Tensorizing Neural Networks

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, Dmitry Vetrov

NIPS, 2015

Presentation by:
Moussa Traore
Mehraveh Javan

Outline

- Motivation
- Tensor Train (TT) Format
  - TT-Format for vectors
  - TT-Format for matrices
- TT-Layer
  - Back propagation
- Experimental Results
- Related Work

Motivation

Deep Neural Networks have achieved state-of-the -art-performance in many domains, however they require large amount of memory, expensive hardware (GPUs) and long processing times, preventing the increase in model size and using mobile devices.

Neural Network compression using matrix and tensor decomposition

Motivation

The huge number of parameters of FC layers is the bottleneck in a typical deep neural network

Fully connected layer is a transformation between two high dimensional vectors:

$$\underbrace{\mathbf{y}}_{M} = \underbrace{\mathbf{W}}_{M \times N} \underbrace{\mathbf{x}}_{N} + \underbrace{\mathbf{b}}_{M}$$

Example:VGG-19 architecture

Layer name	#Filters	#Parameters	#Activations
input			150K
conv1_1	64	1.7K	3.2M
conv1_2	64	36K	3.2M
max pooling			802K
conv2_1	128	73K	1.6M
conv2_2	128	147K	1.6M
max pooling			401K
conv3_1	256	300K	802K
conv3_2	256	600K	802K
conv3_3	256	600K	802K
conv3_4	256	600K	802K
max pooling			200K
conv4_1	512	1.1M	401K
conv4_2	512	2.3M	401K
conv4_3	512	2.3M	401K
conv4_4	512	2.3M	401K
max pooling			100K
conv5_1	512	2.3M	100K
conv5_2	512	2.3M	100K
conv5_3	512	2.3M	100K
conv5_4	512	2.3M	100K
max pooling			25K
fc6		103M	4K
fc7		17M	4K
output		4M	1K

95% of parameters are in fully connected layer

Motivation

TensorNet applies Tensor Train (TT) format to represent weights of fully connected layer, reducing the number of parameters hugely without compromising performance:

- Compatible with the existing training algorithms for neural networks
- Match the performance of the uncompressed counterparts with compression
- Enables using more hidden units than was available before

How About Matrix Rank Decomposition?

$$\underbrace{\mathbf{W}}_{M \times N} = \underbrace{\mathbf{A}}_{M \times r} \underbrace{\mathbf{B}}_{r \times N}$$

Compression:

$$O\Big(MN\Big) \rightarrow O\Big(r(M+N)\Big)$$

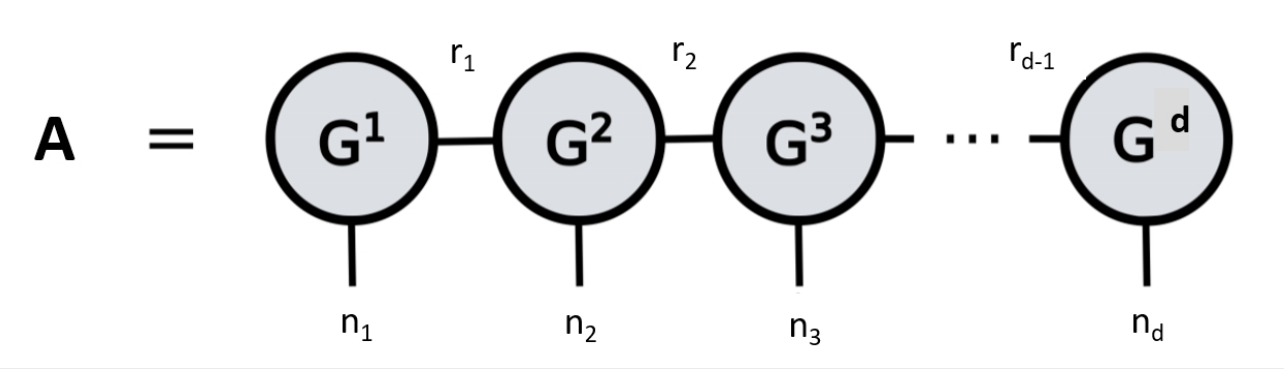
Low-rank representation can reduce number of parameters without compromising accuracy

But can we compress more?

Tensor Train Summary

Tensor **A** can be decomposed to TT-format as**:**

$$\mathbf{A}(i_1, i_2, \ldots, i_d) = \mathbf{G_1}[i_1]\mathbf{G_2}[i_2]\ldots\mathbf{G_d}[i_d]$$



Tensor Train Summary

Tensor **A** can be decomposed to TT format as**:**

$$\mathbf{A}(i_1, i_2, \ldots, i_d) = \mathbf{G_1}[i_1]\mathbf{G_2}[i_2]\ldots\mathbf{G_d}[i_d]$$

Where:

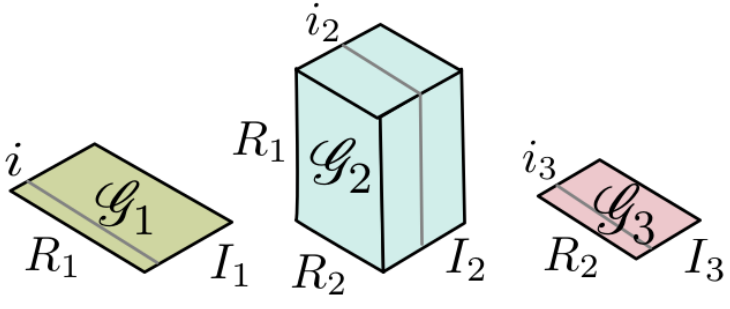
$$\mathbf{G_k}[i_k] \in \mathbb{R}^{r_{k-1} \times r_k} \quad , \quad r_0 = r_d = 1$$

Compression:

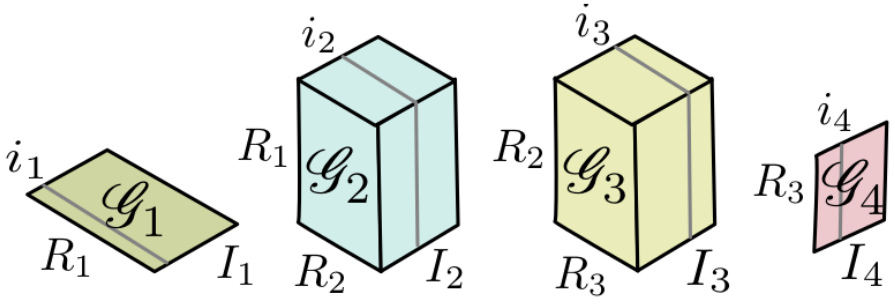
$$O\left(n^d\right) \rightarrow O\left(ndr^2\right)$$

- TT-cores: $\mathbf{G_k}$
- TT-ranks: r_k
- TT max rank $r = max\ r_k \quad , \quad k = 0, \ldots, d$

Tensor Train Summary: Examples



3D



4D

Source: https://www.ifi.uzh.ch/dam/jcr:846e4588-673e-4f55-b531-544a2e1f602e/TA_Tutorial_Part2.pdf

TT-representations for vectors

Consider a vector b: $b \in \mathbb{R}^N$

Where: $N = \prod_{k=1}^{d} n_k$

We can establish a bijection:

$$\mu : l \in \{1, \ldots, N\} \mapsto (\mu_1(l), \ldots, \mu_d(l))$$

Where:

$$\mu_k(l) \in \{1, \ldots, n_k\}$$

We can represent it using a tensor B:

$$B \in \mathbb{R}^{n_1 x n_2 x \ldots x n_d}$$

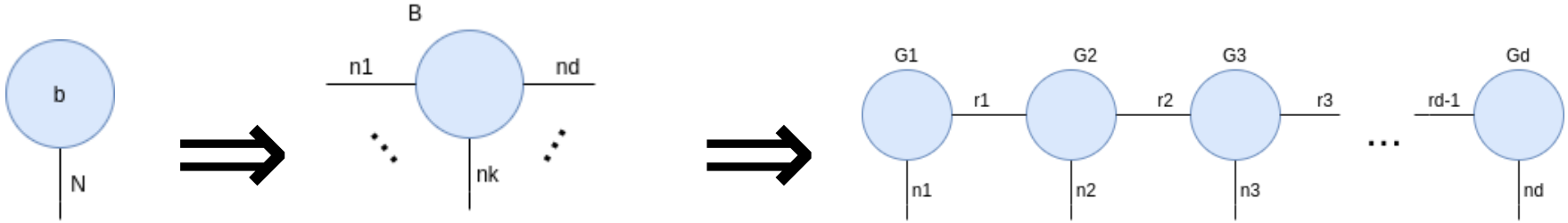
Where:

$$B((\mu_1(l), \ldots, \mu_d(l)) = b_l$$

We can represent the vector in TT-Format

Vector ⇒ TT-Format: Tensor Network Diagram

$$b(l) = B((\mu_1(l), \ldots, \mu_d(l)) = \underbrace{G_1[\mu_1(l)]}_{1Xr_1}\underbrace{G_2[\mu_2(l)]}_{r_1Xr_2}\ldots\underbrace{G_d[\mu_d(l)]}_{r_{d-1}X1}$$



TT-representations for matrices (1)

Consider a matrix A:

Where: $A \in \mathbb{R}^{\mathrm{MxN}}$

And: $M = \Pi_{k=1}^{d} m_k$, $N = \Pi_{k=1}^{d} n_k$

We can establish the bijections:

$$\nu : t \in \{1, \ldots, M\} \mapsto (\nu_1(t), \ldots, \nu_d(t))$$

And:

$$\mu : l \in \{1, \ldots, N\} \mapsto (\mu_1(l), \ldots, \mu_d(l))$$

Where:

$$\mu_k(l) \in \{1, \ldots, n_k\} \qquad \nu_k(t) \in \{1, \ldots, m_k\}$$

And

TT-representations for matrices (2)

We can represent it using a tensor W:

$$W \in \mathbb{R}^{m_1 n_1 x m_2 n_2 x \ldots x m_d n_d}$$

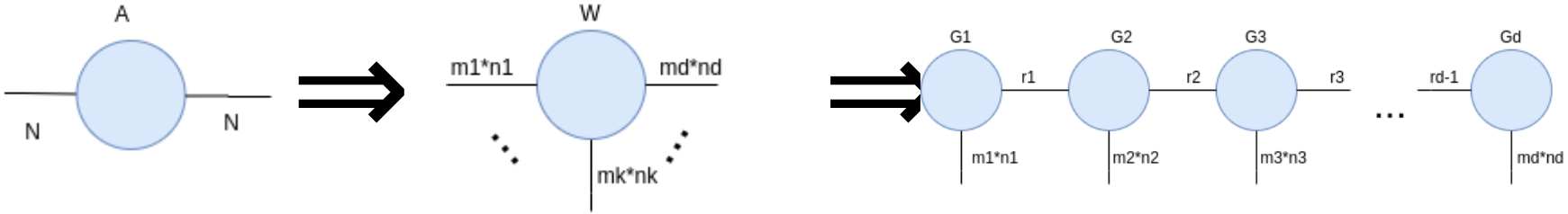
Where:

$$W((\nu_1(t), \mu_1(l)), \ldots, (\nu_d(t), \mu_d(l))) = A(t, l)$$

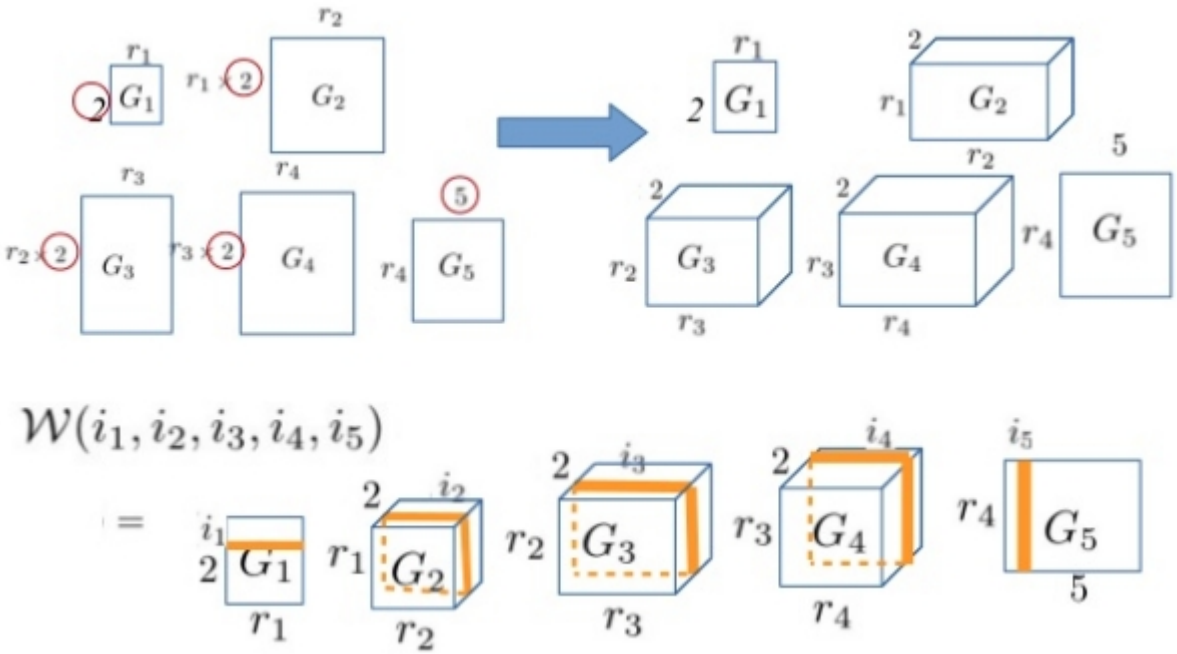
We can represent the matrix in TT-Format

Matrices ⇒ TT-Format: Tensor Network Diagram

$$A(t,l) = W((\nu_1(t),\mu_1(l)),\ldots,(\nu_d(t),\mu_d(l))) = \underbrace{G_1[\nu_1(t),\mu_1(l)]}_{1Xr_1}\underbrace{G_2[\nu_2(t),\mu_2(l)]}_{r_1Xr_2}\ldots\underbrace{G_d[\nu_d(t),\mu_d(l)]}_{r_{d-1}X1}$$



Matrices ⇒ TT-Format: Tensor Network Diagram

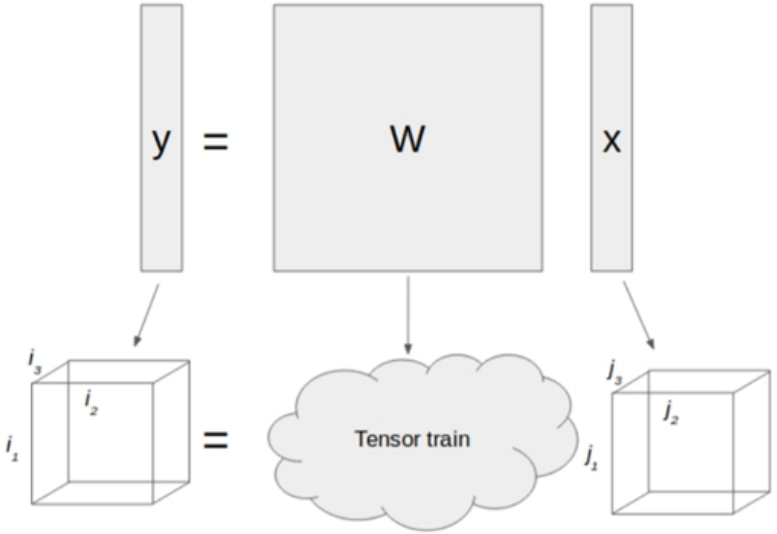


Source: https://www.slideshare.net/RuochunZeung/tensorizing-neuralnetwork-present-72811535

TT-representations of Neural Networks

TT-layer: a fully connected layer with weights stored in TT format

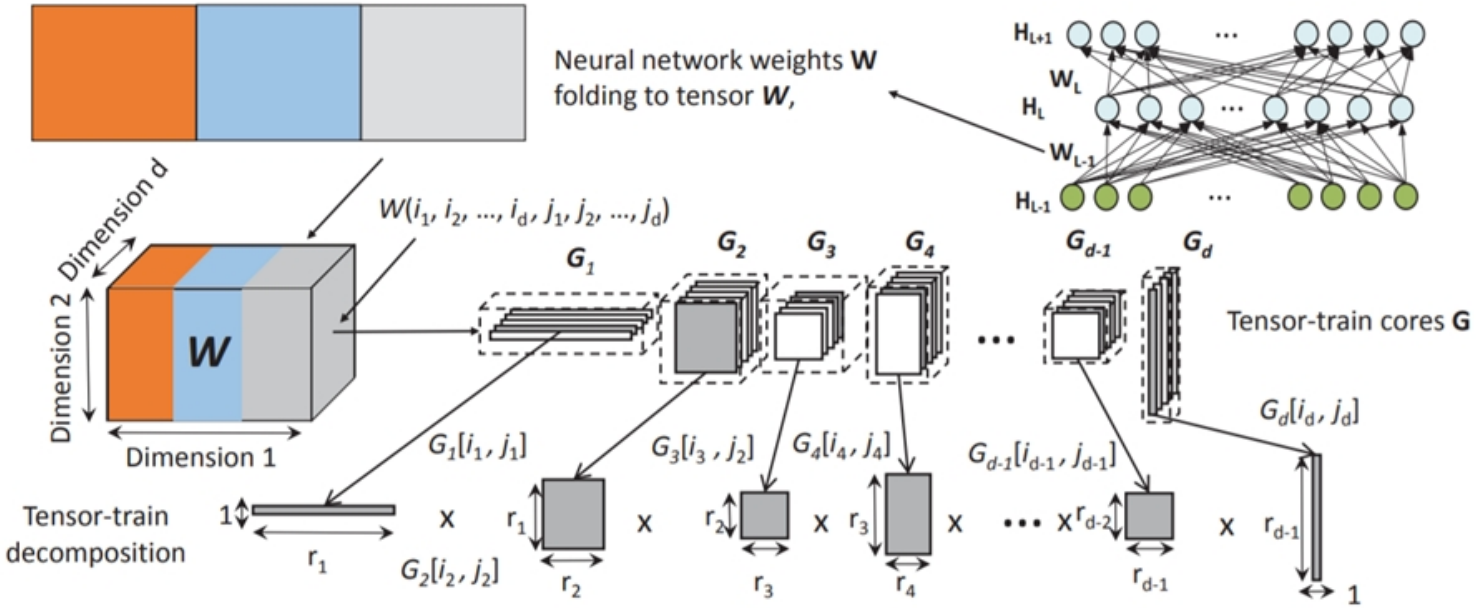
$$\underbrace{\mathbf{y}}_{M} = \underbrace{\mathbf{W}}_{M \times N} \underbrace{\mathbf{x}}_{N} + \underbrace{\mathbf{b}}_{M}$$



Transforming $\mathbf{x}$,$\mathbf{y}$ and $\mathbf{b}$ to d-dimensional tensors, TT layer is expressed as:

$$Y(i_1, \ldots, i_d) = \sum_{j_1, \ldots j_d} \mathbf{G_1}[i_1, j_1] \ldots \mathbf{G_d}[i_d, j_d] X(j_1, \ldots, j_d) + B(i_1, \ldots, i_d)$$

TT-representations of Neural Networks



Source: LTNN: An energy-efficient machine learning accelerator on 3D CMOS-RRAM for layer-wise tensorized neural network, Huang et al. 2017

Learning For TT-Layer | Back-Propagation (1)

$$i_k^- := (i_1, \ldots, i_{k-1})$$
$$i_k^+ := (i_{k+1}, \ldots, i_d)$$

$$\mathbf{i} = (i_k^-, i_k, i_k^+)$$

$$P_k^-[i_k^-, j_k^-] := G_1[i_1, j_1] \ldots G_{k-1}[i_{k-1}, j_{k-1}]$$
$$P_k^+[i_k^+, j_k^+] := G_{k+1}[i_{k+1}, j_{k+1}] \ldots G_d[i_d, j_d]$$

Using these constructs, we can rewrite the TT-Layer as follow:

$$Y(i_1, \ldots, i_d) = \sum_{j_1, \ldots j_d} \mathbf{G_1}[i_1, j_1] \ldots \mathbf{G_d}[i_d, j_d] X(j_1, \ldots, j_d) + B(i_1, \ldots, i_d)$$

Becomes:

$$\Upsilon(\mathbf{i}) = \Upsilon(i_k^-, i_k, i_k^+) = \sum_{(j_k^-, j_k, j_k^+)} P_k^-[i_k^-, j_k^-] G_k[i_k, j_k] P_k^+[i_k^+, j_k^+] \chi(j_k^-, j_k, j_k^+) + B(\mathbf{i})$$

Learning For TT-Layer | Back-Propagation (2)

Postfix sequences of indices:

$$i_k^- := (i_1, \ldots, i_{k-1})$$

$$i_k^+ := (i_{k+1}, \ldots, i_d)$$

$$\mathbf{i} = (i_k^-, i_k, i_k^+)$$

Partial core products:

$$P_k^-[i_k^-, j_k^-] := G_1[i_1, j_1] \ldots G_{k-1}[i_{k-1}, j_{k-1}]$$

$$P_k^+[i_k^+, j_k^+] := G_{k+1}[i_{k+1}, j_{k+1}] \ldots G_d[i_d, j_d]$$

Learning For TT-Layer | Back-Propagation (3)

Learning For TT-Layer | Back-Propagation (3)

$$\Upsilon(\mathbf{i}) = \Upsilon(i_k, i_k, i_k) = \sum_{(j_k^-, j_k, j_k^+)} P_k^-[i_k^-, j_k^-] G_k[i_k, j_k] P_k^+[i_k^+, j_k^+] \chi(j_k^-, j_k, j_k^+) + B(\mathbf{i})$$

The Gradient of the loss function L w.r.t the k-th core in position [î,ĵ]:

$$\frac{\partial L}{\partial G_k[\hat{i_k}, \hat{j_k}]} = \sum_i \frac{\partial L}{\partial \Upsilon(i)} \frac{\partial \Upsilon(i)}{\partial G_k[\hat{i_k}, \hat{j_k}]}$$

Assuming we know:

$$\frac{\partial L}{\partial \Upsilon(i)}$$

Learning For TT-Layer | Back-Propagation (4)

$$\Upsilon(\mathbf{i}) = \Upsilon(i_k^-, i_k, i_k^+) = \sum_{(j_k^-, j_k, j_k^+)} P_k^-[i_k^-, j_k^-] G_k[i_k, j_k] P_k^+[i_k^+, j_k^+] \chi(j_k^-, j_k, j_k^+) + B(\mathbf{i})$$

We need to compute:

$$\frac{\partial \Upsilon(i_k^-, i_k, i_k^+)}{\partial G_k[\hat{i_k}, \hat{j_k}]}$$

For any values of the core $k \in \{1, .., d\}$

and $\hat{i_k} \in \{1, \ldots m_k\}$

and $\hat{j_k} \in \{1, \ldots n_k\}$

Learning For TT-Layer | Back-Propagation (5)

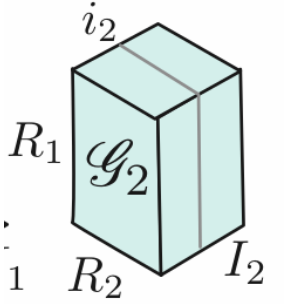
$$\frac{\partial a^T X b}{\partial X} = a b^T$$

Learning | Back-Propagation (6)

$$\Upsilon(\mathbf{i}) = \Upsilon(i_k^-, i_k, i_k^+) = \sum_{(j_k^-, j_k, j_k^+)} P_k^-[i_k^-, j_k^-] G_k[i_k, j_k] P_k^+[i_k^+, j_k^+] \chi(j_k^-, j_k, j_k^+) + B(\mathbf{i})$$

For any $i_k \neq \hat{i_k}$ or $j_k \neq \hat{j_k}$

$$\frac{\partial \Upsilon(i_k^-, i_k, i_k^+)}{\partial G_k[\hat{i_k}, \hat{j_k}]} = 0$$



Otherwise

$$\frac{\partial \Upsilon(i)}{\partial G_k[\hat{i_k}, \hat{j_k}]} = \sum_{(j_k^-, j_k^+)} P_k^-[i_k^-, j_k^-]^T P_k^+[i_k^+, j_k^+]^T \chi(j_k^-, \hat{j_k}, j_k^+)$$

$$\frac{\partial a^T X b}{\partial X} = a b^T$$

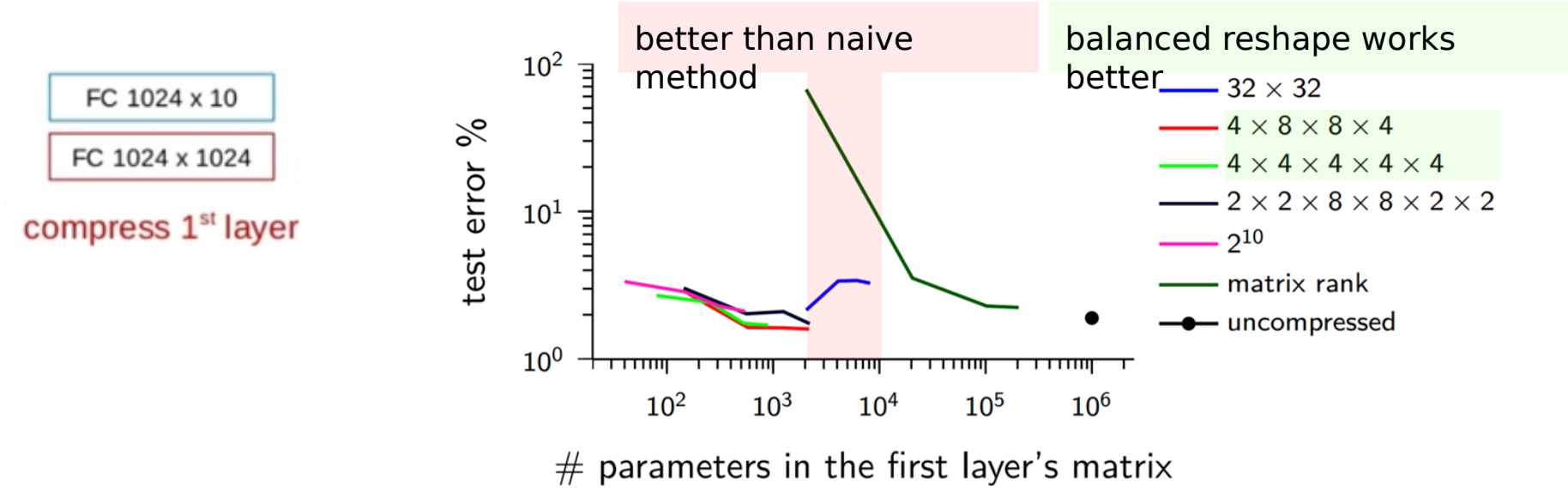
Results: MNIST

Handwritten images

Results: MNIST

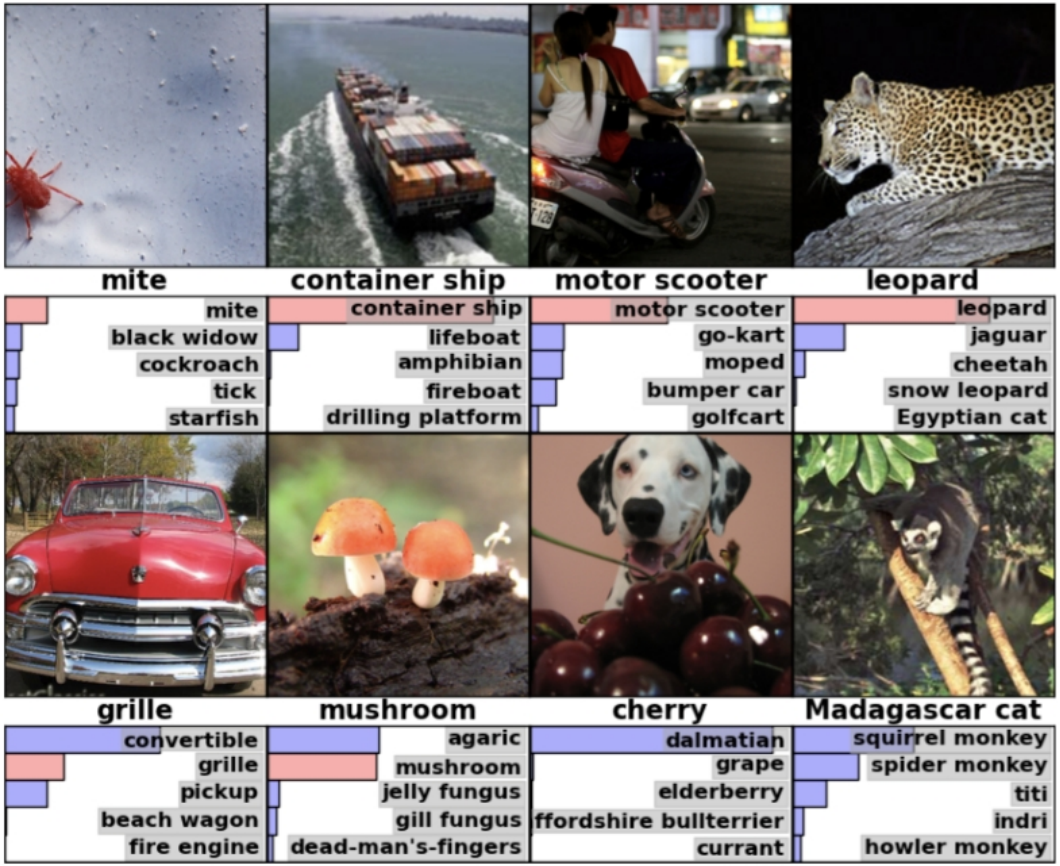
- To investigate properties of TT-layer and different parameters setting using small network with two fully connected layers

Different plot points are obtained by varying TT-rank or matrix rank



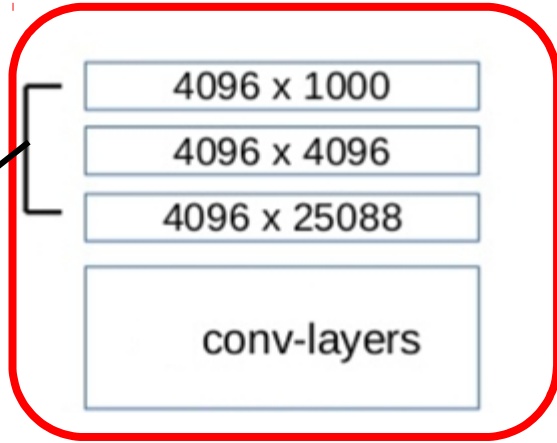
Results: Imag

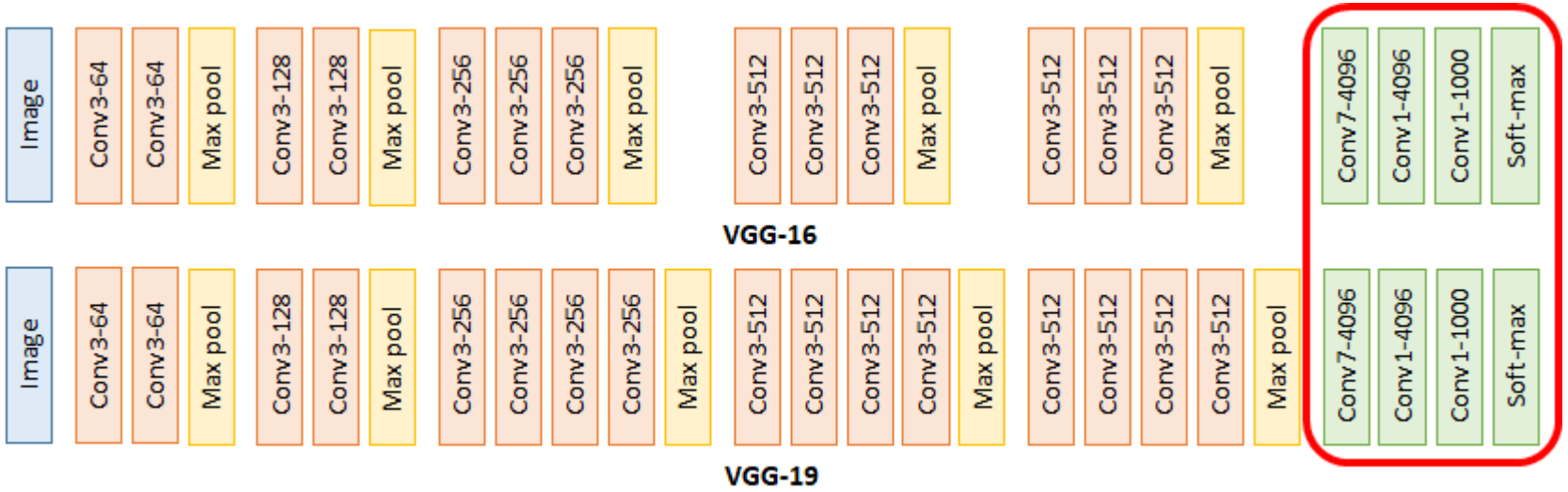
- 1000 classes
- 1.2 M training images



Results: ImageNet

Replacing three fully connected layers in VGG-16 and VGG-19





Results: ImageNet

Architecture	TT-layers compr.	vgg-16 compr.	vgg-19 compr.	vgg-16 top 1	vgg-16 top 5	vgg-19 top 1	vgg-19 top 5
FC FC FC	1	1	1	30.9	11.2	29.0	10.1
TT4 FC FC	50 972	3.9	3.5	31.2	11.2	29.8	10.4
TT2 FC FC	194 622	3.9	3.5	31.5	11.5	30.4	10.9
TT1 FC FC	713 614	3.9	3.5	33.3	12.8	31.9	11.8
TT4 TT4 FC	37 732	7.4	6	32.2	12.3	31.6	11.7
MR1 FC FC	3 521	3.9	3.5	99.5	97.6	99.8	99
MR5 FC FC	704	3.9	3.5	81.7	53.9	79.1	52.4
MR50 FC FC	70	3.7	3.4	36.7	14.9	34.5	15.8

FC: fully connected layer

TT[x]: TT layer with all ranks equal to x

MR[x]: fully connected layer with rank constrained to x

Related Work: Ultimate T

Ultimate tensorization: compressing convolutional and FC layers alike

Timur Garipov[1] **Dmitry Podoprikhin**[1,2] **Alexander Novikov**[3,4] **Dmitry Vetrov**[2,3]
[1]Moscow State University, Moscow, Russia
[2]Yandex, Moscow, Russia
[3]National Research University Higher School of Economics, Moscow, Russia
[4]Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow, Russia
timgaripov@gmail.com podoprikhin.dmitry@gmail.com
novikov@bayesgroup.ru vetrovd@yandex.ru

Related Work: Ultimate Tensorization (2016)

Extension of TT compression to convolutional layers allows for further compression of Network with small performance decrease

Model	top-1 acc.	compr.
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

Conclusion

- TT-decomposition of weight matrix of a fully-connected layer and using the cores of the decomposition as the parameters of the layer
- Train the fully-connected layers compressed by up to 200 000× compared with the explicit parametrization without significant error increase
-

Operation	Time	Memory
FC forward pass	$O(MN)$	$O(MN)$
TT forward pass	$O(dr^2 m \max\{M, N\})$	$O(r \max\{M, N\})$
FC backward pass	$O(MN)$	$O(MN)$
TT backward pass	$O(d^2 \mathrm{r}^4 m \max\{M, N\})$	$O(\mathrm{r}^3 \max\{M, N\})$

Table 1: Comparison of the asymptotic complexity and memory usage of an $M \times N$ TT-layer and an $M \times N$ fully-connected layer (FC). The input and output tensor shapes are $m_1 \times \ldots \times m_d$ and $n_1 \times \ldots \times n_d$ respectively ($m = \max_{k=1\ldots d} m_k$) and r is the maximal TT-rank.

Thank You!