Differentiable Mask for Pruning Convolutional and Recurrent Networks

Ramchalam Kinattinkara Ramakrishnan*, Eyyüb Sari*, Vahid Partovi Nia

Accelerated Neural Technology (Ant), Huawei Noah's Ark Lab

Abstract

Deep networks require massive computation and such models need to be compressed to bring them on edge devices. Most existing pruning techniques are focused on vision-based models like convolutional networks, while text-based models are still evolving. The emergence of multi-modal multi-task learning calls for a general method that works on vision and text architectures simultaneously. We introduce a *differentiable mask*, that induces sparsity on various granularity to fill this gap. We apply our method successfully to prune weights, filters, subnetwork of a convolutional architecture, as well as nodes of a recurrent network.

Introduction

Recent models on machine translation, self-driving cars, Alpha Go have shown game-changing breakthroughs. However, most of these models are highly over-parametrised for variety of reasons, ranging from the increase of computational power to the lack of domain expertise. Subsequently, deploying these models on constrained edge devices is counter-intuitive. For instance, real time updates to mobile phones could be hampered by the model size. Consequently, the training and inference time are impacted. One alternative is to store deep models on the cloud rather than edge devices to overcome many of the edge implementation drawbacks, and perform computation on the cloud server. However, the cons far outweigh the pros, especially in terms of security, and the latency in transferring the data to and from the cloud. Most of the models are preferred to be stored and computed on the edge in real applications. This goal can be achieved only by simplifying neural networks computations. Various techniques have been used for pruning such as Network Slimming [2], similar to our approach as well as Scalpel [3].

Proposed Method

Let $\{(\mathbf{x}_i, \mathbf{y}_i) \mid i \in N\}$ be a dataset of N samples with $\mathbf{x}_i$ representing the input vector and $\mathbf{y}_i$ the output vector. We consider a model M with L layers, where a layer $l \in L$ represents a prunable entity and its parameters denoted by θ_l. We define a prunable entity as a node in the computational graph that does not invalidate the graph upon its parameters being removed (i.e. the forward pass can still be performed). Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be any element-wise transformation mapping on a node's output (eg. ReLU, Batch Normalization, Identity etc.). Let $J(\Theta)$ be the objective to be minimized,

$$J^* = \min_{\theta \in \Theta} J(\theta),$$

where Θ is a set of all learnable parameters and J^* is the optimized loss.

We introduce *Differentiable Mask Pruning* (DMP) for gradual pruning while training a network. Our method can be generalized to unstructured (i.e. weights) or structured (i.e. vector of parameters, filter, subnetwork) sparsity.

Let $\alpha \in {\mathbb{R}_+}^d$ be a strictly positive scaling factor of dimension d for a given prunable entity, f be a scale sensitive differentiable function (i.e. $f(\alpha \odot \mathbf{X}) \neq f(\mathbf{X})$), $I(\alpha)$ be a mask function where

$$I(\alpha) = \begin{cases} 1 & \text{if } |\alpha| > \mathrm{t}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

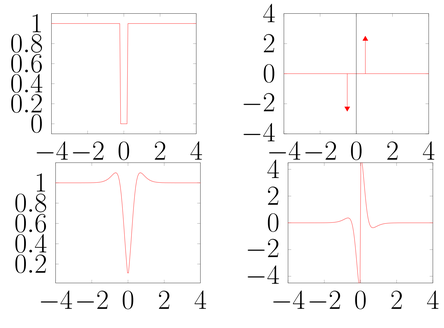

Figure 1: Top: Original mask function (left), derivative (right). Bottom: Approximated mask function (left), derivative (right) [1].

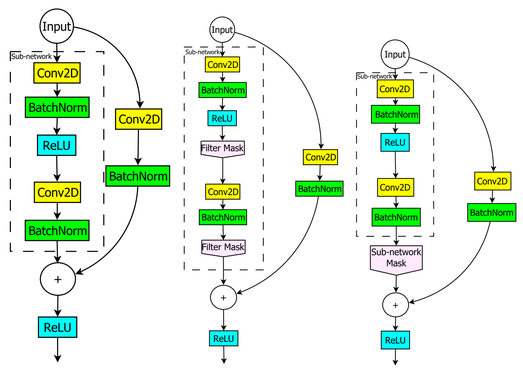

Figure 2: A ResNet style subnetwork (left panel), DMP filter pruning (middle panel), DMP block pruning (right panel).

The intuition behind our approach is to to replace $f(\mathbf{X})$ with $g(\mathbf{X}) = f(\alpha \odot I(\alpha) \odot \mathbf{X})$ and apply ℓ_1-regularization on α to introduce sparsity on its corresponding prunable entity. Formally we propose to replace $J(\theta)$ with

$$J(\theta, \alpha) = C(\theta, \alpha) + \mathcal{R}(\theta) + \lambda \sum_{l \in L} ||\alpha_l||_1, \tag{2}$$

in which $\mathcal{R}(\theta)$ is regularizer, often an ℓ_2 norm on weights.

Results

Method	λ_1 $\times 10^{-3}$	Test Error	Pruned subnets out of 27	Pruned filters out of 2032	Ratio $\times 100$
Unpruned	-	6.53	0	0	0
DMP	1	7.10	5	320	15.70
DMP	5	7.78	12	928	45.60
DMP	10	8.34	17	1152	56.60

Table 1: ResNet-56 run on CIFAR-10 to prune subnetworks with different regularization constant λ_1.

Method	λ_1 $\times 10^{-4}$	Test Error	Pruned filters out of 2032	Ratio $\times 100$
Unpruned	-	6.53	0	0
DMP	1	6.81	264	12
DMP	5	8.48	1227	60
DMP	10	9.50	1599	78

Table 2: ResNet-56 architecture run on CIFAR-10, to prune filters with different regularization constant λ_1.

Method	λ_1 $\times 10^{-7}$	Test perplexity	Pruned nodes out of 5200	Ratio $\times 100$
Unpruned	-	84.9	0	0
DMP	1	85.1	524	10
DMP	5	85.02	672	12
DMP	10	86.1	940	18

Table 3: Penn Tree Bank Dataset Language model using recurrent networks to prune LSTM nodes. (1 LSTM layer)

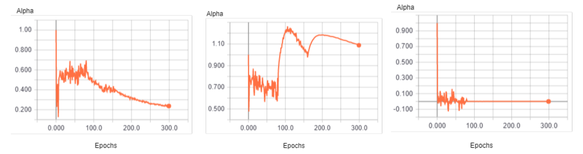

Figure 3: The training process of 3 scaling factors for subnetwork pruning. As one scaling factor gets pruned, another scaling factor compensates for the instability caused and increases in magnitude. The rightmost figure also shows the scaling factor rejuvenating throughout the training process and finally pruned at epoch 80.

Conclusion

We introduced DMP, a new technique that extends pruning on two directions: structured and unstructured. DMP induces sparsity that can be easily extended to prune weights, nodes, vectors, filters and sub-networks. The main shortcoming of pruning is to train the network properly with fewer parameters. We proposed to improve the training procedure by approximating the hard threshold gradient, and updating back-propagation accordingly. We demonstrated the versatility of DMP by easily integrating it into CV and NLP architectures. If pruning entity is a sub-network, DMP can be regarded as a differentiable architecture search method, while spanning always on architectures with lower complexity.

References

[1] Mouloud Belbahri, Eyyüb Sari, Sajad Darabi, and Vahid Partovi Nia. Foothill: A quasiconvex regularization for edge computing of deep neural networks. pages 3–14, 2019.

[2] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763, 2017.

[3] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. Scalpel: Customizing dnn pruning to the underlying hardware parallelism. *SIGARCH Comput. Archit. News*, 45(2):548–560, June 2017.