Learning Sparse Filters In Deep Convolutional Neural Networks With A l_1/l_2 Pseudo-Norm

Anthony Berthelier, Yongzhe Yan, Thierry Chateau, Christophe Blanc, Stefan Duffner and Christophe Garcia

11 January 2021

Introduction

- Deep models have achieved state-of-the-art performance in numerous visual tasks such as face recognition, semantic segmentation, object classification and detection, etc. [1][2][3][4][5].

- This high performance requires high computation capabilities and large memory resources.

Introduction

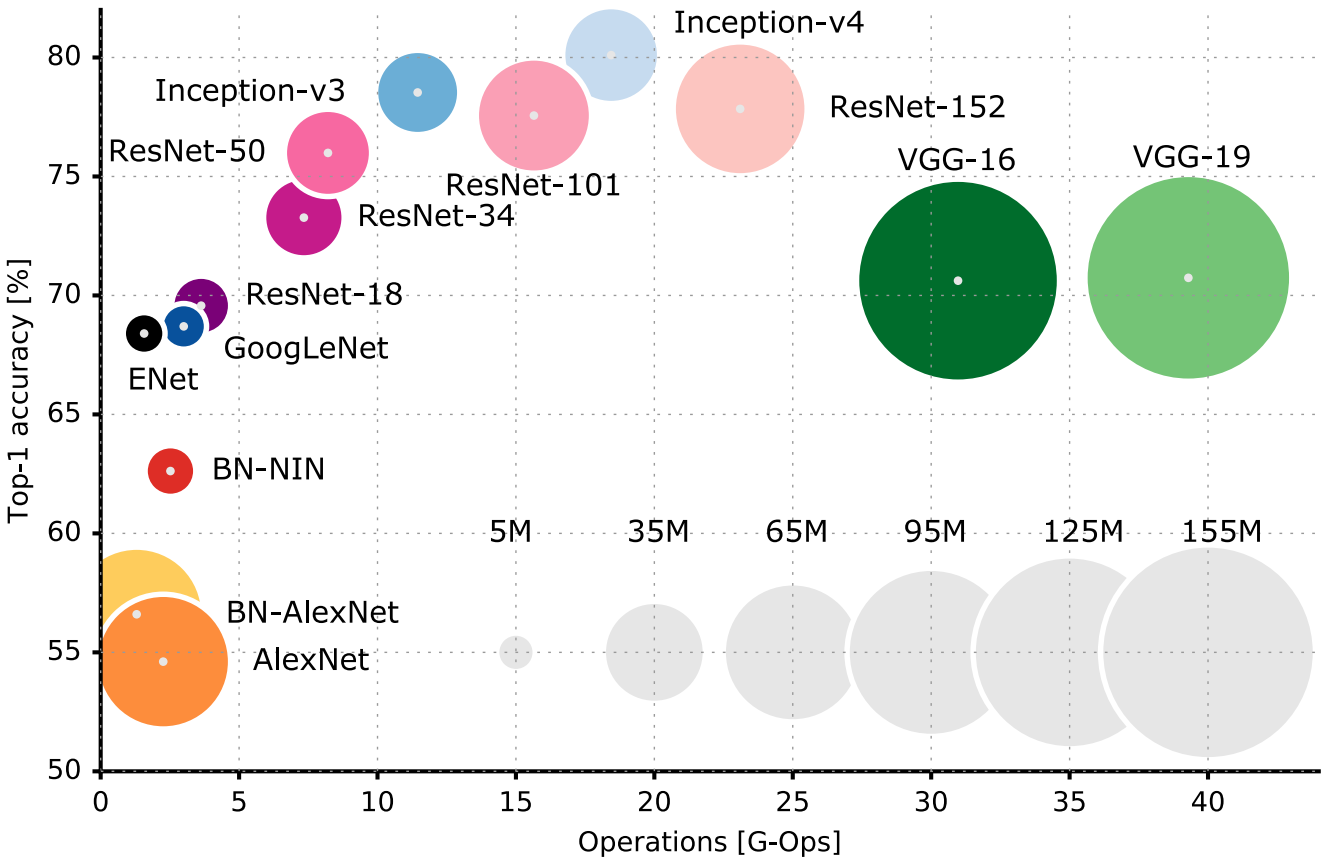

FIGURE – A comparison of the accuracy of famous CNNs against the number of operations needed by these models to classify the ImageNet dataset.

Introduction

- These issues prevent them from running on resource-limited devices such as smartphones or embedded devices.
- Goal : compress deep models.
- Many methods have been developed to obtain compact Deep Neural Networks (Pruning, Weight Sparsifying, Quantization, etc).
- We have developed our own method based on pruning and weight sparsifying.

Motivation

- Redundancy among the weights in a Deep Convolutional Neural Network (DCNN) [6].
- Numerous Sparsity methods in the literature [7].
- These sparsity methods have rarely been used to remove weights during training [8].

Overview

- Removing kernels (filters) in a DCNN using a l_1/l_2-norm.
- Expressing the filter reduction problem by introducing sparsity on a set of pseudo-norms computed on each kernel but not directly on the kernels actual values.

Kernel-sparsity regularization

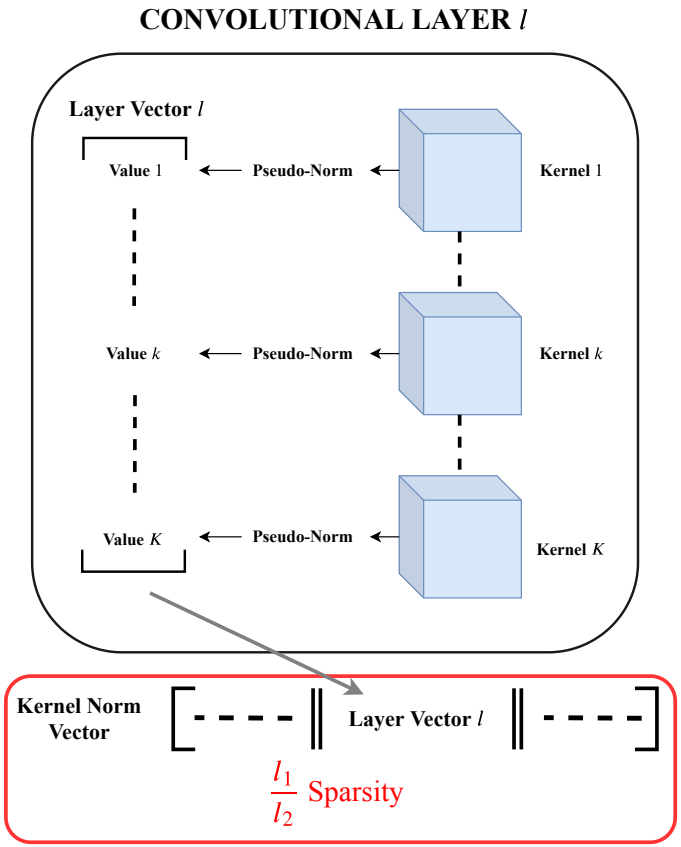

FIGURE – Visual representation of our method and the computation of the kernel norm vector using a pseudo-norm.

Kernel-sparsity regularization

- The global kernel-sparsity is defined by the sparsity of $\vec{N}$ and can be estimated by a l_1/l_2 ratio function :

$$\mathcal{L}_s \doteq \frac{\vec{N}_1}{\vec{N}_2} \, . \tag{1}$$

- The kernel-sparsity regularization term $\mathcal{L}_s$ weighted by the coefficient $\lambda \in \mathbb{R}$ is simply added to loss function $\mathcal{L}_\mathcal{N}$ of the model (e.g. cross entropy) :

$$\mathcal{L}_{all} = \mathcal{L}_\mathcal{N} + \lambda \mathcal{L}_s \, . \tag{2}$$

Training with kernel sparsity regularisation

1. The l_1/l_2 pseudo-norm is computed on each kernel of the model and is integrated to the loss function. This is inducing sparsity at the kernel level during training, pushing some weights to have a near zero value.

2. Sort kernels according to their ascending normalized pseudo-norm. The kernels participating to the cumulative sum under a threshold t are removed.

Experiments

- Tests on *Lenet* and *VGG* on 2 datasets : MNIST and CIFAR-10.
- The models are implemented in Pytorch on various NVIDIA GPUs using CUDA.

Lenet on MNIST

Method	λ	Error	C1 F# (S)	C2 F# (S)	Total Sparsity
Baseline	-	**0.9%**	20	50	**0%**
l_1	0.5	1.2%	4(80%)	5(90%)	87.1%
l_2	0.5	1.2%	3(85%)	5(90%)	88.6%
SSL 1	-	**0.8%**	5(75%)	19(62%)	**65.7%**
SSL 2	-	1.0%	3(85%)	12(76%)	78.6%
NISP	-	**0.8%**	10(50%)	25(50%)	**50.0%**
GAL	-	1.0%	2(90%)	15(70%)	75.7%
l_1/l_2	0.5	**0.7%**	5(75%)	18(64%)	**67.1%**

TABLE – Results after penalizing unimportant filters in *LeNet* on MNIST. Baseline is the simple *LeNet* Caffe model. l_1 and l_2 are the best results found by using the l_1-norm and l_2-norm regularization on the kernels. SSL, NISP and GAL are the pruning methods respectively from [8], [9] and [10]. l_1/l_2 is our method with $\lambda = 0.5$.

Advantages

1. All steps are done during training, i.e. no additional fine-tuning operations are needed.
2. Our method being based on simple l_1 and l_2 norms, is straightforward to implement and compute compared to other methods that remove weights during training.
3. As we are keeping track of the evolution of the network at every step during training, it is possible to choose the best model based on a trade-off between compression and accuracy.

Conclusion

- Compact DCNNs can reach almost the same accuracy as the original models and in some cases even perform better.
- Tests on bigger models and datasets, such as ImageNet are on the way and are showing good results.
- To go beyond, in the future, the method needs to be test on several deeper models. Fully convolutional networks and segmentation problems are also an important focus.

Conclusion

- Thank you for your attention.
- Questions ?

- A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," **NIPS**, pp. 1097–1105, 2012.
- K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale Image recognition," **CoRR**, pp. 1–14, 2015.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," **CVPR**, pp. 1–9, 2015.
- K. He and J. Sun, "Convolutional neural networks at constrained time cost," pp. 5353–5360, 2015.
- K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," **CVPR**, pp. 770–778, 2016.


M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” **BMVC**, 2014.


F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, “Optimization with sparsity-inducing penalties,” **Found. Trends Mach. Learn.**, p. 1–106, 2012.


W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” **NIPS**, p. 2082–2090, 2016.


R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis, “NISP : pruning networks using neuron importance score propagation,” **CVPR**, 2018.


S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. S. Doermann, “Towards optimal structured CNN pruning via generative adversarial learning,” **CVPR**, pp. 2790–2799, 2019.