Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition

Vadim Lebedev Yaroslav Ganin Maksim Rakhuba Ivan Oseledets Victor Lempitsky

Yandex, Moscow, Russia Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia
Moscow Institute of Physics and Technology, Moscow Region, Russia Institute of Numerical Mathematics RAS, Moscow, Russia

Overview

- A new approach to speed-up CNNs is proposed

Method (for one layer)

1. decompose a 4D convolutional tensor into a product of four 2D tensors
2. replace the full convolution with a chain of four reduced convolutions
3. fine-tune the entire CNN on training data

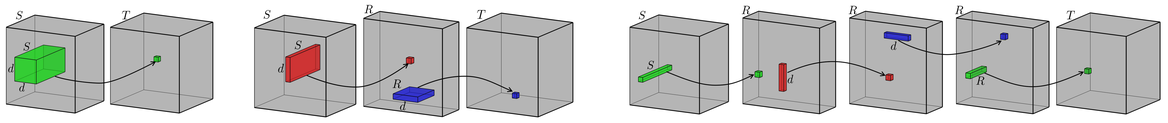
- Advanced non-greedy CP-decomposition in *step 1* gives substantial boost
- Small network (character classification, *CharNet* [2]): whole network 8.5× speedup (accuracy drop 91% → 90%)
- AlexNet: 4× for the second convolution layer at the cost of 1% of the overall top-5 classification error.

Advantages

- **Ease of the decomposition implementation.** CP-decomposition is one of the standard tools in tensor linear algebra. One can use existing efficient algorithms such as NLS or ALS to compute it efficiently.
- **Decomposition accuracy.** Algorithms that optimize the decomposition components simultaneously (ALS, NLS) are known to produce better fit than greedy decompositions algorithms for the tensors of rank $R > 2$
- **Ease of the CNN implementation.** No custom layer implementation is needed. Stride and padding in the original convolution are easily accounted for.
- **Ease of fine-tuning.** Once a convolutional layer is approximated and replaced with a sequence of four convolutional layers with smaller kernels, it is straight-forward to finetune the *entire* network on training data using back-propagation.
- **Efficiency.** Better speed-accuracy trade-off for the approximated CharNet than the previous methods.

Methods Comparison

Full convolution Two component decomposition proposed in [2] Our four component decomposition based on CP-decomposition



Full convolution	Two component decomposition		Our four component decomposition			
num_output : N	num_output : R	num_output : N	num_output : R	num_output : R	num_output : R	num_output : N
kernel_size : d	kernel_h : d	kernel_h : 1	kernel_size : 1	kernel_h : 1	kernel_h : d	kernel_size : 1
	kernel_w : 1	kernel_w : d		kernel_w : d	kernel_w : 1	
				group : R	group : R	

Decomposition

The most time-consuming operation within modern CNNs is generalized convolution with 4D kernel:

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s) \quad (1)$$

The rank-R CP-decomposition of the 4D kernel:

$$K(i,j,s,t) = \sum_{r=1}^{R} K^x(i,r)\, K^y(j,r)\, K^s(s,r)\, K^t(t,r) \quad (2)$$

Tensorlab package (`www.tensorlab.net`) can be used to find CP-decompositions. Substituting 2 into convolution and regrouping terms gives:

$$V(x,y,t) = \sum_{r=1}^{R} K^t(t,r) \left(\sum_{i=x-\delta}^{x+\delta} K^x(i-x+\delta, r) \left(\sum_{j=y-\delta}^{y+\delta} K^y(j-y+\delta, r) \left(\sum_{s=1}^{S} K^s(s,r)\, U(i,j,s) \right) \right) \right) \quad (3)$$

Expression 3 can be computed via a sequence of four convolutions with smaller kernels with the following sizes

1. 1×1 convolution with R output channels
2. vertical filters, separately in every channel
3. horizontal filters, separately in every channel
4. 1×1 convolution with N output channels

Initialization

Accuracy drop for the greedy[1] and the non-linear least-squares (NLS) CP-decomposition. The results are given for the second layer of `CharNet`

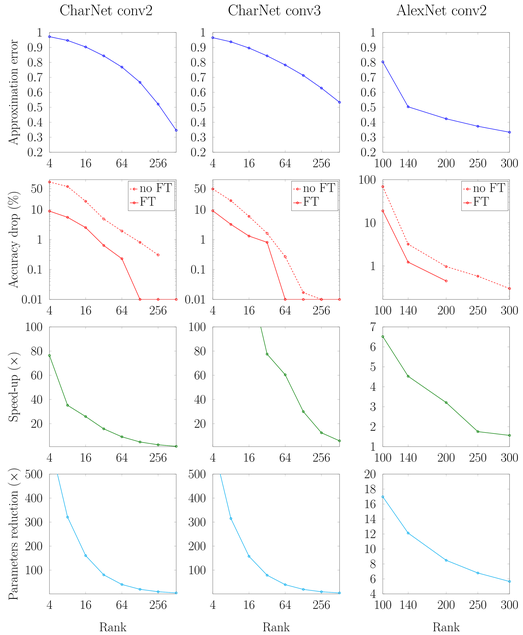
	`CharNet`, R=16		`CharNet`, R=64		`CharNet`, R=256	
	NO FT	FT	NO FT	FT	NO FT	FT
RANDOM	–	9.70	–	7.64	–	6.13
GREEDY	24.15	2.64	4.99	0.46	1.14	-0.31
NLS	**18.96**	**2.16**	**1.93**	**0.09**	**0.31**	**-0.52**

	`AlexNet`, R=140	`AlexNet`, R=200	`AlexNet`, R=300
GREEDY	65.04	7.29	4.76
NLS	3.21	0.97	0.30

Fine-tuning

- Fine-tuning of the entire network is possible through backpropagation
- Four subsequent convolutional layers can lead to exploding gradients. Setting zero learning rate for them accelerates convergence without significant loss of accuracy.
- Even better way: alternating optimization (fix some, learn some).

Experiments



Performance metrics of different approximated CNNs as functions of the approximation rank. **First row:** kernel tensor approximation error. **Second row:** accuracy drop before and after finetuning. **Third row:** empirical speed-up. **Forth row:** ratio between the numbers of parameters in the original and the approximated layers.

References

[1] Denton, Emily, Zaremba, Wojciech, Bruna, Joan, LeCun, Yann, and Fergus, Rob. Exploiting linear structure within convolutional networks for efficient evaluation. arXiv preprint arXiv:1404.0736, 2014

[2] Jaderberg, Max, Vedaldi, Andrea, and Zisserman, Andrew. Speeding up convolutional neural networks with low rank expansions. In Proceedings of the British Machine Vision Conference (BMVC), 2014