanov (Taylor-Oracle criterion), which try to approximate the impact of removing one feature map, we derive it to prune directly filter:

$$\text{TW} = \left|\mathcal{L}(\mathcal{D}|\mathbf{W}_i = 0) - \mathcal{L}(\mathcal{D}|\mathbf{W}_i)\right| \approx \left|\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}\mathbf{W}_i\right| \quad (3)$$

Where $\mathbf{W}_i$ is the weight tensor. By taking in account the progressive soft pruning, the criterion can be simplified to:

$$\text{GN}_i = \left|\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}\right| \quad (4)$$

where GN_i is the criterion for our approach for i channel. There are two ways of calculating our criterion:

- **PGP:** perform a training epoch without updating the model, and compute the pruning criterion.
- **RPGP:** compute the pruning criterion directly during a forward/backward pass of the training loop while updating.

The criterion can also be computed in two different ways as well:

- $\mathbf{GN_G}$: Measure the global changes of a network
- $\mathbf{GN_S}$: Measure the gradual changes of a network

Error rates vs number of parameters on CIFAR 10

For this poster, we show result on the dataset of CIFAR10 using the hyper-parameters that were proposed by the authors.

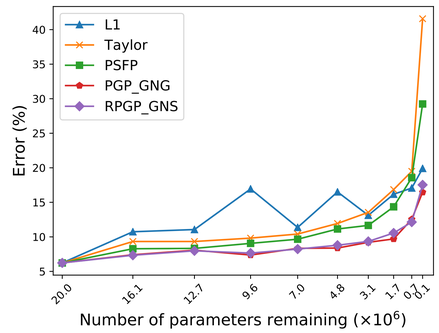

Figure 3: Result of different pruning algorithms on CIFAR10 with VGG19

Other experiments

Pruning on object detection

Table 1: Performance of pruning methods for training Faster R-CNN with VGG16 backbone on the Pascal VOC detection dataset with $t_{\text{pruned}} = 50\%$.

Methods	No. Params	FLOPS	mAP	Training Time
Baseline VGG16	137M	250G	69.6%	428 min
L1 [2]	125M	174G	62.3%	(428) + 31 min
PSFP [1]	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**

Pruning on domain adaptation

Table 2: Performance of our methods for training DAN with VGG16 backbone on Office-31 with $t_{\text{pruned}} = 20\%$.

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

Conclusion

Contributions

- Improved performance on existing progressive pruning techniques
- Allows pruning of supervised or unsupervised tasks
- A pruning criterion suitable for pruning during training
- A faster pruning and training time

References

[1] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, and Yi Yang. Progressive deep neural networks acceleration via soft filter pruning.

[2] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets.