Ultimate tensorization: compressing convolutional and FC layers alike

Timur Garipov[1] Dmitry Podoprikhin[1,2] Alexander Novikov[3] Dmitry Vetrov[2,3]

[1]Moscow state university, Moscow, Russia

[2]Yandex LLC, Moscow, Russia

[3]Higher School of Economics, Moscow, Russia

bayesgroup.ru

Motivation 1

- Convolutional neural networks are powerful but too large for smartphones.
- Fitting the network into on-chip SRAM cache can save battery 10×.
- We focus on compressing convolutional layers, since
  - several modern architectures (Inception, ResNet) lack FC layers;
  - we can compress FC layers to move the bottleneck into convolutional layers.

Summary 2

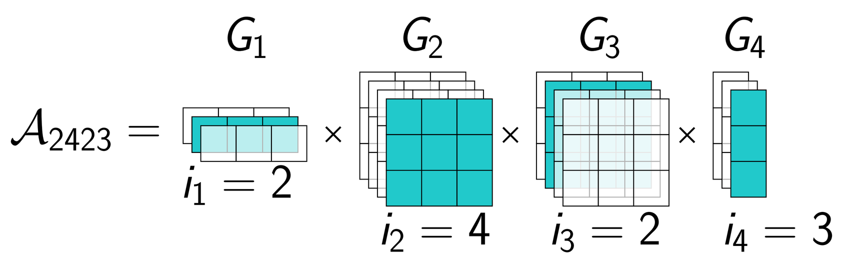
- We factorize the convolutional kernel via the Tensor Train decomposition – a compact format for tensors (=multidimensional arrays).
- We compressed a network that consisted only of convolutions up to 4× with 2% accuracy decrease.
- We combine the proposed approach with the FC layers compression of [1]. Compressing both convolutional and fully-connected layers of a network yielded 82× network compression rate with 1% accuracy drop.
- TensorFlow code: `https://github.com/timgaripov/TensorNet-TF`

Tensor Train 3

- Tensor Train decomposition [2]: reshape a matrix $\boldsymbol{W}$ to a tensor (=multidimensional array) and use the following format:

$$W(t,\ell) = W(i_1,\dots,i_d;\ j_1,\dots,j_d) = \underbrace{G_1[i_1,j_1]}_{1\times r}\underbrace{G_2[i_2,j_2]}_{r\times r}\dots\underbrace{G_d[i_d,j_d]}_{r\times 1}$$

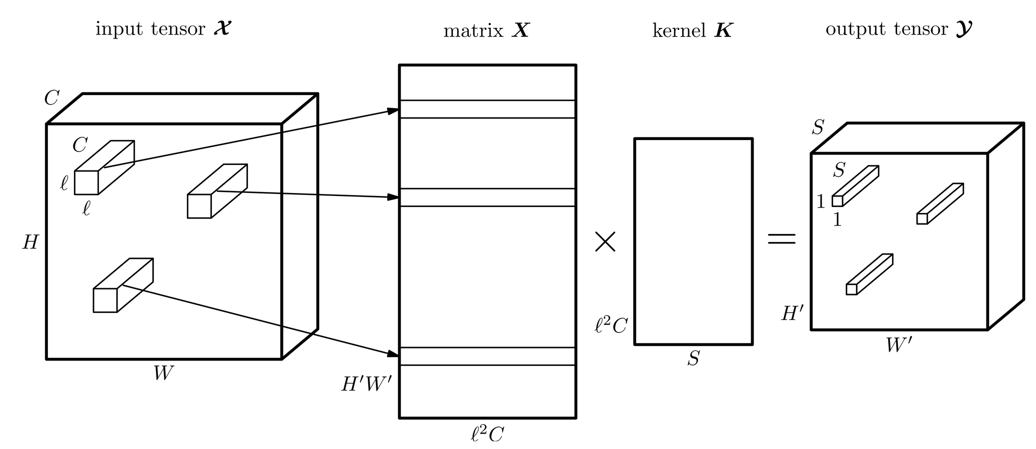
- The number r is called TT-rank.
- An illustration of the TT-format for a $3\times4\times4\times3$ tensor $\boldsymbol{\mathcal{A}}$ with the TT-rank equal 3



Tensor Train properties 4

- Any matrix can be represented in TT-format with sufficient TT-rank r.
- The TT-format is efficient when the TT-rank r is small.
- The TT-format allows to perform linear algebra operations (such as sum of two matrices) efficiently and get the result in the TT-format.
- The TT-format allows to efficiently factorize a given matrix.
- MATLAB and Python libraries for these operations are available online.

Convolution via matrix multiplication 5



A convolution of the input $\boldsymbol{\mathcal{X}}$ with the kernel $\boldsymbol{\mathcal{K}}$

$$\mathcal{Y}(x,y,s) = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\sum_{c=1}^{C}\mathcal{K}(i,j,c,s)\,\mathcal{X}(x+i-1,y+j-1,c).$$

$$\Downarrow$$

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{K}$$

Why direct TT-decomposition of $\mathcal{K}$ is bad? 6

- Direct TT-decomposition of kernel tensor with 1×1 filters coincides with the matrix low-rank decomposition.
- Matrix low-rank proved to be less efficient than matrix TT-format for neural networks [1].

Main idea 7

- We apply TT-decomposition to the matrix $\boldsymbol{K}$ instead of the tensor $\boldsymbol{\mathcal{K}}$.
- We treat 3-dim input and output tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ tensors as tensors of higher-order (e.g. 5-dimensional as in the example below).
- $\mathcal{X}(x,y,\mathbf{c}) \xrightarrow{\text{reshape}} \widetilde{\mathcal{X}}(x,y,c_1,c_2,c_3)$

  $\mathcal{Y}(x,y,\mathbf{s}) \xrightarrow{\text{reshape}} \widetilde{\mathcal{Y}}(x,y,s_1,s_2,s_3)$

  $\mathcal{K}(x,y,\mathbf{c},\mathbf{s}) \xrightarrow{\text{reshape}} K(x+\ell(y-1)+\ell^2(\mathbf{c}-1),\mathbf{s}) \xrightarrow{\text{TT-format}} G_0[x,y]G_1[c_1,s_1]G_2[c_2,s_2]G_3[c_3,s_3]$
- TT-convolution

$$\widetilde{\mathcal{Y}}(x,y,s_1,s_2,s_3) = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\sum_{c_1,c_2,c_3}\widetilde{\mathcal{X}}(i+x-1,j+y-1,c_1,c_2,c_3)\cdot$$
$$\cdot\, G_0[i,j]G_1[c_1,s_1]G_2[c_2,s_2]G_3[c_3,s_3]$$

Learning 8

- Instead of $\mathcal{K}$ optimize w.r.t. $\{\boldsymbol{G}_k\}_{k=1}^d$ with stochastic gradient descend.
- Compute the necessary gradients with automatic differentiation implemented in TensorFlow.
- Initialize $\{\boldsymbol{G}_k\}_{k=1}^d$ with gaussian noise.
- Apply BatchNormalization after each TT-conv and TT-fc layer.

Experiments 9

- Experiments on CIFAR-10 dataset.
- We used two baseline networks:
  1. fully convolution network (557 thousand of parameters, 6 convolutional layers);
  2. convolution network with FC layers (14 millions of parameters, 6 convolutional and 3 FC layers).
- Different rows with the same model name in the tables below correspond to different choices of the TT-ranks.

Convolutional network 10

Model	top-1 accuracy	compression
conv (baseline)	90.7	1
TT-conv	89.9	2.02
TT-conv	89.2	2.53
TT-conv	89.3	3.23
TT-conv	88.7	4.02
TT-conv (naive)	88.3	2.02
TT-conv (naive)	87.6	2.90

'TT-conv': the proposed compression method;**'TT-conv (naive)'**: direct application of the TT-decomposition to convolutional kernels

Network with convolutional and FC layers 11

Model	top-1 accuracy	compression
conv-fc (baseline)	90.5	1
conv-TT-fc	90.3	10.72
conv-TT-fc	89.8	19.38
conv-TT-fc	89.8	21.01
TT-conv-TT-fc	90.1	9.69
TT-conv-TT-fc	89.7	41.65
TT-conv-TT-fc	89.4	82.87

'conv-TT-fc': only the fully-connected part of the network is compressed; **'TT-conv-TT-fc'**: fully-connected and convolutional parts are compressed.

Future work 12

- Provide experiments with modern architectures (ResNet, Inception, VGG-16) on ImageNet dataset.
- Provide a strategy to select ranks and modes.
- Compare with other works.

References

[1] Alexander Novikov et al. "Tensorizing neural networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 442–450

[2] Ivan V Oseledets. "Tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317