Automated search for optimal convolutional neural network factorization

Frank Mokadem
Vision and Image Processing Lab, University of Waterloo
Alex Wong
Vision and Image Processing Lab, University of Waterloo
Email: {fmokadem, alexander.wong}@uwaterloo.ca

Abstract

Deep Neural networks (DNNs) are the state of the art technique when it comes to artificial intelligence tasks relating to computer vision. Usage of DNNs is wide spread across multiple industries, and entertainment. Most notably is the use of Convolutional Neural Networks(CNNs) architectures for object detection and classification, and even more recently information retrieval. However, one downfall of CNNs is their computational cost, even on trivial tasks. The reason for such high computational cost lies in the high flop number of kernel convolution, the core operation which is built upon a CNN. Hence presenting the need for compression of floating number information in CNNs. In this work we explore the techniques of CNN compression using tensor decompositions. Furtheremore, we aspire to build an automated tool to execute a grid search through the space of all possible factorizations of a CNN and pick an optimal compressed network representation with respect to performance requirements.

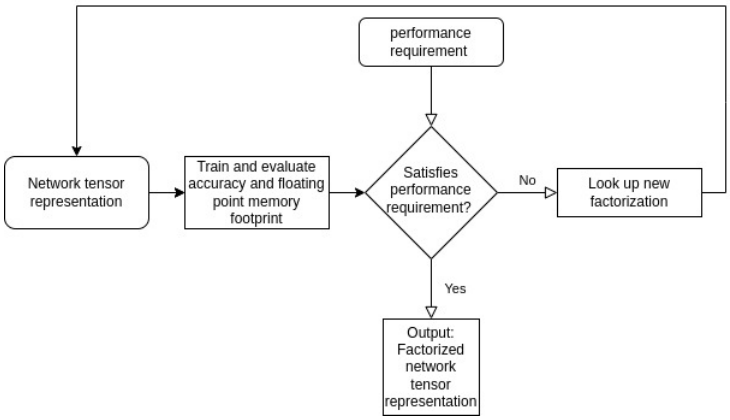

Fig. 1: Flow chart of automated grid search tool.

1 Introduction

In Deep learning applications for computer vision, the state of the art is to use convolutional layers. Convolutional layers are kernels (weighted sum on regions of the image) that are applied on input images. These layers are mathematically equivalent to linear maps (convolutional layer is linear, non-linearity is introduced elsewhere on Neural Networks) between the input image and the output image. The convolution of an input image via a kernel is a multilinear map. Indeed, because this is a point wise operation, i.e. operating on each pixel at a time, and is a weighted sum of the image pixels. It is natural then to represent these operations using the mathematical object: tensor. Similar to a matrix, in a given basis a tensor is a tabular description of a multilinear relationship represented with an n-dimensional array. Also similar to the matrix case, there exists decompositions that reduce a tensor into a lower order (i.e. lower sizes in each of the n dimensions). We are interested in the Tucker decomposition (TD) that decomposes a tensor into a core tensor and n matrices. The final result is given in (1).

$$X = G \times_1 U(1) \times_2 U(2) \times_3 ... \times_N U(N) \quad (1)$$

Where X is the original tensor. G is the core tensor, and U(1),...,U(N) are the matrices. $\times_i$ denotes the mode product of a tensor by a matrix along its i'th dimension.

2 Automated grid search

In this work and following advancements we are building an automated grid-search tool to systematically explore the space of possible factorization configurations of a CNN while maintaining a set of performance requirements imposed by the the user.

1 illustrates the flow of operations in the automated grid search tool. A user provides a trained network, i.e. architecture and parameters, represented in tensor format and specifies a set of performance requirements. Mainly, a max cut-off for floating number memory footprint and a min cut-off in loss of network accuracy. The user will also provide the train and test data used to train the network, and evaluate its performance.
The automated grid search tool will generate a search space of possible tensor factorizations that respond to the memory cost requirement and subsequently build, train and test a sequence of factorized neural networks evaluating every one if it satisfies the user performance requirements. Finally, should the tool find a successful factorized network, it outputs its tensor representation.

3 Network factorization and evaluation of performance

We study the difference in accuracy and flop counts for two neural network architectures. ResNet50 [1], and AlexNet [2]. Both networks trained and tested on the imagenet data set [3]. These preliminary results are referenced from the work of ruihangdu [4]. The tensor decomposition and network factorizations tasts are implemented using the Tensorly framework [5]. The theoretical framework for tensor decomposition for purpose of factorization of neural network is based on the discussion and theoretical findings in [6]
Choice of AlexNet and ResNet50 is driven by difference in network size. We try to conclude if there is a pattern of effect on error when tensor decomposition is used.

We observe a cost reduction of an average two fold for the two networks, coupled with a drop in accuracy of around a unit percentage. These results do confirm our conjecture that the Tucker decomposition induced error will affect accuracy of prediction. The choice to opt for this compression relies on the task and business requirement in hand, because in reality a unit percentage drop in accuracy could very well be unacceptable for some applications, especially when the deployment of the network is on a large scale and for continuous usage.

Table 1: accuracy loss after uniform tucker decomposition on AlexNet [2] and ResNet50 [1]. Training and tests performed using imagenet [3]

Network	accuracy drop	drop in flops in convolutions (Giga)
AlexNet	1.19	.86
ResNet50	0.48	2.3

4 Conclusion

It is worth noting that Tucker Decomposition does offer a computational advantage that could be exploited for edge devices requiring deep learning technologies. However, a study of induced error on prediction power of the model must be conducted to observe if the trade off is still within business requirement. On the other hand, Tucker decomposition only performs approximation on the linear components of the network, hence leaving room for further numerical optimizations on the nonlinear components. Furthermore, One might speculate that performing approximation on individual layers, might result in loss of information coded in between layers, and hence, an objective of a global numerical approximation of the

layers might be reasonable to conserve prediction power of the network

Acknowledgments

I must sincerely thank my supervisor Alexander Wong, professor of Computer Science at University of Waterloo and co-director of the Vision and Image Processing Lab.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[2] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[4] ruihangdu, "Decompose cnn," https://github.com/ruihangdu/Decompose-CNN.

[5] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1–6, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-277.html

[6] M. Astrid and S. Lee, "Cp-decomposition with tensor power method for convolutional neural networks compression," *CoRR*, vol. abs/1701.07148, 2017. [Online]. Available: http://arxiv.org/abs/1701.07148