Convolutional network pruning with matrix factorization

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Introduction

Properties of convolutional networks (layers...)... convolutional layers take time, fully connected layers take space (importantly in test time). Pruning of convolutional network. Problems that can be solved with model: generalization, time and storage reduction, better interpretation. Introduction to our model, same-time matrix trifactorization on weights (on covolutional (time) and fully connected (size) layers separately). From almost keeping the performance of convolutional network to improvement of generalization.

2 Related work

For a typical convolutional neural network, about 90% of the model size is taken up by the dense connected layers and more than 90% of the running time is taken by the convolutional layers [10]. Running time is depended from the computation which is dominated by the convolution operations in the lower layers of the model. There is significant redundancy in the parametrization of most of deep learning models. Compressing the parameters to reduce model size brings the focus upon how to compress the dense connected layers since the vast majority of weights reside in these layers which results in significant savings.

In comparison to model size reduction, a smaller number of literature on compressing the convolutional layers to improve running time was found. In article [4], they exploited the redundancy with linear compression techniques, resulting in significant speedups for the evaluation of trained large scale networks, with minimal compromise to performance. They did it by compressing each convolutional layer by finding an appropriate low-rank approximation, and then fine-tune the upper layers until the prediction performance is restored.

Compressing the most storage demanding dense connected layers is dominated by matrix factorization methods [CITE THEM]. In [5], they presented vector quantization methods for which they said have a clear gain over existing matrix factorization methods and compressing the convolutional layers with vector quantization methods for running time reduction can also be applied. The effort at reducing the model size while keeping the accuracy improvements applied with singular value decomposition (SVD) on the weight matrices in deep neural network was presented in article [8]. The reconstruction of the model reduced the model size significantly with negligible accuracy loss. In both solutions, fine-tuning to compressed layers improves performance.

Hashing is also an effective strategy for dimensionality reduction while preserving generalization performance [7, 6]. The strategy used on neural networks named HashedNets [1] uses a low-cost hash function to randomly group connection weights into hash buckets where all connection inside

share a single and tuned parameter value. The big problem in this approach is that the optimization of the reduced set of vectors is rather nontrivial.

Another solution in model size reduction and preserving the generalization ability is to train models that have a constant number of simpler neurons, presented by [2], or whereas in [9] they replaced the fully connected layers of the network with an Adaptive Fastfood transform, resulting in a deep fried convnet. The Fastfood transform allows for a theoretical reduction in computation. However, the computation in convolutional neural networks is dominated by the convolutions, and hence the deep fried convnets are not necessarily faster in practice.

In article [3] they said that giving only a few weight values for each feature it is possible to accurately predict the remaining values while many of them dont need to be learned at all. They exploit the fact that the weights in learned networks tend to be structured.

3 Method description

Collecting the weight matrices from convolutional layers and from fully connected layers, perform pruning (description of pruning with same-time matrix trifactorization), setting the pruned weights to zero, tuning the parameters with iterations after pruning.

4 Experiments

Description and preparation of datasets, parameters, results, analysis. Datasets: mnist, imageNet, CIFAR Convnets: classical, alexnet, deep fried nets?

5 Discussion and conclusion

Acknowledgments

References

References

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

[2] Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

[3] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

[4] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[5] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.

[6] Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. Hash kernels for structured data. *The Journal of Machine Learning Research*, 10:2615–2637, 2009.

[7] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

[8] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *INTERSPEECH*, pages 2365–2369, 2013.

[9] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. *arXiv preprint arXiv:1412.7149*, 2014.

[10] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.