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

Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation [4] [TIME AND STORAGE REDUCTION, time: on convolutional layers, storage: on fully connected layers, compromise to accuracy, might improve generalization too]

The computation is dominated by the convolution operations in the lower layers of the model. Redundancy can be exploited with linear compression techniques, resulting in significant speedups for the evaluation of trained large scale networks, with minimal compromise to performance. Compressing each convolutional layer by finding an appropriate low-rank approximation, and then fine-tune the upper layers until the prediction performance is restored. Since the vast majority of weights reside in the fully connected layers, compressing only these layers translate into a significant savings. The low-rank projections effectively decrease number of learnable parameters, suggesting that they might improve generalization ability.

Network In Network [7] [better interpretation, overfitting reduction]

The conventional convolutional layer uses linear filters followed by a nonlinear activation function to scan the input. Instead, we build micro neural networks with more complex structures to abstract the data within the receptive field. With enhanced local modeling via the micro network, we are able to utilize global average pooling over feature maps in the classification layer, which is easier to interpret and less prone to overfitting than traditional fully connected layers. In traditional CNN, it is difficult to interpret how the category level information from the objective cost layer is passed back to the previous convolution layer due to the fully connected layers which act as a black box in between. In contrast, global average pooling is more meaningful and interpretable as it enforces correspondance between feature maps and categories, which is made possible by a stronger local modeling using the micro network. Furthermore, the fully connected layers are prone to overfitting and heavily depend on dropout regularization, while global average pooling is itself a structural regularizer, which natively prevents overfitting for the overall structure.

Compressing deep convolutional networks using vector quantization [5] [storage reduction, possible performance improvement, possible to apply compression on convolutional layers for time reduction]

Compressing the most storage demanding dense connected layers, vector quantization methods have a clear gain over existing matrix factorization methods. Compressing the parameters to reduce storage instead of speeding up the testing time (Denton et al., 2014 [Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation] [4]). For a typical network described in (Zeiler and Fergus, 2013 [Visualizing and understanding convolutional neural networks] [14]), about 90% of the storage is taken up by the dense connected layers; more than 90% of the running time is taken by the convolutional layers. Therefore, we shall focus upon how to compress the dense connected layers to reduce storage of neural networks. Simply applying kmeans-based scalar quantization achieves very impressive results. It will also be interesting to apply fine-tuning to the compressed layers to improve performance. Whereas this paper mainly focused on compressing dense connected layers, it will be interesting to investigate if we can apply the same vector quantization methods to compress convolutional layers.

OBD, OBS, MF pruning

Dropout

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [9] [generalization improvement, minus: increse off time]

The key idea is to randomly drop units (along with their connections) from the neural network during training. It prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently. The term dropout refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections. One of the drawbacks of dropout is that it increases training time.

Regularization of Neural Networks using DropConnect [10] [like dropconnect, slower but better generalization]

DropConnect, a generalization of Dropout (Hinton et al., 2012), for regularizing large fully-connected layers within neural networks. DropConnect instead sets a randomly selected subset of weights within the network to zero. Each unit thus receives input from a random subset of units in the previous layer. Current implementation of DropConnect is slightly slower than No-Drop or Dropout, in large models models the feature extractor is the bottleneck, thus there is little difference in overall training time. DropConnect allows us to train large models while avoiding overfitting.

HashedNets [storage reduction, preserving of generalization]

Feature Hashing for Large Scale Multitask Learning [11], Hash Kernels for Structured Data [8], Compressing Neural Networks with the Hashing Trick [1]

Hashing is an effective strategy for dimensionality reduction and practical nonparametric estimation. Kernel methods have slow runtime performance if the number of kernel functions used in the expansion is large. Very good generalization performance is reported. The big problem in this approach is that the optimization of the reduced set of vectors is rather nontrivial. HashedNets uses a low-cost hash function to randomly group connection weights into hash buckets, and all connections within the same hash bucket share a single parameter value. These parameters are tuned to adjust to the HashedNets weight sharing architecture with standard backprop during training. HashedNets shrink the storage requirements of neural networks substantially while mostly preserving generalization performance.

vanishing gradient problem 26 other models

AlexNet 29 Memory Bounded Deep Convolutional Networks [increase of generalization [2], (parameter) storage reduction]

Rather than having a smaller number of complex units, we try to train models that have a constant number of simpler units. This may be a key tool in training deep learning vision models that are meant to be deployed in resource-constrained environments, or to increase the generalizability of existing models. Using sparsity-inducing regularization provides a significantly improved method

for reducing the parameters of a model as compared with baselines including reducing the number of units in the network and simple thresholding. An interesting empirical observation we see is that very sparse models manage to do surprisingly well on benchmark image classification tasks. We leverage this not only to construct very simple models, but also to build ensembles of these sparse models that outperform the baseline dense models while still staying within fixed resources.

Deep fried convnets 28

Deep Fried Convnets [13] [(parameter) storage reduction, without sacrificing performace, in theory faster, but because computation dominated by the convolutions, not visible in practice]

Adaptive Fastfoodtransform to reparameterize the matrix-vector multiplication of fully connected layers. The number of parameters required to represent a deep convolutional neural network can be substantially reduced without sacrificing predictive performance. Our approach works by replacing the fully connected layers of the network with an Adaptive Fastfood transform, which is a generalization of the Fastfood transform for approximating kernels. At test and also at train time. The Fastfood transform allows for a theoretical reduction in computation from O(nd) to O(n log d). However, the computation in convolutional neural networks is dominated by the convolutions, and hence deep fried convnets are not necessarily faster in practice.

tensor train TT NN rank 5,18,22

Predicting Parameters in Deep Learning [3] [parameter (storage) reduction with prediction]

There is significant redundancy in the parametrization of several deep learning models. Given only a few weight values for each feature it is possible to accurately predict the remaining values. Moreover, we show that not only can the parameter values be predicted, but many of them need not be learned at all. We study techniques for reducing the number of free parameters in neural networks by exploiting the fact that the weights in learned networks tend to be structured. Brings into question whether we have the right parameterizations in deep learning.

Maxout Networks [6] [with dropout, better generalization]

designed to both facilitate optimization by dropout and improve the accuracy of dropouts fast approximate model averaging technique. uses a new type of activation function: the maxout unit. In a convolutional network, a maxout feature map can be constructed by taking the maximum across k affine feature maps (i.e., pool across channels, in addition spatial locations). Maxout networks learn not just the relationship between hidden units, but also the activation function of each hidden unit.

hash 3 lower numerical precision 1,9, fewer parameters 7

Restructuring of Deep Neural Network Acoustic Models with Singular Value Decomposition [12] [storage (size) reduction, keeping the performance]

effort on DNN aiming at reducing the model size while keeping the accuracy improvements. We apply singular value decomposition (SVD) on the weight matrices in DNN, and then restructure the model based on the inherent sparseness of the original matrices. After restructuring we can reduce the DNN model size significantly with negligible accuracy loss. We also fine-tune the restructured model using the regular back-propagation method to get the accuracy back when reducing the DNN model size heavily.

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] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[7] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[8] 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.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[10] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[11] 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.

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

[13] 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.

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