Dynamic Network Surgery for Efficient DNNs
06/02/2022

1 Authors, Conference/Journal, Year, Num Citations

- Authors: Yiwen Guo, Anbang Yao, Yurong Chen (Intel Labs, China)
- Conference: NeurIPS
- Year: 2016
- Num Citations: 776

2 Problem Statement

- Deep neural network (DNN) algorithms are very complex, which make them difficult to deploy on platforms with limited computational, memory and battery power (such as mobile phones). Authors proposes a network compression algorithm which reduces the DNNs complexity without significant decrease in performance measure (here they have used accuracy measure).

3 Contribution

- A network compression method called Dynamic Network Surgery (DNS) is proposed, which reduces the complexity by pruning the layers' node to node connections in DNNs.
- If an important connection has been pruned then the method re-forms the pruned connection.

4 How is the work different from related works?

- [6] improve the speed of CNNs by performing convolutions calculation in frequency domain.
- [2], [3], [5] are based on the idea of matrix decomposition.
- [1] compresses network by grouping its parameters into hash buckets.
- [4] is based on network pruning idea. Their work is based on this paper. Issues:

– Irretrievable Network: This work is also based on network pruning. Once the connection is pruned, there is no chance of recovery. As a consequence, incorrect pruning may result in severe drop in accuracy.

– Learning Inefficiency: Several iterations of alternate pruning and re-training are necessary to achieve compression. On each iteration training DNNs with millions of parameters is time consuming.

5 Main Idea

- There are redundancies in the DNN parameters [2], therefore with a proper strategy it is possible to compress the model without significant reduction in prediction accuracies.

- Authors note that a network connection may be redundant due to the existence of other connections, but may become important if some other connections are removed. Therefore, they proposes to continually maintain network structure and perform two key operations:

  – Pruning: Removing connections in the DNN.

  – Splicing: Connection recovery if the pruned connections are found to be important.

- Notations:

  – Suppose DNN parameters can be represented by $\{W_k : 0 \leq k \leq C\}$, where W_k denotes a matrix with weights in the kth layer. Consider a fully connected layer with p-dimensional input and q-dimensional ouput, the size of W_k matrix will be $p \times q$. Similarly for CNNs with learn-able kernels, unfold the kernel into a vector and concatenate all of them to form W_k matrix.

  – To represent connection pruning, authors use $\{W_k, T_k : 0 \leq k \leq C\}$ matrices, where W_k matrix is defined as above and each T_k is a binary matrix with entries indicating the state of the network connections, i.e. whether the connections are pruned or not.

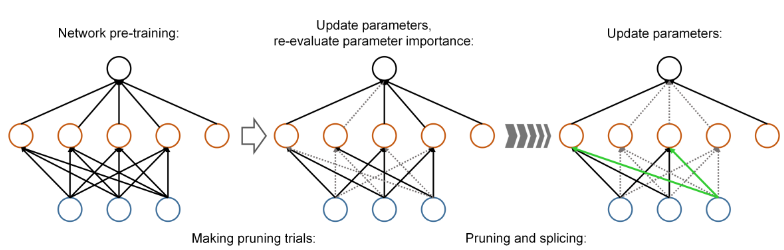

Figure 2: Overview of the dynamic network surgery for a model with parameter redundancy.

- Algorithm formulation:

– Consider the optimization problem for kth layer:
$\min_{W_k, T_k} L(W_k \odot T_k)$ s.t $T_k^{(i,j)} = h_k(W_k^{(i,j)}), \forall(i,j)$,
where $\odot$ denotes hadamard product operator and $h_k(.)$ is a discriminative function, which satisfy $h_k(w) = 1$ if parameter w is important in the kth layer else 0.

– Discriminative function $h_k(W_k^{(i,j)})$ is defined as:

$$\begin{aligned} h_k(W_k^{(i,j)}) &= 0, \text{ if } a_k > |W_k^{(i,j)}| \\ &= T_k^{(i,j)}, \text{ if } a_k \leq |W_k^{(i,j)}| < b_k \\ &= 1, \text{ if } b_k < |W_k^{(i,j)}| \end{aligned}$$

– The above optimization problem is solved by alternately updating W_k and T_k through stochastic gradient descent method. Weight update described as below :
$W_k^{(i,j)} \leftarrow W_k^{(i,j)} - \beta \dfrac{\partial L(W_k \odot T_k)}{\partial (W_k^{(i,j)} T_k^{(i,j)})}, \forall(i,j)$

6 Evaluation

- Comparison on MNIST dataset with LeNet variants, with the then state of the art method [4]. Prediction accuracies are very close to the non-compressed version of LeNets. Here [9] refers to [4].

Model	Layer	Params.	Params.% [9]	Params.% (Ours)
LeNet-5	conv1	0.5K	$\sim$ 66%	14.2%
	conv2	25K	$\sim$ 12%	3.1%
	fc1	400K	$\sim$ 8%	0.7%
	fc2	5K	$\sim$ 19%	4.3%
	Total	431K	$\sim$ 8%	**0.9%**
LeNet-300-100	fc1	236K	$\sim$ 8%	1.8%
	fc2	30K	$\sim$ 9%	1.8%
	fc3	1K	$\sim$ 26%	5.5%
	Total	267K	$\sim$ 8%	**1.8%**

- Comparison on ImageNet with AlexNet, with the then state of the art method [4].

Model	Top-1 error	Top-5 error	Epochs	Compression
Fastfood 32 (AD) [21]	41.93%	-	-	2×
Fastfood 16 (AD) [21]	42.90%	-	-	3.7×
Naive Cut [9]	57.18%	23.23%	0	4.4×
Han et al. [9]	42.77%	19.67%	$\geq$ 960	9×
Dynamic network surgery (Ours)	43.09%	19.99%	$\sim$ 140	**17.7×**

- Overall performance of DNS method

model	Top-1 error	Parameters	Iterations	Compression
LeNet-5 reference	0.91%	431K	10K	
LeNet-5 pruned	0.91%	4.0K	16K	**108**×
LeNet-300-100 reference	2.28%	267K	10K	
LeNet-300-100 pruned	1.99%	4.8K	25K	**56**×
AlexNet reference	43.42%	61M	450K	
AlexNet pruned	43.09%	3.45M	700K	**17.7**×

7 Conclusion

- Current mobile phone comes with both CPU and GPU installed. How can we use mobile GPU efficiently?

References

[1] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. *ICML*, abs/1504.04788, 2015.

[2] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. *NeurIPS*, abs/1306.0543, 2013.

[3] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *NeurIPS*, abs/1404.0736, 2014.

[4] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *NeurIPS*, abs/1506.02626, 2015.

[5] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *ICLR*, 2015.

[6] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. arXiv:1312.5851, 2013.