

Tensor train for machine learning and everything else

Skoltech

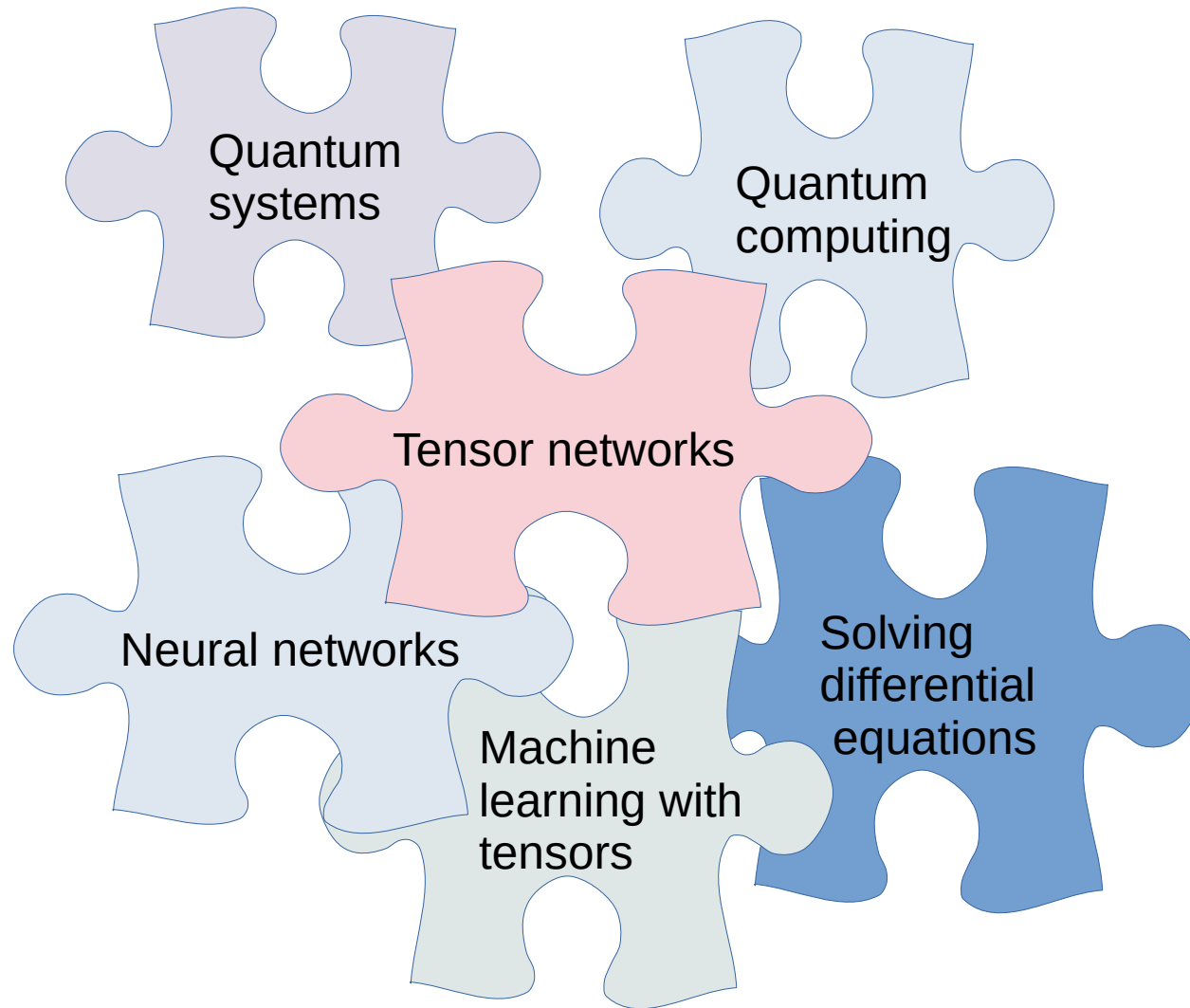
Skolkovo Institute of Science and Technology



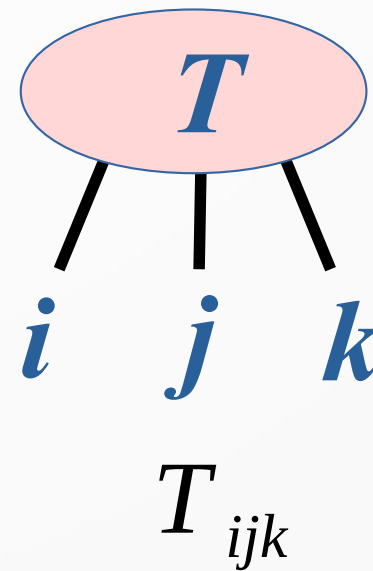
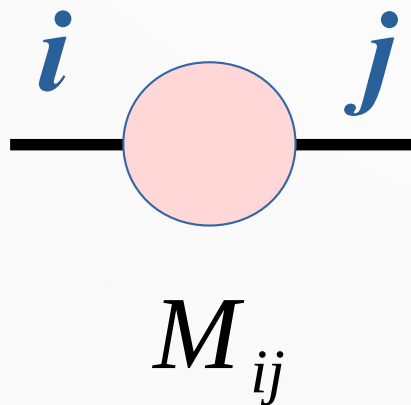
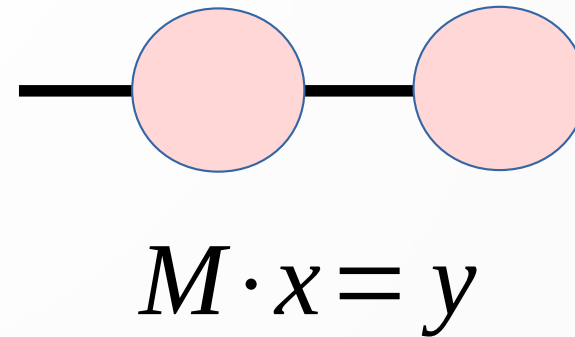
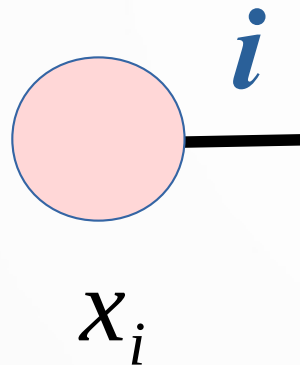
HUAWEI

**Roman Schutski,
Sergey Matveev
CDISE
Skoltech**

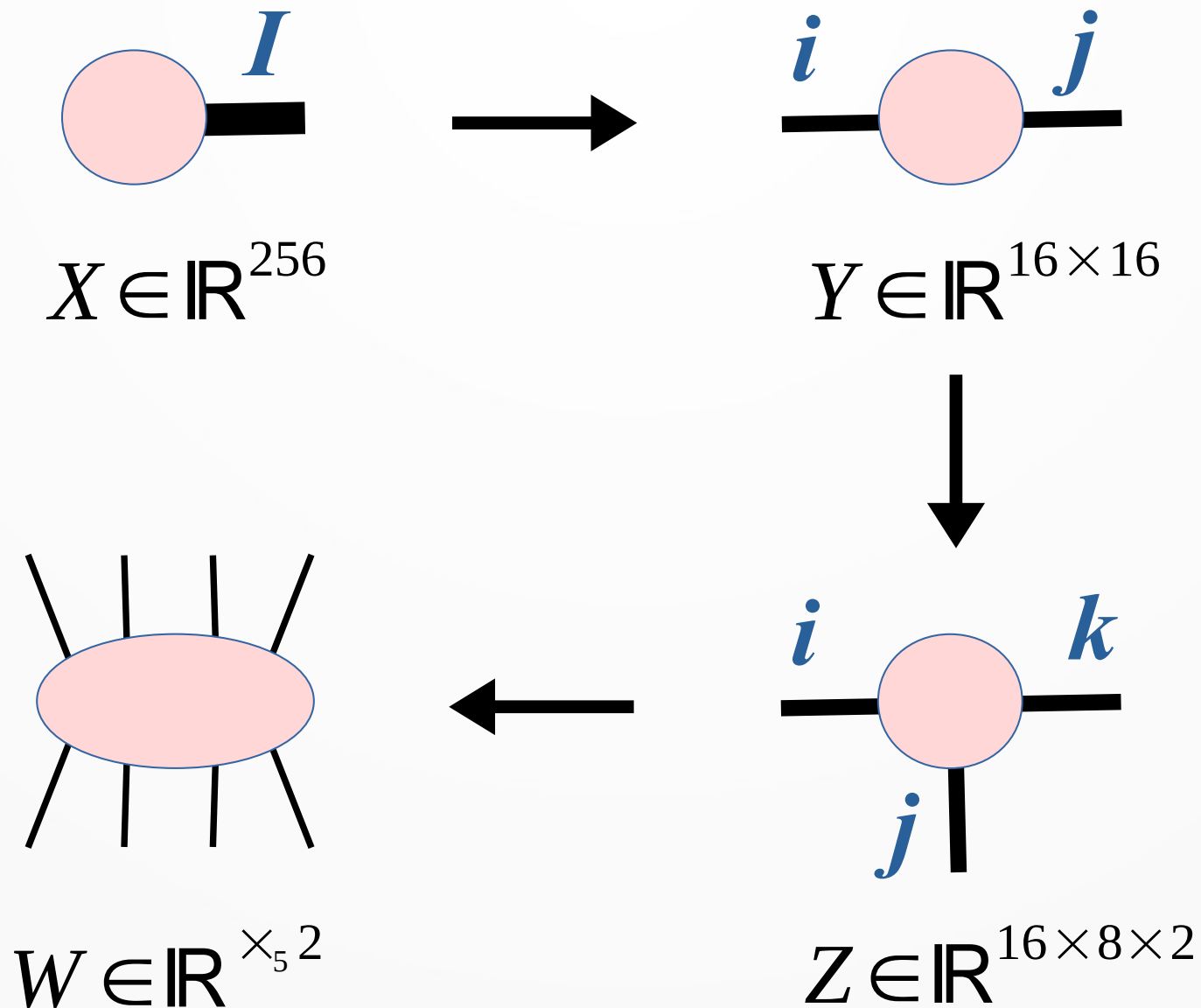
Tensor networks



Step aside: graphical notation

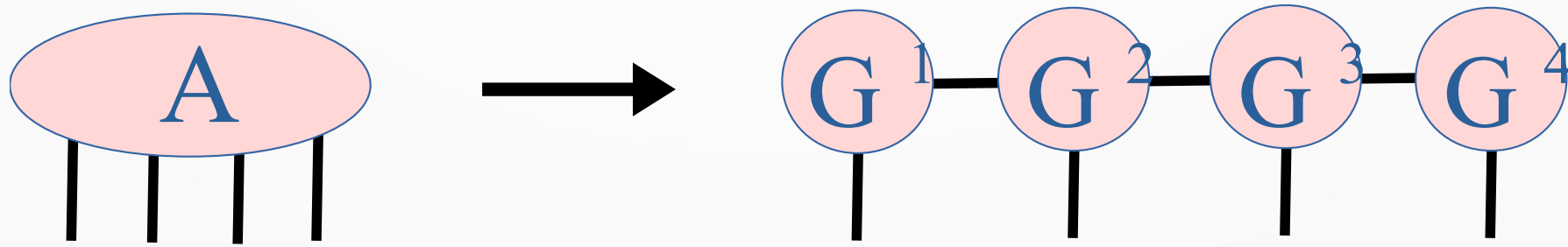
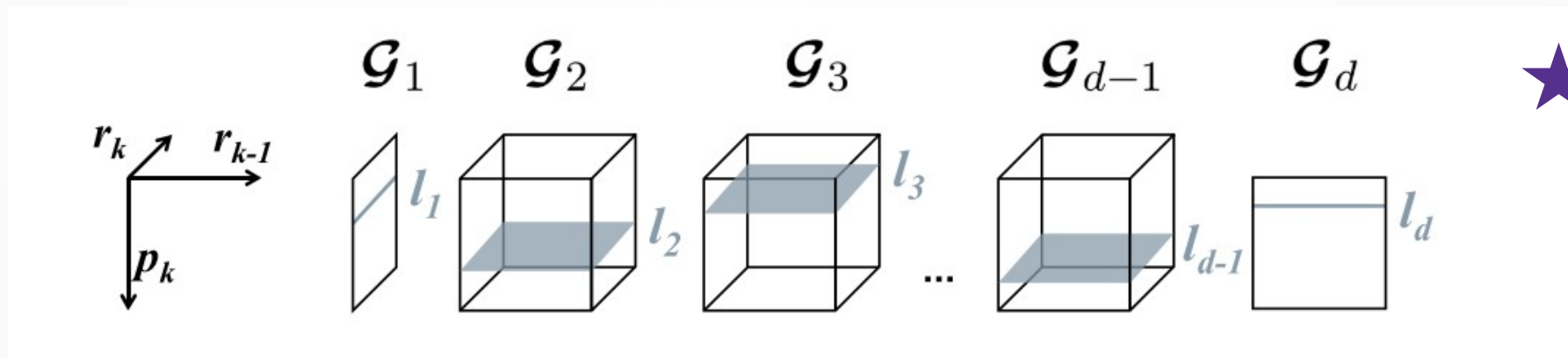


Step aside: tensor reshaping

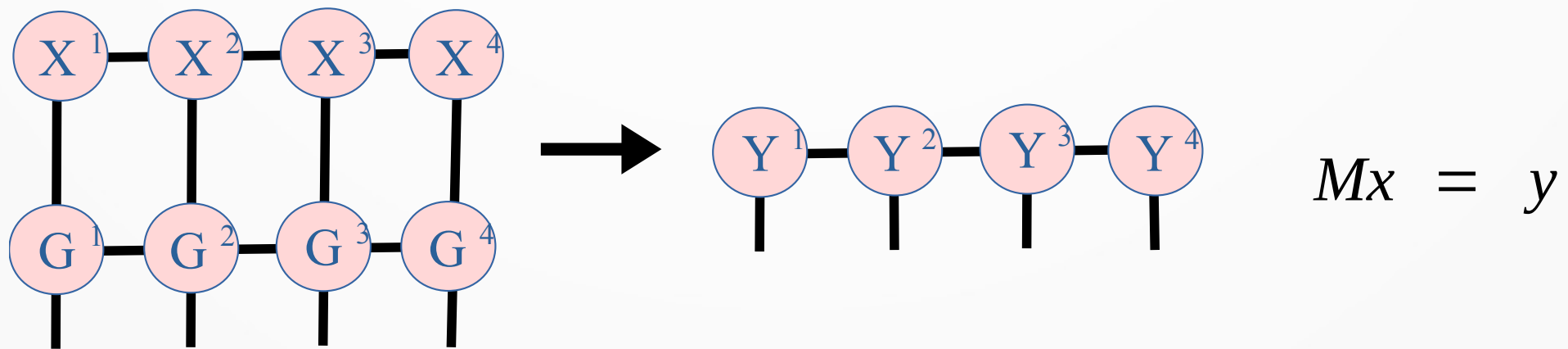
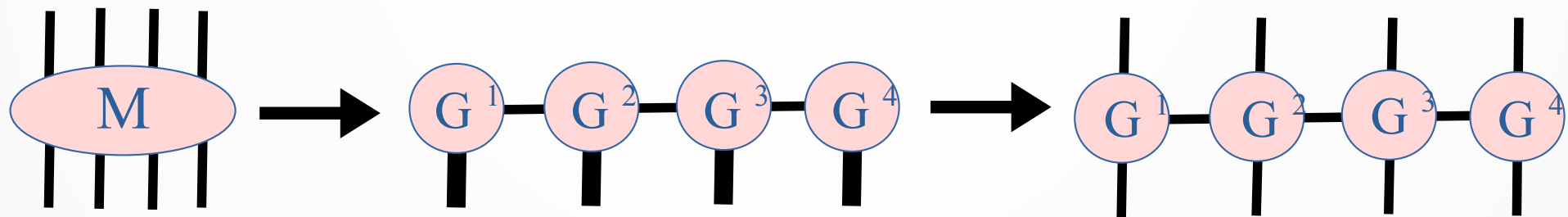


Tensor Train in pictures

$$A(i_1, i_2, i_3, i_4) = G^1(r_0, i_1, r_1) \cdot G^2(r_1, i_2, r_2) \cdot G^3(r_2, i_3, r_3) \cdot G^4(r_3, i_4, r_4)$$
$$r_0 = 1, \quad r_4 = 1$$



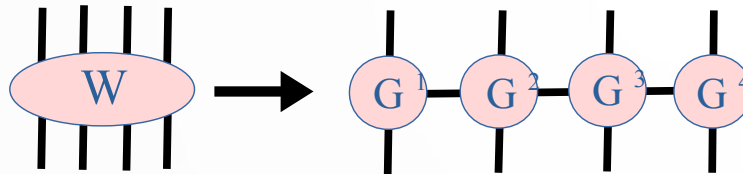
TT matrices



Plan

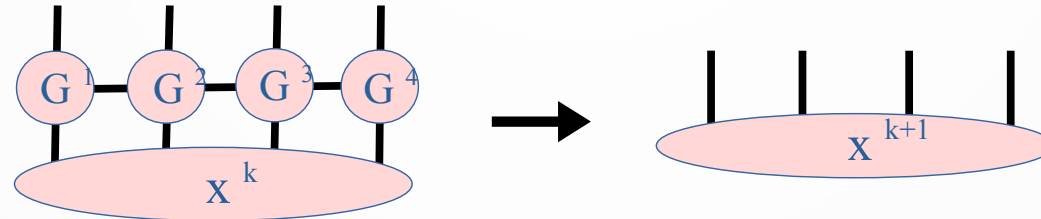
- **TT for compression of feed forward neural networks:
faster inference and/or less memory**
 - Fully connected layers
 - Convolutional layers
 - RNN layers
 - Tricks
- **TT as machine learning models**
 - Generalization properties
 - Connections with physics
- **Final remarks**

TT compression of FC layers



$$Wx^k = x^{k+1}$$

$$f(x^{k+1}) = a^{k+1}$$

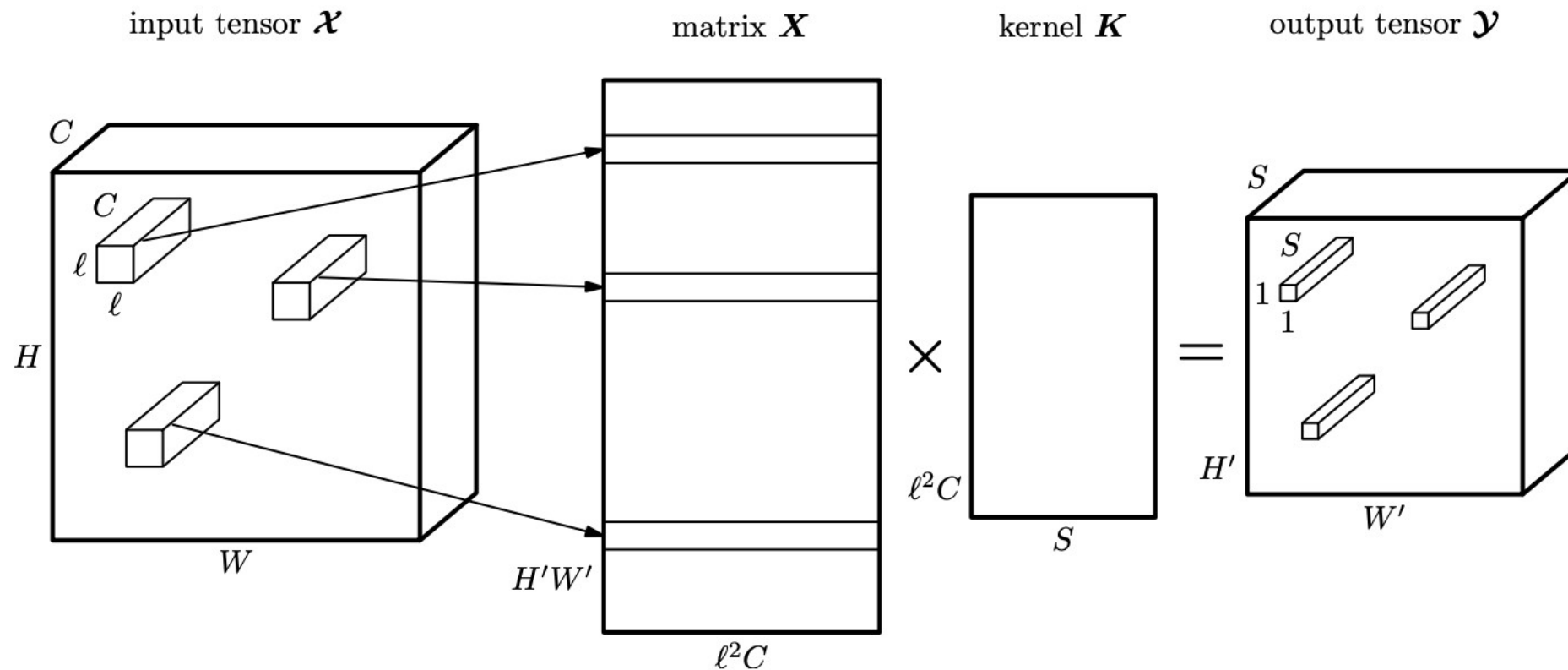


Operation	Time	Memory
FC forward pass	$O(MN)$	$O(MN)$
TT forward pass	$O(dr^2m \max\{M, N\})$	$O(r \max\{M, N\})$
FC backward pass	$O(MN)$	$O(MN)$
TT backward pass	$O(d^2 r^4 m \max\{M, N\})$	$O(r^3 \max\{M, N\})$



TT compression of convolutional layers

Figure 1: Reducing convolution (1) to a matrix-by-matrix multiplication.



Garipov, Podoprikin, Novikov, Vetrov, arXiv:1611.03214.

TT compression of convolutional layers

(a) Compressing the first baseline ('conv'), which is dominated by convolutions. 'TT-conv': the proposed compression method; 'TT-conv (naive)': direct application of the TT-decomposition to convolutional kernels.

Model	top-1 acc.	compr.
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

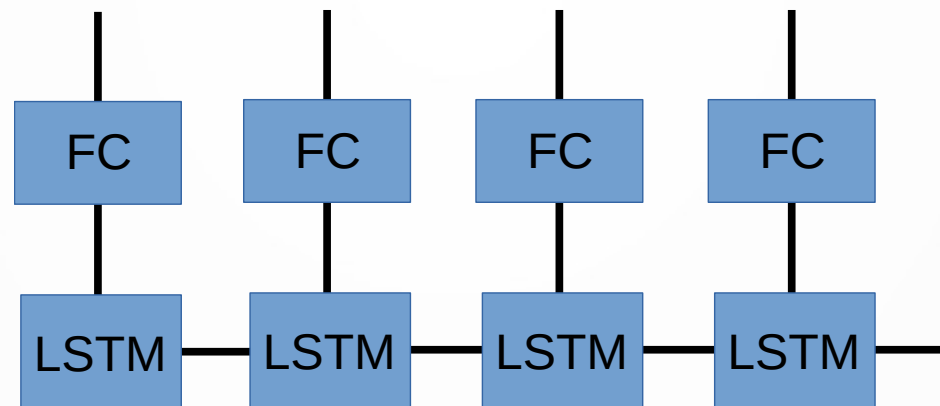
(b) Compressing the second baseline ('conv-fc'), which is dominated by fully-connected layers. 'conv-TT-fc': only the fully-connected part of the network is compressed; 'TT-conv-TT-fc': fully-connected and convolutional parts are compressed.

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



Garipov, Podoprikin, Novikov, Vetrov, arXiv:1611.03214.

TT compression of RNNs



Dataset	Model	Embedding shape	Test acc.	Compr.
IMDB	Full	25000×256	0.886	1
	TT1	$(25, 30, 40) \times (4, 8, 8)$	0.871	93
	TT2	$(10, 10, 15, 20) \times (4, 4, 4, 4)$	0.886	232
	TT3	$(5, 5, 5, 5, 6, 8) \times (2, 2, 2, 2, 4, 4)$	0.888	441
SST	Full	17200×256	0.374	1
	TT1	$(24, 25, 30) \times (4, 8, 8)$	0.415	78
	TT2	$(10, 10, 12, 15) \times (4, 4, 4, 4)$	0.411	182
	TT3	$(4, 5, 5, 5, 6, 6) \times (2, 2, 2, 2, 4, 4)$	0.399	307



Khrulkov, Hrinchuk, Mirvakhabova, Oseledets, arXiv:1901.10787.

TT compression of RNNs

TT-GRU:

$$\begin{aligned} \mathbf{r}^{[t]} &= \sigma(TTL(\mathbf{W}^r, \mathbf{x}^{[t]}) + \mathbf{U}^r \mathbf{h}^{[t-1]} + \mathbf{b}^r) \\ \mathbf{z}^{[t]} &= \sigma(TTL(\mathbf{W}^z, \mathbf{x}^{[t]}) + \mathbf{U}^z \mathbf{h}^{[t-1]} + \mathbf{b}^z) \\ \mathbf{d}^{[t]} &= \tanh(TTL(\mathbf{W}^d, \mathbf{x}^{[t]}) + \mathbf{U}^d(\mathbf{r}^{[t]} \circ \mathbf{h}^{[t-1]})) \\ \mathbf{h}^{[t]} &= (1 - \mathbf{z}^{[t]}) \circ \mathbf{h}^{[t-1]} + \mathbf{z}^{[t]} \circ \mathbf{d}^{[t]}, \end{aligned} \quad (11)$$

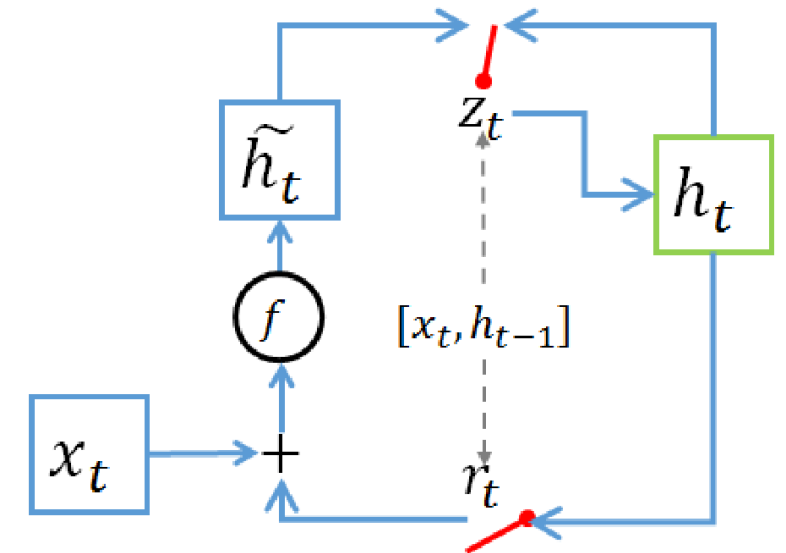



Figure 3 Gated Recurrent Unit

FC layers can be stacked together!



Yang, Krompass, Tresp, Proc. ICML 70 (pp. 3891-3900), 2017

TT compression of RNNs



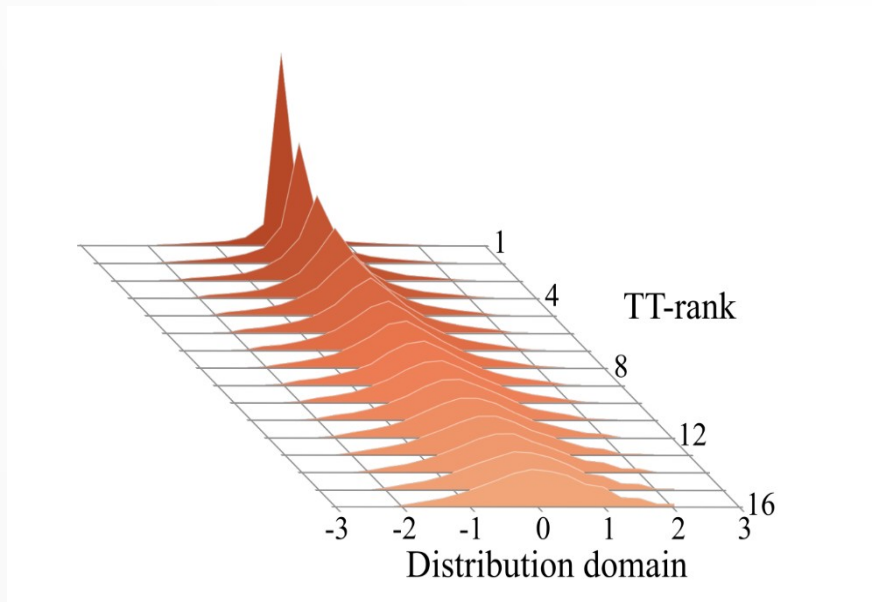
	Accuracy	# Parameters	Runtime
TT-MLP	0.427 ± 0.045	7,680	902s
GRU	0.488 ± 0.033	44,236,800	7,056s
LSTM	0.492 ± 0.026	58,982,400	8,892s
TT-GRU	0.813 ± 0.011	3,232	1,872s
TT-LSTM	0.796 ± 0.035	3,360	2,160s

Original: (Liu et al., 2009)	0.712
(Liu et al., 2013)	0.761
(Hasan & Roy-Chowdhury, 2014)	0.690
(Sharma et al., 2015)	0.850
Our best model (TT-GRU)	0.813



Trick: proper initialization

$$\mathcal{G}^{(k)}(r_{k-1}, i_k, r_k) \sim \mathcal{N}\left(0, \left(\frac{\sigma}{R}\right)^{2/N}\right)$$



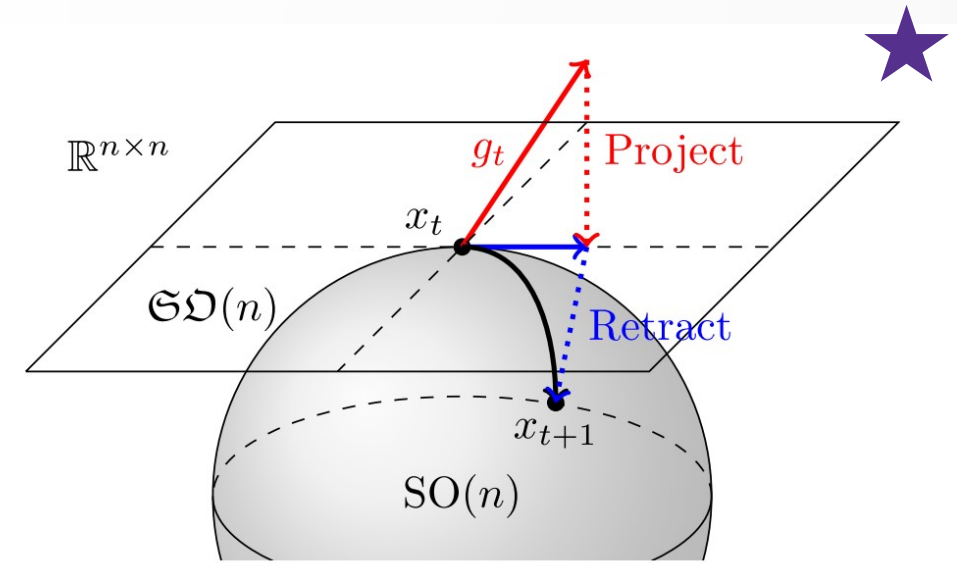
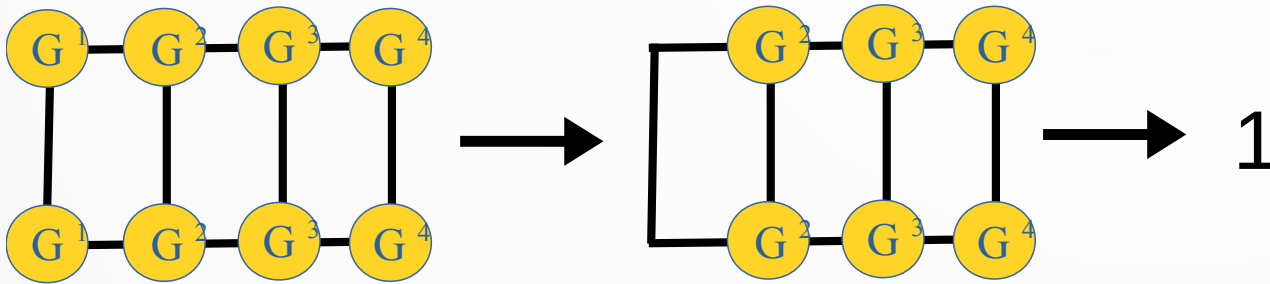
$$\sigma_g = \sqrt[4d-2]{\frac{\sigma_w^2}{\prod_{k=0}^{2d} r_k}}$$

and initialize $\mathcal{G}_k \sim \mathcal{N}(0, \sigma_g^2)$.

★ Khrulkov, Hrinchuk, Mirvakhabova, Oseledets, arXiv:1901.10787.

★ TJANDRA, SAKTI, NAKAMURA, Recurrent Neural Network
Compression based on Low-Rank Tensor Representation.

Trick: orthogonal TT

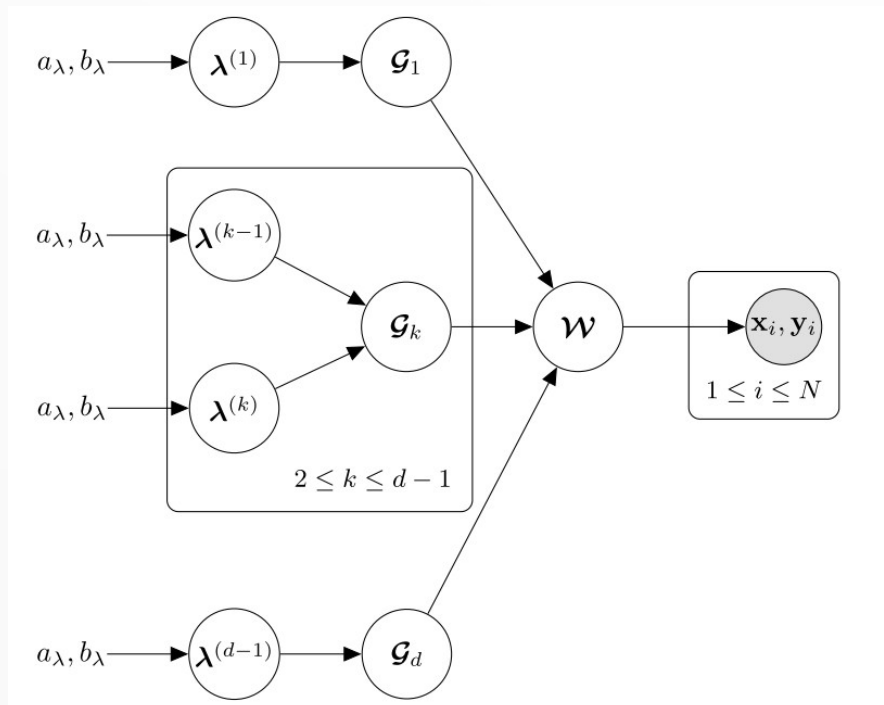


- Twice fewer parameters with almost same accuracy
- Bounded gradients

Trick: automatic rank selection

$$p(\boldsymbol{\lambda}^{(k)}) = \prod_{r_k=1}^{R_k} \text{Ga}(\lambda_{r_k}^{(k)} | a_\lambda, b_\lambda).$$

$$p(\mathcal{G}_k | \boldsymbol{\lambda}^{(k-1)}, \boldsymbol{\lambda}^{(k)}) = \prod_{r_{k-1}, m_k, j_k, r_k} \mathcal{N}(\mathcal{G}_k(r_{k-1}, m_k, j_k, r_k) | 0, \lambda_{r_{k-1}}^{(k-1)} \cdot \lambda_{r_k}^{(k)}), \quad 2 \leq k \leq d-1.$$



★ Hawkins, Zhang, 2019,
arXiv preprint
arXiv:1905.10478

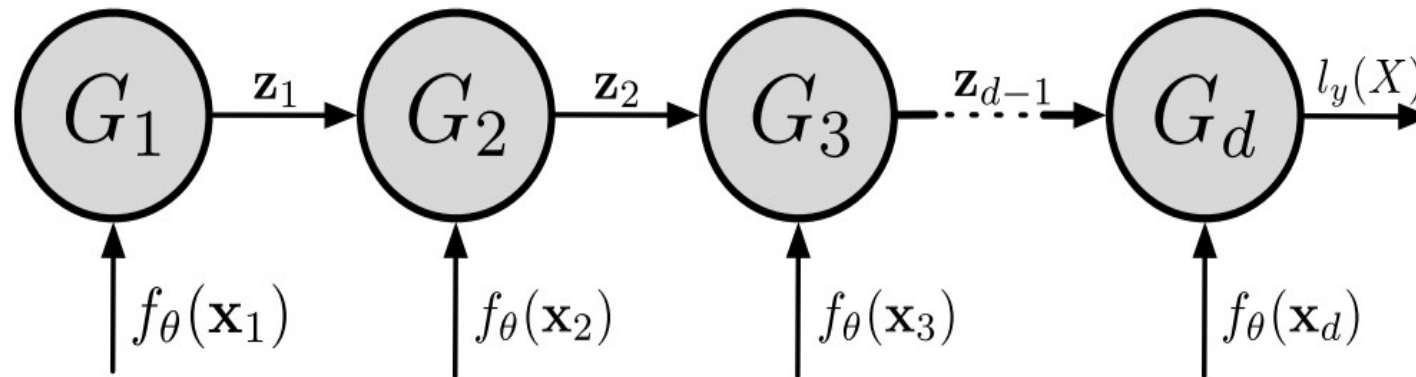
Trick: automatic rank selection

Table 1: Results of a standard Bayesian neural network (BNN), a Bayesian tensorized neural network (BTNN) with Gaussian prior, and our low-rank Bayesian tensorized neural network (LR-BTNN).

Example	Model	SVGD Param #	MAP Param #	Test LL	MAP Accuracy
MNIST (w/ 2 FC layers)	BNN	24,844,250	496,885	-0.267	92.1%
	BTNN	1,361,750	27,235	-0.188	97.7%
	LR-BTNN	181,250 (137\times)	3,625 (137\times)	-0.106	97.8%
CIFAR-10 (CNN, w/ 4 Conv +2 FC)	BNN	31,388,360	1,569,418	-0.497	89.0%
	BTNN	13,737,360	686,868	-0.463	88.8%
	LR-BTNN	1,987,520 (15.8\times)	99,376 (15.8\times)	-0.471	87.4%
CIFAR-10 (ResNet-110, w/ 109 Conv+1 FC)	BNN	34,855,240	1,742,762	-0.506	92.6%
	BTNN	12,895,800	644,790	-0.521	91.1%
	LR-BTNN	4,733,020 (7.4\times)	236,651 (7.4\times)	-0.515	90.4%



TT as RNNs with linear activations



Khrulkov, Novikov, Oseledets, ICLR 2018

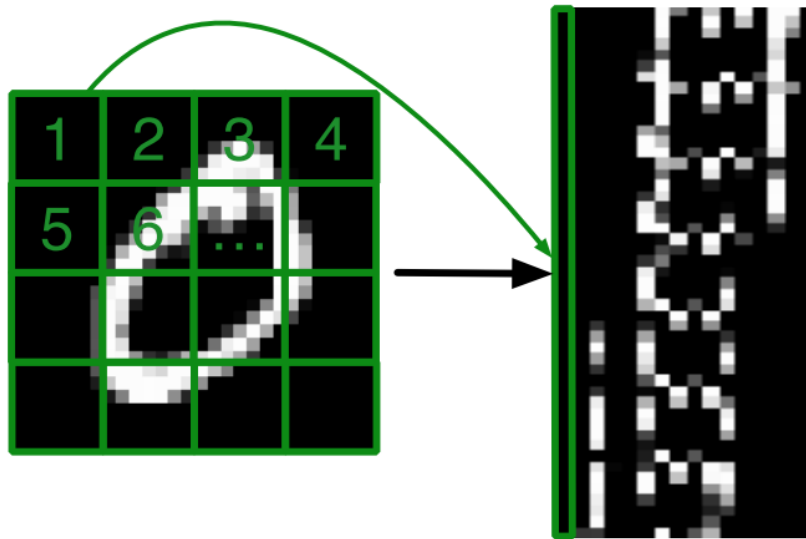


Yu, Zheng, Anandkumar, Yue, 2017, arXiv:1711.00073

Machine learning from quantum physicists

- Bradley, Stoudenmire, Terilla: Modeling Sequences with Quantum States: A Look Under the Hood, arXiv:1910.07425
- Stoudenmire, Schwab: Supervised learning with tensor networks, NIPS 2016
- Han, Wang, Fan, Wang, Zhang, 2018. Unsupervised generative modeling using matrix product states. Physical Review X, 8(3), p.031012.
- Levine, Sharir, Ziv, Shashua, 2017. On the Long-Term Memory of Deep Recurrent Networks. ArXiv:1710.09431.
- Huggins, Patil, Mitchell, Whaley, Stoudenmire: Towards quantum machine learning with tensor networks, arXiv:1803.11537
- Levine, Yakira, Cohen, Shashua, 2017. Deep learning and quantum entanglement: Fundamental connections with implications to network design. ArXiv:1704.01552.

Expressive power of TT networks



$$y = f(W^2(f(W^1(x)))) , \quad W^1 \in \mathbb{R}^{I \times h}, \quad W^2 \in \mathbb{R}^{h \times O}$$

$$y = f(G^d(x_d, f(G^{d-1}(x_{d-1}, \dots f(G^0(x_0)))))),$$

$$G^k \in \mathbb{R}^{r_{k-1} \times i_k \times r_k},$$

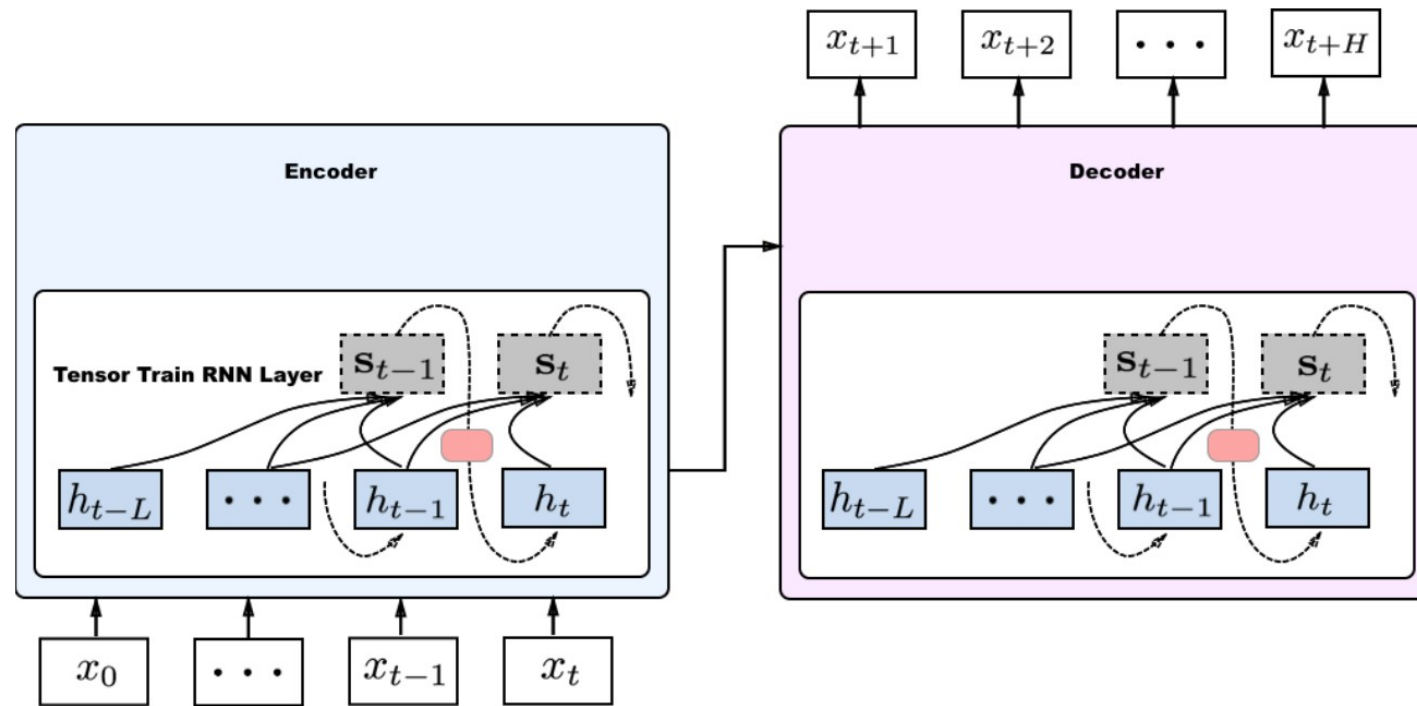
$$\prod i_k = I, \quad r_d = O$$

$$r \sim \log(h) \quad \text{typically}$$



Non-linear dynamics

$$[\mathbf{h}_t]_{\alpha} = f(W_{\alpha}^{hx} \mathbf{x}_t + \sum \mathcal{W}_{\alpha i_1 \dots i_P} \mathbf{s}_{t-1; i_1} \otimes \dots \otimes \mathbf{s}_{t-1; i_P})$$



Yu, Zheng, Anandkumar, Yue, 2017, arXiv:1711.00073

Non-linear dynamics

Theorem 4.2. *Let the state transition function $f \in \mathcal{H}_\mu^k$ be a Hölder continuous function defined on a input domain $\mathcal{I} = I_1 \times \cdots \times I_d$, with bounded derivatives up to order k and finite Fourier magnitude distribution C_f . Then a single layer Tensor Train RNN can approximate f with an estimation error of ϵ using with h hidden units:*

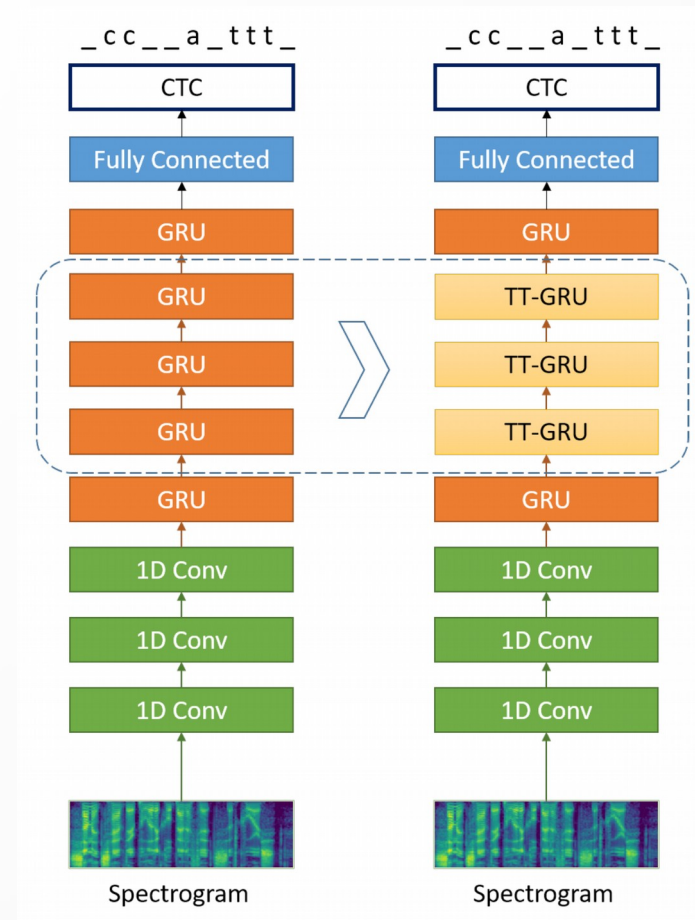
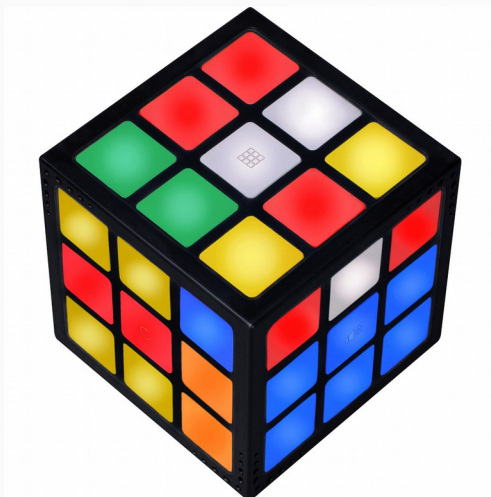
$$h \leq \frac{C_f^2}{\epsilon} (d-1) \frac{(r+1)^{-(k-1)}}{(k-1)} + \frac{C_f^2}{\epsilon} C(k) p^{-k} \quad (14)$$

where $C_f = \int |\omega|_1 |\hat{f}(\omega)| d\omega$, d is the size of the state space, r is the tensor-train rank and p is the degree of high-order polynomials i.e., the order of tensor.



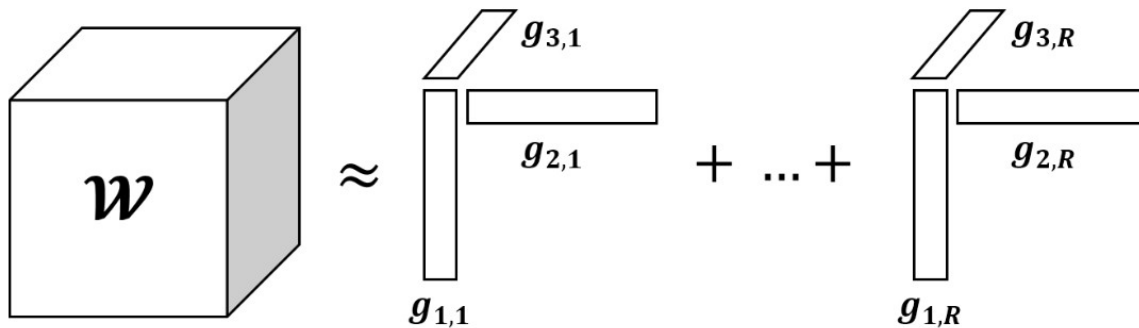
General remark: locality

- “Inputs” to each “core” should have bounded derivatives
- The “edges” between “inputs” should be sharp

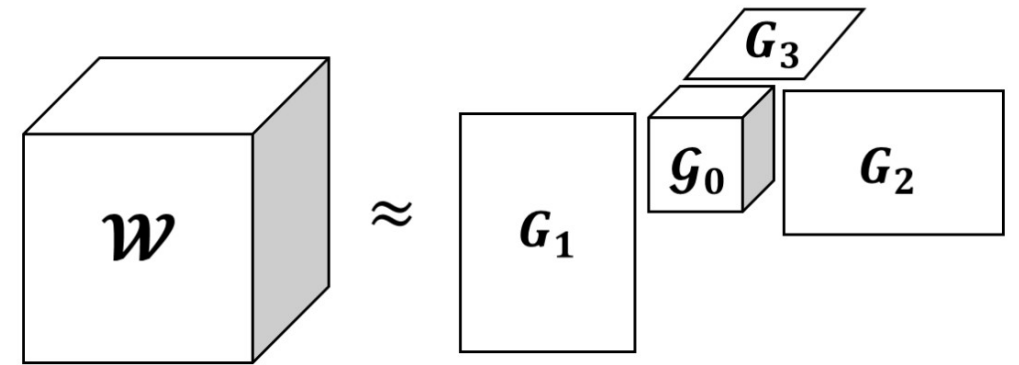


JANDRA, SAKTI, NAKAMURA, Recurrent Neural Network Compression based on Low-Rank Tensor Representation.

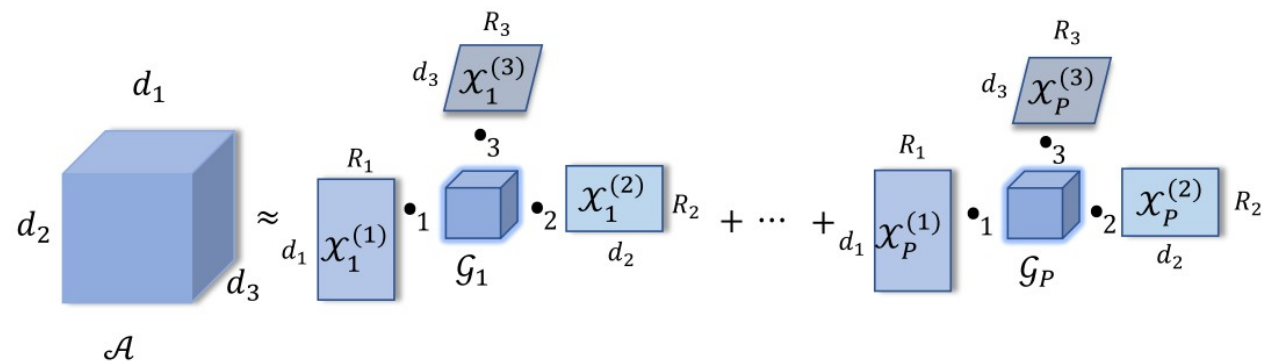
Other decompositions



Canonical Polyadic decomposition



Tucker decomposition



Block-term decomposition

Speeding up inference with decomposition

- **MUSCO** - framework for model compression using tensor decompositions (PyTorch)
- **Distiller** - package for compression using pruning and low-precision arithmetic (PyTorch)
- **MorphNet** - framework for neural networks architecture learning (TensorFlow)
- **Mayo** - deep learning framework with fine- and coarse-grained pruning, network slimming, and quantization methods
- **PocketFlow** - framework for model pruning, sparcification, quantization (TensorFlow implementation)
- **Keras compressor** - compression using low-rank approximations, SVD for matrices, Tucker for tensors.
- **Caffe compressor** K-means based quantization

Summary

- 1) Tensor decompositions are neural networks with linear/multiplicative activations**
- 2) Automatic ranks for TT were formulated only recently. Analogous results exist for Tucker**
- 3) Choosing network structure is non-trivial (and it is my research topic)**

感谢您的关注

Thank you for your attention!