Dynamic parameter reallocation improves trainability of deep convolutional networks

Hesham Mostafa
Artificial Intelligence Products Group
Intel Corporation
hesham.mostafa@intel.com

Xin Wang
Artificial Intelligence Products Group
Intel Corporation
xin3.wang@intel.com

Abstract

Network pruning has emerged as a powerful technique for reducing the size of deep neural networks. Pruning uncovers high-performance subnetworks by taking a trained dense network and gradually removing unimportant connections. Recently, alternative techniques have emerged for training sparse networks directly without having to train a large dense model beforehand, thereby achieving small memory footprints during both training and inference. These techniques are based on dynamic reallocation of non-zero parameters during training. Thus, they are in effect executing a training-time search for the optimal subnetwork. We investigate a most recent one of these techniques and conduct additional experiments to elucidate its behavior in training sparse deep convolutional networks. Dynamic parameter reallocation converges early during training to a highly trainable subnetwork. We show that neither the structure, nor the initialization of the discovered high-performance subnetwork is sufficient to explain its good performance. Rather, it is the dynamics of parameter reallocation that are responsible for successful learning. Dynamic parameter reallocation thus improves the trainability of deep convolutional networks, playing a similar role as overparameterization, without incurring the memory and computational cost of the latter.

1 Introduction

Training high-performance compact networks is often a two-step process. A large network is first trained, then compressed using techniques such as pruning, distillation, or low-rank decomposition. Training a compact network from scratch typically fails to reach the same level of accuracy achieved by compressing a larger network [Zhu and Gupta, 2017].

Network pruning is a common compression method that yields a high-performance subnetwork of an original network. A natural question is whether such high-performance subnetworks can be uncovered by a direct search over the space of subnetworks, without the two-step process of training a large network first and then pruning it down. The advantage of such a search-based procedure is that the full dense model need not be trained; instead, we start with an initial subnetwork and continuously modify it during training until we find a high-performance subnetwork. [Anonymous, 2019] described such a search procedure. Their scheme starts with a sparse network and continuously reallocates during training its non-zero parameters throughout the network based on a simple heuristic. The resulting subnetworks perform on par with, and often better than, subnetworks obtained by iteratively pruning a large overparameterized model. This calls into question the belief that overparameterization is essential to successful learning. Here, we argue that dynamic parameter reallocation (DPR) is an equally effective approach to overparameterization to improving the trainability of deep convolutional networks (CNNs). We present results for a wide Resnet WRN-28-2 [Zagoruyko and Komodakis,

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2016] trained on CIFAR10 and show that DPR is needed for effective learning, even when the structure and initialization of the high-performance subnetwork is given *a priori*.

2 Related work

Several recently proposed methods used DPR for direct training of sparse networks: Bellec et al. [2017] used a random walk in parameter space to explore different sparse parameterizations; Mocanu et al. [2018] used alternating steps of magnitude-based pruning and random growth to search for efficient sparse networks. Anonymous [2019] extended the method in Mocanu et al. [2018] by allowing parameters to be moved across layers. It outperformed the previous two methods when training deep CNNs and it is the method we use here to investigate how DPR trains high-performance sparse networks. Neural architecture search (NAS) techniques [Elsken et al., 2018] bear some similarities to DPR methods in the sense that they also search for compact high-performance networks, but not simultaneous to training. A closely-related NAS technique is one-shot architecture search [Bender et al., 2018] where a large network is trained and then the performance of different subnetworks are evaluated to find the one with highest performance.

3 Methods

Algorithm 1 is a high-level summary of the DPR mechanism introduced by Anonymous [2019]. Training starts with a compact network with sparse parameter tensors $\{\mathbf{W}_i\}$. During training, Algorithm 1 is invoked once every few hundred training iterations. Each invocation reallocates parameters throughout the network through a two-step procedure: weights of absolute values under an adaptive threshold are pruned, followed by an immediate redistribution of an equal number of weights across parameter tensors, resulting in a new subnetwork with exactly the same number of parameters as the original (see Anonymous [2019] for details of the algorithm).

Algorithm 1: Reallocate non-zero parameters within and across parameter tensors

```
for each sparse parameter tensor $\mathbf{W}_i$ do
    $(\mathbf{W}_i, k_i) \leftarrow$ prune_by_threshold($\mathbf{W}_i, H$)          ▷ $k_i$ is the number of pruned weights
    $l_i \leftarrow$ number_of_nonzero_entries($\mathbf{W}_i$)   ▷ Number of surviving weights after pruning
end for
$(K, L) \leftarrow (\sum_i k_i, \sum_i l_i)$                      ▷ Total number of pruned and surviving weights
$H \leftarrow$ adjust_pruning_threshold($H, K, \delta$)                        ▷ Adjust pruning threshold
for each sparse parameter tensor $\mathbf{W}_i$ do
    $\mathbf{W}_i \leftarrow$ grow_back($\mathbf{W}_i, \frac{l_i}{L}K$)        ▷ Grow $\frac{l_i}{L}K$ zero-initialized weights at random in $\mathbf{W}_i$
end for
```

4 Results

We present results of WRN-28-2 [Zagoruyko and Komodakis, 2016] trained on CIFAR10, with standard data augmentation, training and DPR hyperparameters adapted from Anonymous [2019]. We trained WRN-28-2 for 200 epochs at two levels of global sparsity: 0.9 and 0.8. First, DPR was actively invoked during the entire course of training and continuously changed the subnetwork structure by reallocating parameters according to Algorithm 1, yielding sparse networks with high generalization performance (yellow bars in Figure 1a).

To investigate whether the structure of the subnetwork discovered by DPR contributed to its trainability, we did the following experiment: after training with DPR, the structure (i.e. positions of non-zero entries in sparse parameter tensors) of the final subnetwork was retained, and this subnetwork was randomly re-initialized and re-trained without DPR (green bars in Figure 1a). Even though the subnetwork has the same structure as the final subnetwork found by DPR, its training failed to reach the same accuracy as the identical subnetwork discovered by active DPR.

One might argue that it is not just the network structure, but also its initialization that allow it to reach high accuracies. To assess this argument, we used the final subnetwork structure found by DPR as described above, and initialized it with the same initial values used in the DPR training instance. As

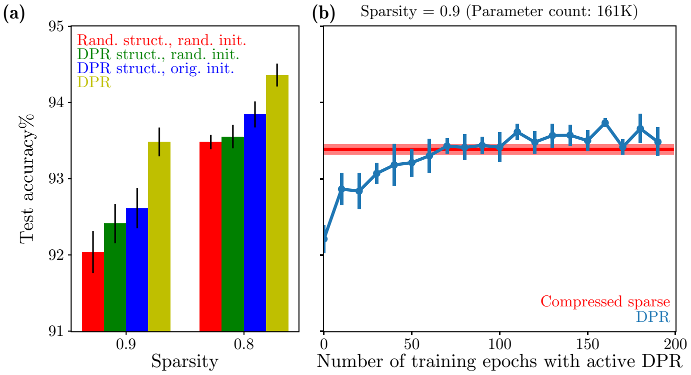

Figure 1: Test accuracies of sparse WRN-28-2 trained on CIFAR10. All plotted data show mean and standard deviation of 5 independent trials. (a) Comparison of DPR against a number of related statically trained networks. (b,c) Results of experiments on active initial DPR turned off at certain stages of training. For all data points, we ran training for 200 epochs (independently of when DPR was stopped).

shown in Figure 1a (blue bars), the combination of final structure and original initialization still fell significantly short of the level of accuracy achieved by DPR training, though it performed better than training the same subnetwork with random initialization (green bars).

Finally, we ran control trials where the subnetwork and its initialization were both random (red bars in Figure 1a) and not surprisingly, these subnetworks performed the worst.

Further, we asked at what stage of training DPR discovered a highly trainable subnetwork. We ran training trials where, in each trial, we stopped DPR after a certain number of epochs (Figures 1b & 1c). The horizontal red band marks the baseline performance of sparse compression of a fully trained dense model iteratively pruned and retrained (see Zhu and Gupta [2017], Anonymous [2019]), and the blue symbols represent DPR training up to a certain stage of training. Interestingly, DPR discovered effectively trainable network structures at relatively early stages of training.

5 Discussion

The need for overparameterization during learning has often been attributed to the reduced likelihood of stochastic gradient descent (SGD) being trapped in bad local optima as the dimensionality of the loss surface (number of parameters) increases. An alternative hypothesis, “the lottery ticket hypothesis" [Frankle and Carbin, 2018], argues that starting with large, overparameterized networks simply provides more candidate subnetworks, making it more likely that one of these candidates becomes a “winning lottery ticket", i.e, having the right structure and initialization needed to learn the task. Our results did not support this hypothesis. We showed that structure and initialization alone or in combination were not sufficient to train compact sparse CNNs to high performance. Rather, successful learning seemed to depend on the dynamics and the extra degrees of freedom provided by DPR. Note that our results are not at odds with those in Frankle and Carbin [2018], which reported negative results on finding “winning tickets" in deep residual networks.

DPR seems to play an analogous role to overparameterization when it comes to improving network trainability. Like overparameterization, DPR allows training to explore more degrees of freedom than those strictly necessary to solve the task. Unlike overparameterization where these degrees of freedom are extra parameters, DPR introduces extra degrees of freedom by simultaneous exploration of different subnetwork structures during training. In terms of computational and memory resources, DPR is a more attractive method than overparameterization in improving network trainability because it obviates the need to maintain or operate on large models, and requires a smaller memory footprint that is the same during training and inference.

References

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. 2017. URL `http://arxiv.org/abs/1710.01878`.

Anonymous. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Submitted to International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1xBioR5KX`. under review.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. may 2016. URL `http://arxiv.org/abs/1605.07146`.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, dec 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-04316-3. URL `http://www.nature.com/articles/s41467-018-04316-3`.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep Rewiring: Training very sparse deep networks. nov 2017. URL `http://arxiv.org/abs/1711.05136`.

Thomas Elsken, J.H. Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Small, Trainable Neural Networks. mar 2018. URL `http://arxiv.org/abs/1803.03635`.