ACML 2020 TUTORIAL

Tensor Networks in Machine Learning: Recent Advances and Frontiers

Qibin Zhao

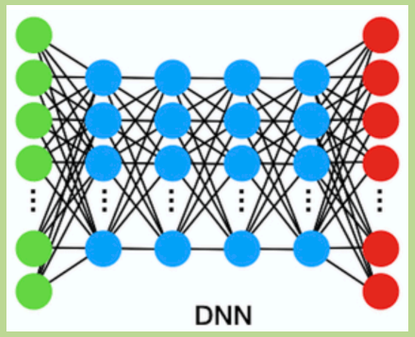
Tensor Learning Team
RIKEN AIP
https://qibinzhao.github.io

Success of Machine Learning

Big Data

Computation

Model & Algorithm



Tensor Networks

Mathematics

Quantum Physics

Role of Tensor and Tensor Networks

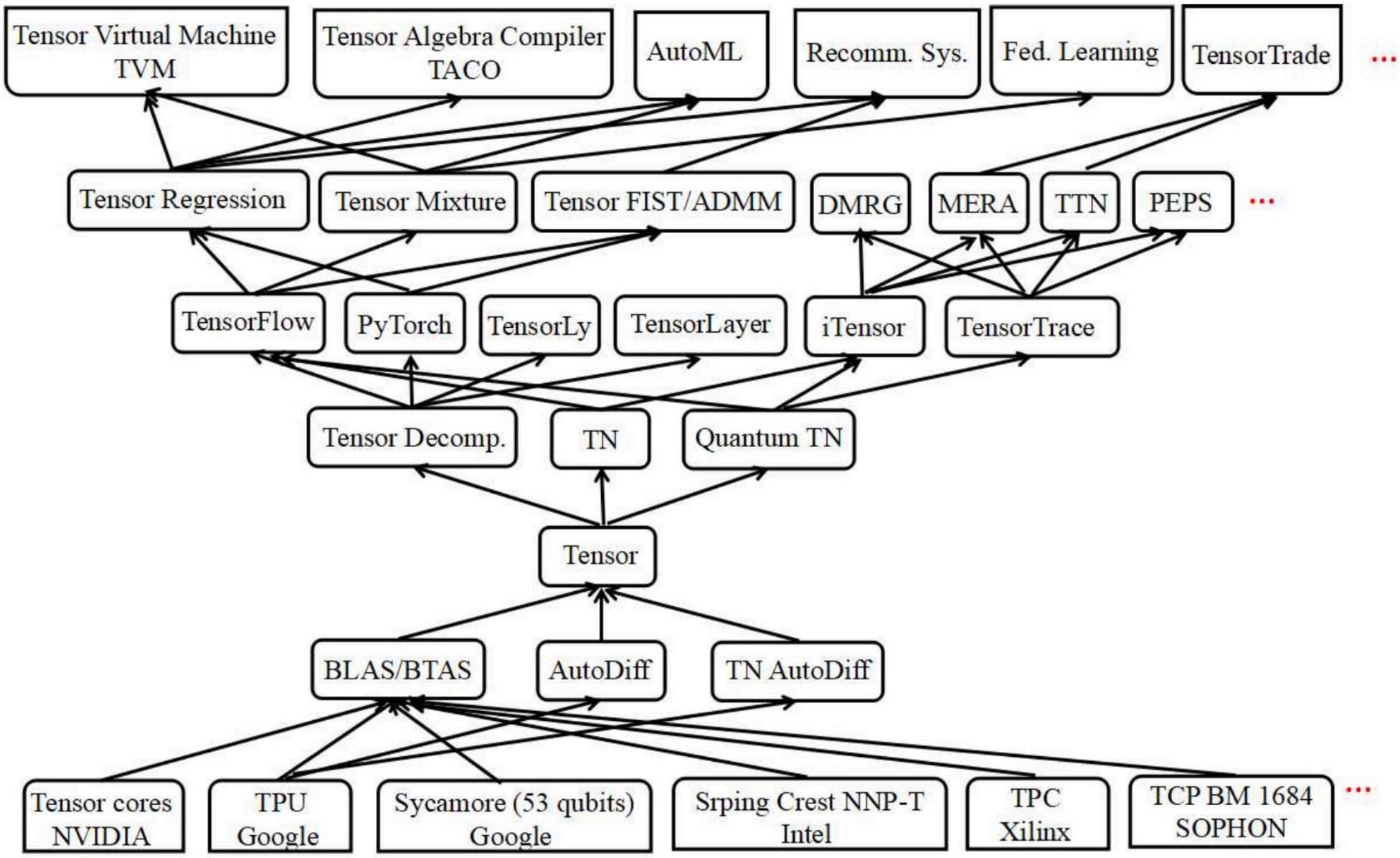


Figure 1: The hourglass architecture of tensor, tensor networks and quantum tensor networks in machine learning.

(Liu et al. IJCAI Workshop 2020)

Agenda

This tutorial aims to introduce tensor networks fundamental, and to overview recent progress of tensor networks applied to machine learning.

Part I

Tensor Methods for Data Representation

Part II

Tensor Networks in Deep Learning Modeling

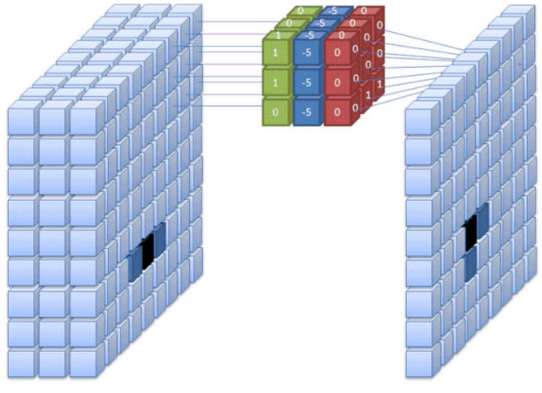
Part III

Frontiers and Future Trends

Part 1:

Tensor Networks for Data Representation

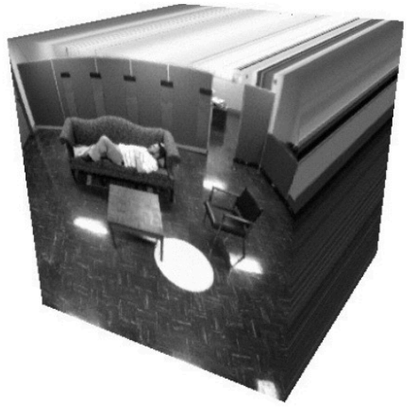
Multiway Structured Data



Feature Maps in CNN

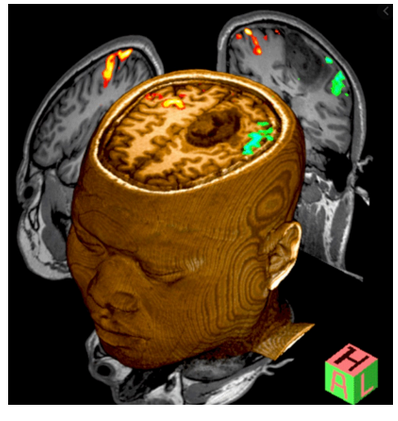
(Spatial x Spatial x Filter)

WIKIMEDIA COMMONS



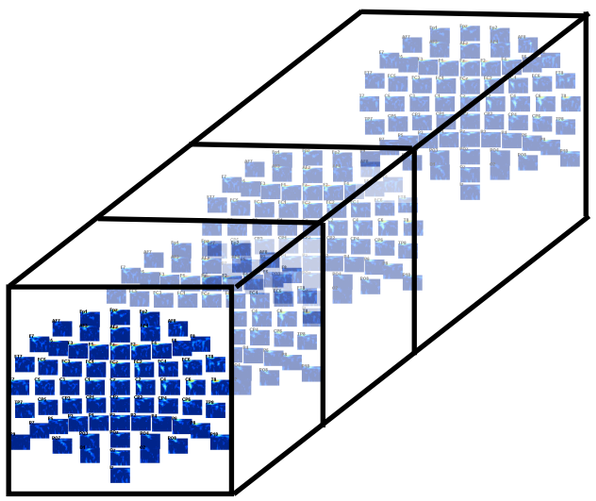
Video Data

(Spatial x Spatial x Time)



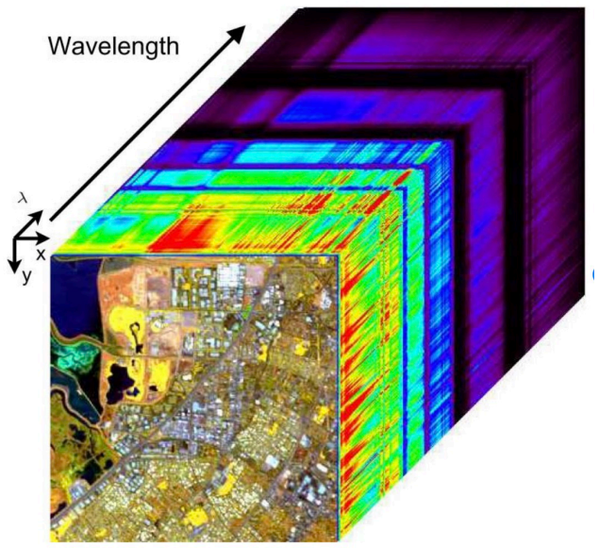
fMRI Data

(Spatial x Spatial x Spatial)



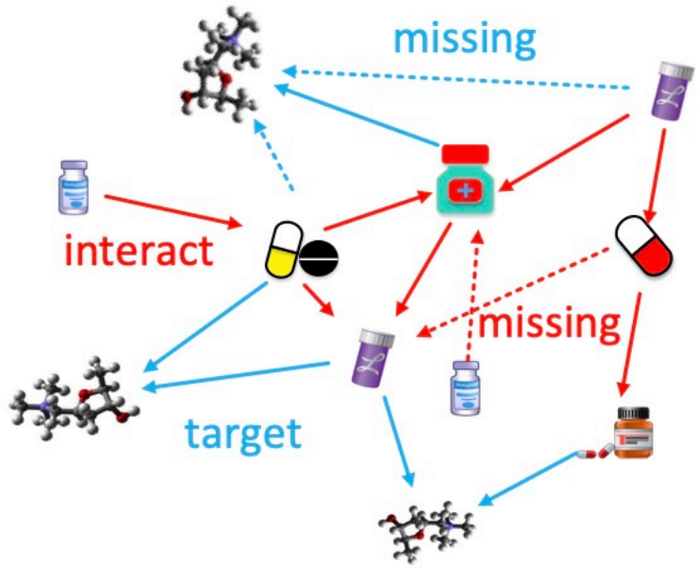
EEG

(Spatial x Time x Frequency)



Hyperspectral Image

(Spatial x Spatial x Spectral)

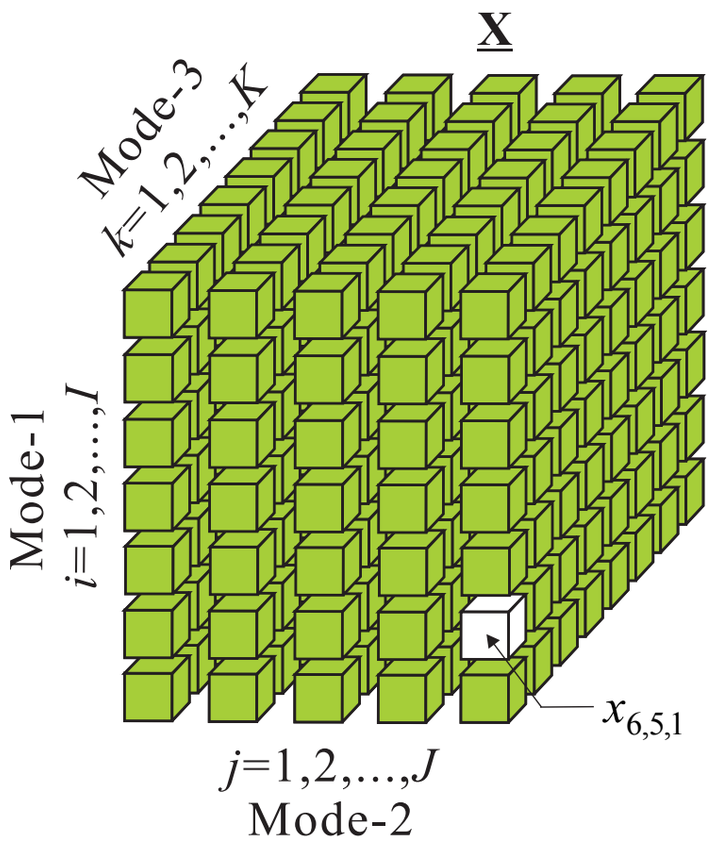


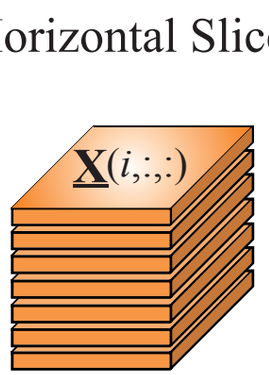
Medical Knowledge Graph

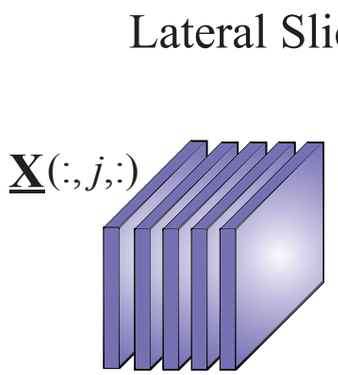
(Entity x Entity x Relation)

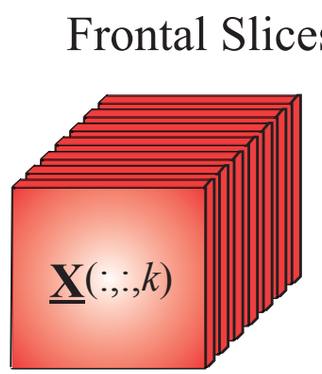
(Wang et al., 2017)

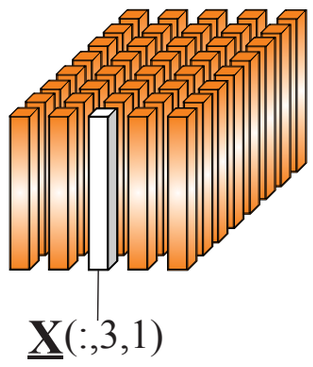
Tensor: Generalization of Vector and Matrix

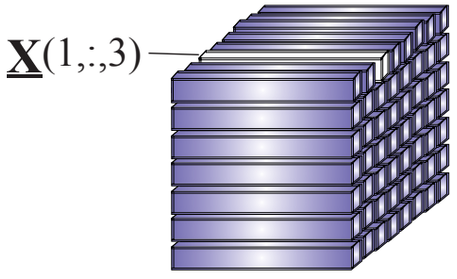


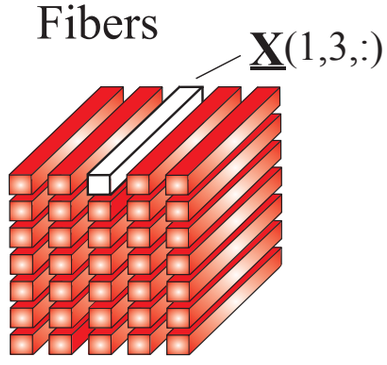




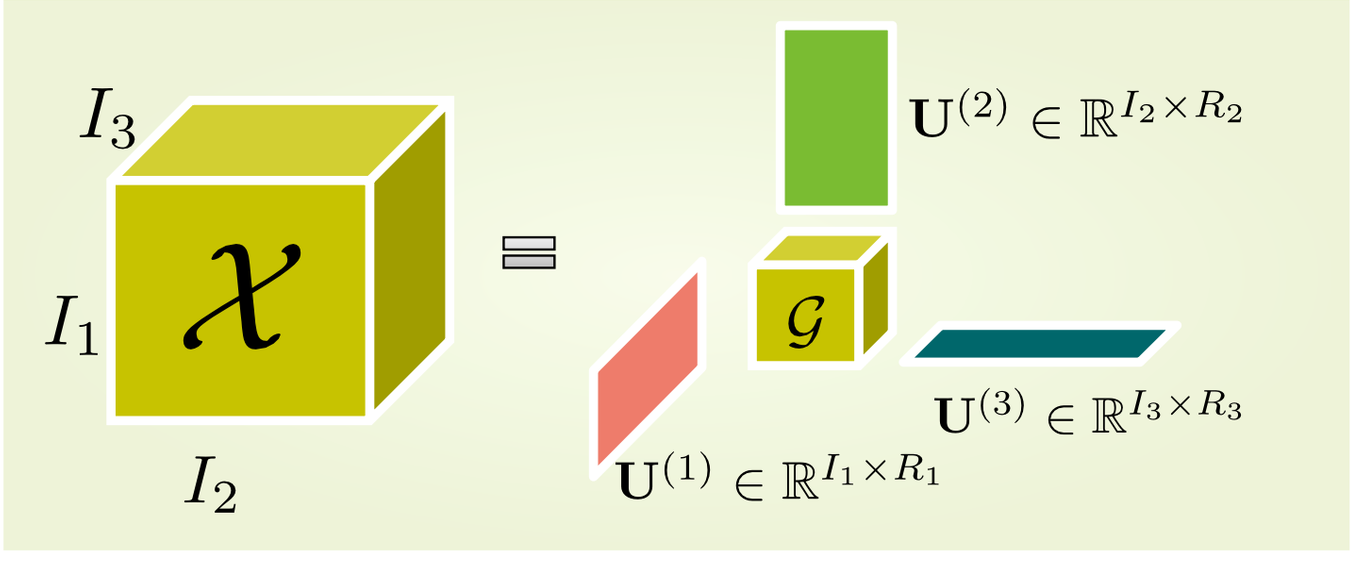








Tensor Decomposition



Tucker Decomposition (HOSVD):

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_M \mathbf{U}^{(M)},$$

$$\text{rank}_{ML}(\underline{\mathbf{X}}) = \{\text{rank}(\mathbf{X}_{(1)}), \text{rank}(\mathbf{X}_{(2)}), \ldots, \text{rank}(\mathbf{X}_{(N)})\},$$

Canonical Polyadic Decomposition (CPD) :

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \, \mathbf{u}_r^{(1)} \circ \mathbf{u}_r^{(2)} \circ \cdots \circ \mathbf{u}_r^{(M)}, \qquad R_1 \leqslant \text{rank}_{CP}(\underline{\mathbf{X}}) \leqslant R_2 R_3 \cdots R_N.$$

Tensor Methods

- HOSVD: A method for tensor denosing and dimension reduction
- Tensor completion
- Tensor imaging and sensing
- Multi-linear blind source separation (BSS)

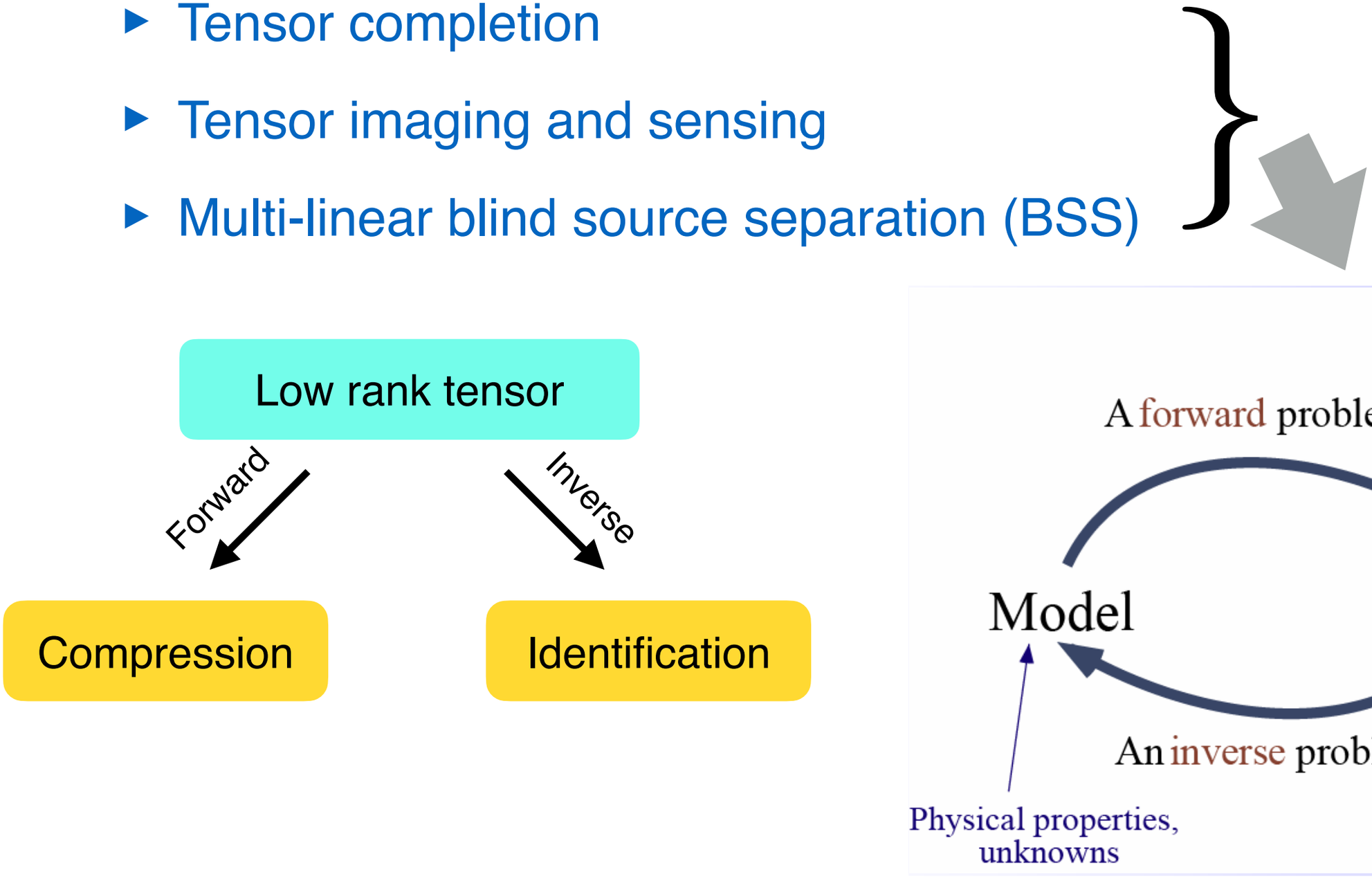
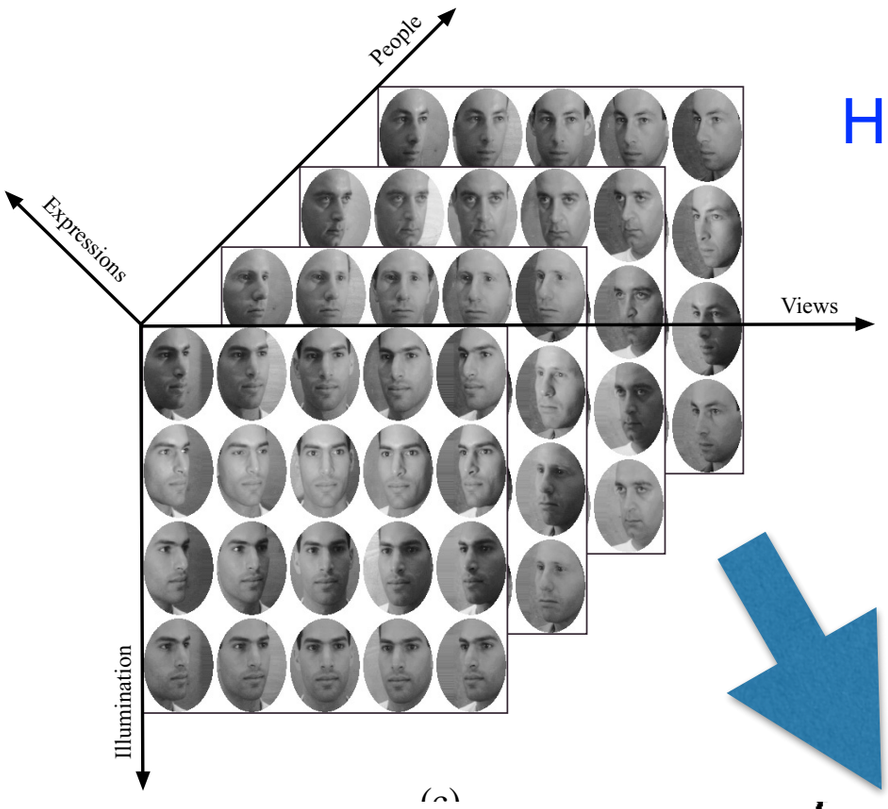


fig: (Malcolm Sambridge, 2016)

Multilinear Dimension Reduction

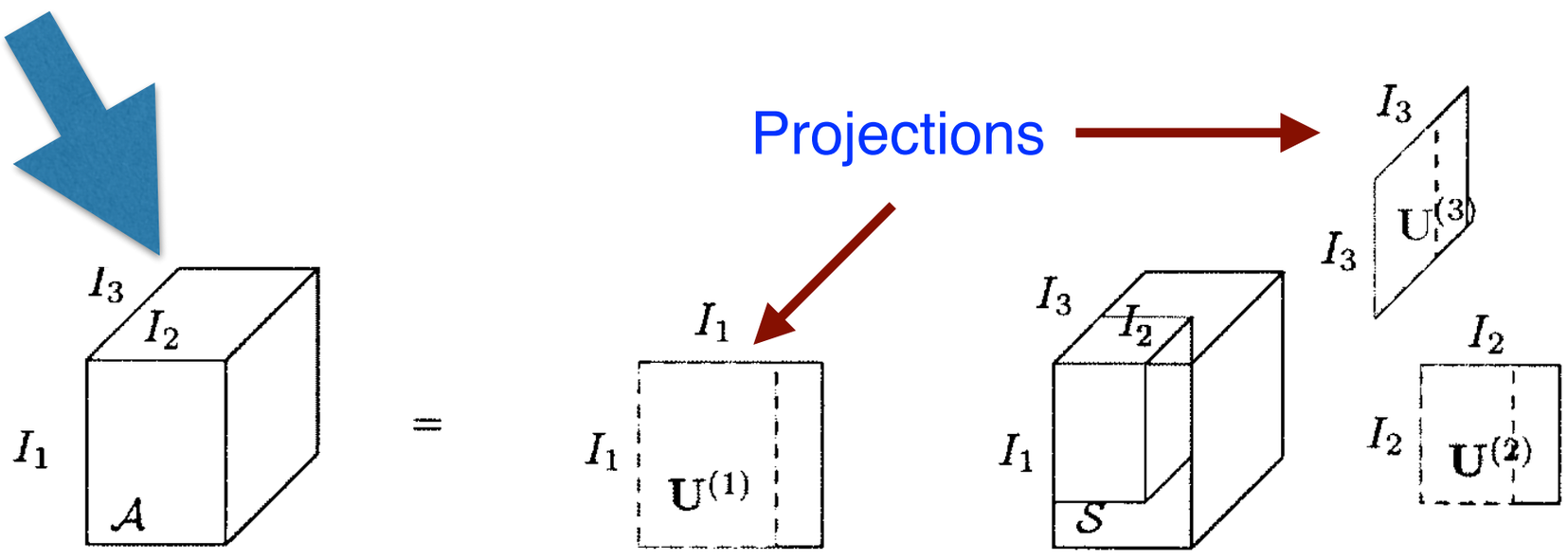
- Multi-linear extension of SVD, PCA and ICA



Higher-order singular value decomposition

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}$$

Orthonormal

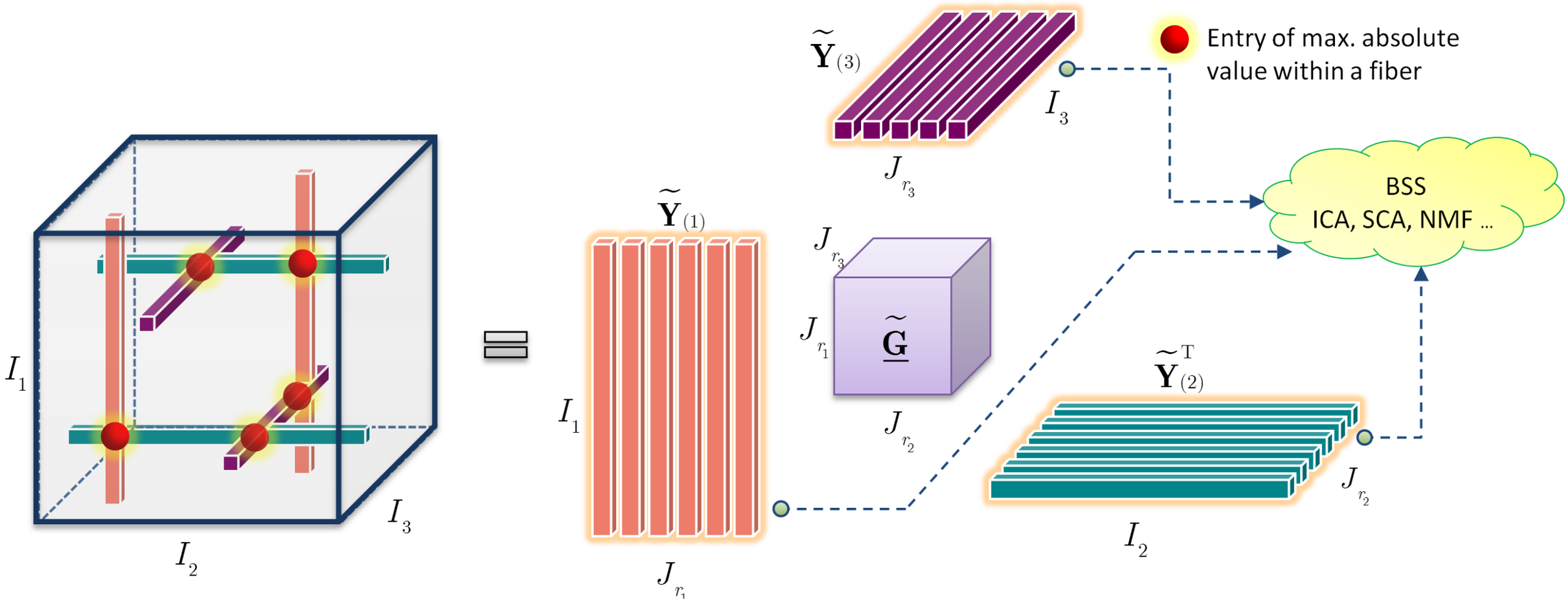


Multilinear Analysis of Image Ensembles: TensorFaces (Vasilescu et al., ECCV 2002)

A Multi-linear Singular Value Decomposition (Lathauwer et al , SIAM J. Matrix Anal. Appl., 2000)

Tensor Blind Source Separation

- Blind source separation (BSS) is applied to multiple latent factors



G. Zhou and A. Cichocki, “Fast and unique Tucker decompositions via multiway blind source separation”, 2012

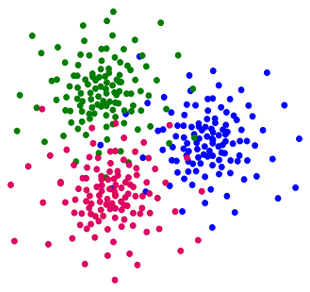
Tensor Decomposition for Efficient Learning Algorithm

[A. Anandkumar, JMLR 2014]

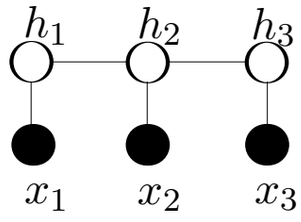
Unsupervised learning

- Estimate moments $\mathbb{E}[x_i]$, $\mathbb{E}[x_i \otimes x_j]$, $\mathbb{E}[x_i \otimes x_j \otimes x_k]$
- Learning through higher order moments
- Tensor decomposition via power method

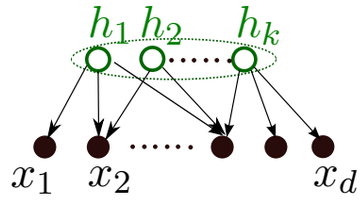
GMM



HMM

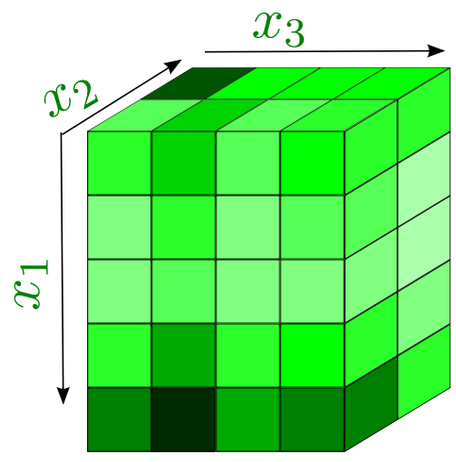


ICA



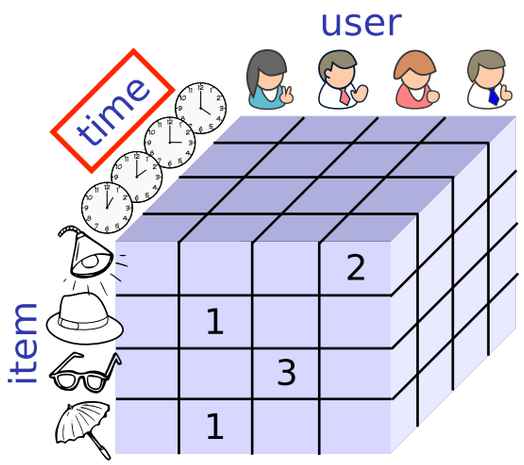
Higher order moments

- $T = \mathbb{E}[x_1 \otimes x_2 \otimes x_3]$.
- $T(i, j, k) = \mathbb{E}[x_1(i)x_2(j)x_3(k)]$

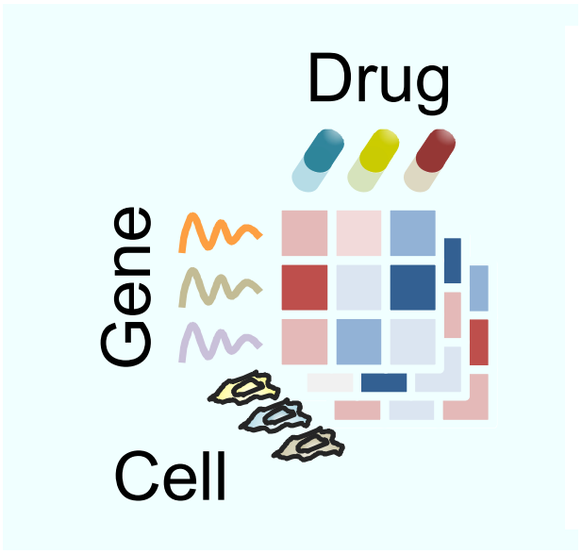


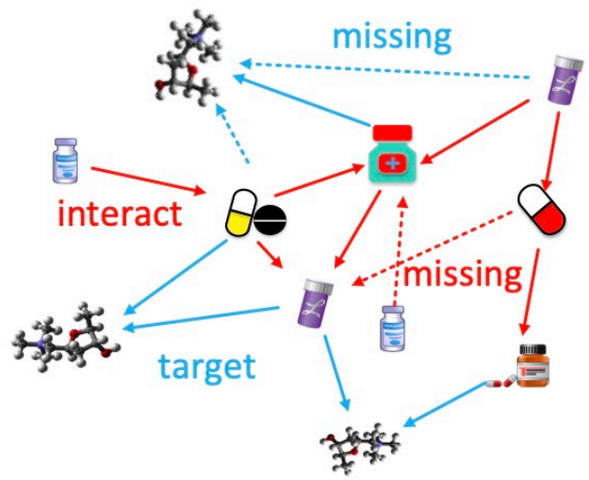
Missing Values Problems in ML

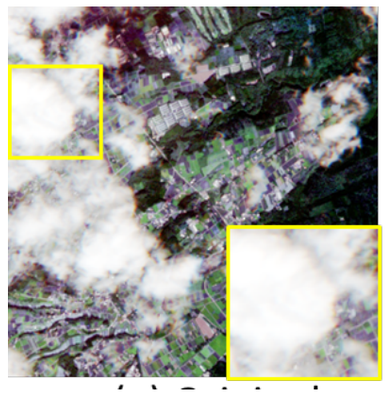
- Recommender system (user x item x time), Knowledge graph prediction
- Image inpainting/denoising, drug repositioning

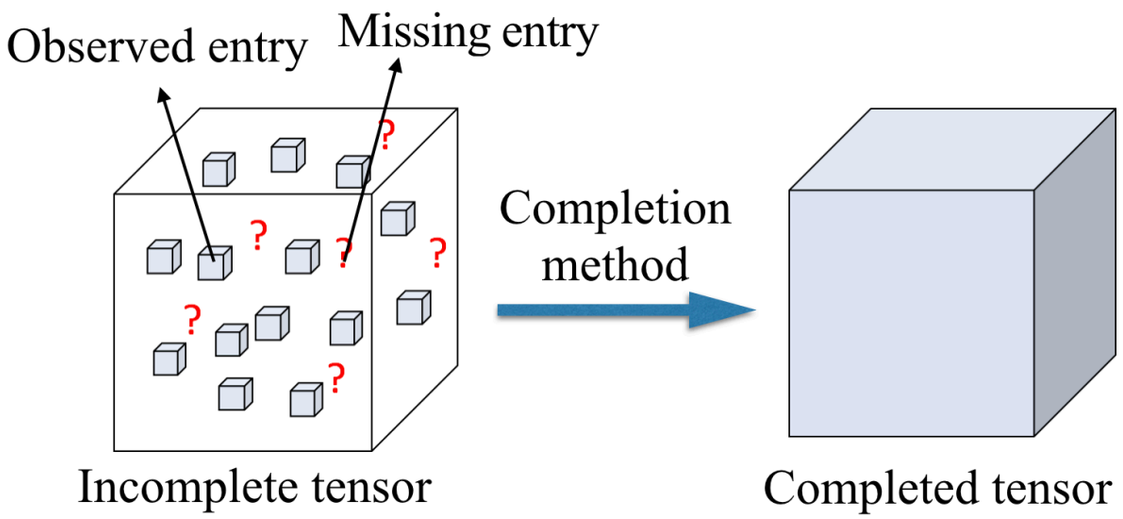


3 order tensor









Tensor Completion

Problem formulation (low-rank approximation)

$$\min_{\mathcal{X}} \mathrm{Rank}(\mathcal{X}),\ s.t.\ \mathcal{Y}_\Omega = \mathcal{X}_\Omega$$

Main Approaches:

- Low-rank approximation via convex optimization (high computation cost)

$$\min_{\boldsymbol{\mathcal{X}}} \|\Omega * (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}})\|_F^2 + \lambda \|\boldsymbol{\mathcal{X}}\|_*$$

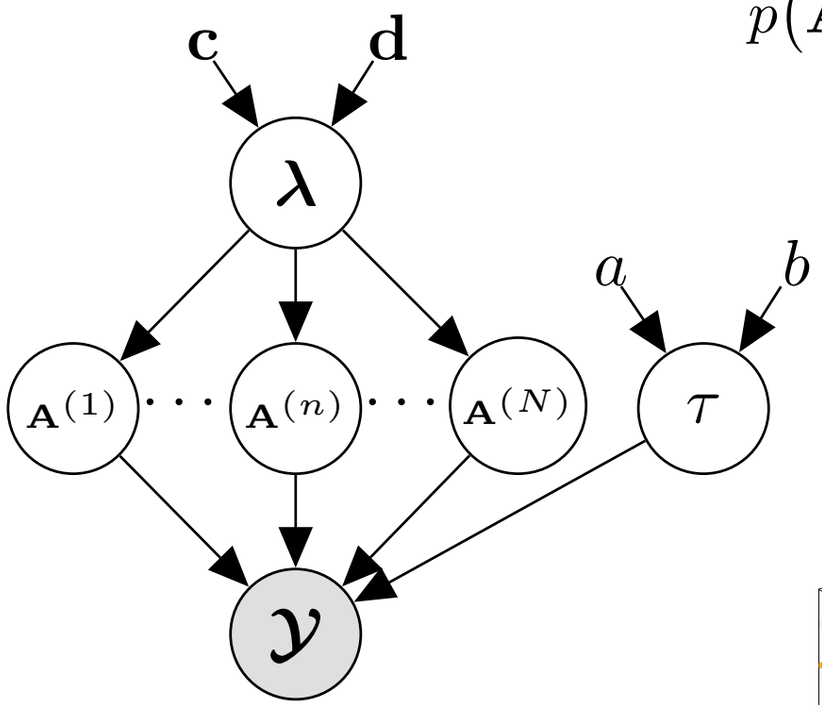
- Decomposition based approach (rank selection problem)

$$\min_{\boldsymbol{\mathcal{G}}} \|\Omega * (\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}})\|_F^2, \quad s.t. \quad \boldsymbol{\mathcal{X}} = \mathrm{TN}(\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_d).$$

- Combining convex optimization with decomposition approach
- Incorporating priori knowledge, such as auxiliary information from other data, smoothness, sparsity, non-negativity and etc.

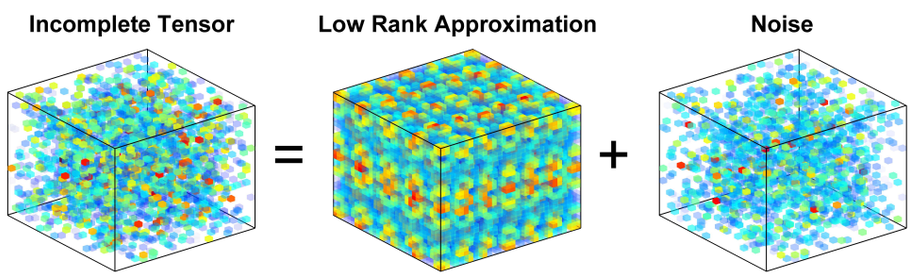
Learning Optimal Tensor Rank

- Probabilistic modeling of tensor decomposition
- Group sparsity prior imposed on factor matrices
- Bayesian inference for posteriors of model parameters



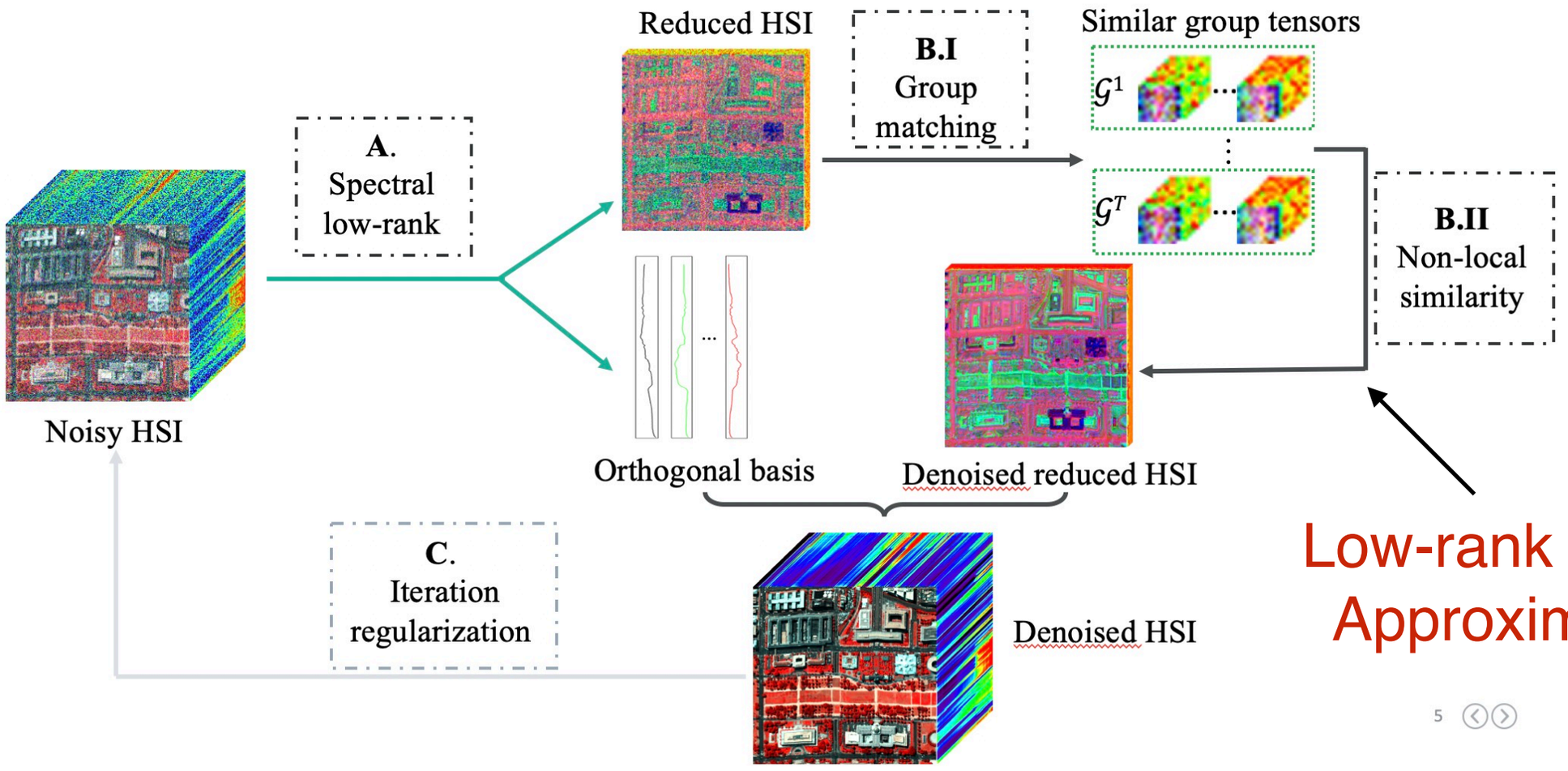
$$p\left(\mathbf{A}^{(n)}|\boldsymbol{\lambda}\right) = \prod_{i_n=1}^{I_n} \mathcal{N}\left(\mathbf{a}_{i_n}^{(n)}|\mathbf{0}, \mathbf{\Lambda}^{-1}\right), \forall n \in [1, N],$$

$$p(\boldsymbol{\lambda}) = \prod_{r=1}^{R} \mathrm{Ga}(\lambda_r|c_0^r, d_0^r),$$



(Zhao et al, TPAMI 2015)

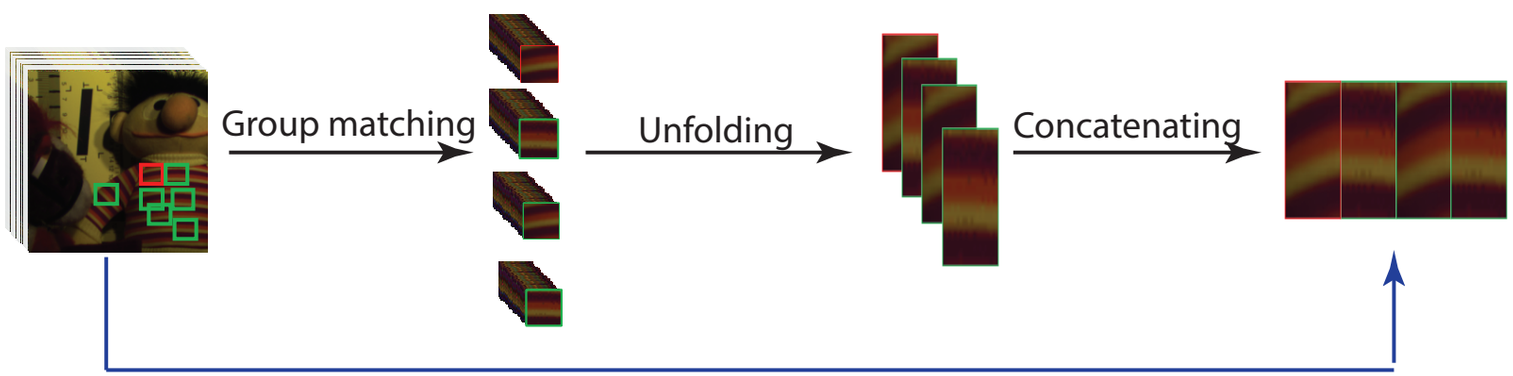
Applications to Hyperspectral Image Denoising



Non-Local Meets Global: An Integrated Paradigm for Hyperspectral Denoising (He et al., CVPR 2019)

Tensor Completion Under Multiple Transformation

- Image is not always globally low-rank
- Non-local similar patches are often low-rank
- Lack of theoretical analysis



A linear transformation from high-rank to low-rank structure

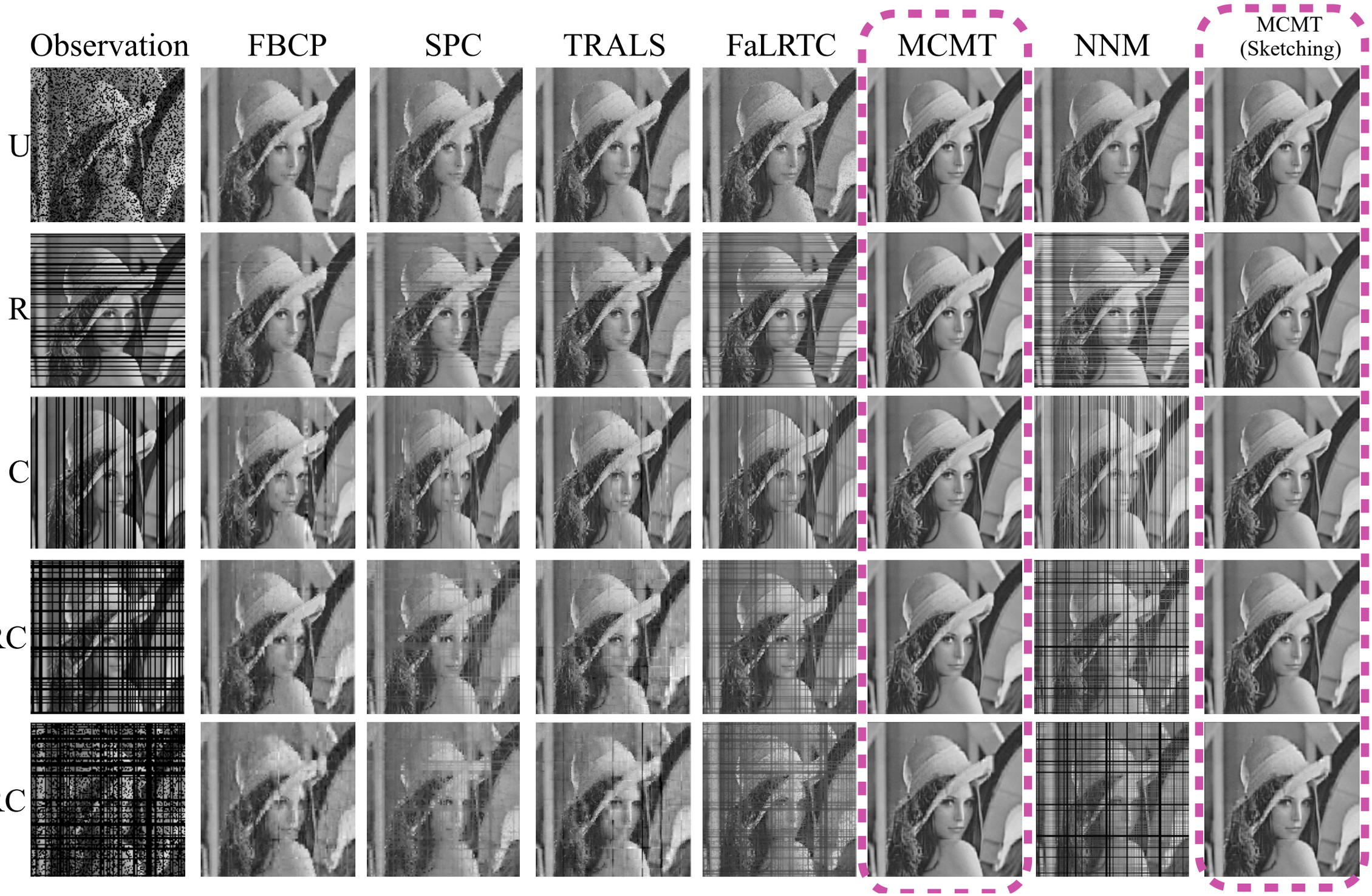
Low-rankness under linear transformation

$$\min_{\mathbf{X}\in\mathbb{R}^{m_1\times m_2}} \|\mathcal{Q}(\mathbf{X})\|_* \quad s.t.\ \|\mathcal{P}_\Omega(\mathbf{X}) - \mathcal{P}_\Omega(\mathbf{Y})\|_F \le \delta,$$

Linear transformation

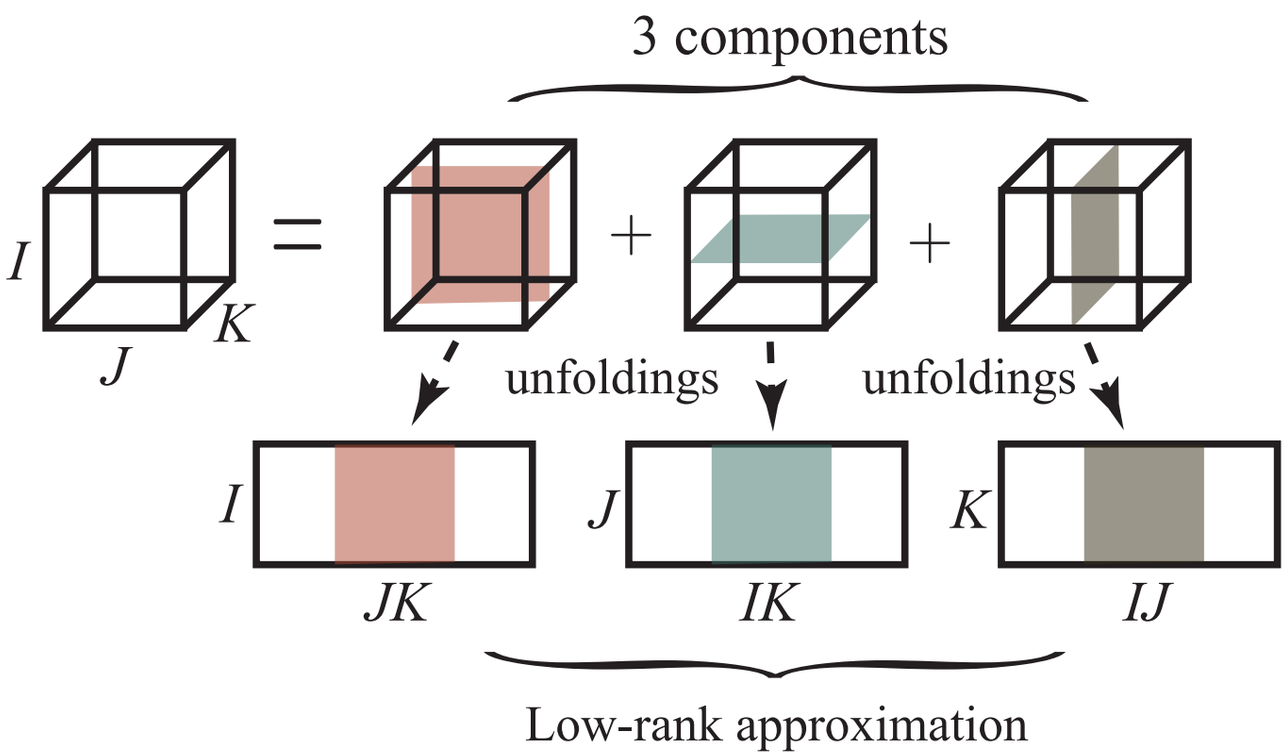
Guaranteed Matrix Completion under Multiple Linear Transformations (Li et al, CVPR 2019)

Illustrative Experiment



Latent Convex Tensor Decomposition

- Most tensor decompositions are non-convex
- Latent convex tensor decomposition is to optimize low-rank components



- Latent convex tensor decomposition cannot ensure exact recovery

(Tomioka, Hayashi, Kashima, Low-rank tensor via convex optimization)

(Tomioka, Suzuki, Convex tensor decomposition via structured schatten norm regularization)

Reshuffled Tensor Decomposition

- unfolding *v.s.* reshuffling.

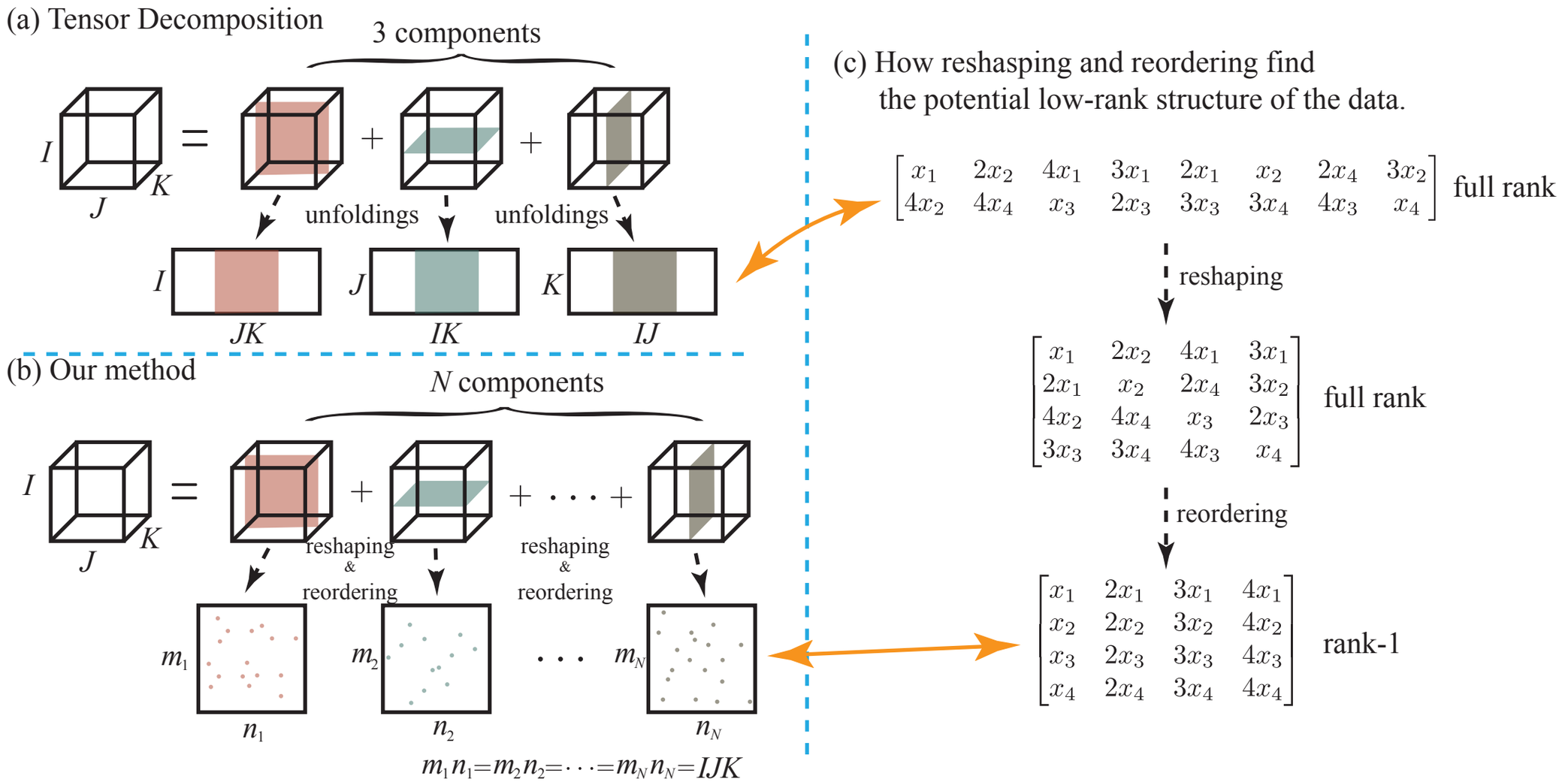


Figure Illustration of the difference with the "latent" model.

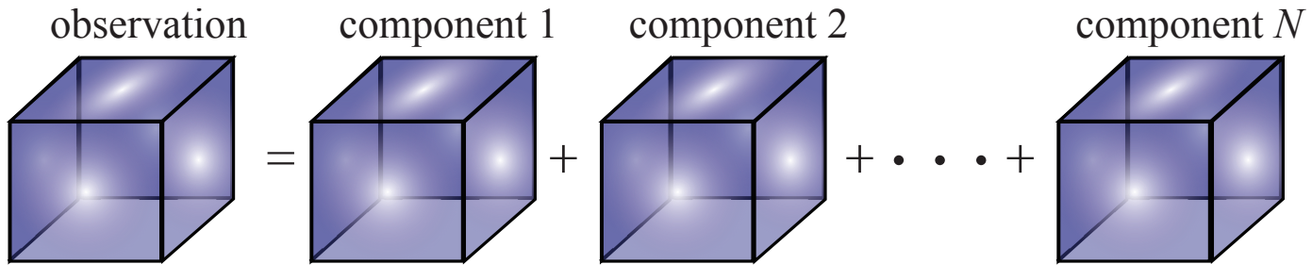
Beyond Unfolding: Exact Recovery of Latent Convex Tensor Decomposition under Reshuffling (Li et al, AAAI 2020)

Reshuffled Tensor Decomposition

(Li et al, AAAI 2020)

- Formulation

$$\mathcal{X} = R_1(\mathbf{A}_1) + R_2(\mathbf{A}_2) + \cdots + R_N(\mathbf{A}_N).$$



- Optimization objective: $\min_{\mathbf{A}_i,\, i\in[N]} \sum_{i=1}^{N} \|\mathbf{A}_i\|_*, \quad s.t.,\ \mathcal{X} = \sum_{i=1}^{N} R_i(\mathbf{A}_i),$

- Theoretical guarantee

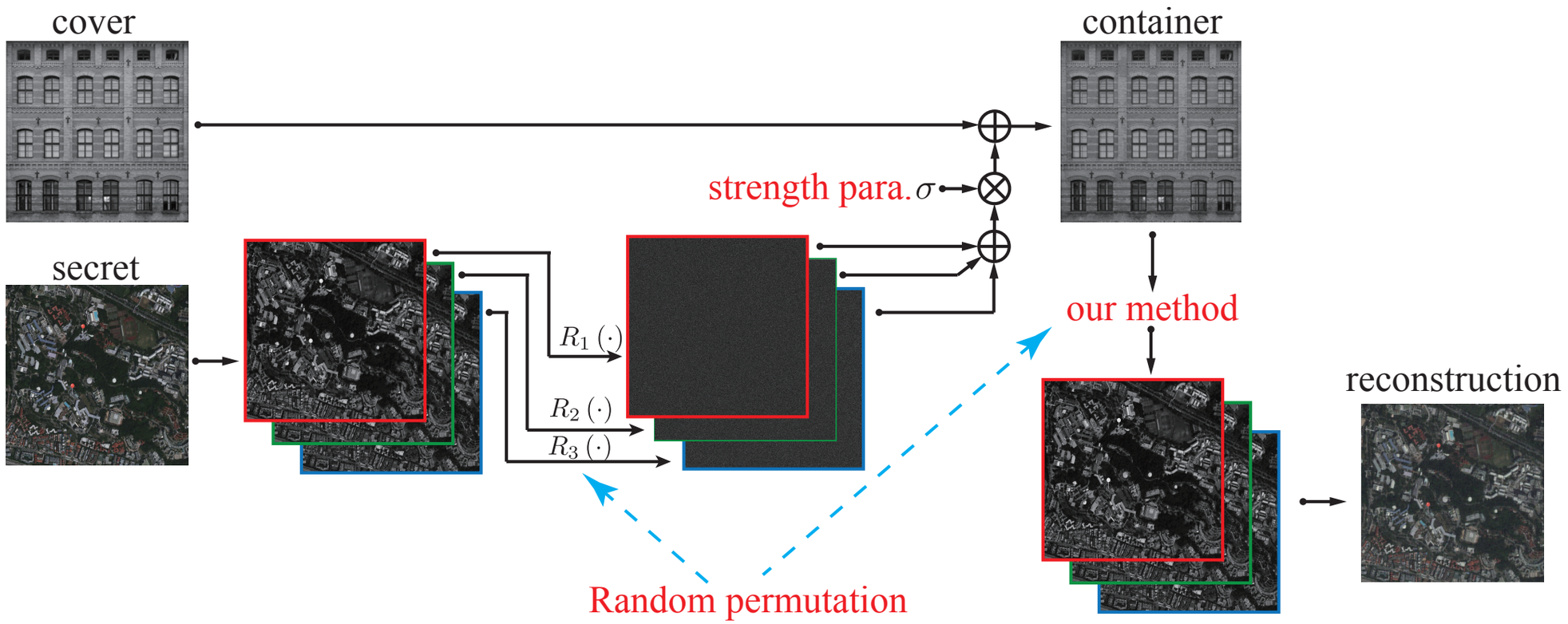
Theorem (Exact-Recovery Condition)

The estimated $\hat{\mathbf{A}}_i$, obtained by Reshuffled-TD, are equal to the true $\mathbf{A}_i^*$ for all i, when

$$\max_{i=1,\ldots,N} \mu_i(\mathbf{A}_i^\star) < \frac{1}{3N-2}, \tag{7}$$

where N denotes the number of the components.

Reshuffled Tensor Decomposition



Application: Image steganography

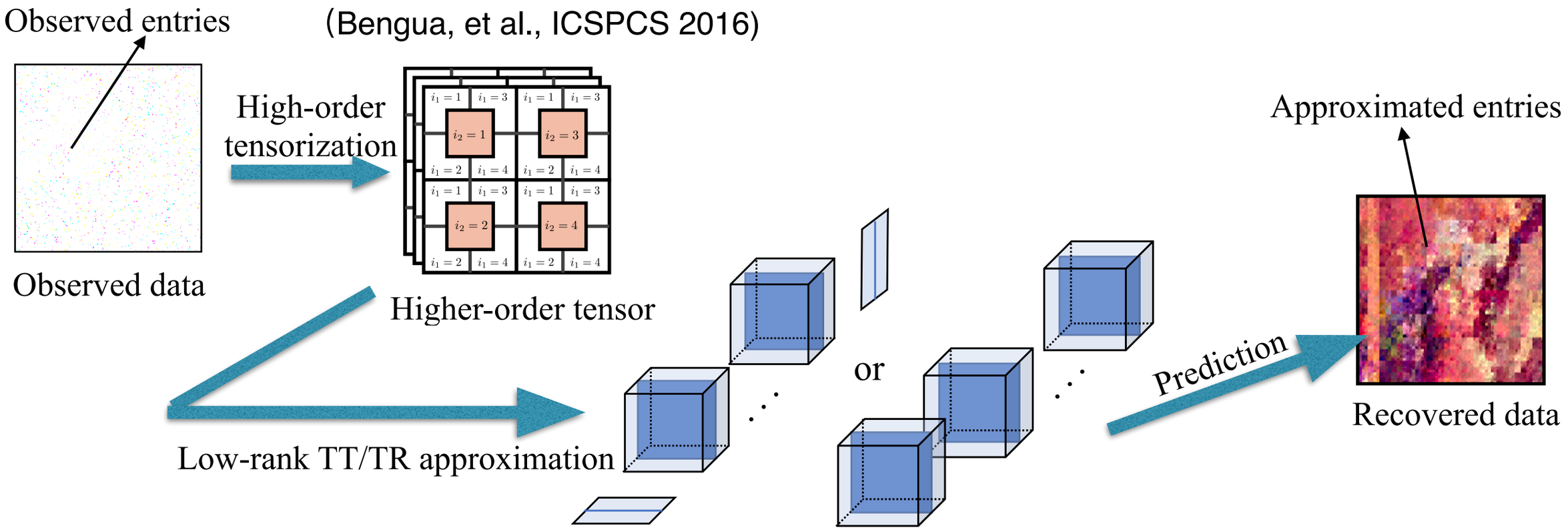
Convex Formulation of TT Rank Minimization

- Schatten TT norm: $\|\boldsymbol{\mathcal{X}}\|_{s,T} := \frac{1}{d-1}\sum_{n=1}^{d-1}\left\|\boldsymbol{X}_{[n]}\right\|_*$.
- Optimizaiton problem (TT-ADMM): $\min_{\boldsymbol{\mathcal{X}}}\|\boldsymbol{\mathcal{Y}}-\boldsymbol{\mathcal{X}}\|+\lambda\|\boldsymbol{\mathcal{X}}\|_{s,T}$.
- Preserve TT-format (TT-RAM): $\min_{\boldsymbol{\mathcal{G}}}\|\boldsymbol{\mathcal{Y}}-\mathrm{TT}(\boldsymbol{\mathcal{G}})\|+\lambda\|\mathrm{TT}(\boldsymbol{\mathcal{G}})\|_{s,T}$.

Method	Global Convergence	Rank Adaptivity	Time Complexity	Space Complexity	Statistical Bounds
TCAM-TT[30]			$O(nIKR^4)$	$O(I^K)$	
ADF for TT[6]		(search)	$O(KIR^3+nKR^2)$	$O(I^K)$	
SiLRTC-TT[22]		✓	$O(I^{3K/2})$	$O(KI^K)$	
TT-ADMM	✓	✓	$O(KI^{3K/2})$	$O(I^K)$	✓
TT-RAM		✓	$O((n+KD^2)KI^2R^4)$	$O(n+KI^2R^4)$	✓

On tensor train rank minimization: Statistical efficiency and scalable algorithm (Imaizumi, et al., NeurIPS 2017)

Tensor Ring with Low-rank Cores



$$\min_{\mathcal{G}} \quad \left\| \Omega * (\mathcal{Y} - \hat{\mathcal{Y}}) \right\|_F^2 + \lambda \sum_{n=1}^{d} \sum_{i=1}^{3} \left\| \boldsymbol{G}_{(i)}^{(n)} \right\|_*, \quad s.t. \quad \hat{\mathcal{Y}} = \mathrm{TR}(\mathcal{G}^{(1)}, \cdots, \mathcal{G}^{(d)})$$

- High computational efficiency, rank-robustness, theoretical support

L. Yuan et al. , “Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion ”, (AAAI 2019)

What Is Tensor Network?

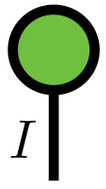
- A powerful tool to describe strongly entangled quantum many-body systems in physics

- Approximation of N-order tensor as contractions of O(N) smaller tensors

Tensor Network Diagram

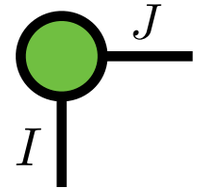
(1×1)

Scalar



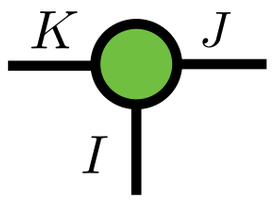
$(I \times 1)$

Vector



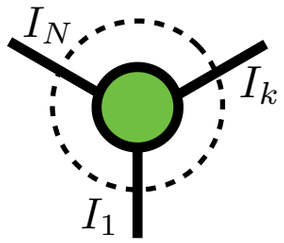
$(I \times J)$

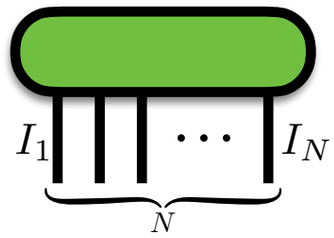
Matrix



$(I \times J \times K)$

3-order Tensor

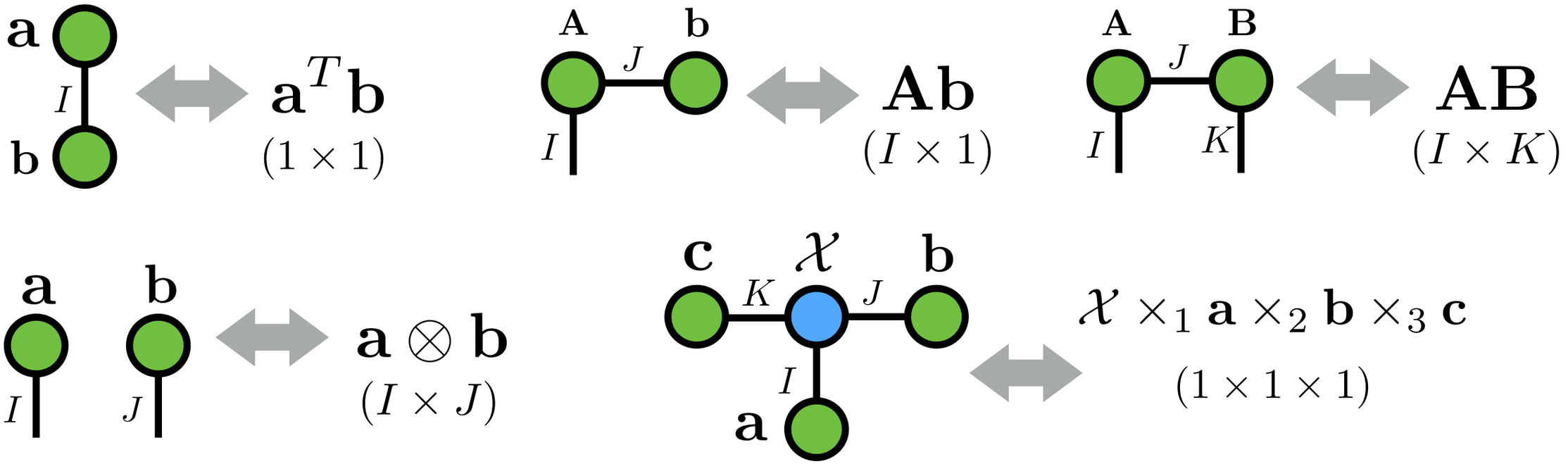


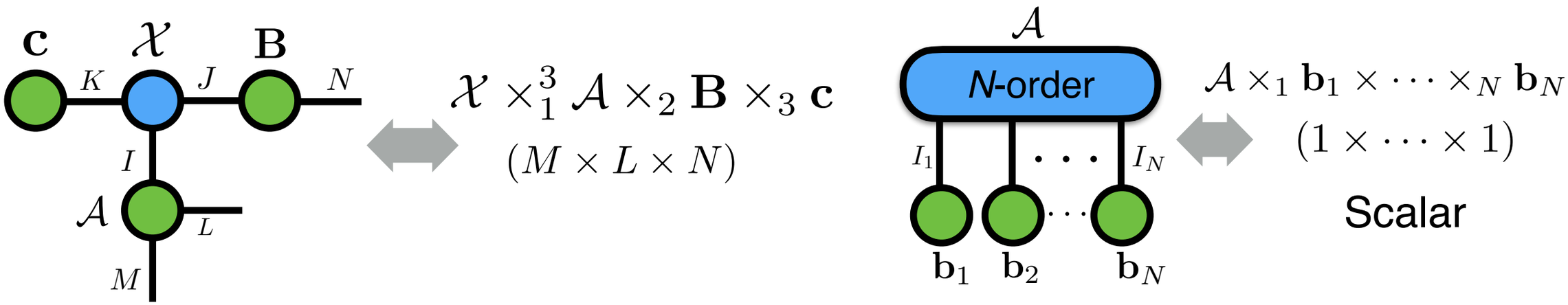


$(I_1 \times I_2 \times \cdots \times I_N)$

N-order Tensor

Tensor Network Operations

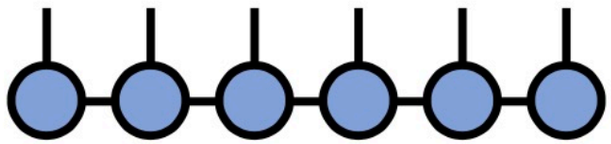




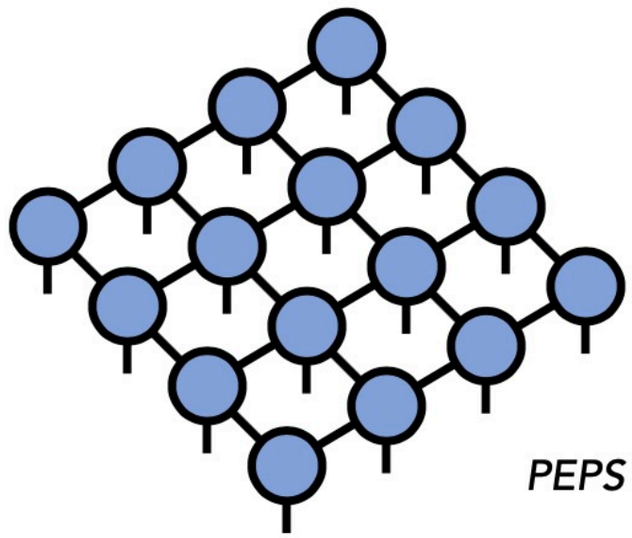
- Basic linear and multilinear algebra operations
- Convenient to denote complex operations

Tensor Network Models

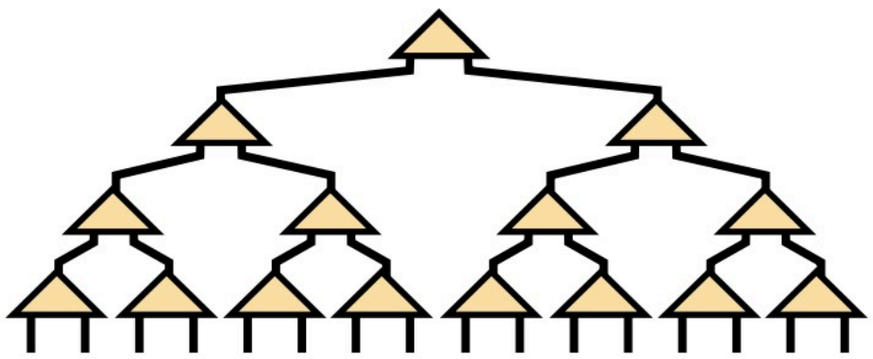
≈



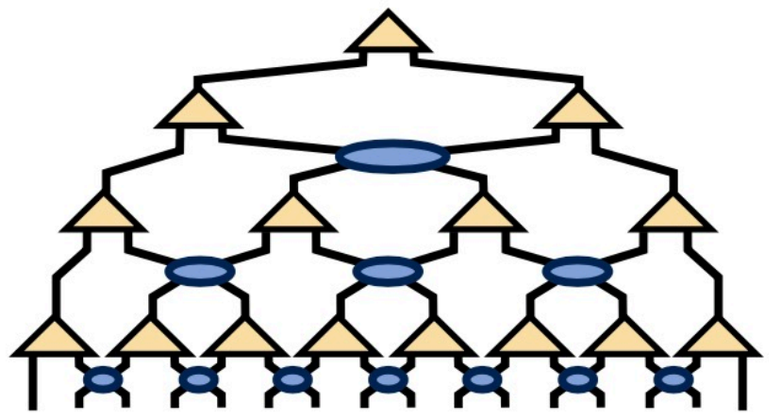
matrix product state / tensor train



PEPS network



tree tensor network / hierarchical Tucker

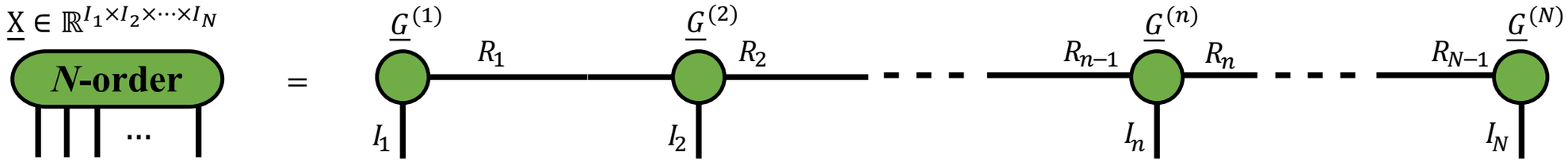


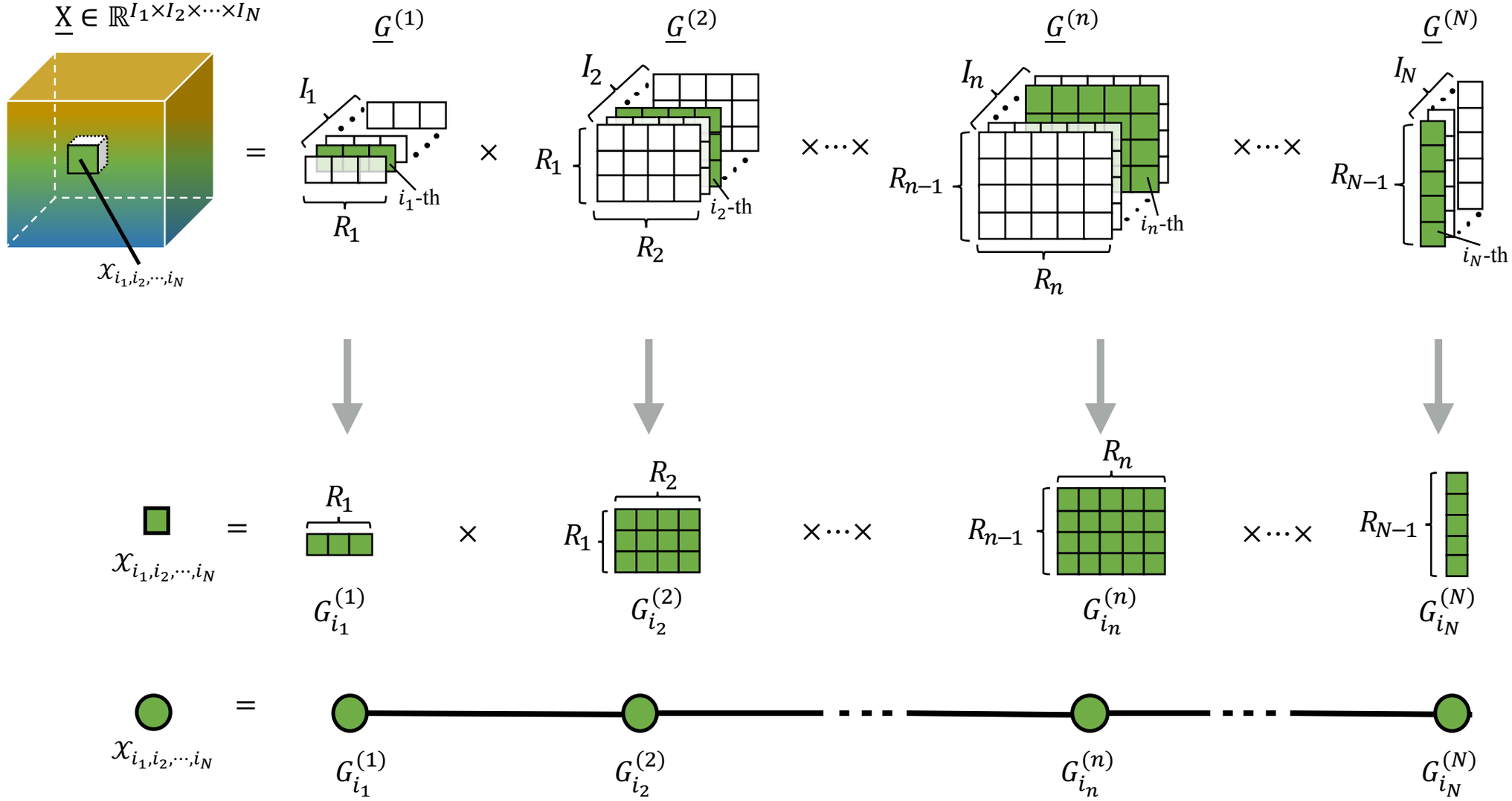
MERA network

R. Orus, Ann. of Phys. 349, 117–158 (2014)

Tensor Train Decomposition

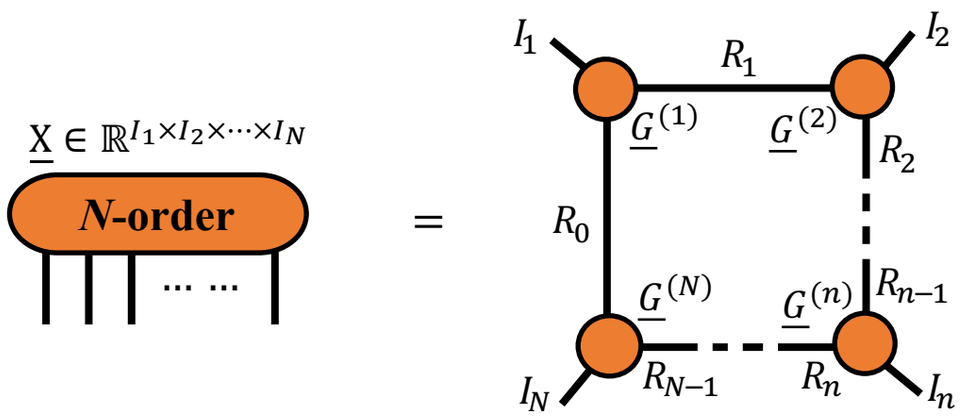
(Oseledets, SIAM J. Sci. Comput. 2011)

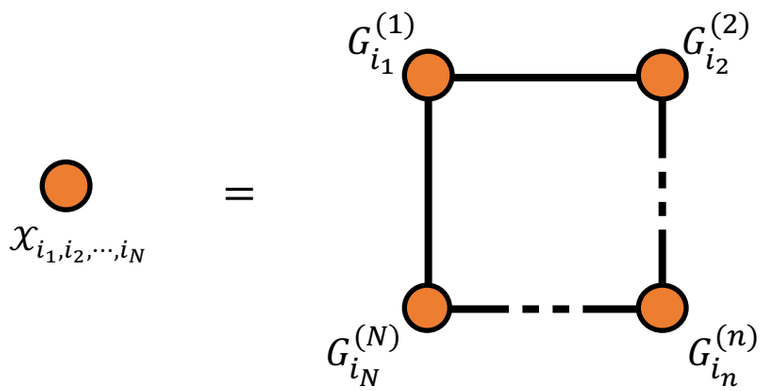


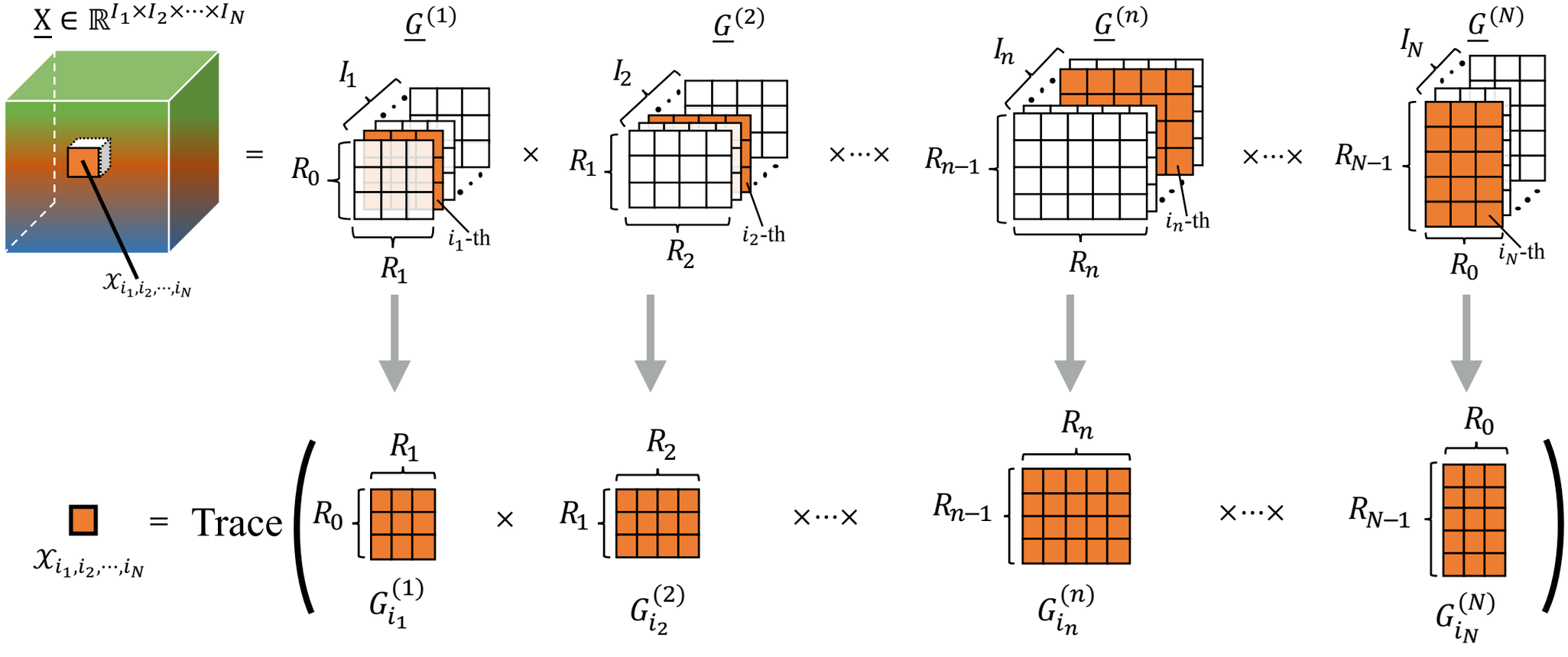


Tensor Ring Decomposition

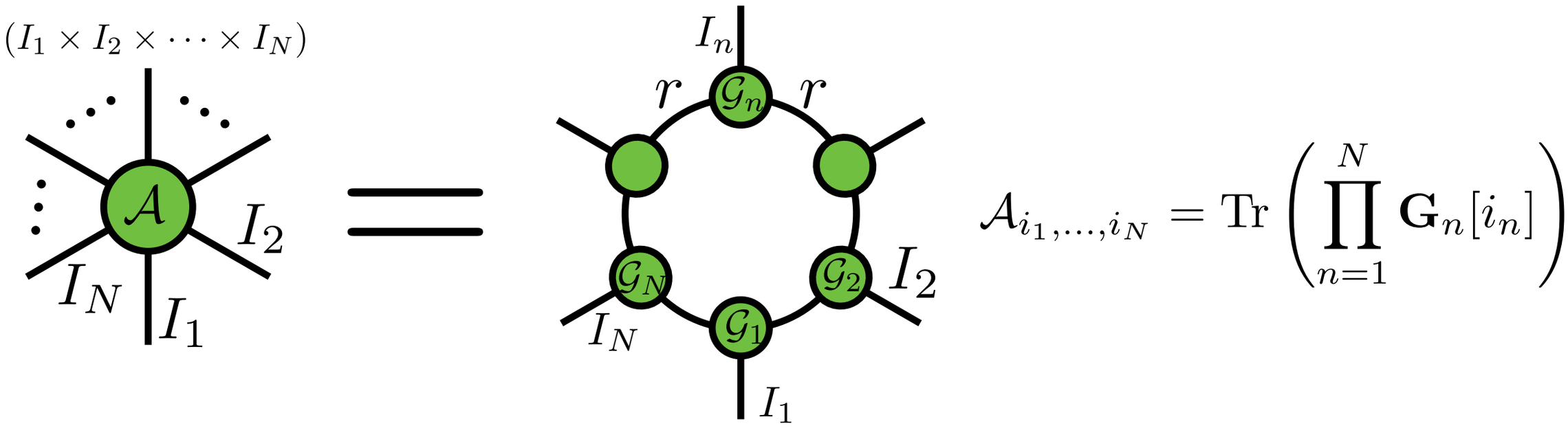
(Zhao et al., arXiv 2016, ICASSP 2019)







Tensor Ring Decomposition



$$\mathcal{A}_{i_1,\ldots,i_N} = \mathrm{Tr}\left(\prod_{n=1}^{N} \mathbf{G}_n[i_n]\right)$$

- Highly compact representation

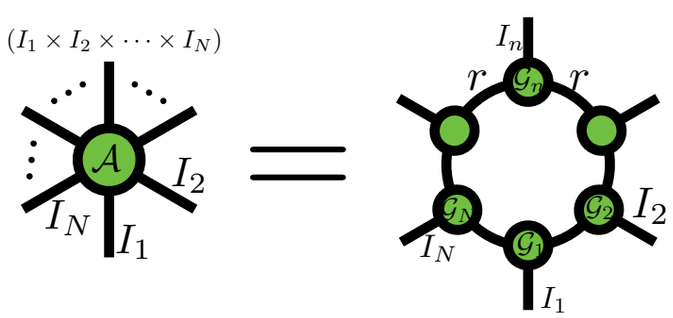
$$\mathcal{O}(I^N) \Rightarrow \mathcal{O}(NIr^2)$$

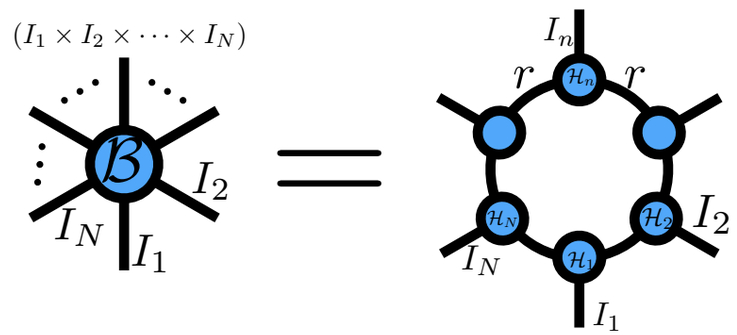
- Circular permutation invariance

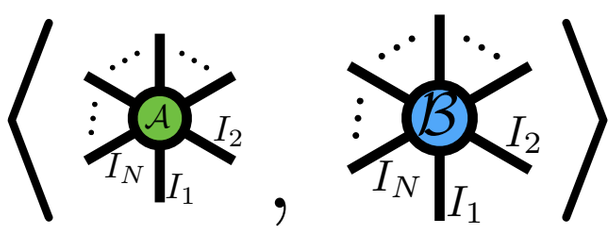
$$\widetilde{\mathcal{A}}(I_n \times \cdots \times I_N \times I_1 \times \cdots I_{n-1}) \Rightarrow \{\mathcal{G}_n, \ldots, \mathcal{G}_N, \mathcal{G}_1, \ldots, \mathcal{G}_{n-1}\}$$

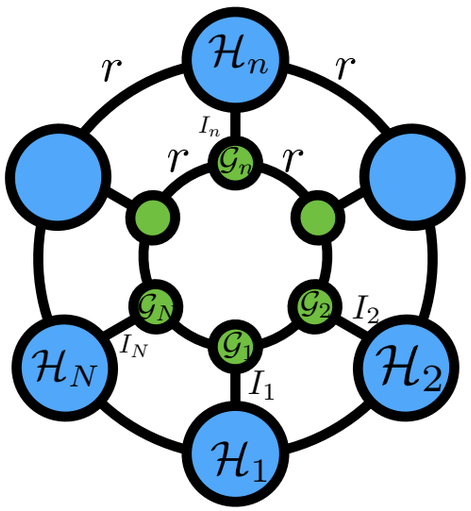
- If any r =1, TR becomes TT.

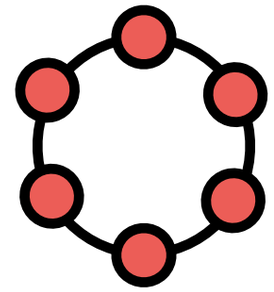
Efficient Computation









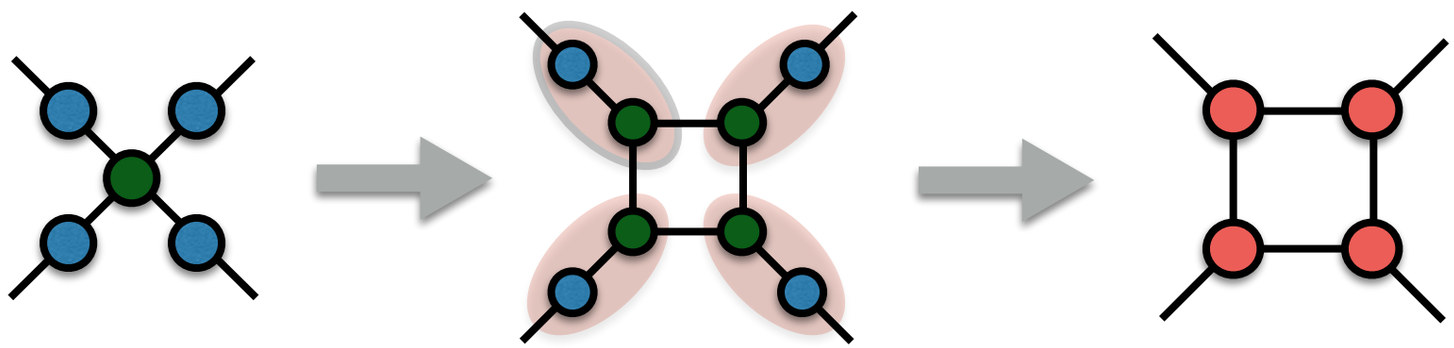


Computation

$\mathcal{O}(I^N)$

Computation

$\mathcal{O}(NIr^4)$



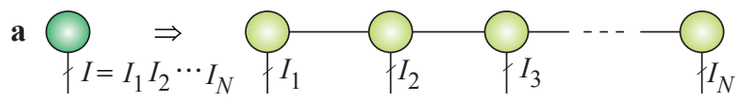
Compact Representations

Table 4: Image representation by using tensorization and TR decomposition. The number of parameters is compared for SVD, TT and TR given the same approximation errors.

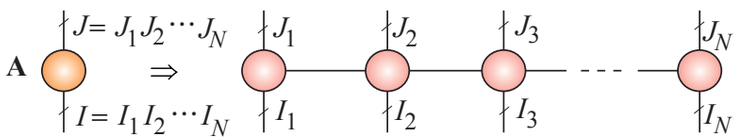
Data	$\epsilon = 0.1$		$\epsilon = 0.01$		$\epsilon = 9e-4$		$\epsilon = 2e-15$	
$n = 256, d = 2$	SVD	TT/TR	SVD	TT/TR	SVD	TT/TR	SVD	TT/TR
	9.7e3	9.7e3	7.2e4	7.2e4	1.2e5	1.2e5	1.3e5	1.3e5
Tensorization	$\epsilon = 0.1$		$\epsilon = 0.01$		$\epsilon = 2e-3$		$\epsilon = 1e-14$	
	TT	TR	TT	TR	TT	TR	TT	TR
$n = 16, d = 4$	5.1e3	3.8e3	6.8e4	6.4e4	1.0e5	7.3e4	1.3e5	7.4e4
$n = 4, d = 8$	4.8e3	4.3e3	7.8e4	7.8e4	1.1e5	9.8e4	1.3e5	1.0e5
$n = 2, d = 16$	7.4e3	7.4e3	1.0e5	1.0e5	1.5e5	1.5e5	1.7e5	1.7e5

Tensor Networks for Large-Scale Optimization Problems

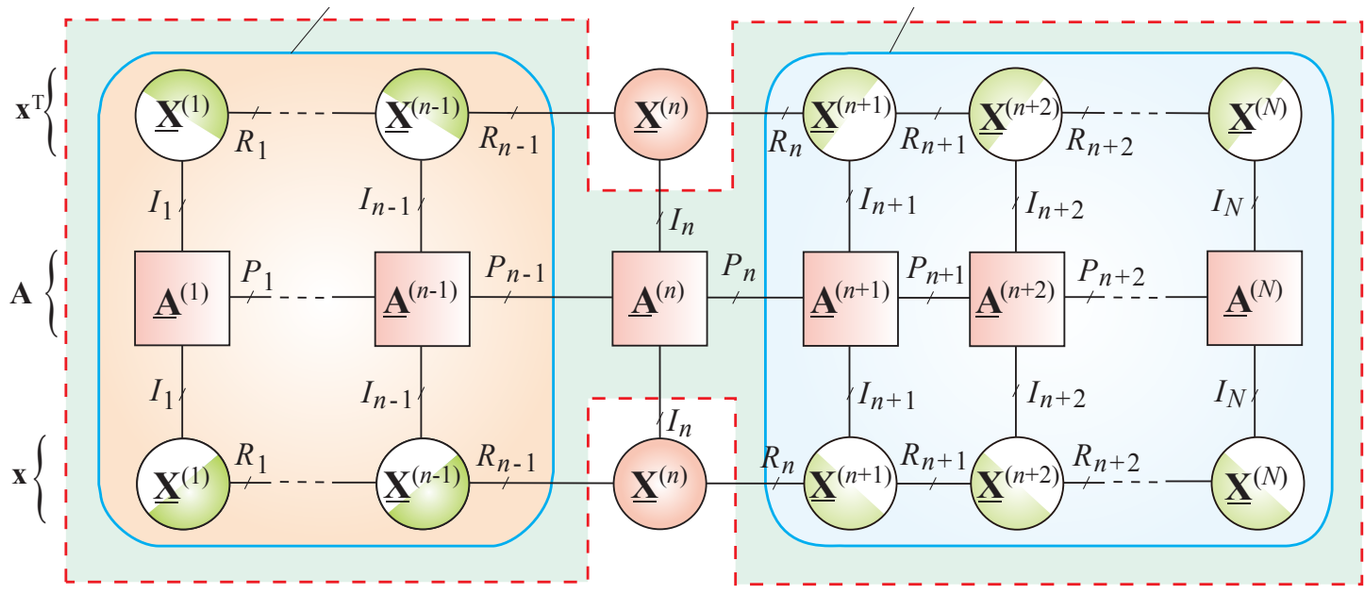
- TT format of a large vector

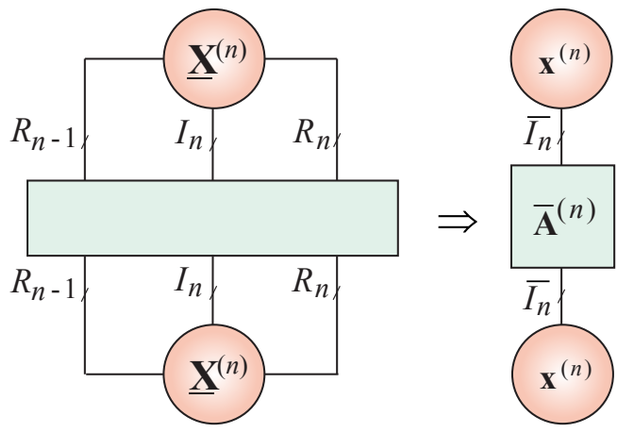


- TT format of a large matrix



Optimization problem: $\max x^T A x$





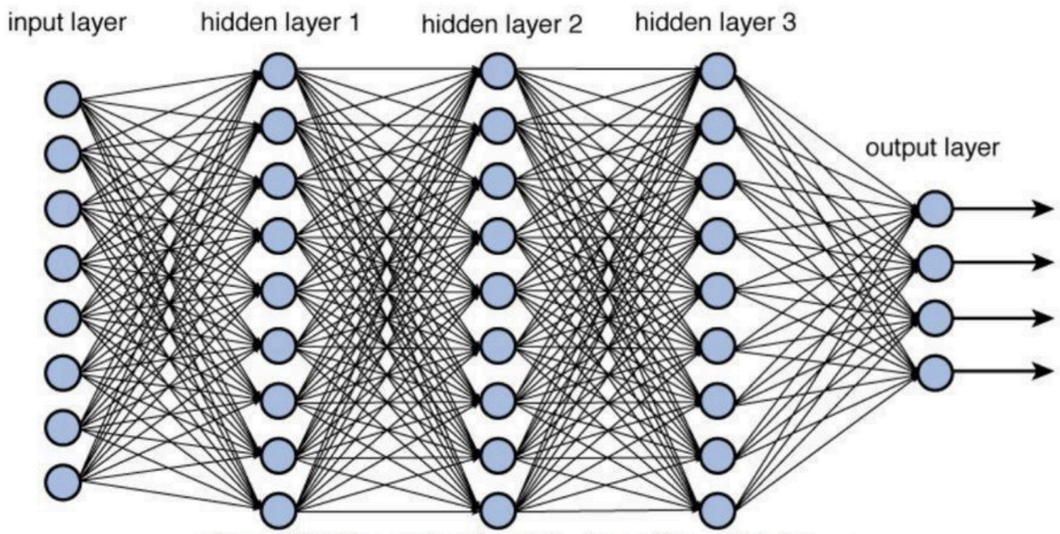
- Fast ALS/DMRG algorithm
- Large-scale SVD/PCA/CCA
- Parallel computing

(Cichocki et al., 2016)

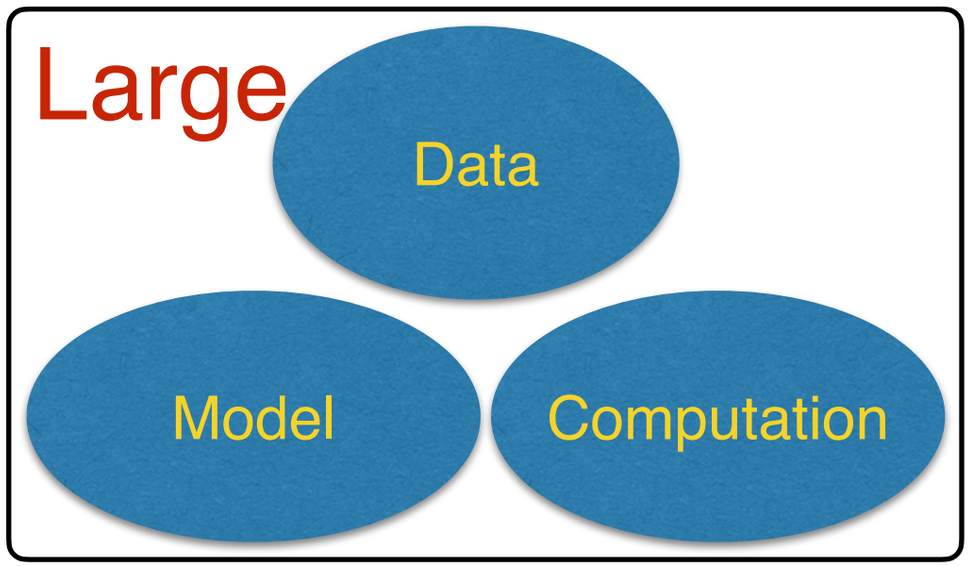
Part 2:

Tensor Networks in Deep Learning

Challenges in DNNs



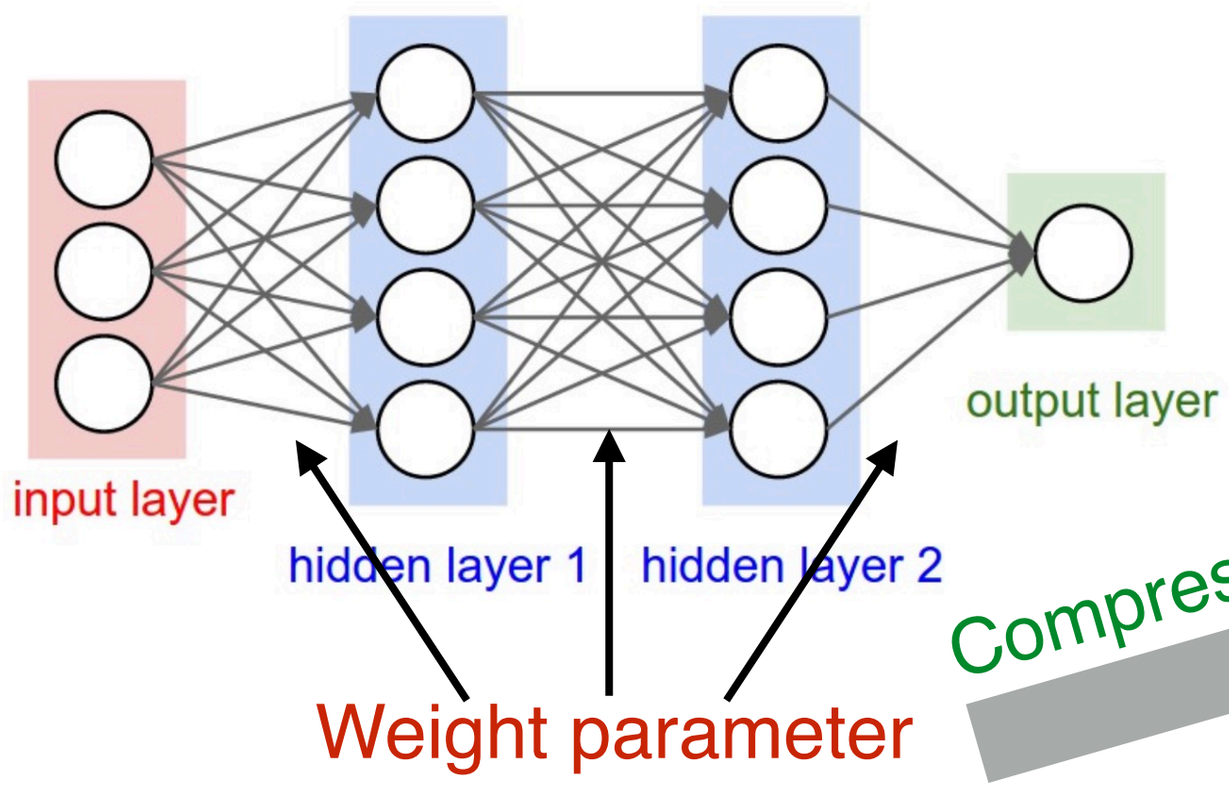
- Dimension of input increases for more complex data and task
- Wider layer improves model capacity but parameters increase dramatically
- Large number of training samples, large number of model parameters, large computation cost



- Huge storage (memory, disk requirement)
- Slow inference and energy costly
- Hard to deploy models on small devices (e.g. smart phone)

Model Compression

Goal: Make a lightweight model that is fast, memory-efficient and energy-efficient

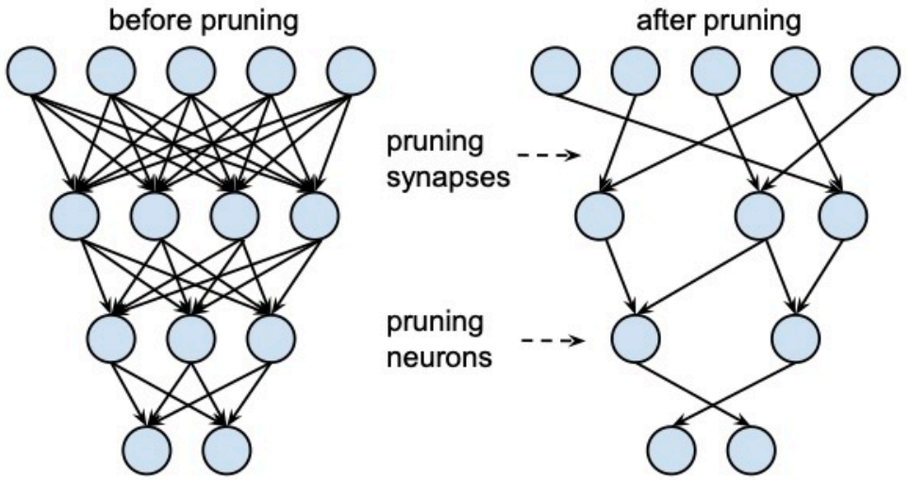


Compression

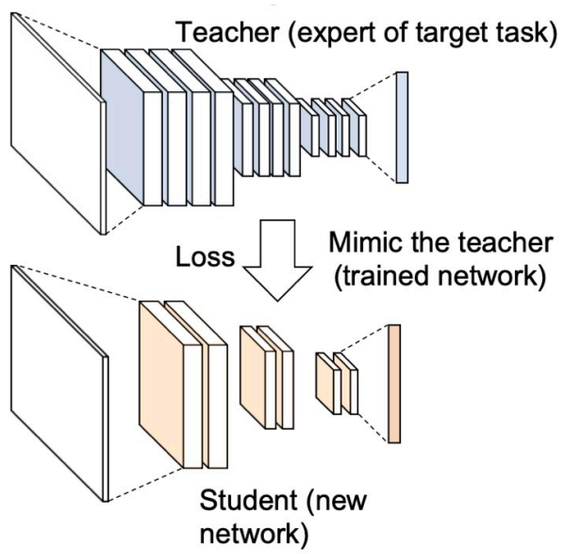
During training
Or
After training

- To compress model parameters with comparable performance
- To increase capacity without increasing model parameters
- To improve computation efficiency

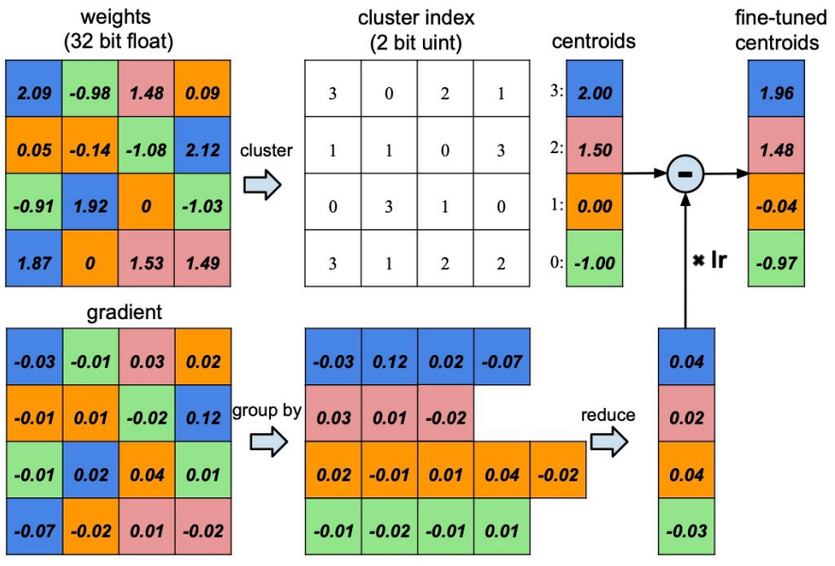
Taxonomy of Model Compression



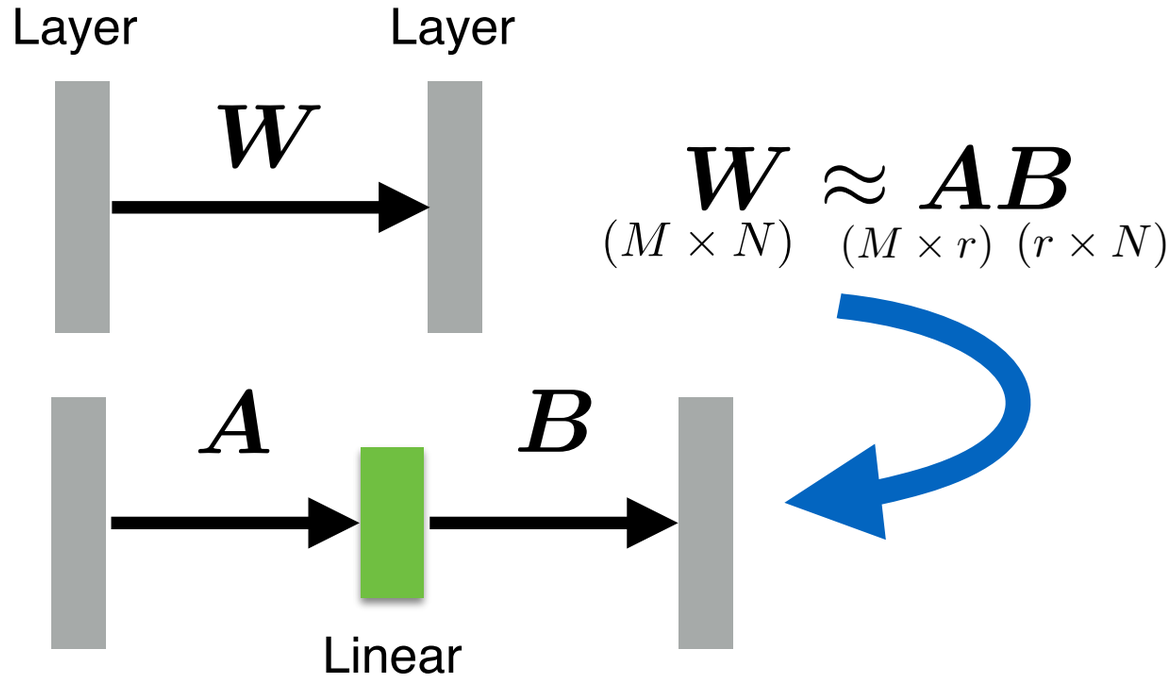
Weight Pruning
(Han et al., NIPS 2015)



Knowledge Distillation
(Hinton et al., NIPS 2014 Workshop)



Quantization and Sharing of Weight
(Han et al., ICLR 2016)



Low-rank Matrix/Tensor Factorization
(Novikov et al., NIPS 2015)

Compression

Low-rank matrix factorization of weights in fully connected layer

$$\underset{(M \times N)}{\boldsymbol{W}} \approx \underset{(M \times r)}{\boldsymbol{A}} \underset{(r \times N)}{\boldsymbol{B}} \qquad \text{Compression: } \mathcal{O}(MN) \to \mathcal{O}(r(M+N))$$

Low-rank tensor network factorization of weights

- Step 1: $\boldsymbol{W} \to \mathcal{W}$ (Matrix to *d*-order tensor)
- Step 2: $\mathcal{W} \approx \mathrm{TT}(\mathcal{G}_1 \cdots \mathcal{G}_d)$ (Tensor network representation)

Loss function:

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{x}, \boldsymbol{y}) \to \mathcal{L}(\{\mathcal{G}_1, \cdots, \mathcal{G}_d\}, \boldsymbol{x}, \boldsymbol{y})$$

Compression:

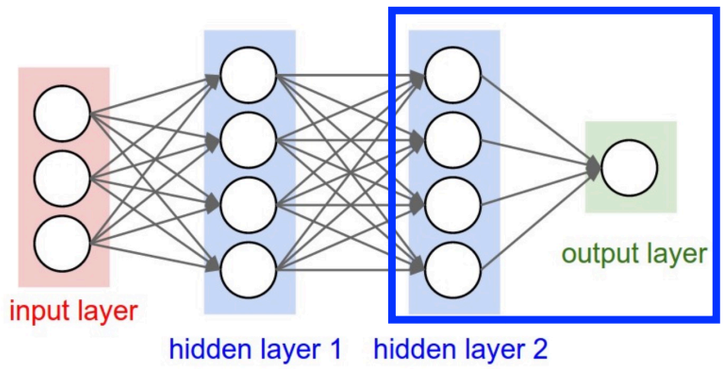
$$\mathcal{O}(MN) \to \mathcal{O}(dr^2 \sqrt[d]{M} \sqrt[d]{N})$$

Order of tensor TT-rank

Train very “wide” model

Tensorizing Neural Networks (Novikov et al., NIPS 2015)

Vector to Scalar Transformation via TN



Hidden layer

$$\underset{(M\times 1)}{\boldsymbol{y}} = \sigma(\boldsymbol{w}^T \boldsymbol{h} + \boldsymbol{b})$$

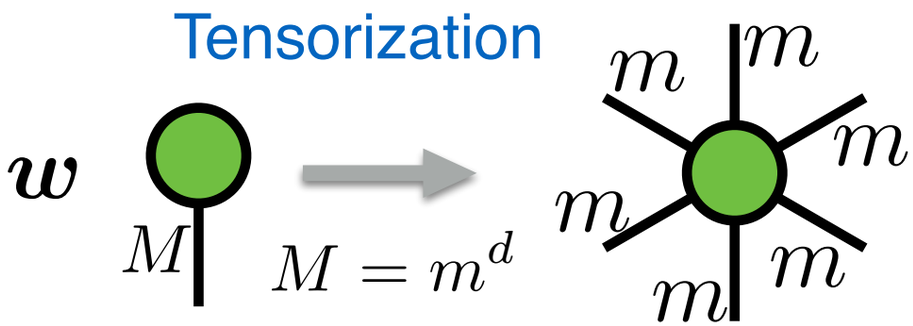
Output

Weights

Number of parameters $d\sqrt[d]{M}r^2$

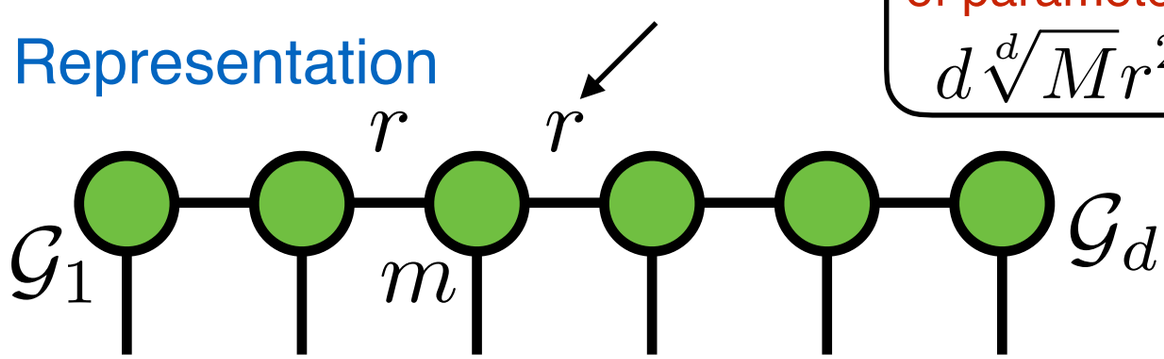
d-order Tensor

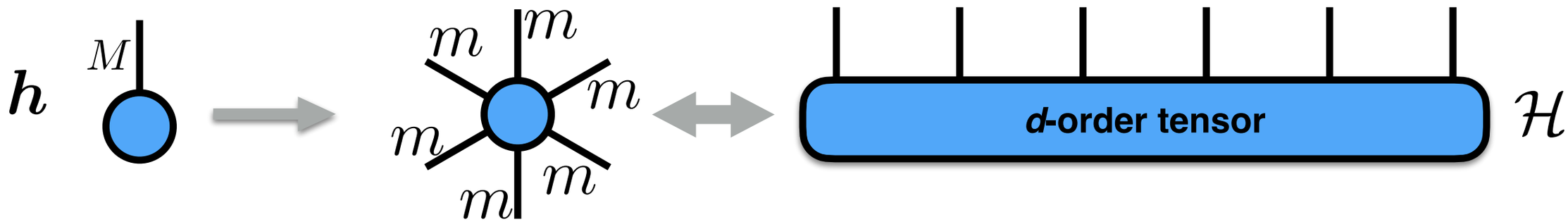
Tensorization



TT Representation

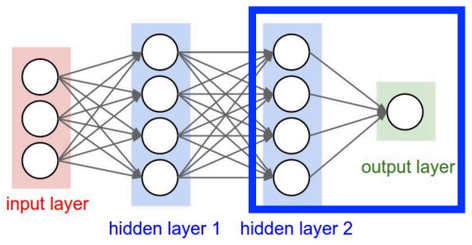
TT-rank





$$< \boldsymbol{w}, \boldsymbol{h} > \quad = \quad < \mathrm{TT}(\mathcal{G}_1, \cdots, \mathcal{G}_d), \mathcal{H} >$$

Vector to Scalar Transformation via TN



$$y = \sigma(\boldsymbol{w}^T \boldsymbol{h} + b)$$

$(M \times 1)$

Hidden layer

Output

Weights

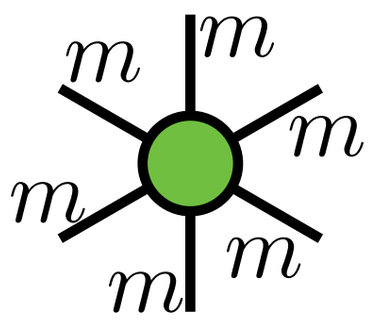
Tensorization

$\boldsymbol{w}$

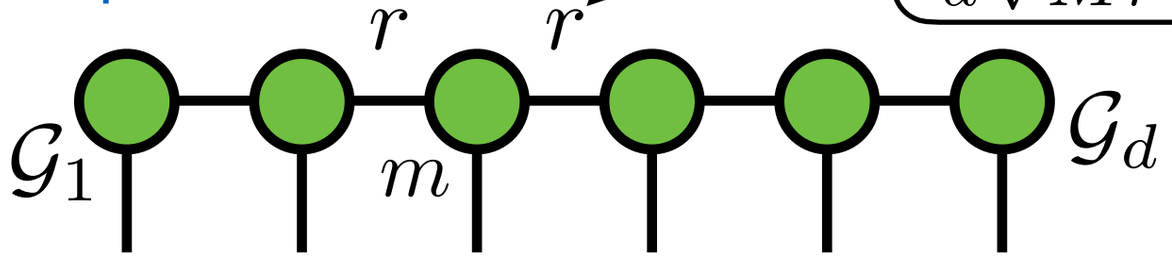
M

$M = m^d$

d-order Tensor



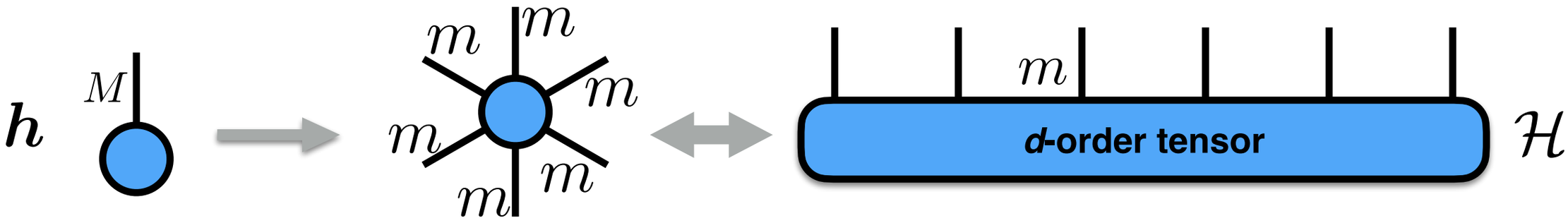
TT Representation

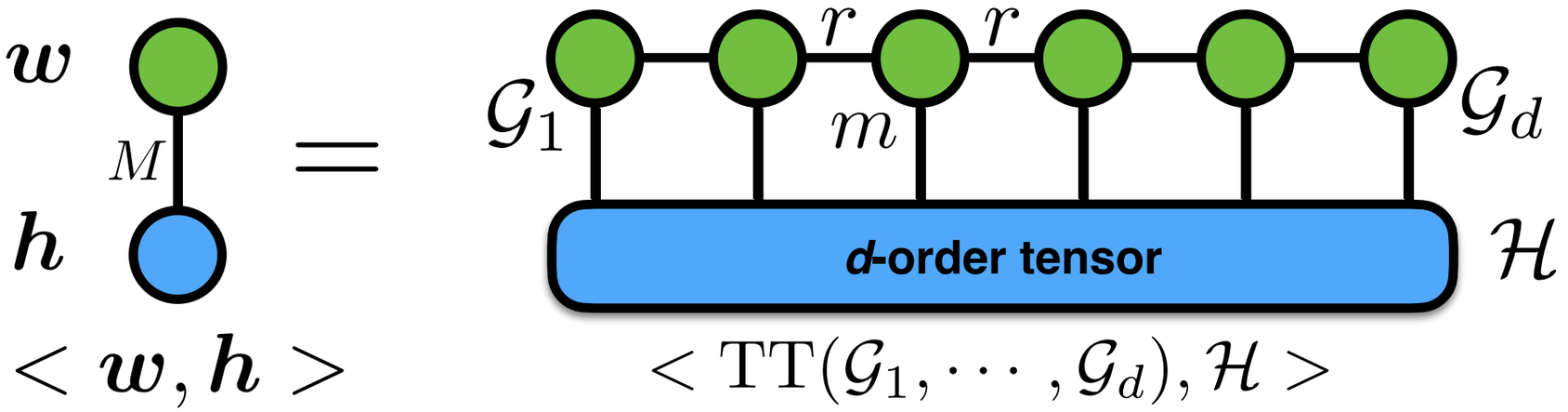


TT-rank

Number of parameters

$d\sqrt[d]{M}r^2$



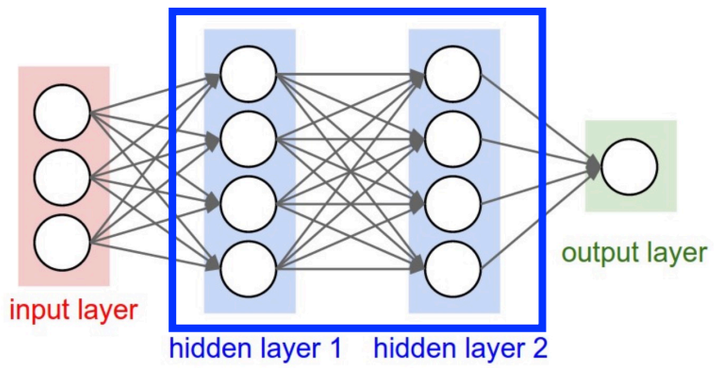


TN Contraction

$< \boldsymbol{w}, \boldsymbol{h} >$

$< \mathrm{TT}(\mathcal{G}_1, \cdots, \mathcal{G}_d), \mathcal{H} >$

Vector to Vector Transformation via TN



$$\boldsymbol{h}_2 = \sigma(\underset{(N \times M)}{\boldsymbol{W}^T} \boldsymbol{h}_1 + \boldsymbol{b})$$

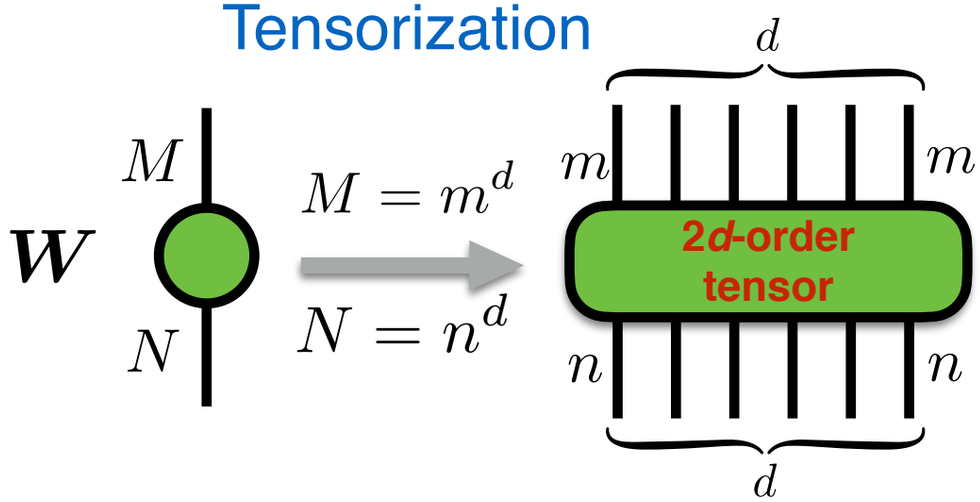
Hidden layer

Hidden layer

Weights

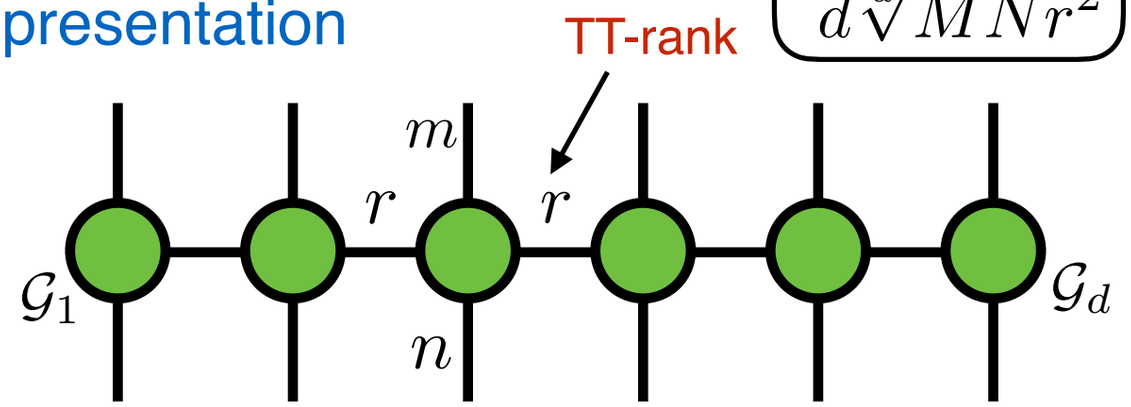
Number of parameters $d\sqrt[d]{MN}r^2$

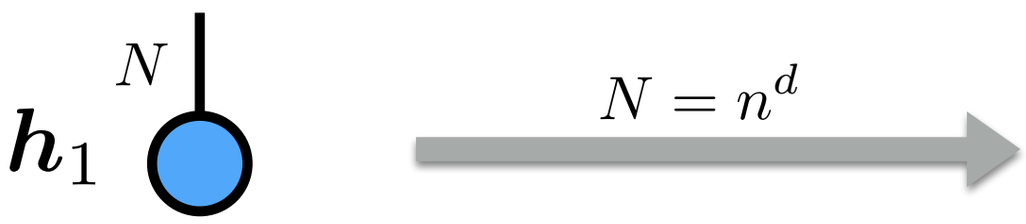
Tensorization

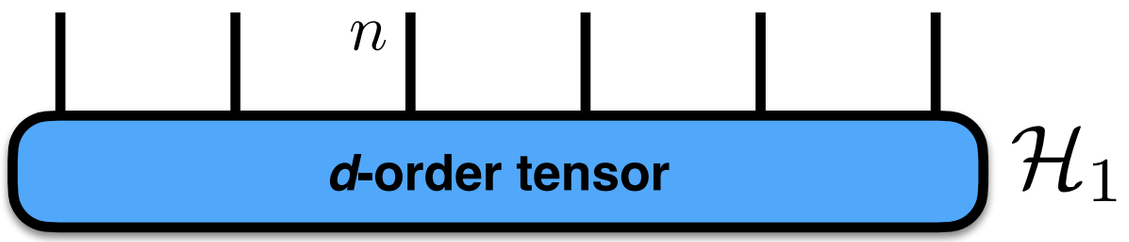


TT Representation

TT-rank

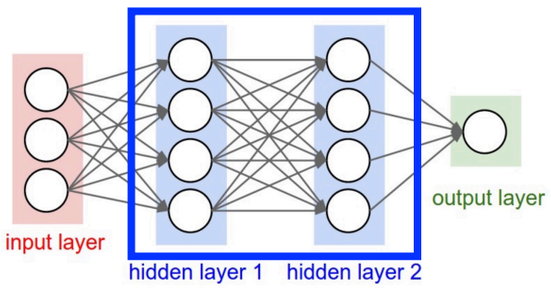






$$\boldsymbol{W}^T\boldsymbol{h}_1 = \mathrm{TT}(\mathcal{G}_1, \cdots, \mathcal{G}_d) \times_{1,\ldots,d} \mathcal{H}_1$$

Vector to Vector Transformation via TN



$$\boldsymbol{h}_2 = \sigma(\underset{(N \times M)}{\boldsymbol{W}^T} \boldsymbol{h}_1 + \boldsymbol{b})$$

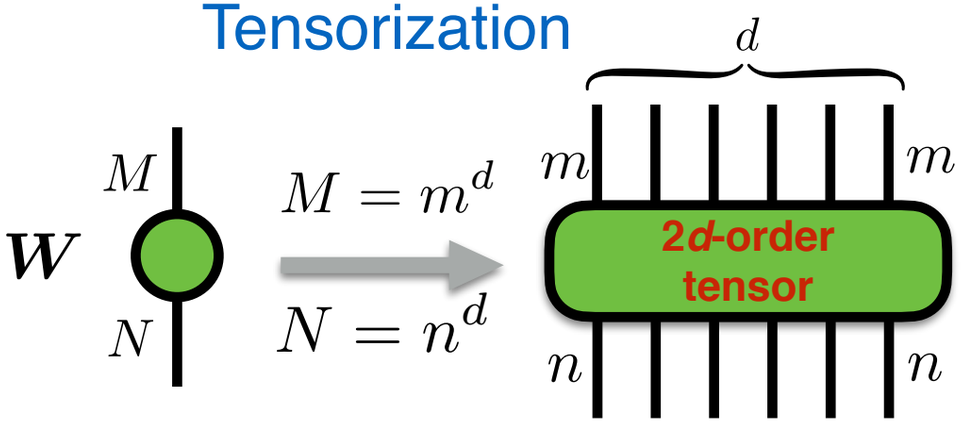
Hidden layer (output)

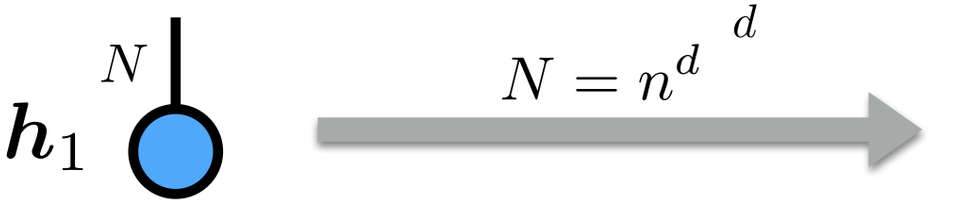
Hidden layer (input)

Weights

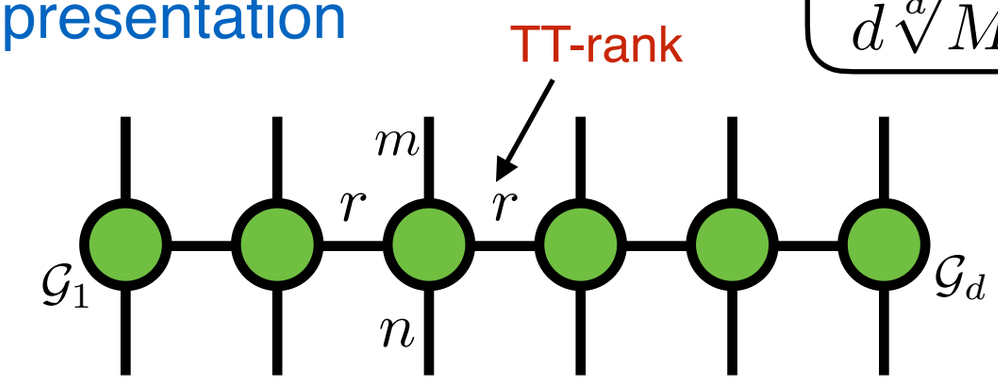
Number of parameters $d\sqrt[d]{MN}r^2$

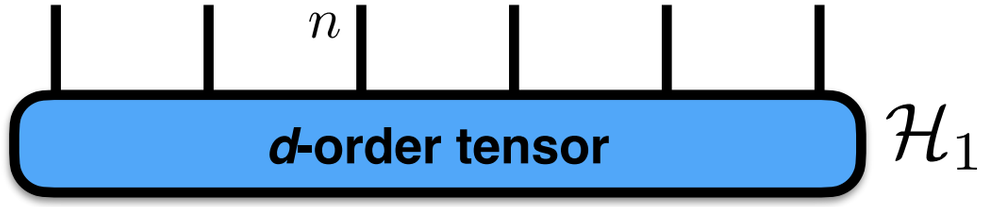
Tensorization

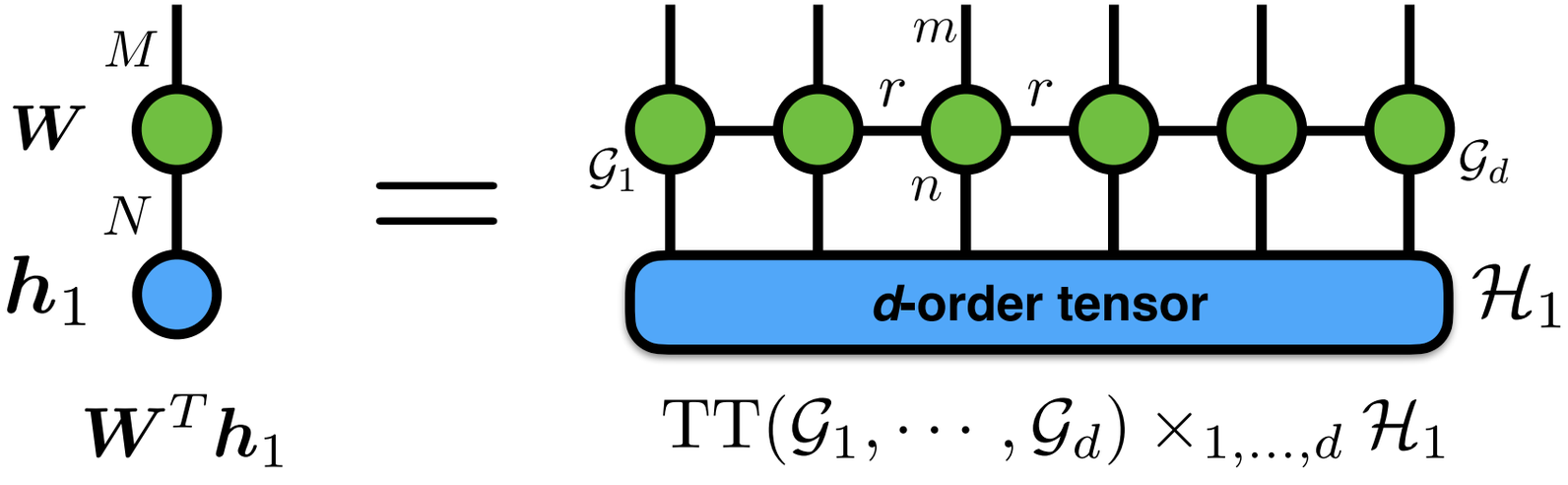




TT Representation





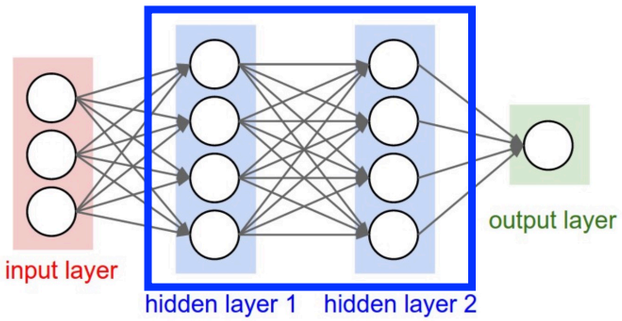


$\boldsymbol{W}^T\boldsymbol{h}_1$

$\mathrm{TT}(\mathcal{G}_1, \cdots, \mathcal{G}_d) \times_{1,\ldots,d} \mathcal{H}_1$

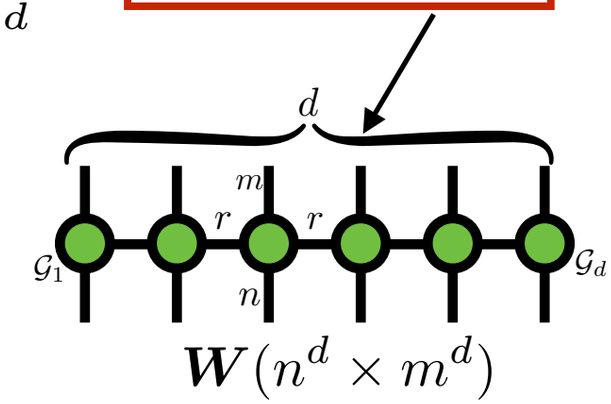
TN Contraction

Learning of TN Weights



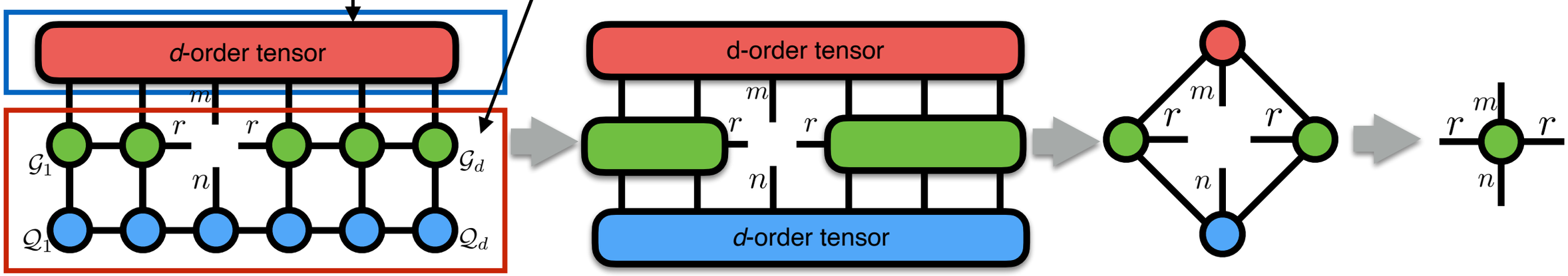
$$\boldsymbol{h}_2 = \sigma(\underbrace{\boldsymbol{W}^T \boldsymbol{h}_1 + \boldsymbol{b}}_{\boldsymbol{o}_2})$$

- Loss: $\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{x}, y) \Rightarrow \min_{\mathcal{G}_1, \cdots, \mathcal{G}_d} \mathcal{L}(\{\mathcal{G}_1, \cdots, \mathcal{G}_d\}, \boldsymbol{x}, y)$



- Gradients over TN core tensors

$$\frac{\partial \mathcal{L}}{\partial \mathcal{G}_k} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_2} \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{o}_2} \frac{\partial \boldsymbol{o}_2}{\partial \mathcal{G}_k}$$



Compression Performance

- Model compression on ImageNet

Architecture	Matrices compr.	Network compr.	Error
FC FC FC	1	1	11.2
TT4 FC FC	50 972	3.9	11.2
TT2 FC FC	194 622	3.9	11.5
TT1 FC FC	713 614	3.9	12.8
TT4 TT4 FC	37 732	7.4	12.3
LR1 FC FC	3 521	3.9	97.6
LR5 FC FC	704	3.9	53.9
LR50 FC FC	70	3.7	14.9

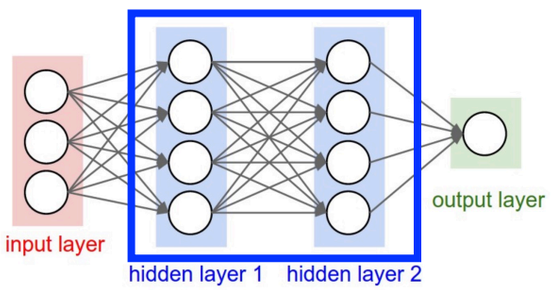
- Speed

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	$O(d^2 \mathrm{r}^4 m \max\{M, N\})$	$O(\mathrm{r}^3 \max\{M, N\})$

Type	1 im. time (ms)	100 im. time (ms)
CPU fully-connected layer	16.09	97.2
CPU TT-layer	1.24	94.7
GPU fully-connected layer	2.7	33
GPU TT-layer	1.92	12.86

Tensorizing Neural Networks (Novikov et al., NIPS 2015)

Ideal Case: TN to TN Transformation



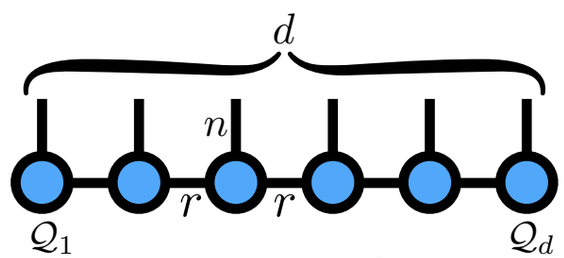
$$\boldsymbol{h}_2 = \sigma(\boldsymbol{W}^T \boldsymbol{h}_1 + \boldsymbol{b})$$

Hidden layer

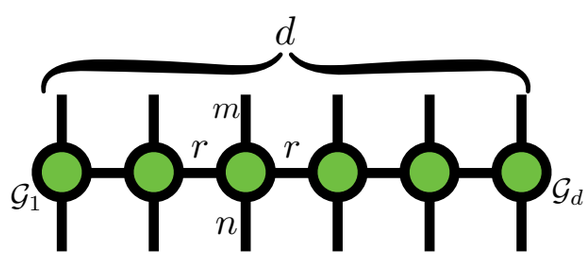
Hidden layer

Weights

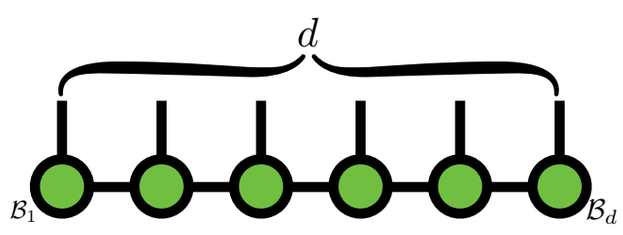
TT Representation



$\boldsymbol{h}_1(n^d)$

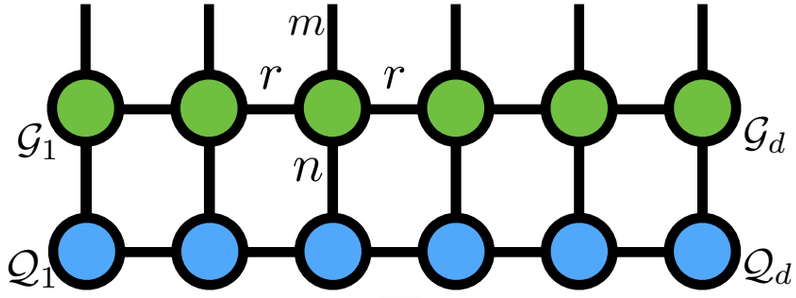


$\boldsymbol{W}(n^d \times m^d)$

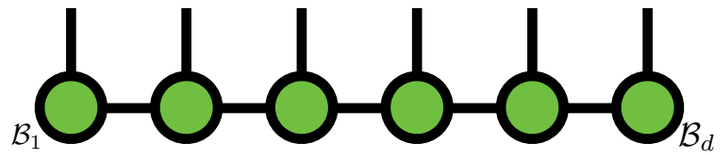


$\boldsymbol{b}(m^d)$

$$\boldsymbol{h}_2 = \sigma\Big($$



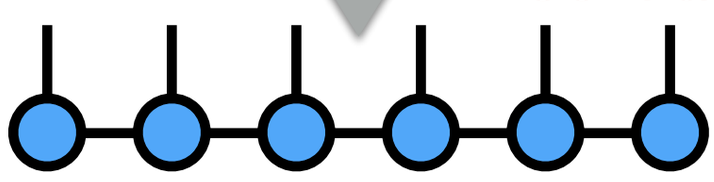
$+$



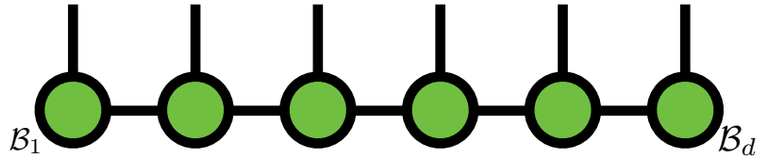
$\Big)$

TN Contraction

$$= \sigma\Big($$

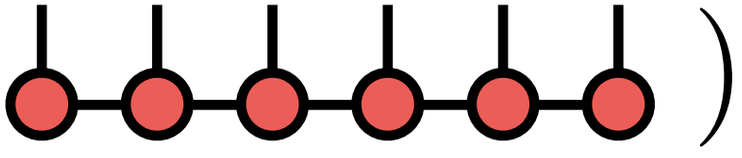


$+$



$\Big)$

$$= \sigma\Big($$



$\Big)$

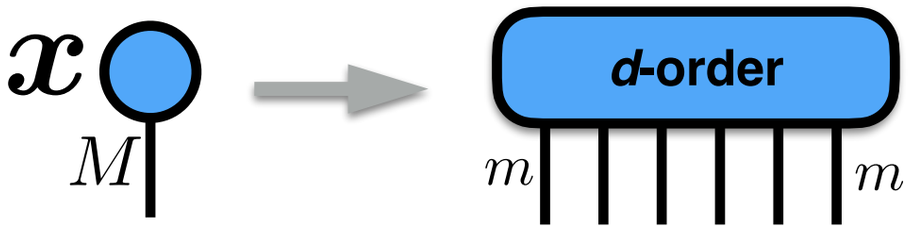
- Very efficient computation

  $\mathcal{O}(d\sqrt[d]{M}\sqrt[d]{N}r^4)$

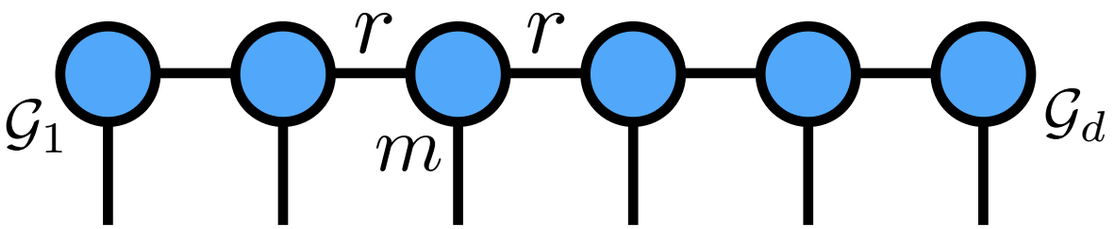
- Convenient for parallel computing

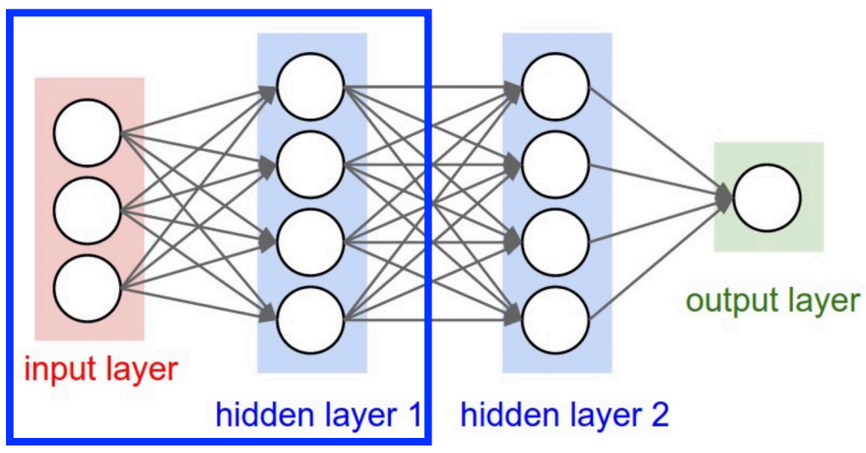
Challenges: Input to Hidden Layer via TN

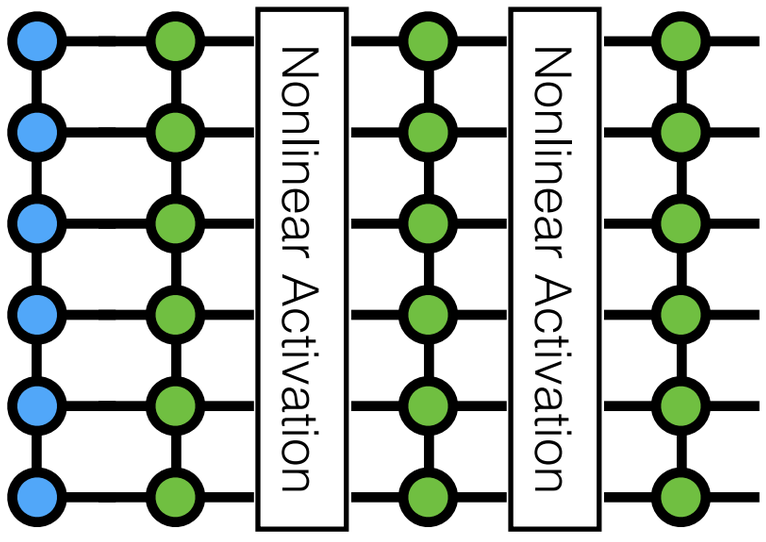
Tensorization



TT Decomposition



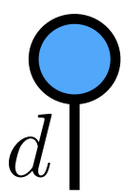




- Input data may not admit low-rank TT approximation (small r)
- TT decomposition is computational costly
- Nonlinear activation destroy TT format,

Feature Mapping

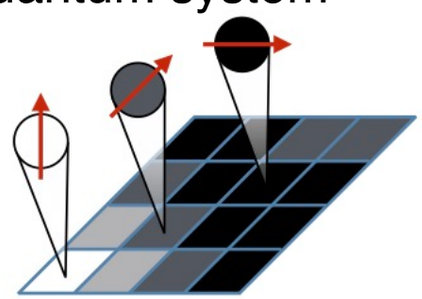
- Mapping input data into TN representation



$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$$

$$\phi(x_i) = \left[\cos\left(\frac{\pi}{2}x_i\right), \sin\left(\frac{\pi}{2}x_i\right)\right]^T$$

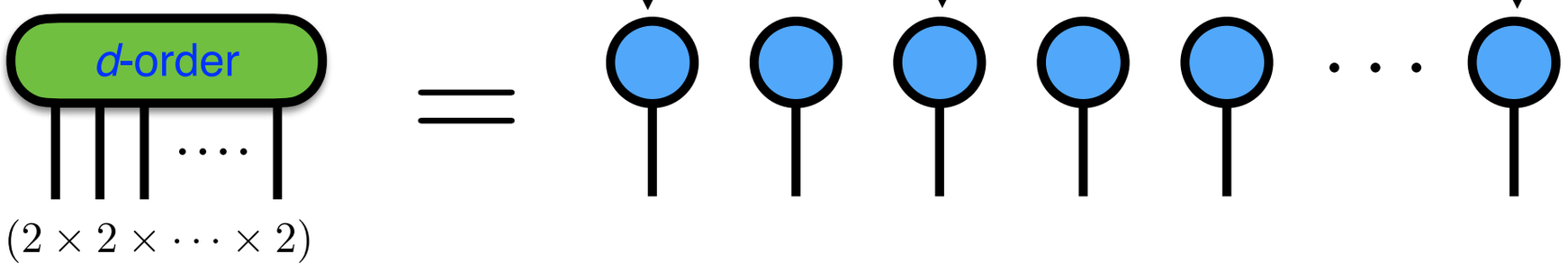
Inspired by “spin” vectors in quantum system



$$d \mapsto 2^d$$

$$\Phi(\mathbf{x}) = \phi(x_1) \otimes \phi(x_2) \otimes \cdots \otimes \phi(x_d)$$

Local feature map



Rank-1 Tensor

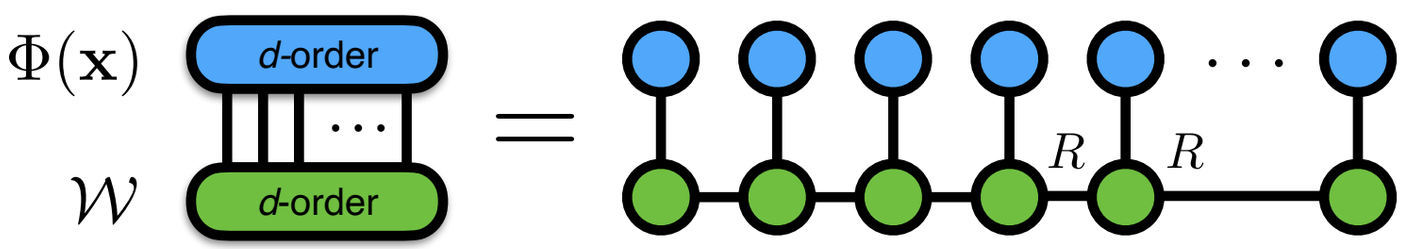
- Different mapping for each feature is possible
- $\phi(x_i)$ can be generalized to I-dimensional, e.g., $\Phi(\mathbf{x}) : d \mapsto I^d$

Supervised Learning with Quantum-Inspired Tensor Networks (Stoudenmire et al., NIPS 2016)

Classification Layer

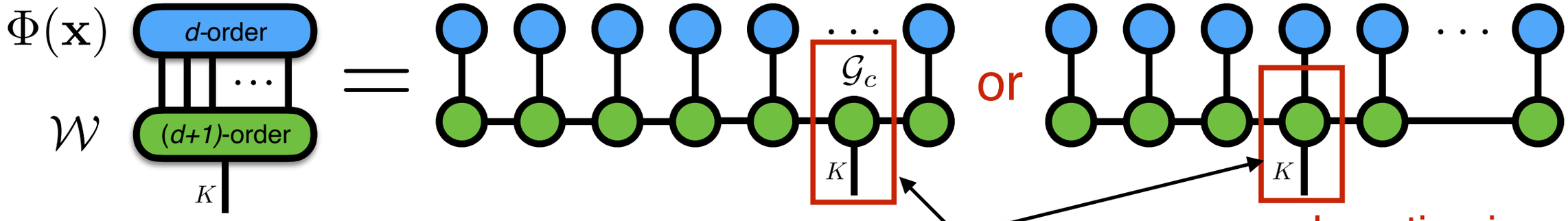
- Binary classification: $f(\mathbf{x}) = \langle \mathcal{W}, \Phi(\mathbf{x}) \rangle$

Number of features 2^d



Number of parameters $2dR^2$

- Multi-class model: $f_c(\mathbf{x}) = \langle \mathcal{W}_c, \Phi(\mathbf{x}) \rangle, \ \mathcal{W} = \{\mathcal{W}_c\}_{c=1}^{K}$



Assumption: $\mathcal{W}_c = \mathrm{TT}(\mathcal{G}_1, \cdots, \mathcal{G}_c(c), \cdots, \mathcal{G}_d,)$

Location is arbitrary

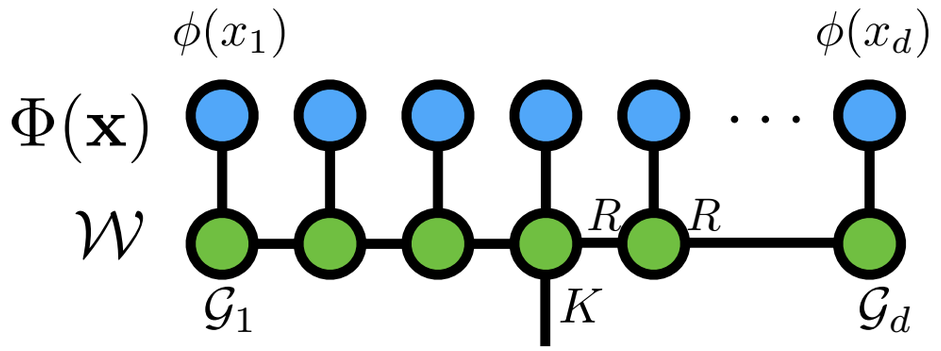
- TN allows highly efficient optimization and computation
- Weight compression: $2^d K \mapsto 2dR^2 + KR^2$, TT-rank (R) controls complexity of TT representation, small R is expected

Supervised Learning with Quantum-Inspired Tensor Networks (Stoudenmire et al., NIPS 2016)

Learning Algorithm: DMRG[1]

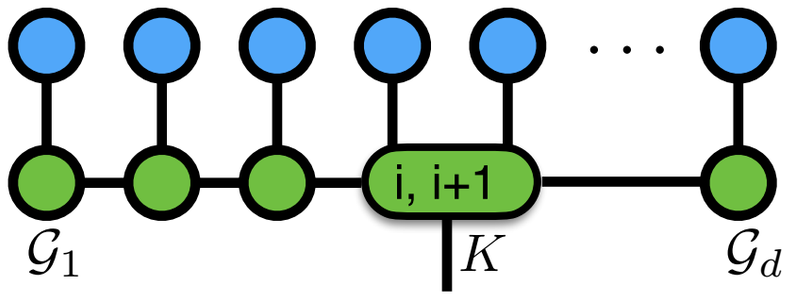
- Optimization: $\min_{\mathcal{W}} \mathbb{E}[\ell(\mathcal{W}, \mathbf{x}, \mathbf{y})] \Rightarrow \min_{\{\mathcal{G}_i\}} \mathbb{E}[\ell(\{\mathcal{G}_1, \ldots, \mathcal{G}_d\}, \mathbf{x}, \mathbf{y})]$
- Standard: computing gradient over each core tensor $\mathcal{G}_i$
- Model:

$$f(\mathbf{x}) =$$



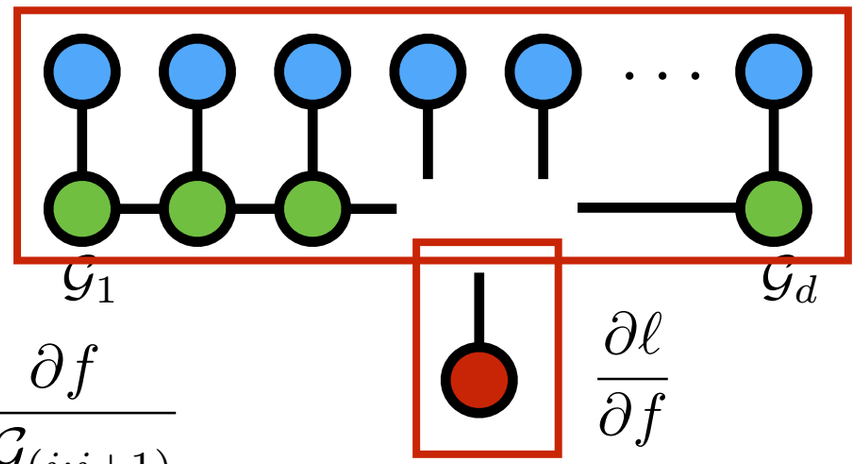
- DMRG:

Step 1: Merge two adjacent cores



Step 2: Compute gradient over block cores

$$\frac{\partial f}{\partial \mathcal{G}_{(i:i+1)}}$$



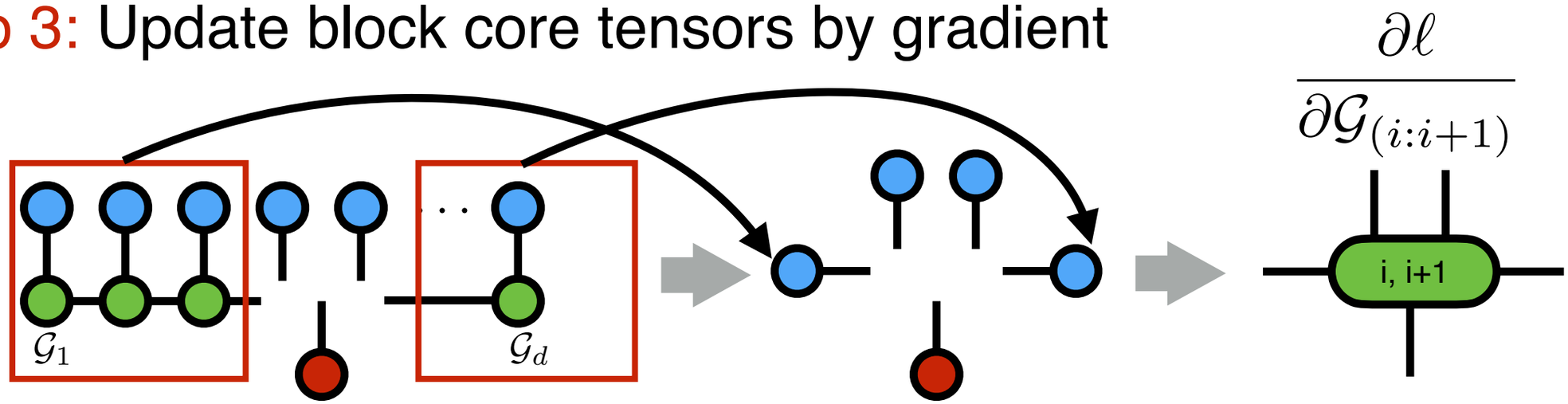
$$\frac{\partial \ell}{\partial \mathcal{G}_{(i:i+1)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial \mathcal{G}_{(i:i+1)}}$$

Supervised Learning with Quantum-Inspired Tensor Networks (Stoudenmire et al., NIPS 2016)

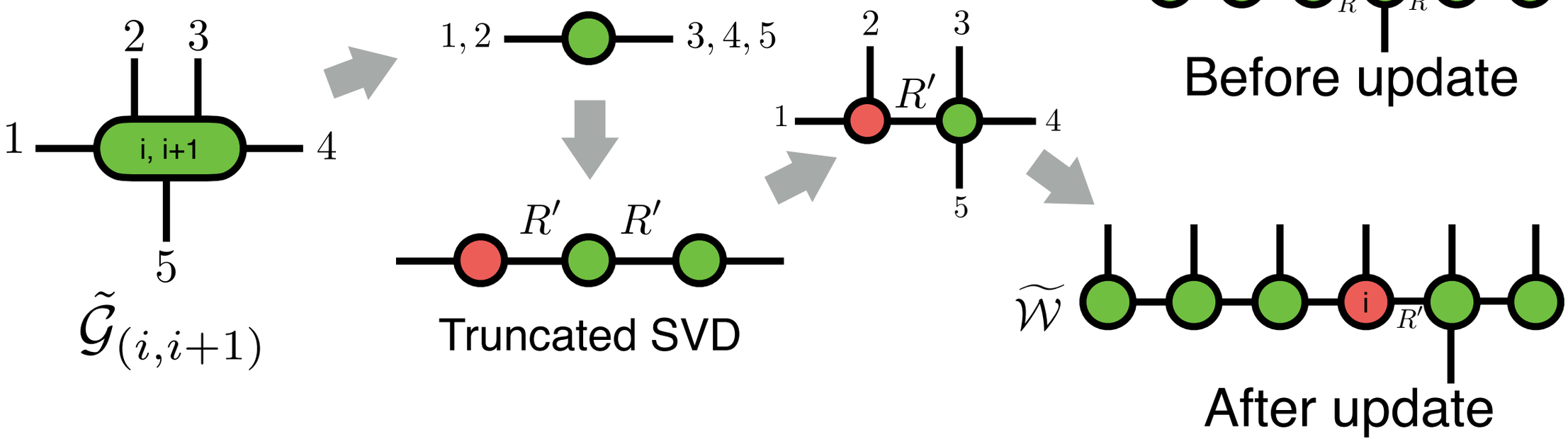
[1]DMRG: Density Matrix Renormalization Group

Learning Algorithm: DMRG

Step 3: Update block core tensors by gradient



Step 4: TT-rank is chosen adaptively



Supervised Learning with Quantum-Inspired Tensor Networks (Stoudenmire et al., NIPS 2016)

Discussions

Advantages：

- Single and wide layer with comparable accuracy
- Highly efficient model compression and computation
- Better interpretability due to multilinear nature
- Provide a simple and efficient way to represent data input as TN format

Challenges:

- Mixing SGD with DMRG algorithm is non-trivial, batch gradient is needed
- More hyper-parameters: dimension of nonlinear mapping, TT-ranks

Exponential Machines

- Mapping of input data

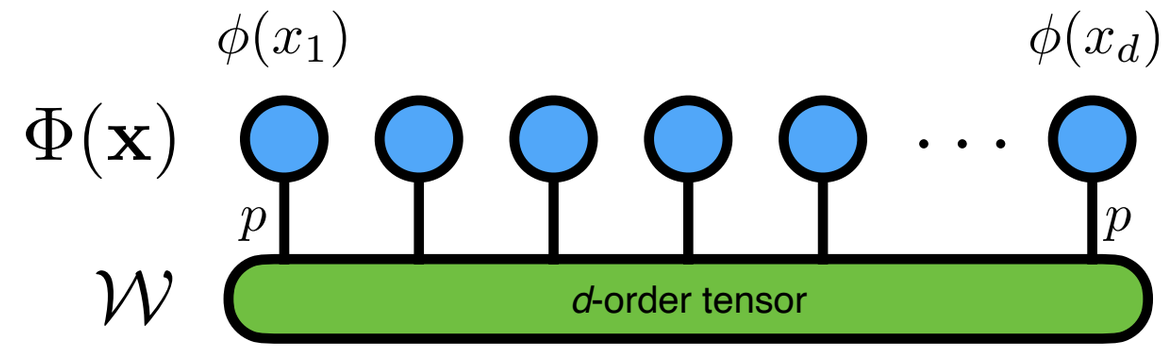
$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$$

$$\phi(x_i) = [1, x_i, x_i^2, \ldots, x_i^{p-1}]^T$$

$$d \mapsto p^d \qquad \Phi(\mathbf{x}) = \phi(x_1) \otimes \phi(x_2) \otimes \cdots \otimes \phi(x_d)$$

- *p-degree* polynomial function

$$f(\mathbf{x}) = \langle \mathcal{W}, \Phi(\mathbf{x}) \rangle$$



Example: $d = 3, p = 2$

$$f(\mathbf{x}) = \mathcal{W}_{000} + \mathcal{W}_{100}x_1 + \mathcal{W}_{010}x_2 + \mathcal{W}_{001}x_3 + \mathcal{W}_{110}x_1x_2 + \mathcal{W}_{101}x_1x_3 + \mathcal{W}_{011}x_2x_3 + \mathcal{W}_{111}x_1x_2x_3$$

- Interactions of features are fully captured
- $\mathcal{W}$ contains weights of interactions and increases exponentially with the dimension of input

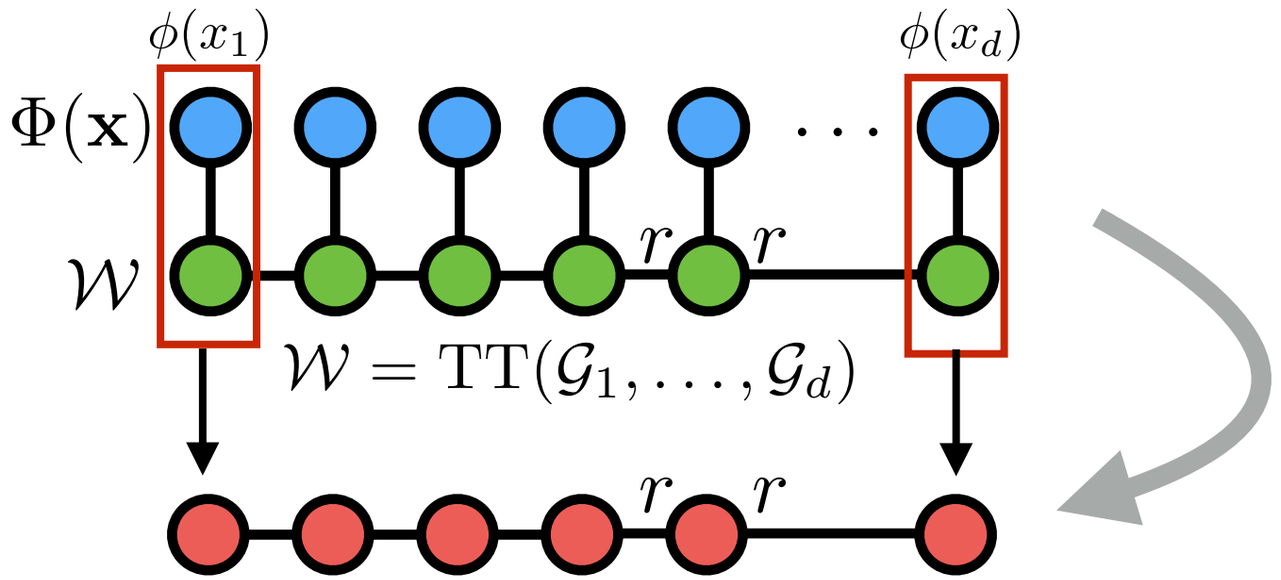
Exponential Machines (Novikov et al., ICLR workshop 2017)

Exponential Machines

$$f(\mathbf{x}) = \sum_{i_1=1}^{p} \cdots \sum_{i_d=1}^{p} \mathcal{W}_{i_1,\ldots,i_d} \prod_{k=1}^{d} x_k^{i_k - 1}$$

	Memory	Computation
Normal	p^d	$\mathcal{O}(dp^d)$
With TN	dpr^2	$\mathcal{O}(dpr^2)$

$f(\mathbf{x}) =$



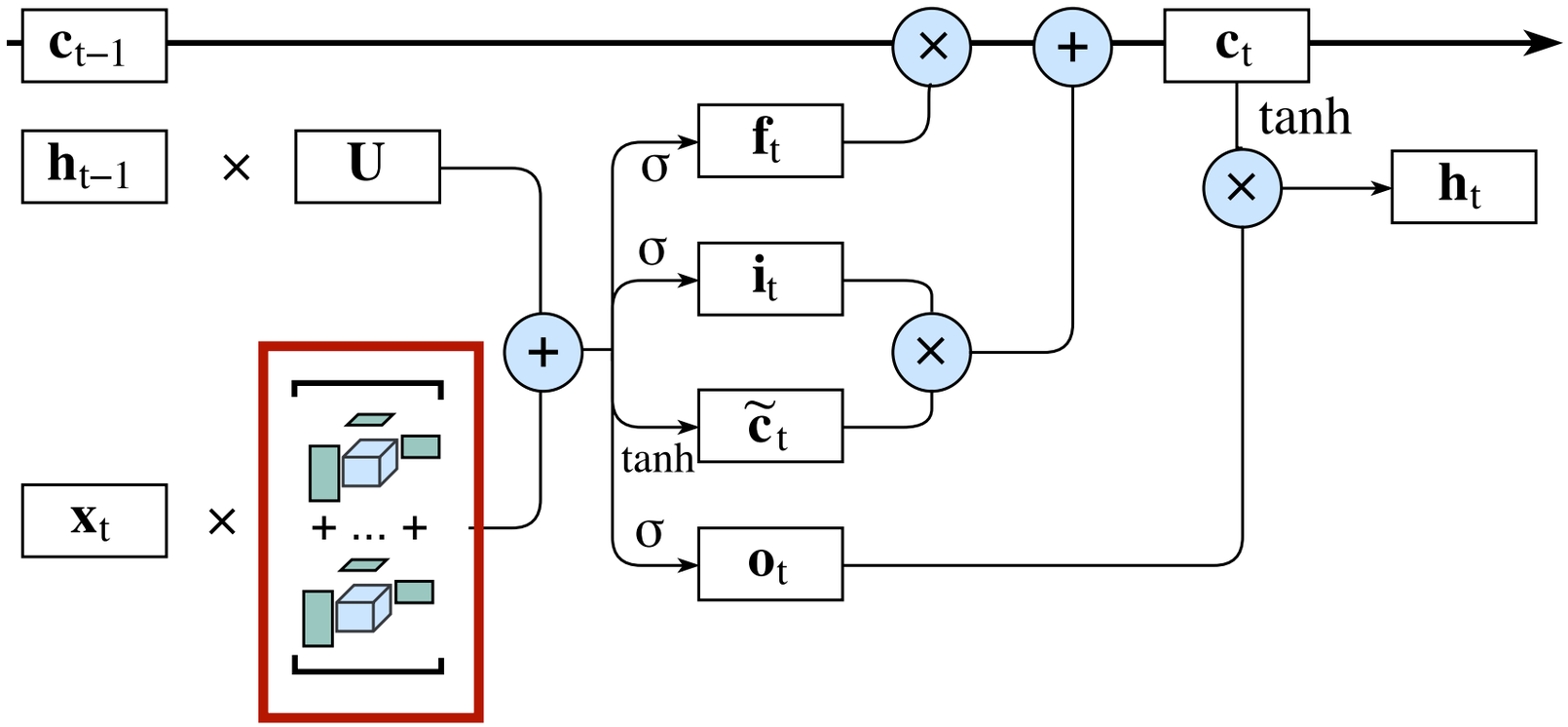
$$f(\mathbf{x}) = \left(\sum_{i_1=1}^{p} x_i^{i_1-1} \mathcal{G}_1[i_1]\right) \cdots \left(\sum_{i_d=1}^{p} x_d^{i_d-1} \mathcal{G}_d[i_d]\right) = \widetilde{\mathbf{G}}_1 \widetilde{\mathbf{G}}_2 \cdots \widetilde{\mathbf{G}}_d$$

- TN representation is highly efficient for very large d, p
- Low-rank structure of TN (i.e., TT-rank) controls expressive power

Exponential Machines (Novikov et al., ICLR workshop 2017)

Model Compression via Different TN Models

- Various DL models + Various TF models



Block term decomposition (Ye et al., 2018)

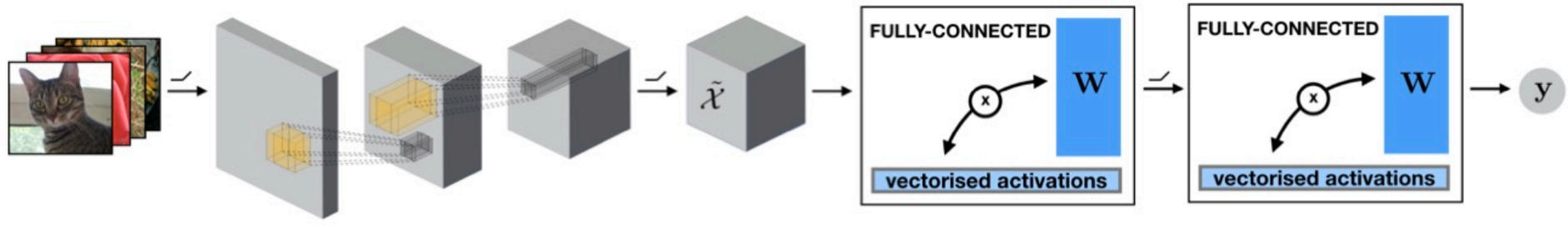
Tesnor ring decomposition (Pan et al., 2019)

…

Learning compact recurrent neural networks with block-term tensor decomposition (Ye et al. CVPR 2018)
Compressing recurrent neural networks with tensor ring for action recognition (Pan et al. AAAI 2019)

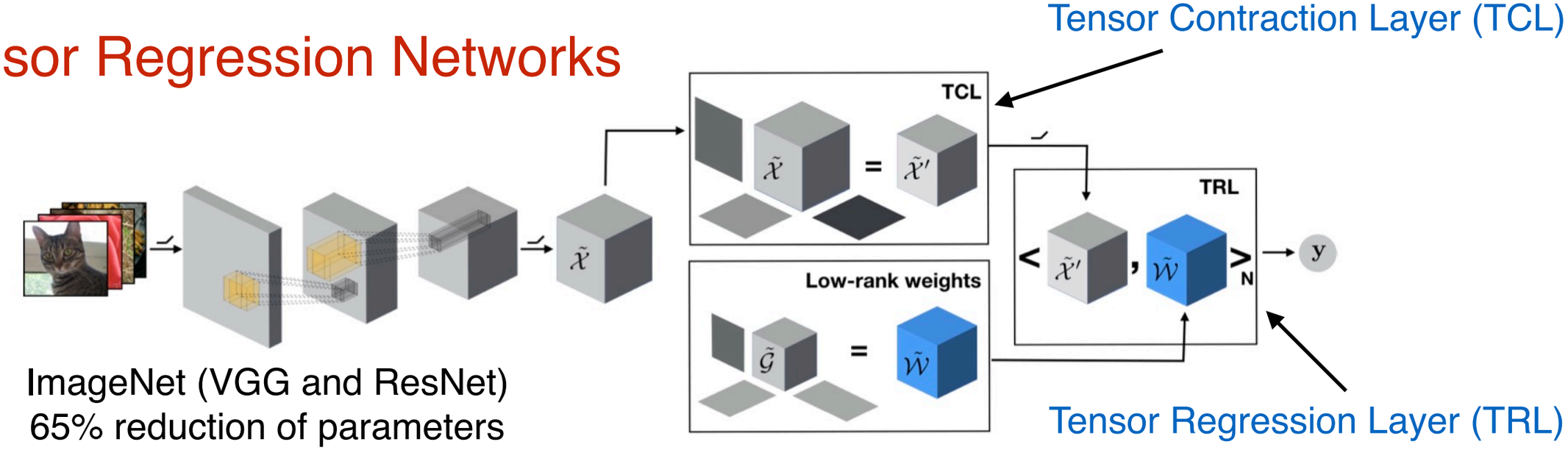
Model Compression: Tensor Contraction

Standard CNN



- Flattening followed by fully-connected layers discards multilinear structure in the activations and require many parameters

Tensor Regression Networks

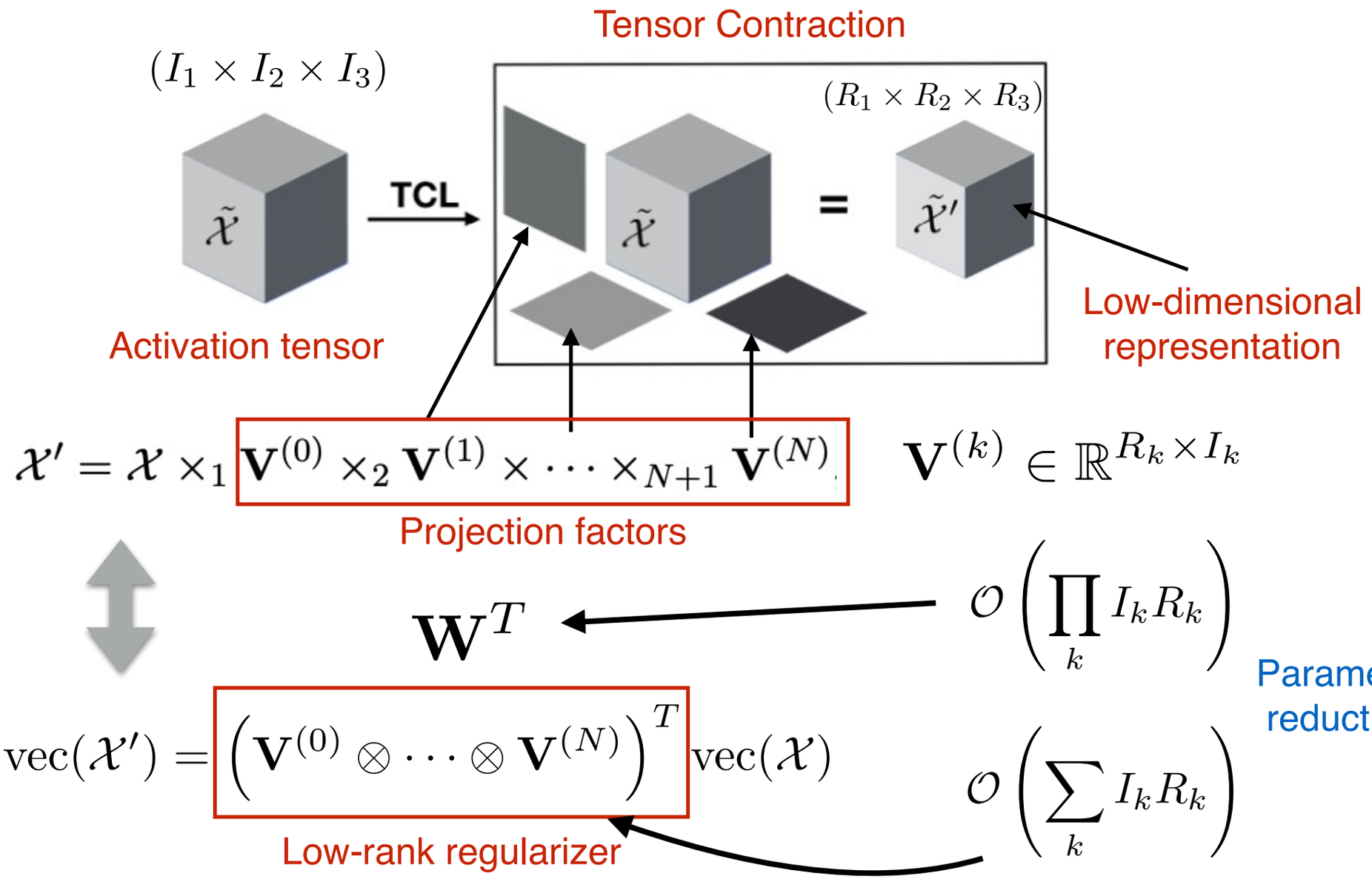


ImageNet (VGG and ResNet)
65% reduction of parameters

- Preserving multilinear structure by TCL, low-rankness on weights by TRL
- Reduce parameters while maintaining or increasing accuracy

Tensor Regression Networks (Kossaifi et al., JMLR 2020)

Tensor Contraction Layer

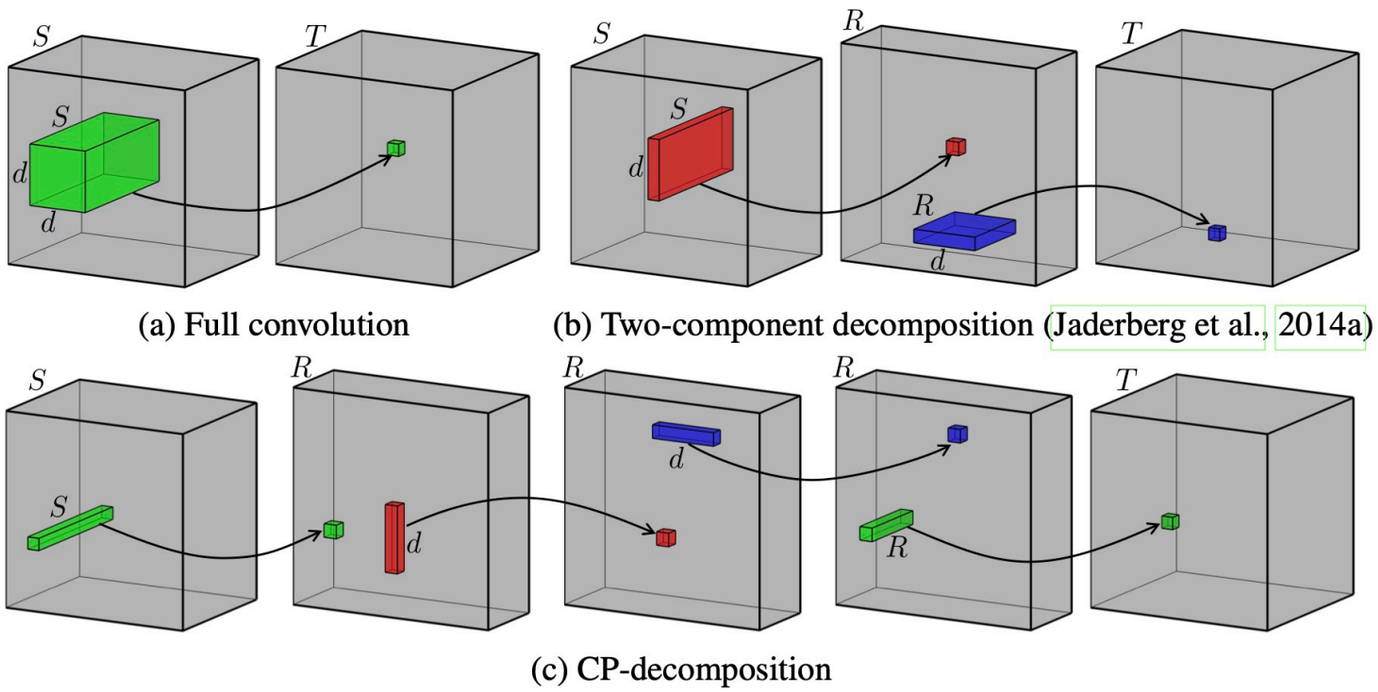


- TCLs require fewer parameters and less computation, while preserving the multilinear structure of the activation tensor

Speed-up and Compression of CNNs

- 4-order kernel tensor is represented by CPD

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



(a) Full convolution (b) Two-component decomposition (Jaderberg et al., 2014a)

(c) CP-decomposition

ImageNet (AlexNet)
4x speedup
1% accuracy drop

Kernel Size Input channels

$$(d \times d \times S \times T)$$

Spatial Output channels

Number of Parameters & Computations

$$STd^2 \Rightarrow R(S+2d+T)$$

Speeding-up convolutional neural networks using fine-tuned cp-decomposition (Lebedev et al. ICLR 2015)

CNN with Low-rank Regularization

- Standard: Pre-trained network + Tensor Approximation with fine-tuning

- Re-parametrize kernel naturally enforces low-rank constraint
- Convolutional kernel in CNN is 4D tensor

$$\mathcal{W} \in \mathbb{R}^{N \times d \times d \times C}$$

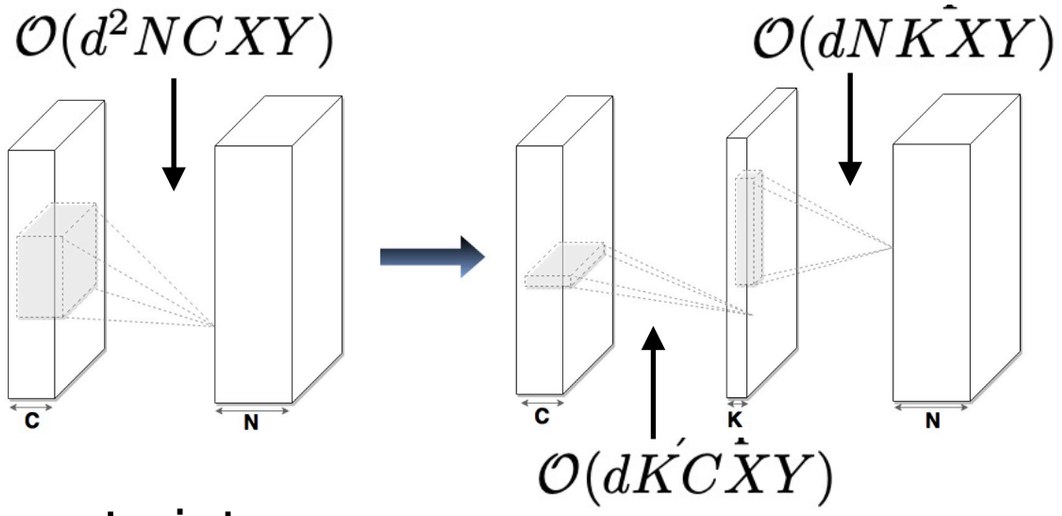
N: Number of output,
C: number of input feature maps

- Train low-rank CNNs from scratch

$$\tilde{\mathcal{W}}_n^c = \sum_{k=1}^{K} \mathcal{H}_n^k (\mathcal{V}_k^c)^T$$

$$\mathcal{H} \in \mathbb{R}^{N \times 1 \times d \times K}$$

$$\mathcal{V} \in \mathbb{R}^{K \times d \times 1 \times C}$$



- Lower expressive power due to low-rank constraints

- Significant speedup, sometimes even improved accuracy (91.31% on CIFAR-10)

Convolutional neural networks with low-rank regularization (Tai et al., ICLR 2016)

CNN Compression by Tucker

Convolution Layer

$$\mathcal{Y}_t = \mathcal{X} * \mathcal{K}_t \quad t \in [1, T]$$

$\mathcal{X}$: $(H \times W \times S)$, $\mathcal{Y}_t$: $(H \times W \times T)$, $\mathcal{K}_t$: $(d \times d \times S)$

Size of Kernel $\mathcal{K}$

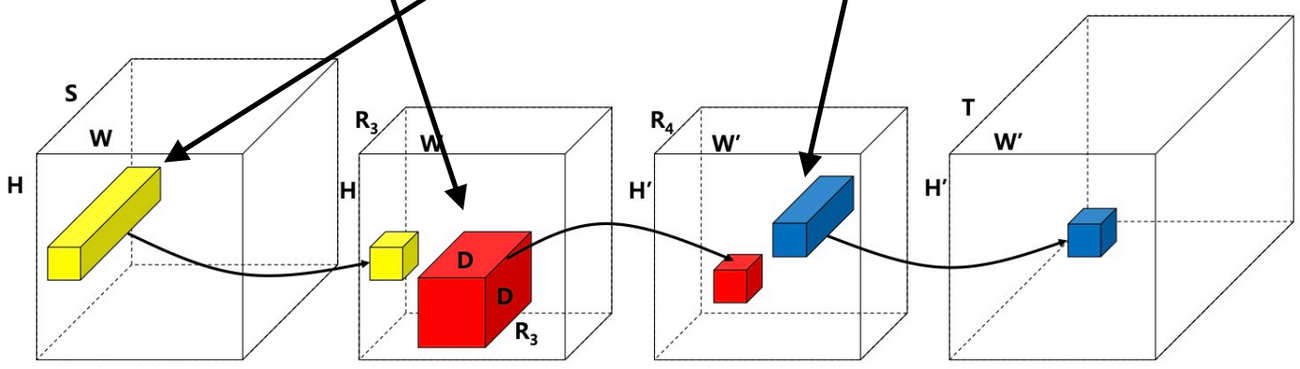
Input channels

$(d \times d \times S \times T)$

Spatial: $d \times d$; Output channels: T

Tucker 2 decomposition

$$\mathcal{K} = \mathcal{G} \times_3 \mathbf{V}_S \times_4 \mathbf{V}_T$$

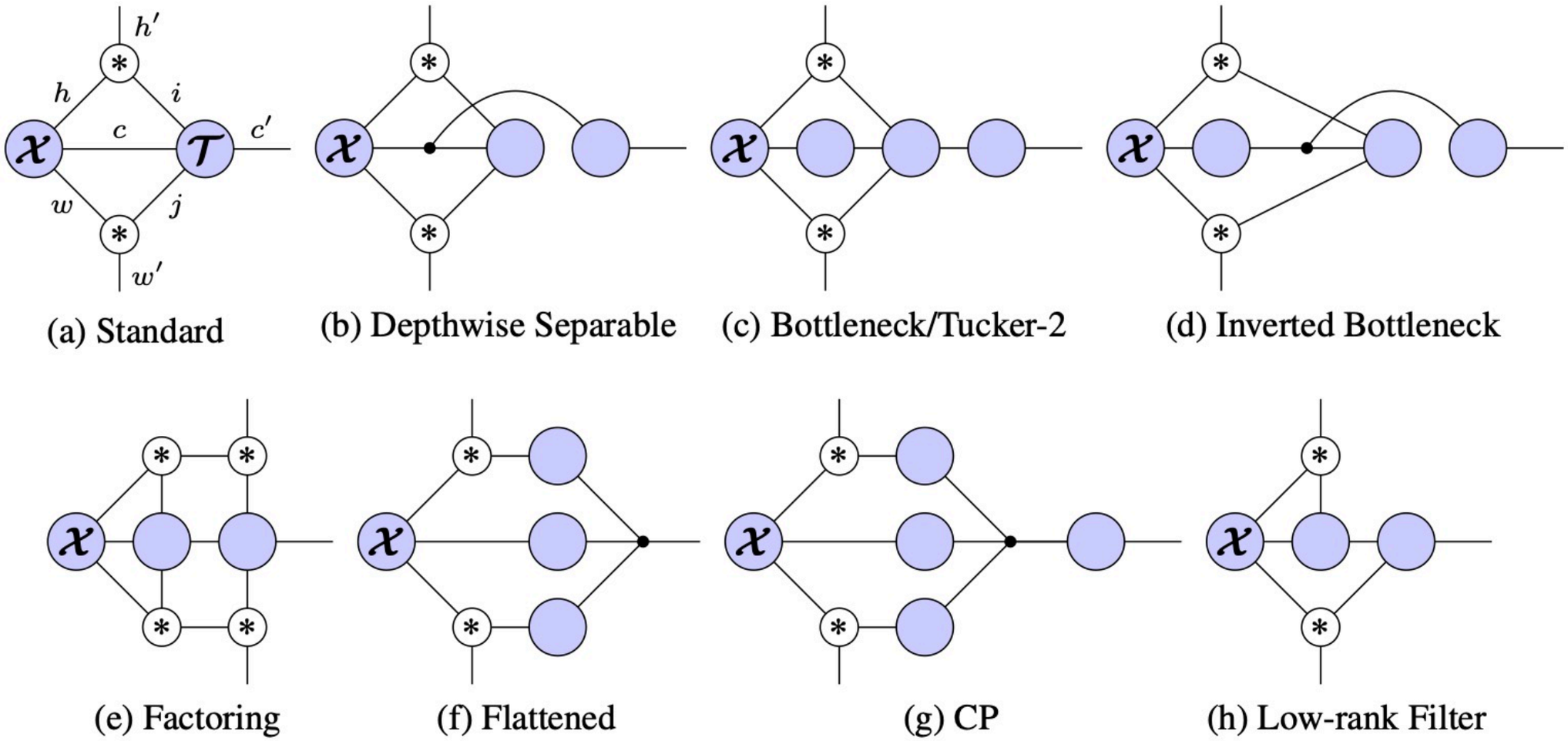


3.68x speedup for VGG

- Tensor Ranks are hyper-parameters
- Control trade-off between kernel size and accuracy loss
- Kernel size corresponds to speed and model size

Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications (Kim et al., ICLR 2016)

TN Decompositions for CNN Kernels



- Interpret algorithms as different TN decompositions
- Explore unexplored TN decompositions

Einconv: Exploring Unexplored Tensor Network Decompositions for Convolutional Neural Networks (Hayashi et al., NeurIPS 2019)

Is Weight Kernel Low-rank?

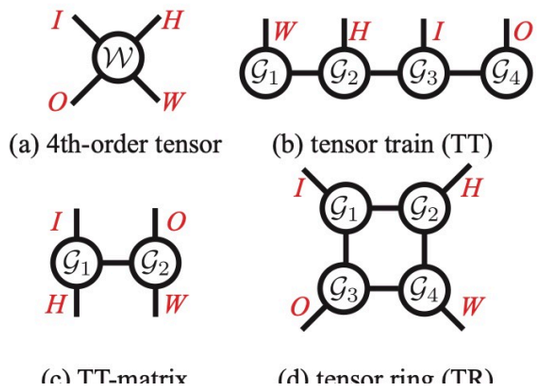
Low-rank regularizer

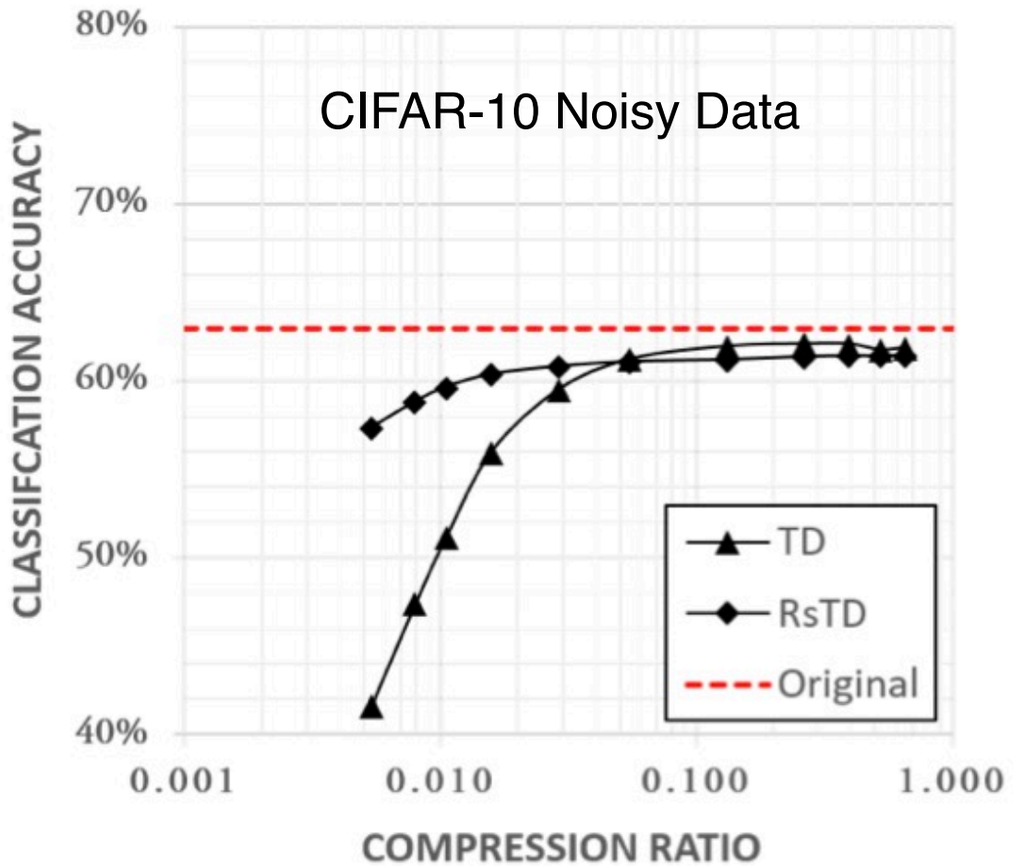
$$\mathcal{W} = \mathrm{TD}(\mathcal{G}_1, \ldots, \mathcal{G}_N)$$

Random shuffling (Rs) low-rank

$$\mathcal{W} = R(\mathrm{TD}(\mathcal{G}_1, \ldots, \mathcal{G}_N))$$

- RsTD performs better in higher compression rate
- More robust to noisy data
- CNN kernels have inherent general low-rank structure

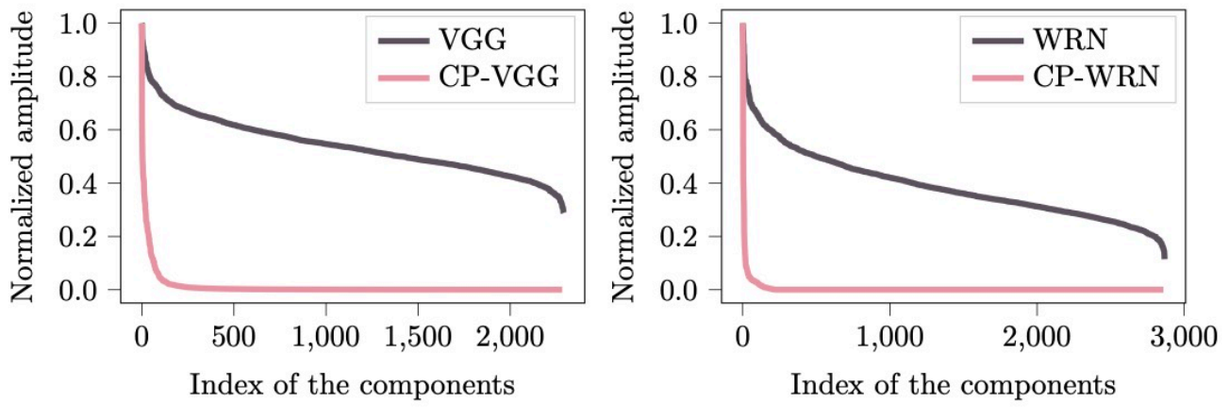




Low-rank embedding of kernels in convolutional neural networks under random shuffling (Li et al., ICASSP 2019)

Generalization of Compressed CNN

- Weight matrices/kernels of well-trained models are not necessarily low-rank
- Re-parametrizes weight tensors as CPD
- CP layer reduces the generalization error
- Higher compression -> smaller generalization error bound



$$\mathcal{K} = \sum_{r=1}^{R} \lambda_r \mathbf{v}_r^{(1)} \otimes \cdots \otimes \mathbf{v}_r^{(N)}$$

Generalization Error

Empirical Loss

CP Rank

$$L_0(\hat{\mathbb{M}}) \leq \hat{L}_\gamma(\mathbb{M}) + \tilde{O}\left(\sqrt{\frac{\sum_{k=1}^{n} \hat{R}^{(k)}\left(s^{(k)} + o^{(k)} + k_x^{(k)} k_y^{(k)} + 1\right)}{m}}\right)$$

Understanding Generalization in Deep Learning via Tensor Methods (Li et al., AISTATS 2020)

Tensor Networks for Theoretical Study of DNNs

Theoretical understanding of DNNs is limited

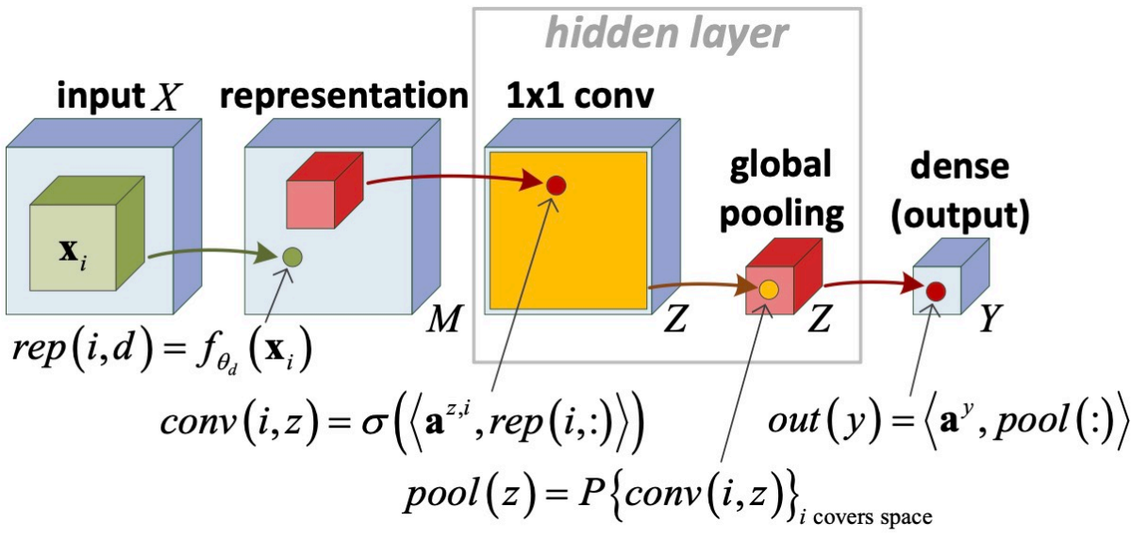
- How to analyze expressive power of DNNs?
- How is depth efficiency of DNNs?
- Generalization error bound
- …

Tensor Networks can be used as a tool for theoretical analysis of DNNs

- Relation and equivalence between TNs and DNNs
- Expressive power of network can be measured by rank of tensor networks
- Expressive analysis of DNNs by analyzing TN ranks
- Provide valuable insights on how they can be improved

Equivalence between CNNs and TNs

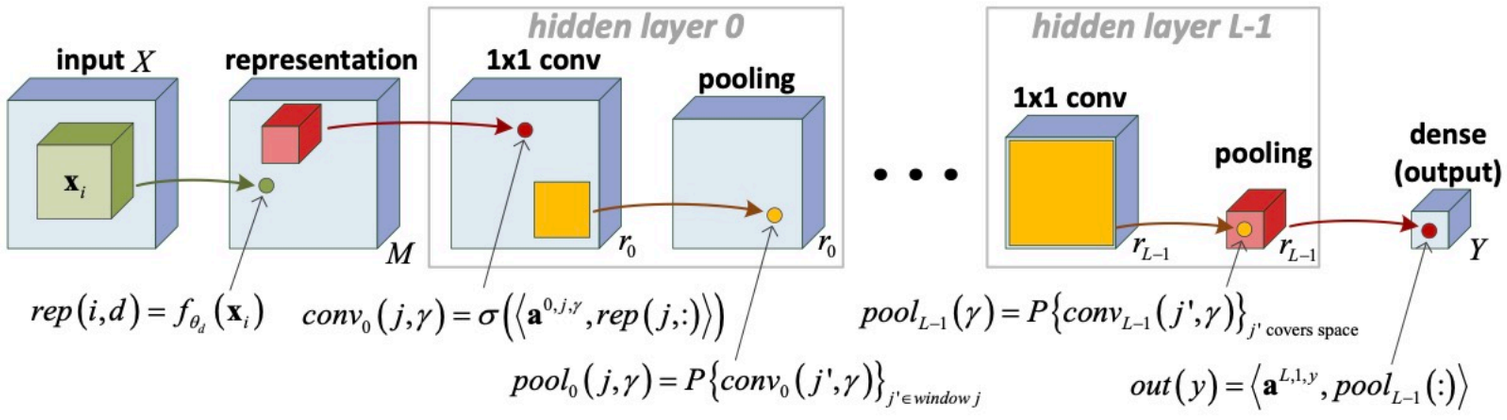
- Shallow ConvNet corresponds to CP decomposition

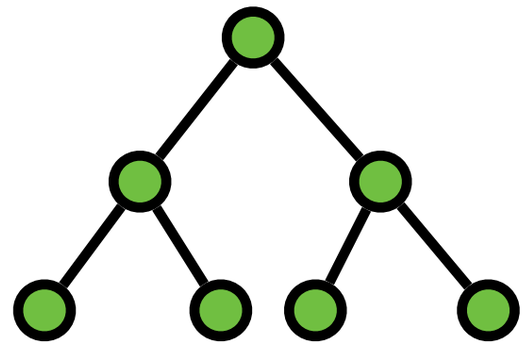


Score function:

$$\mathcal{A}\left(h_y^S\right) = \sum_{z=1}^{Z} a_z^y \cdot \left(F\mathbf{a}^{z,1}\right) \otimes_g \cdots \otimes_g \left(F\mathbf{a}^{z,N}\right)$$

- Deep CNN network corresponds to hierarchical Tucker decomposition



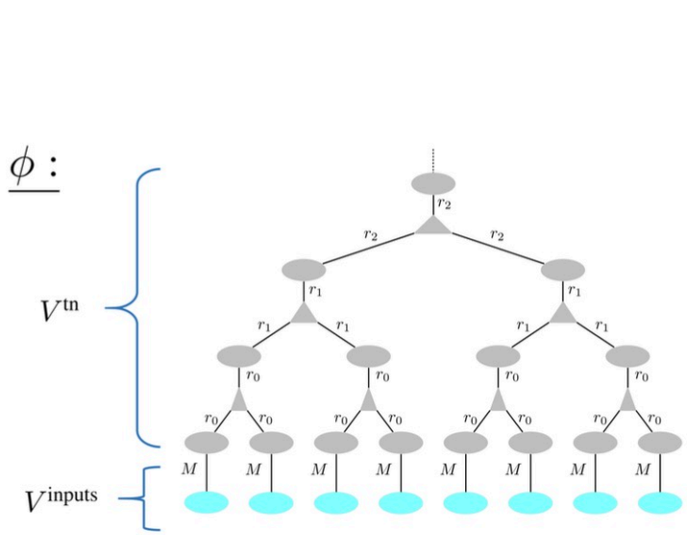


hierarchical Tucker

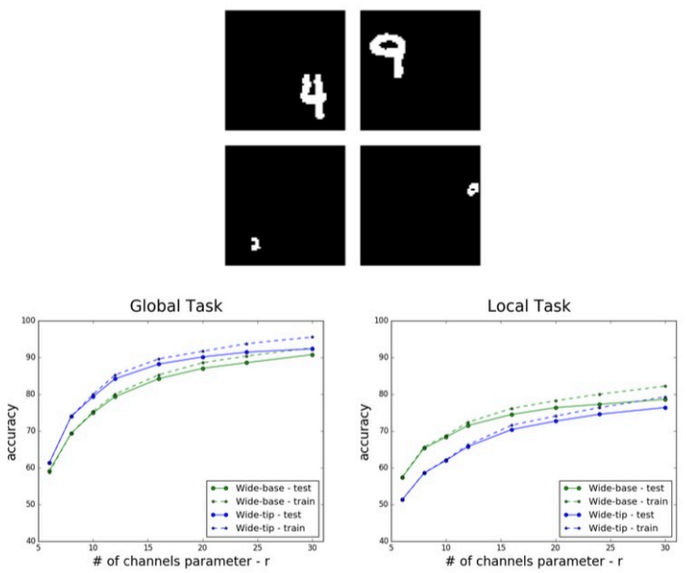
Convolutional Rectifier Networks as Generalized Tensor Decompositions (Cohen et al., ICML 2016)

Deep Learning and Quantum Entanglement

- ConvAC (linear activations + product pooling layers introducing non-linearity) vs. CNN (ReLU + average/max pooling)
- Tensor Network rank as capacity of model
- Experiment on “inductive bias” of model architecture



Tree Network as a Deep Neural Net



Inductive Bias Experiment

Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design (Levine et al., ICLR 2018)

Theoretical Understanding of CNN

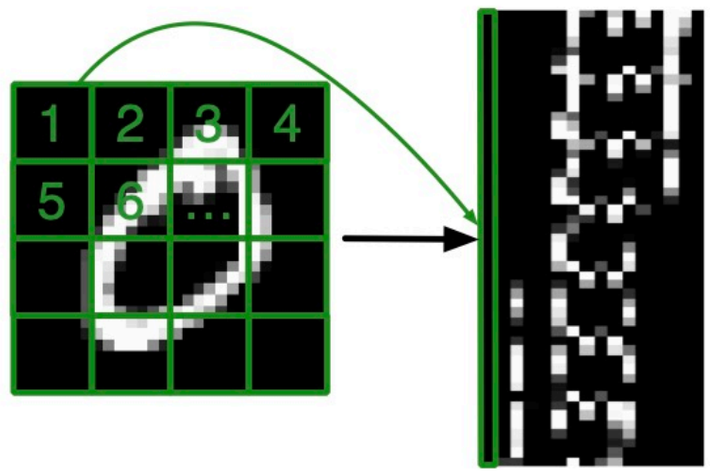
- Deep CNNs are exponentially more expressive than their shallow counterpart
- Depth efficiency: deep network of polynomial size = shallow network of exponential size
- Convolutional rectifier networks are universal with max pooling, but not with average pooling
- SimNets (linear activation + product pooling) have an advantage over the prevalent convolutional rectifier networks

On the expressive power of deep learning: A tensor analysis (Cohen et al., COLT 2016)

Relations between TNs and DNNs

- Representing an image as sequential data

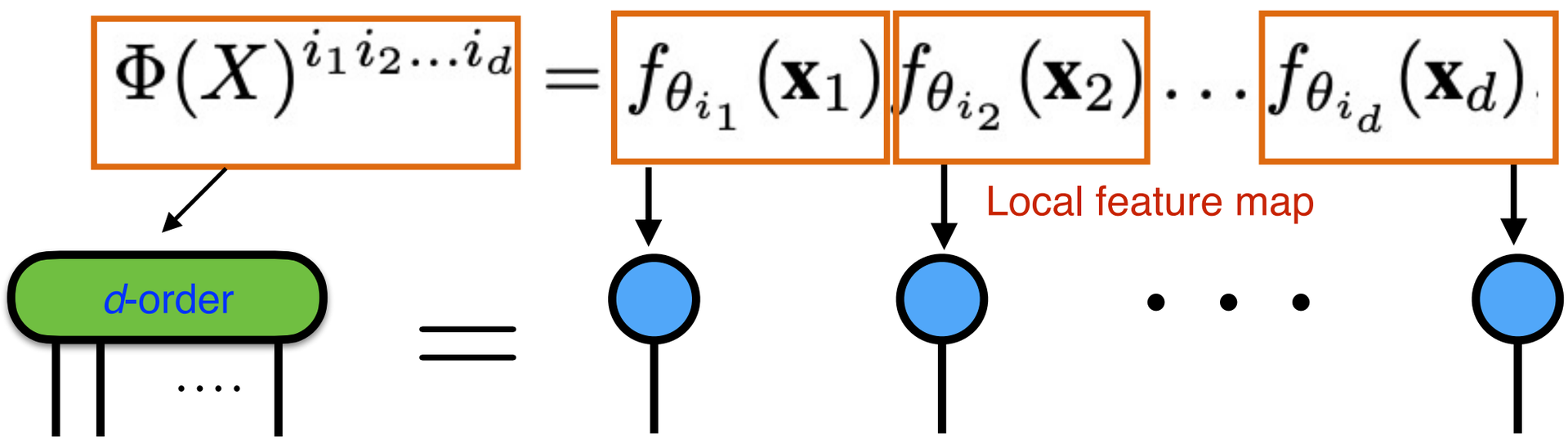
$$X^{(b)} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_d), \quad \mathbf{x}_k \in \mathbb{R}^n$$



- Representation mapping

$$f_\theta(\mathbf{x}) = \sigma(A\mathbf{x} + b)$$

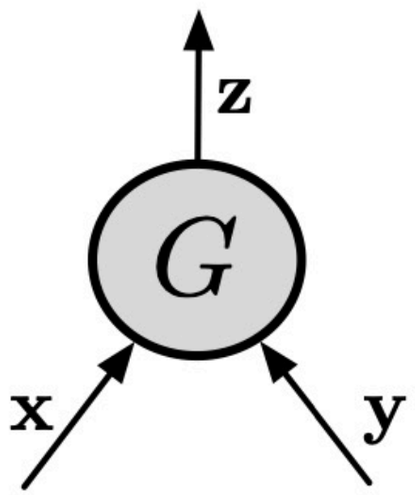
- Feature tensor



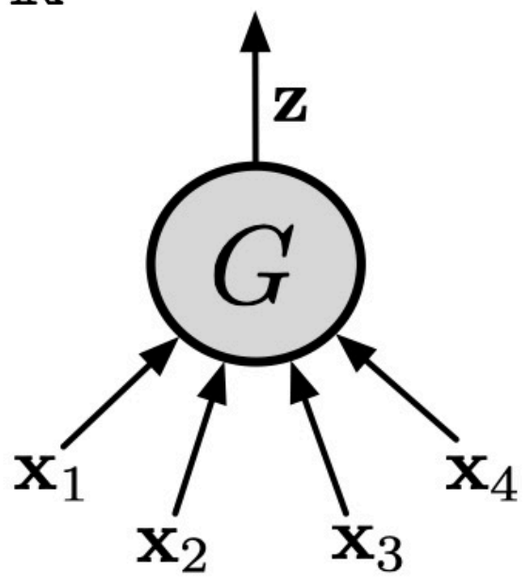
Expressive Power of Recurrent Neural Networks (Khrulkov et al., ICLR 2018)

Multilinear Unit in Tensorial Networks

$$G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^\kappa$$



(a) Bilinear unit



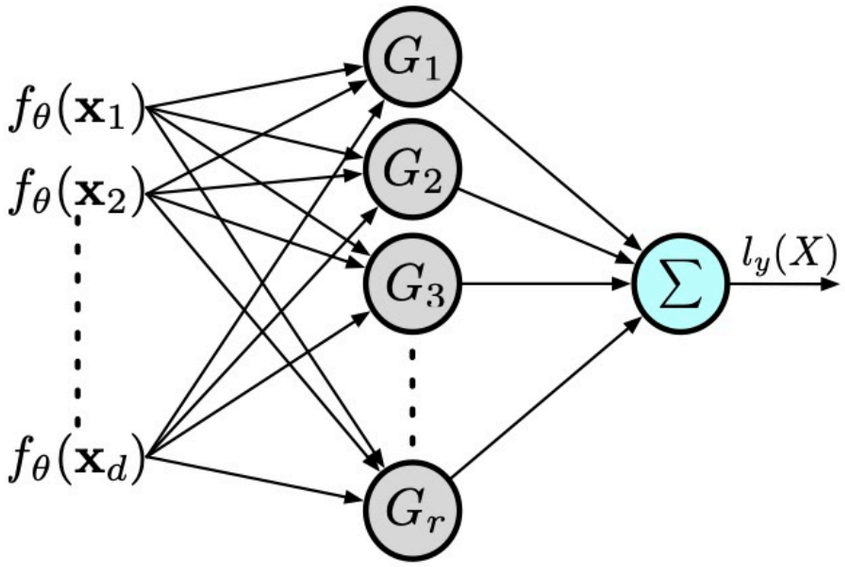
(b) Multilinear unit

$$G(\mathbf{x}, \mathbf{y}) = \mathbf{z},$$
$$\mathbf{z}^k = \sum_{i,j} G^{ijk} \mathbf{x}^i \mathbf{y}^j.$$

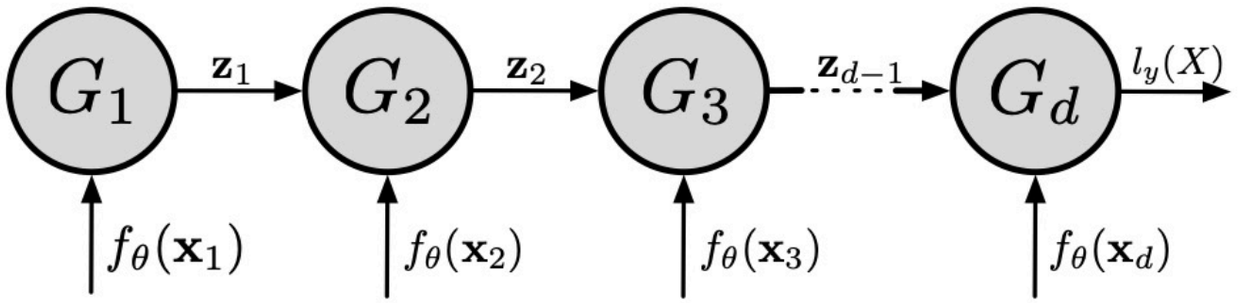
$$G(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d) = \mathbf{z}$$
$$\mathbf{z}^j = \sum_{i_1, i_2, \ldots, i_d} G^{i_1 i_2 \ldots i_d j} \mathbf{x}_1^{i_1} \mathbf{x}_2^{i_2} \ldots \mathbf{x}_d^{i_d}.$$

Expressive Power of Recurrent Neural Networks (Khrulkov et al., ICLR 2018)

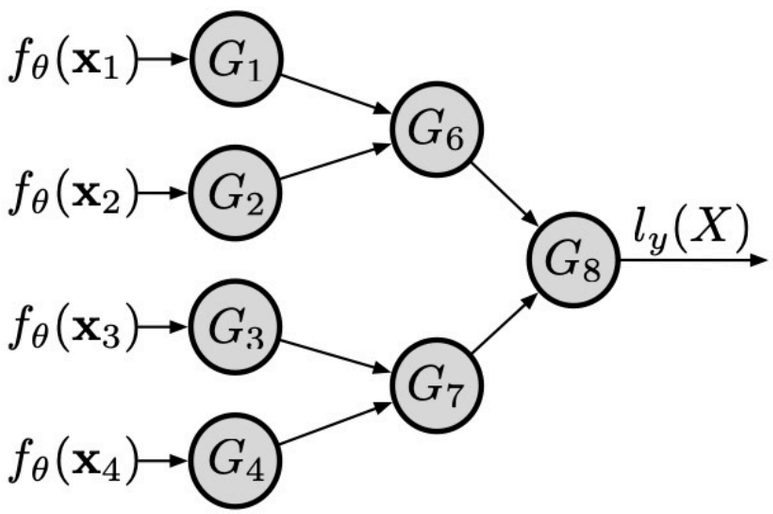
Relations between TNs and DNNs



(a) CP-Network



TT-Network



(b) HT-Network

Score function: $l_y(X) = \langle \mathcal{W}_y, \Phi(X) \rangle$

Tensor Decompositions	**Deep Learning**
CP-decomposition	shallow network
TT-decomposition	RNN
HT-decomposition	CNN
rank of the decomposition	width of the network

Expressive Power of Recurrent Neural Networks (Khrulkov et al., ICLR 2018)

Theoretical Results

	TT-Network	**HT-Network**	**CP-Network**
TT-Network	r	$r^{\log_2(d)/2}$	r
HT-Network	r^2	r	r
CP-Network	$\geq r^{\frac{d}{2}}$	$\geq r^{\frac{d}{2}}$	r

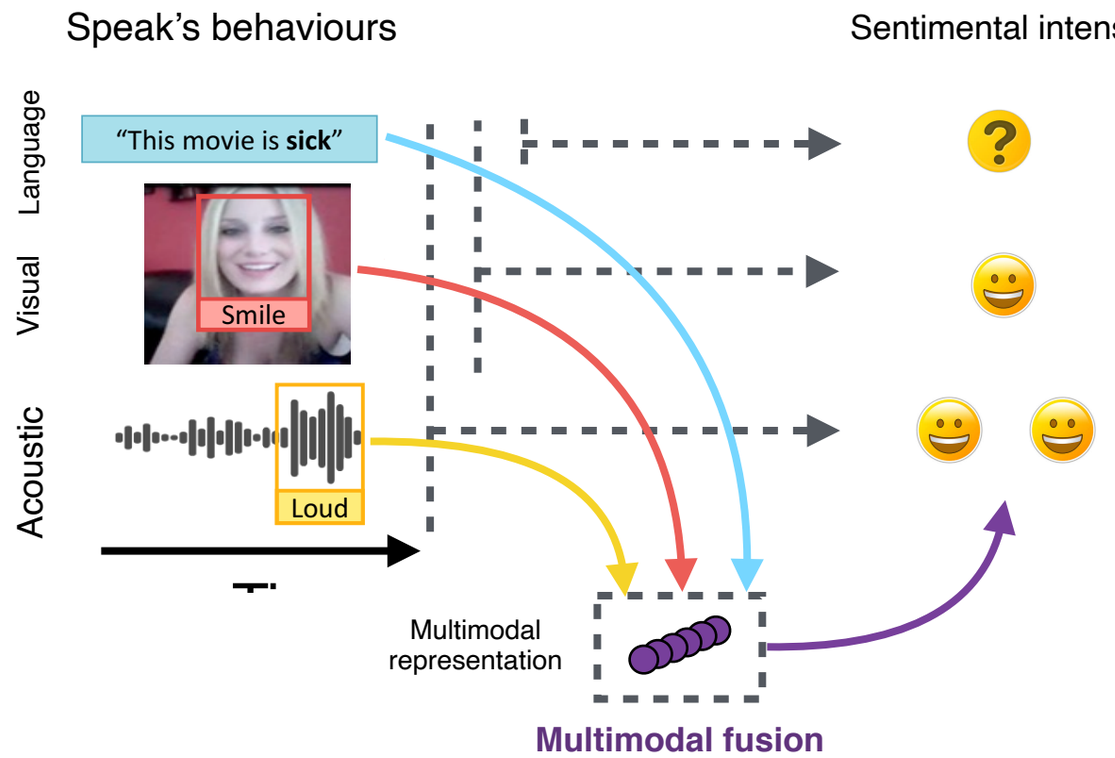
Given a network of width r shown in a column, what is the upper bound on the width of equivalent networks shown in rows

- TT is exponentially more expressive than CP
- A shallow network of exponentially large width is required to mimic a recurrent neural network

Expressive Power of Recurrent Neural Networks (Khrulkov et al., ICLR 2018)

Multimodal Learning

- Multimodal sentimental classification (Acoustic, Visual, Language)

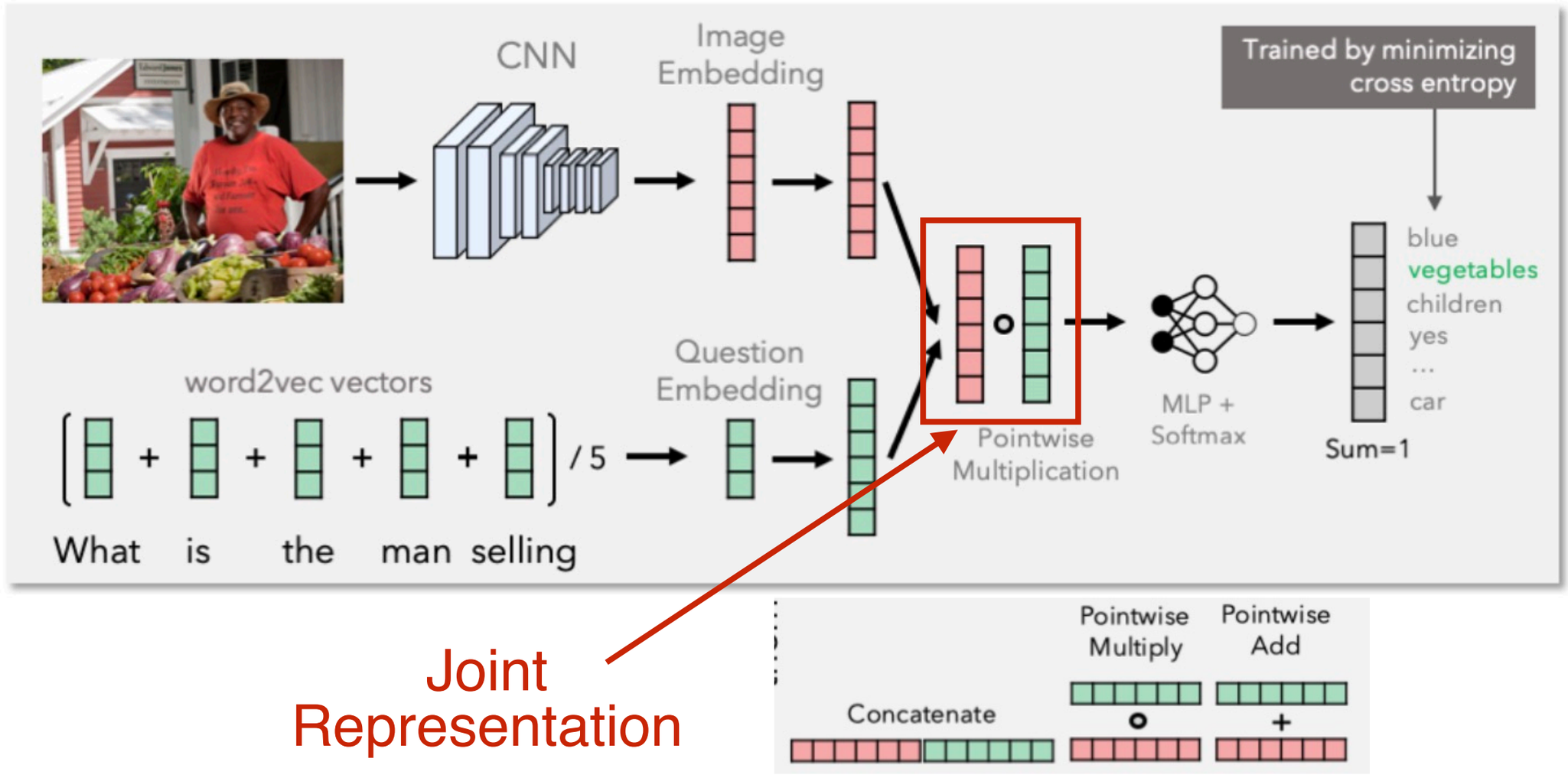


- Visual question answering (Image + Language)

Q : "What do you see?" (Ground Truth : a_3)
a_1 : "A courtyard with flowers"
a_2 : "A restaurant kitchen"
a_3 : "A family with a stroller, tables for dining"
a_4 : "People waiting on a train"

VQA: Visual Question Answering

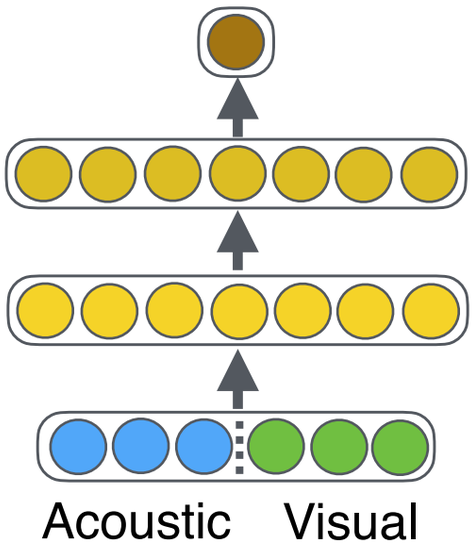
- Given an image and a natural language question about the image, the task is to provide an accurate natural language answer.



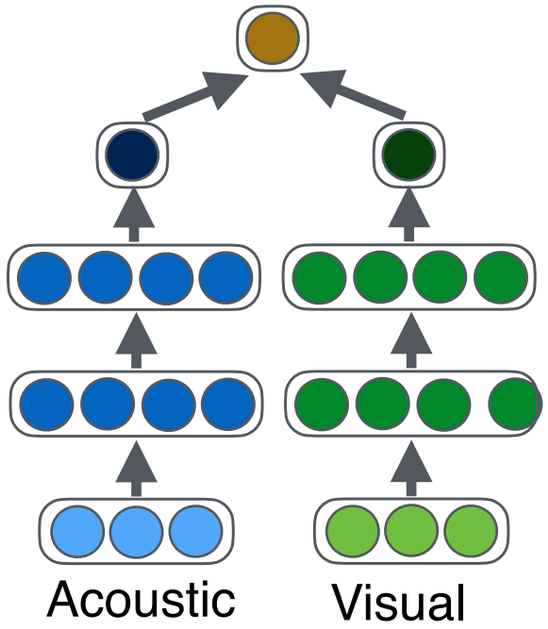
VQA: Visual Question Answering (Agrawal et al., ICCV 2015)

Types of Multimodel Fusion

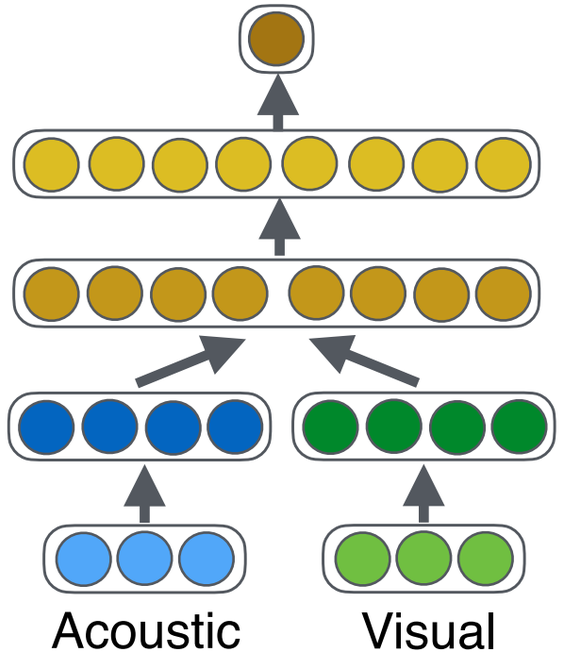
Data-level fusion



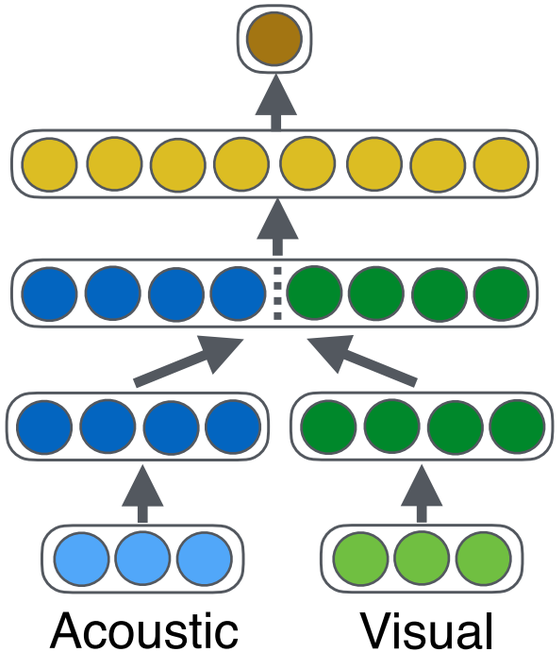
Decision-level fusion



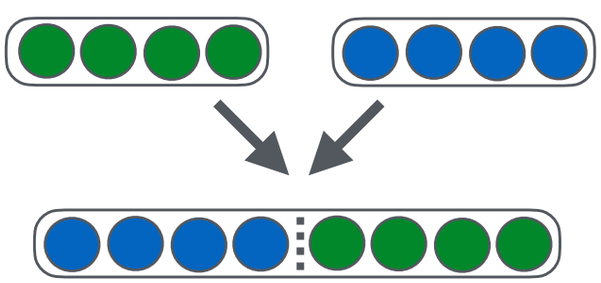
Shared Representation



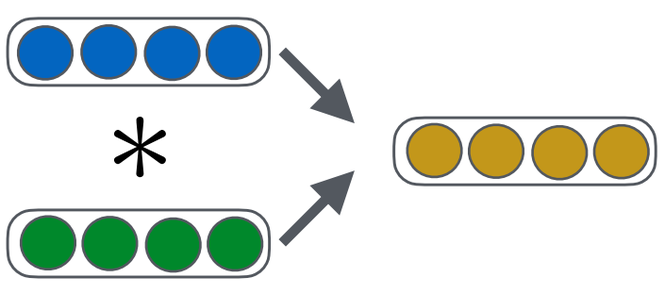
Feature-level fusion



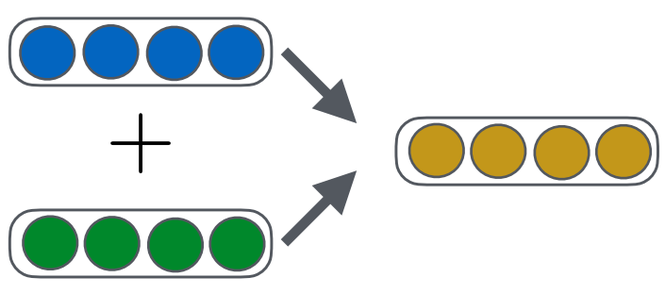
Concatenation



Element-wise multiplication

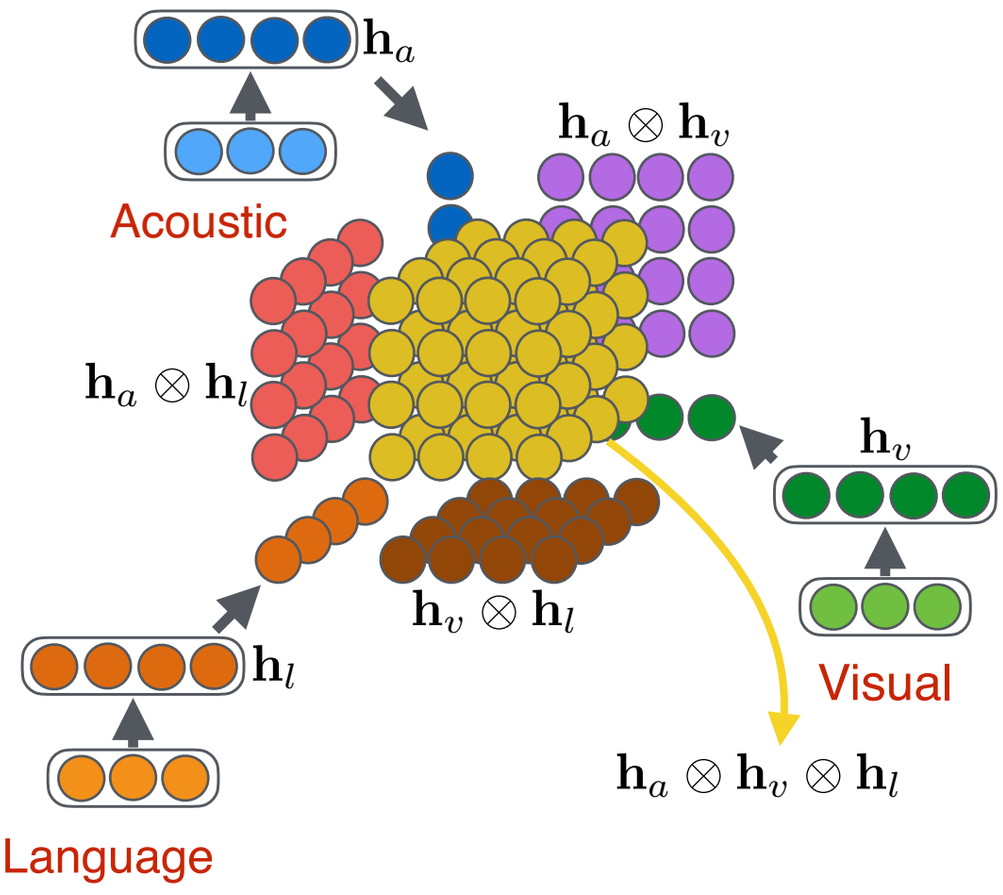


Element-wise add



Tensor Fusion Network for Multimodal Learning

- Trilinear fusion: linear, bilinear and trilinear interactions
- Capture cross-modality nonlinear interaction information



$$\mathbf{z}^m = \begin{bmatrix} \mathbf{z}^l \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^v \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^a \\ 1 \end{bmatrix}$$

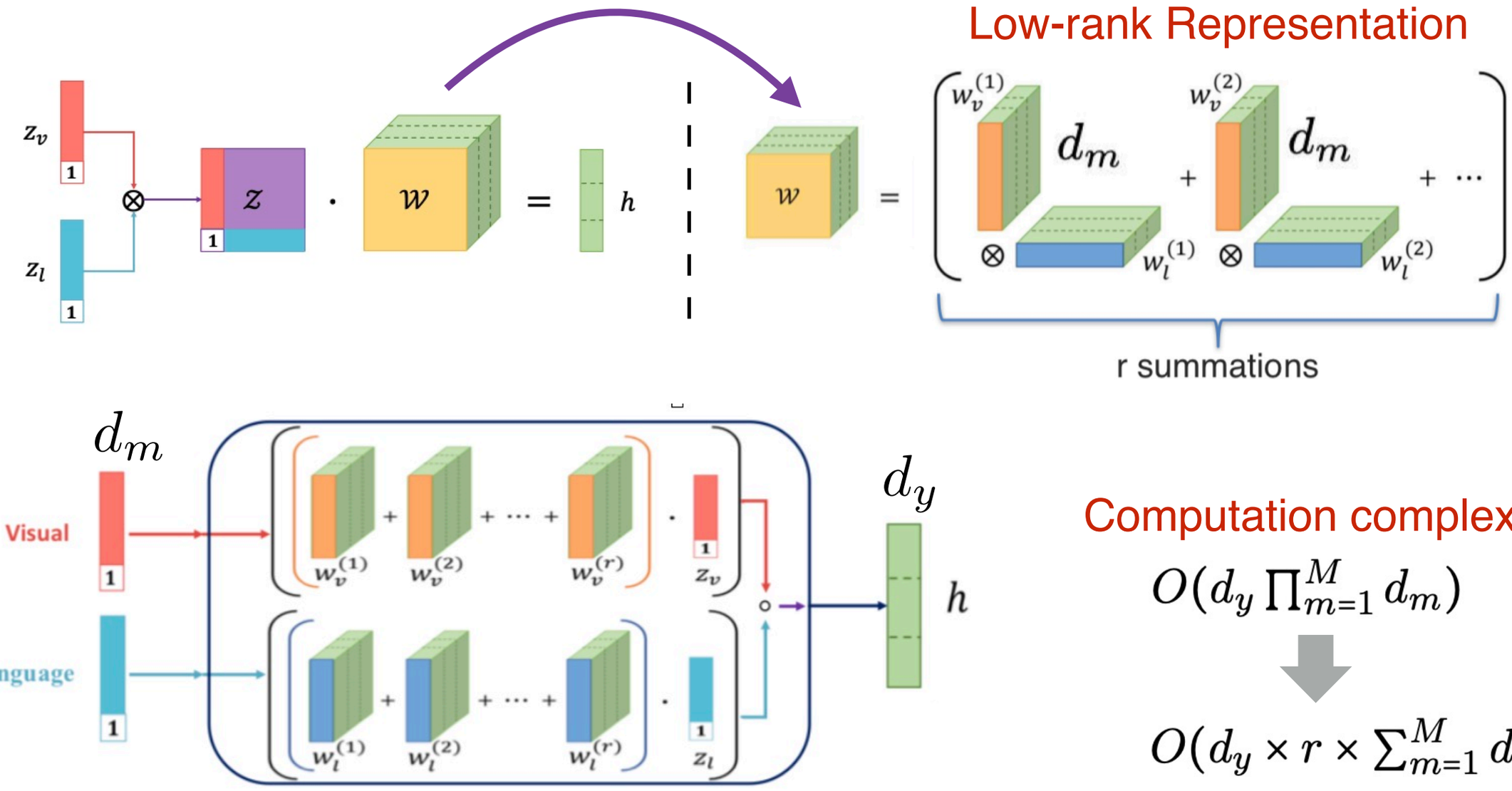
Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
$\text{TFN}_{language}$	74.8	75.6	38.5	0.99	0.61
TFN_{visual}	66.8	70.4	30.4	1.13	0.48
$\text{TFN}_{acoustic}$	65.1	67.3	27.5	1.23	0.36
$\text{TFN}_{bimodal}$	75.2	76.0	39.6	0.92	0.65
$\text{TFN}_{trimodal}$	74.5	75.0	38.9	0.93	0.65
$\text{TFN}_{notrimodal}$	75.3	76.2	39.7	0.919	0.66
TFN	**77.1**	**77.9**	**42.0**	**0.87**	**0.70**
TFN_{early}	75.2	76.2	39.0	0.96	0.63

- Exponential increase of dimensionality and complexity

Tensor Fusion Network for Multimodal Sentiment Analysis (Zadeh et al., EMNLP 2017)

Low-rank Multimodal Fusion

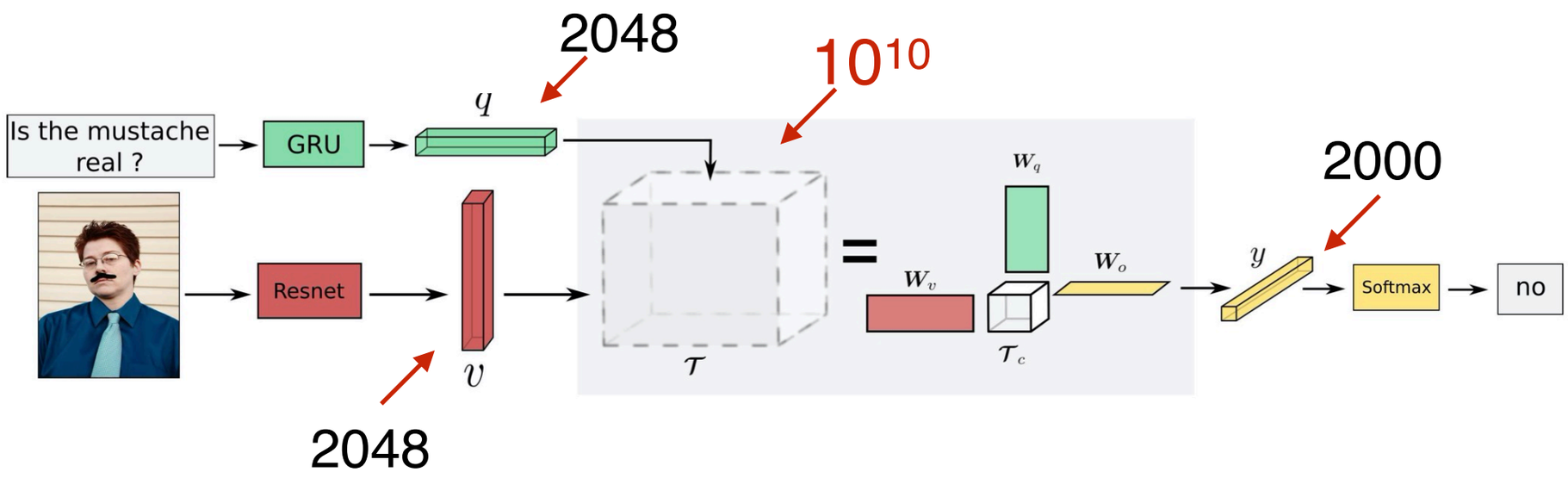
- Efficient computation based low-rank representation



Efficient Low-rank Multimodal Fusion with Modality-Specific Factors (Liu et al., ACL 2018)

Multimodal Tucker Fusion

- Bilinear fusion suffer from huge dimensionality issue



- Tensor decomposition of model parameter $\mathcal{T}$

$$\mathbf{y} = (\boldsymbol{\mathcal{T}} \times_1 \mathbf{q}) \times_2 \mathbf{v} \quad \Rightarrow \quad y = \left((\boldsymbol{\mathcal{T}}_c \times_1 (\mathbf{q}^\top \boldsymbol{W}_q)) \times_2 (\mathbf{v}^\top \boldsymbol{W}_v)\right) \times_3 \boldsymbol{W}_o$$

Tucker Decomposition:

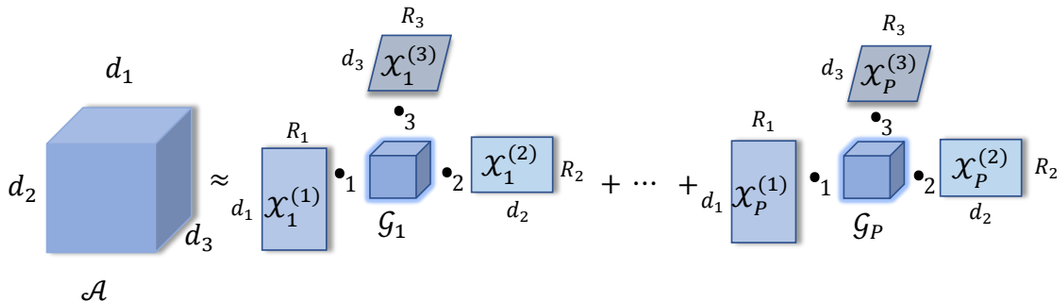
$$\boldsymbol{\mathcal{T}} = ((\boldsymbol{\mathcal{T}}_c \times_1 \boldsymbol{W}_q) \times_2 \boldsymbol{W}_v) \times_3 \boldsymbol{W}_o$$

Tensor Rank is key to balance expressivity and complexity

MUTAN: Multimodal Tucker Fusion for Visual Question Answering (Ben-younes et al., ICCV 2017)

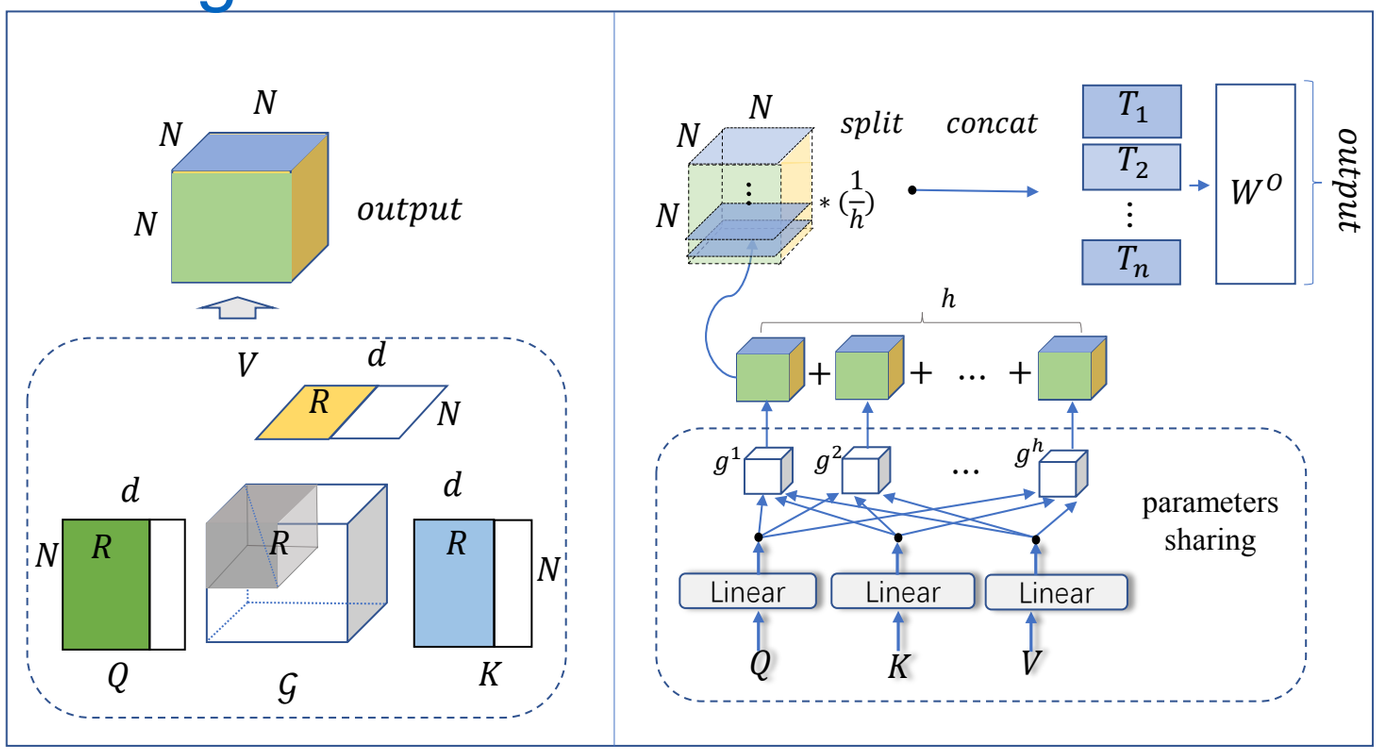
Tensorized Transformer

▸ The BTD model



Sum of Tucker decomposition

▸ Single/multi-head attention



▸ Interaction among (Q)uery, (K)ey and (V)ariable for each head

$$Atten_{TD}(\mathcal{G}; Q, K, V) = \mathcal{G} \bullet_1 Q \bullet_2 K \bullet_3 V$$

$$= \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{m=1}^{M} \mathcal{G}_{ijm} Q_i \circ K_j \circ V_m$$

Ma, Xindian, et al. “A Tenderized Transformer for Language Modeling.”, Neurips 2019

Imperfect Multimodal Time Series Data

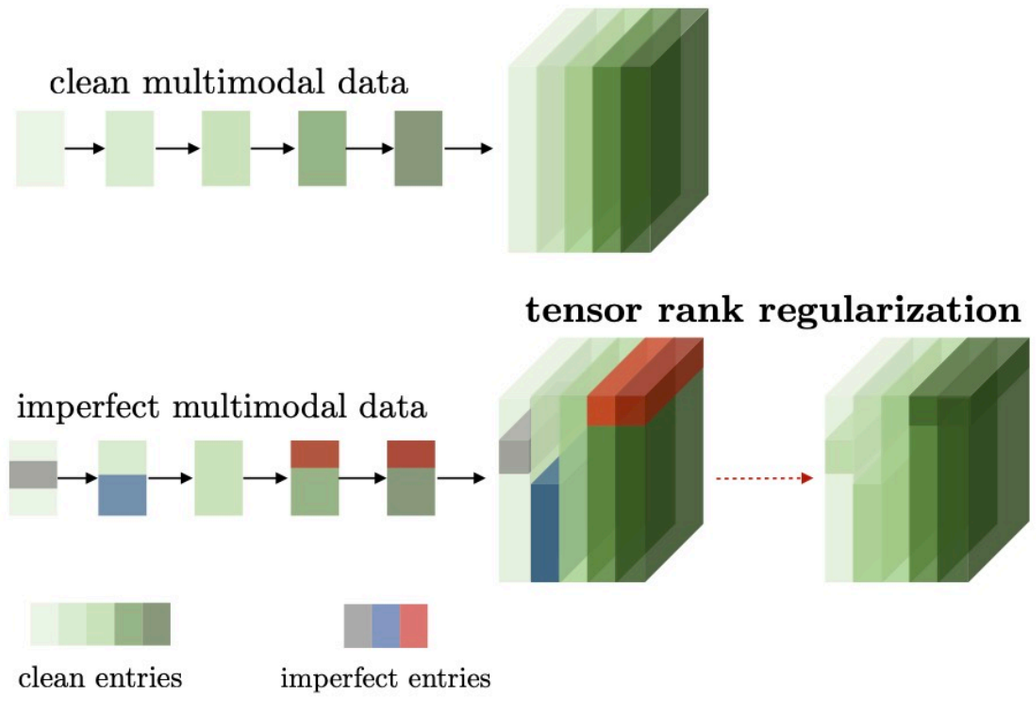
Imperfect data:

- Incomplete due to sensor failure
- Corrupted by random or structured noises

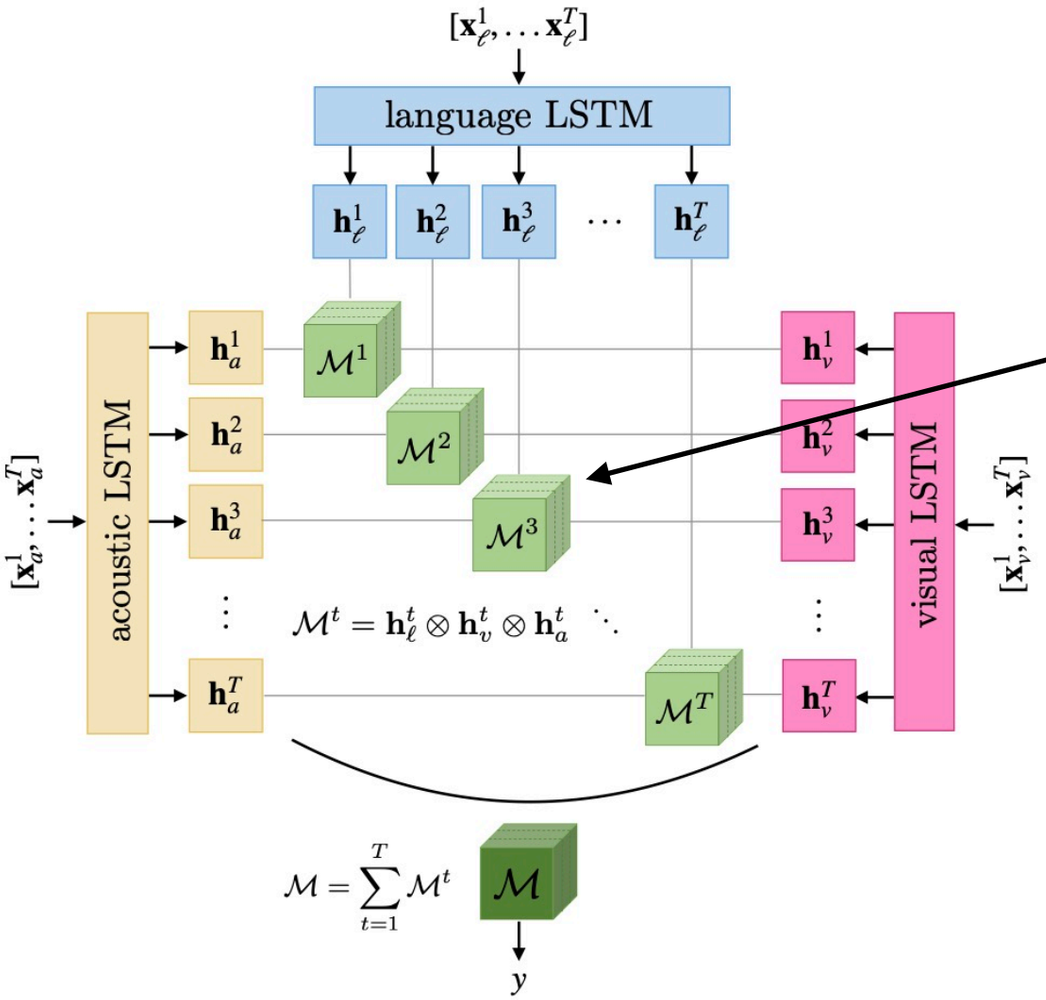
How to learn robust representation from imperfect multimodal data?

- Clean data: multimodal fused tensor exhibits low-rankness across time and modality

- Noisy and incomplete data breaks low-rank structure



Multimodal Learning with Low-rank Regularization



$$\mathcal{M} = \sum_{t=1}^{T} \begin{bmatrix} \mathbf{h}_\ell^t \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{h}_v^t \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{h}_a^t \\ 1 \end{bmatrix}$$

Tensor fusion (Rank-1 tensor)

Low-rank regularizer

$$\|\mathcal{X}\|_* = \inf \left\{ \sum_{i=1}^{r} |\lambda_i| : \mathcal{X} = \sum_{i=1}^{r} \lambda_i \left(\bigotimes_{m=1}^{M} \mathbf{w}_m^i \right), \|\mathbf{w}_m^i\| = 1, r \in \mathbb{N} \right\}$$

Upper bounds on nuclear norm

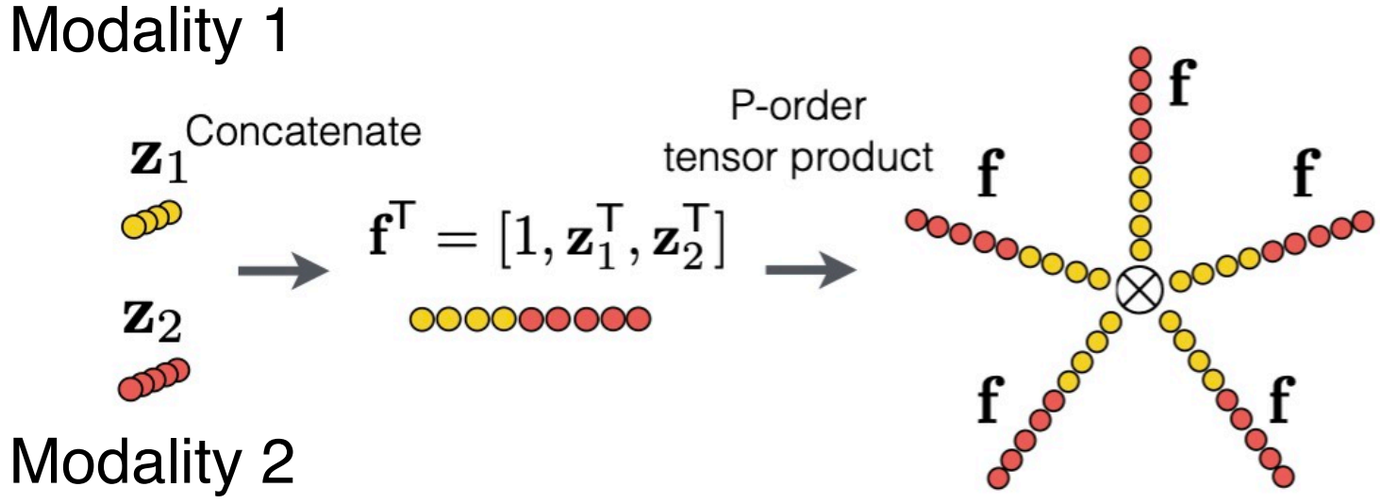
$$\|\mathcal{M}\|_* \leq \sqrt{\frac{\prod_{i=1}^{M} d_i}{\max\{d_1, \ldots, d_M\}}} \|\mathcal{M}\|_F$$

Low-rankness regularizer improves robustness to imperfect data

Learning Representations from Imperfect Time Series Data via Tensor Rank Regularization (Liang et al., ACL 2019)

High-order Tensor Fusion

- Expressive power of tensor fusion is limited
- From first-order to high-oder intra-modal and cross-modal feature interactions
- Tensor Polynomial Pooling (PTP)



Feature Interactions

- Linear
- Bilinear
- Trilinear
- Intra-modal
- High-order

$$\mathcal{F} = \underbrace{\mathbf{f} \otimes \mathbf{f} \otimes \cdots \otimes \mathbf{f}}_{\text{P-order}}$$

Dimensionality increases exponentially with *P*

Example: $\mathcal{F}_{1,1,1} = f_1^3,\ \mathcal{F}_{1,2,1} = f_1^2 f_2,$
$\mathcal{F}_{1,2,3} = f_1 f_2 f_3,\ \mathcal{F}_{1,2,2} = f_1 f_2^2$

$$(md)^P$$

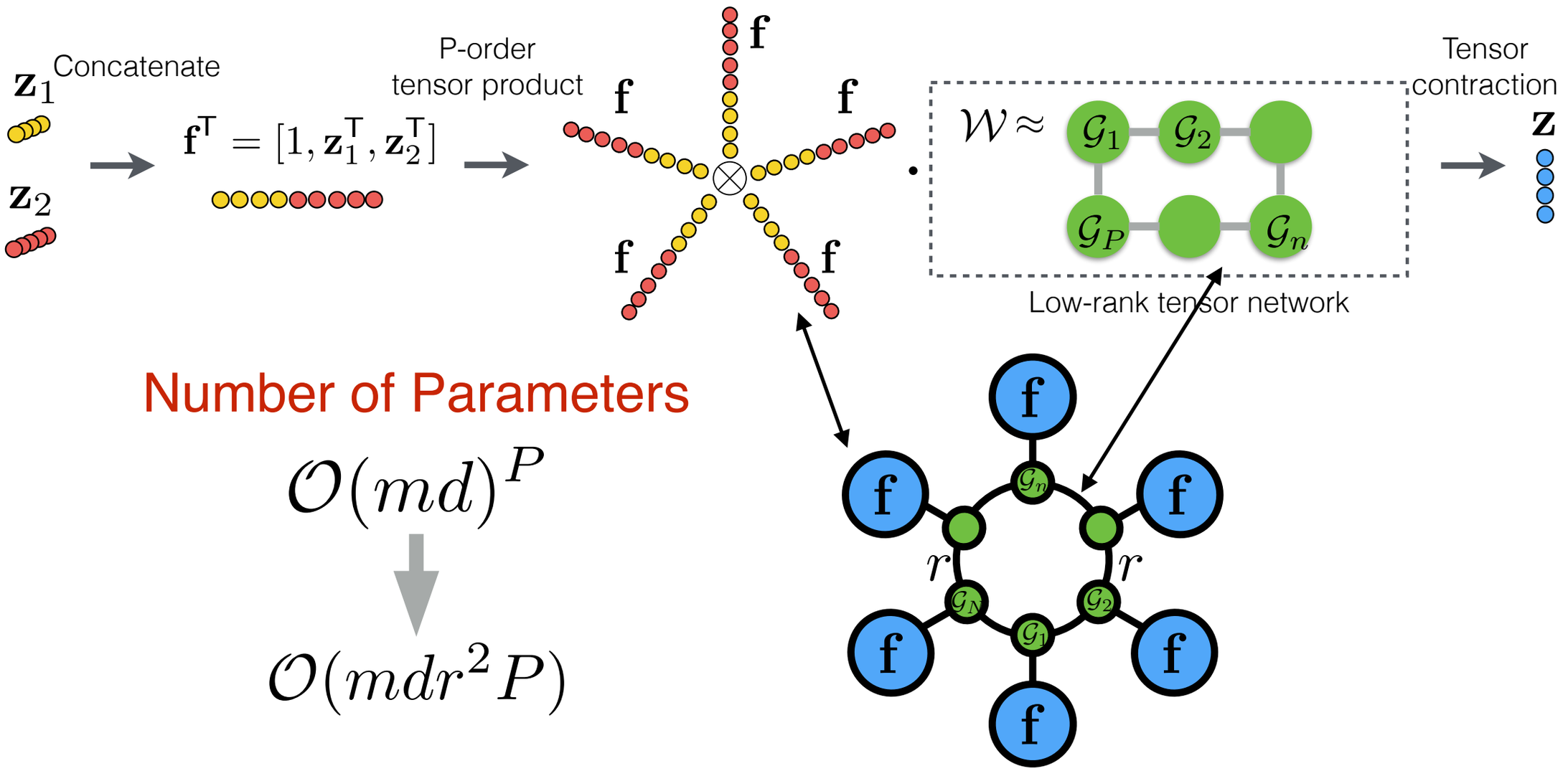
Order

Number of modality

Dimension of feature

Deep Multimodal Multilinear Fusion with High-order Polynomial Pooling (Hou et al., NeurIPS 2019)

Tensor Polynomial Pooling (PTP)

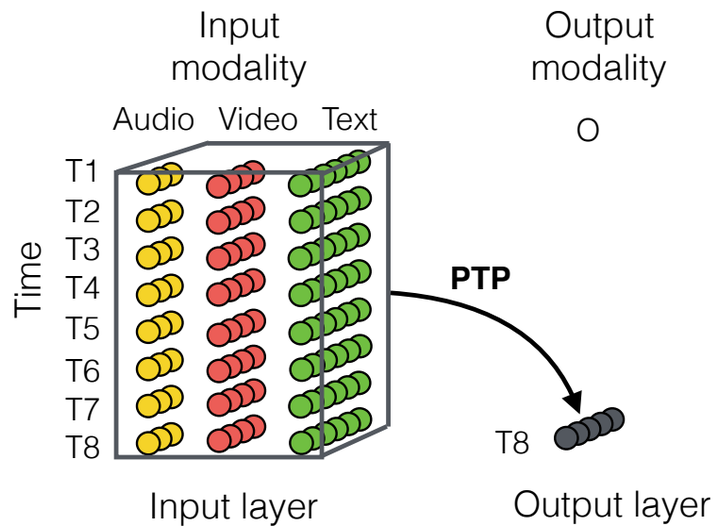


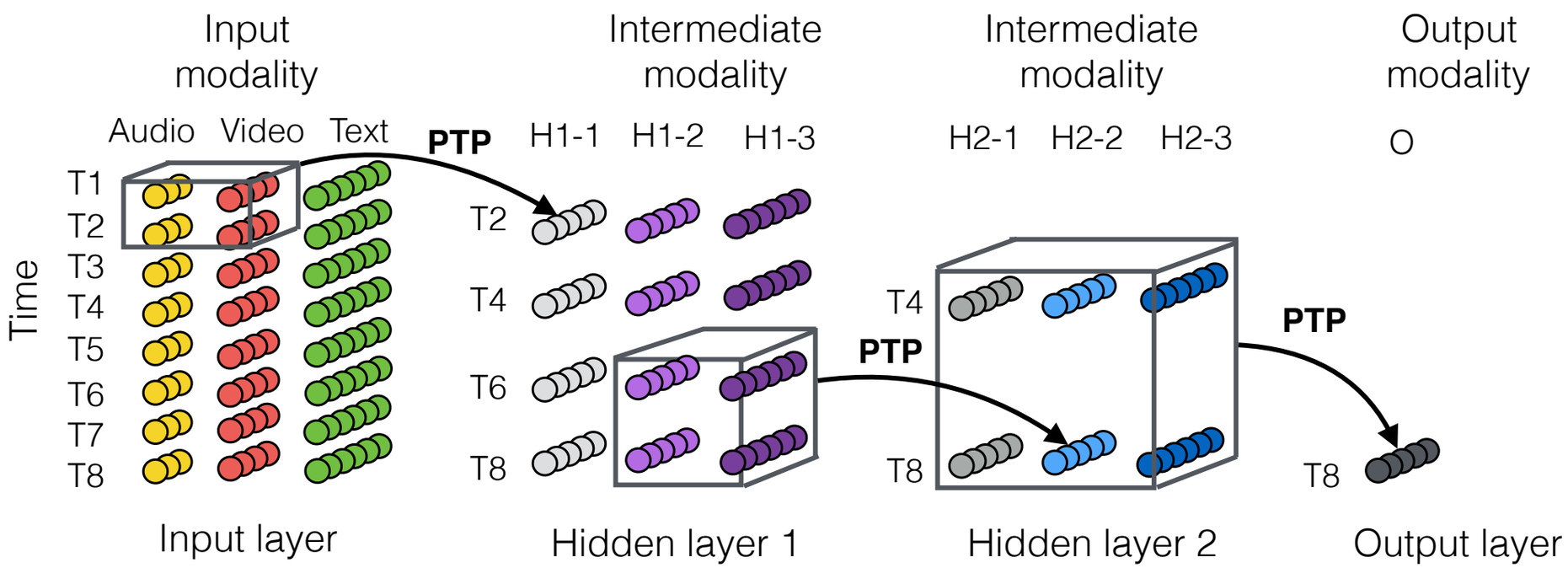
- Tensor network representation of $\mathcal{W}$ is crucial

Deep Multimodal Multilinear Fusion with High-order Polynomial Pooling (Hou et al., NeurIPS 2019)

Hierarchical Polynomial Fusion Network (HPFN)

- Interactions between modality and time steps are modeled explicitly
- Local temporal-modality correlations are fused and multi-layers are designed
- PTP resembles convolutional filter





Tensor-Power Recurrent Model

- Mapping of hidden state: degree-p tensor power

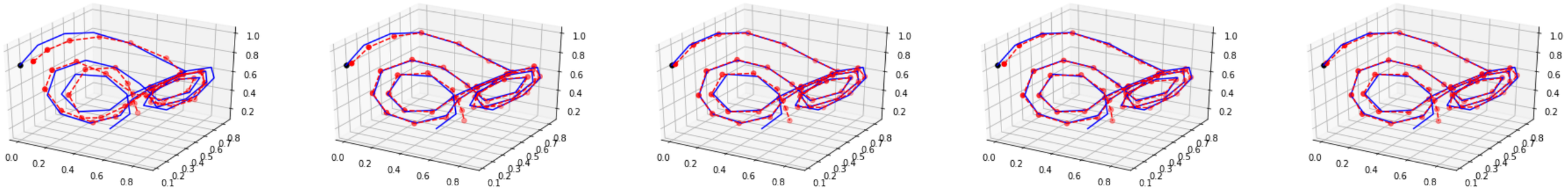
$$\phi(\mathbf{h}^{(t)}) = \underbrace{\mathbf{h}^{(t)} \otimes \mathbf{h}^{(t)} \otimes \cdot \otimes \mathbf{h}^{(t)}}_{p\ repeated} \in \mathbb{R}^{\underbrace{I \times I \times \cdots \times I}_{p}}$$

- Tensor-power recurrent model

$$\mathbf{h}_i^{(t+1)} = \left\langle \mathcal{W}_i, \phi(\mathbf{h}^{(t)}) \right\rangle + \left\langle \mathbf{W}_i, \mathbf{x}^{(t+1)} \right\rangle$$

- Approximation error $\epsilon \leq \frac{1}{h}\left(C_f^2 \frac{d-1}{(k-1)(r+1)^{k-1}} + C(k)p^{-k}\right)$

(TT-)ranks

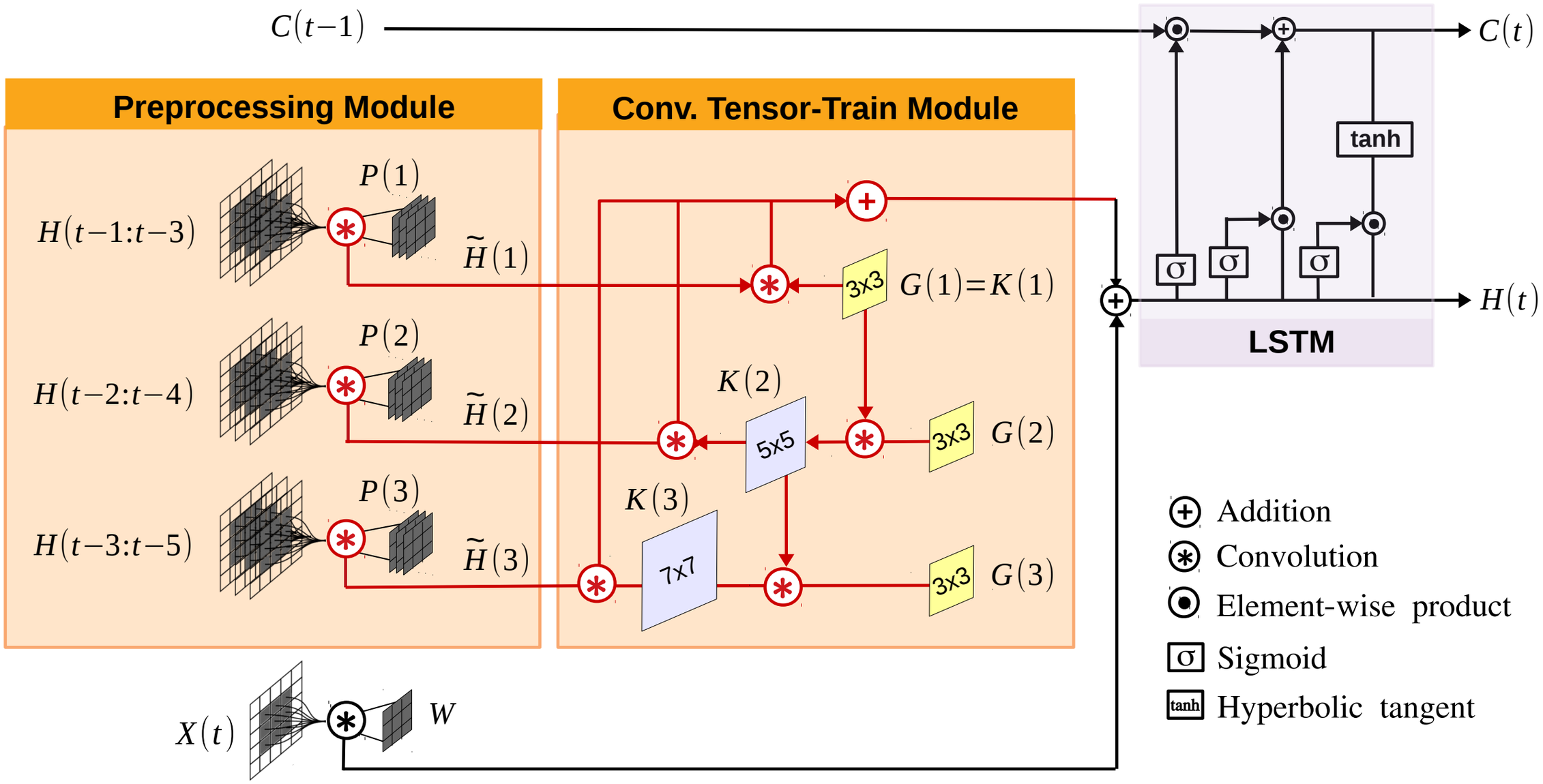


(a) RNN (b) MRNN (c) HORNN (d) HOT-RNN (e) HOT-LSTM

Yu, Rose, et al. "Long-term forecasting using higher order tensor RNNs." *JMLR (2019)*

Tensor-Power Recurrent Model

- Extension to spatio-temporal recurrent model



Su, Jiahao, et al. "Convolutional Tensor-Train LSTM for Spatio-temporal Learning.", Neurips 2020

Software for Tensor Networks

- **Google TensorNetwork** (https://github.com/google/TensorNetwork)
- **Tensorly** (http://tensorly.org/)
- **ITensor** (https://itensor.org)
- **TeNPy** (https://tenpy.readthedocs.io/)
- **Tntorch** (https://tntorch.readthedocs.io/)
- **TT-Toolbox** (https://github.com/oseledets/TT-Toolbox)
- Tensor network machine learning (**TNML)** (https://github.com/emstoudenmire/TNML)
- **TT_RNN** (https://github.com/Tuyki/TT_RNN)
- **TensorNet** (https://github.com/Bihaqo/TensorNet)
- More …

Part 3
Frontier and Future Direction

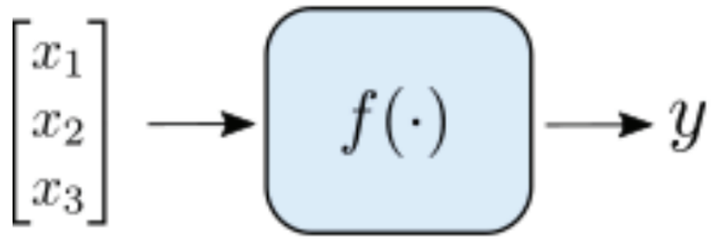
TNs for Function Approximation

Supervised learning

- Identify a nonlinear function from training dataset

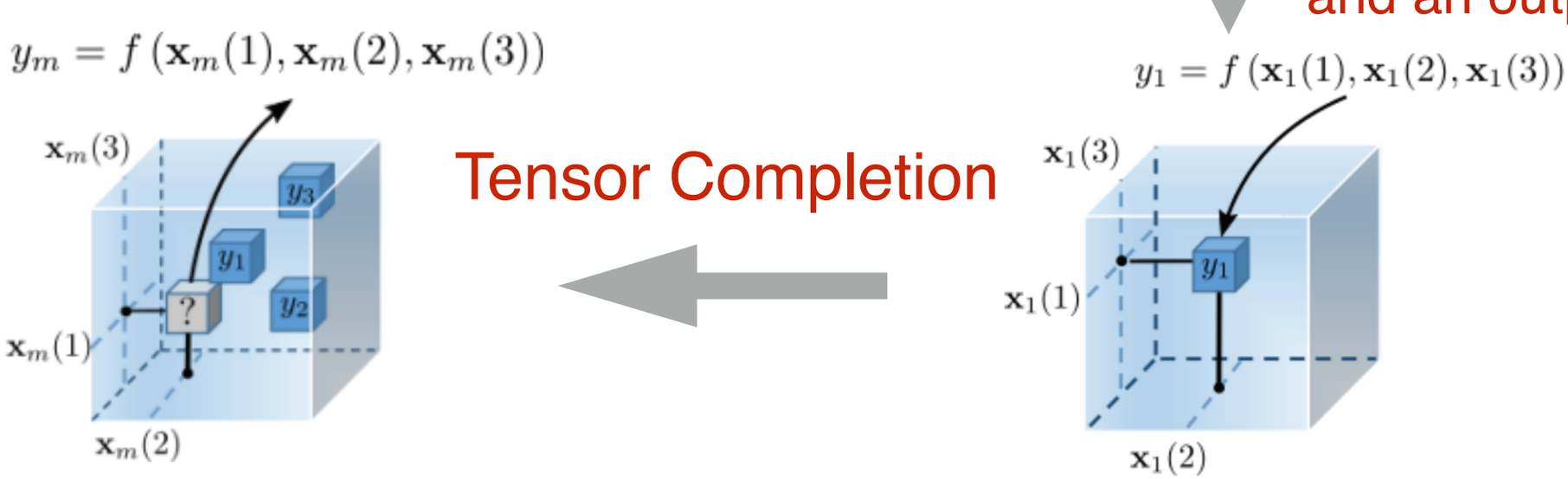
$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$$

$$y = f(x_1, \ldots, x_N)$$



- The function space is modeled as tensor

N discrete inputs and an output



Inference = Missing value prediction

Training points are observed entries

- Supervised learning to tensor completion

Nonlinear System Identification via Tensor Completion (Kargas et al. AAAI 2020)

Learning Low-rank Approximation to Function

- Optimization problem of tensor completion

Fidelity Regularization

$$\min_{\mathcal{X},\{\mathbf{A}_n\}_{n=1}^N} \frac{1}{M}\left\|\sqrt{\mathcal{W}} \circledast (\mathcal{Y}-\mathcal{X})\right\|_F^2 + \sum_{n=1}^{N} \rho \|\mathbf{A}_n\|_F^2$$

$$\text{subject to} \quad \mathcal{X} = \sum_{f=1}^{F} \mathbf{A}_1(:,f) \odot \cdots \odot \mathbf{A}_N(:,f),$$

Low-rank CP representation as constraint

- Pros.
  - Non-parametric model without specific form of function
  - Complexity of function = Tensor Rank, can be learned from dataset
  - Rank-1 can be a complex nonlinear function

A Simple Example

$h(x_2)$

Highly nonlinear

$h(x_1)$

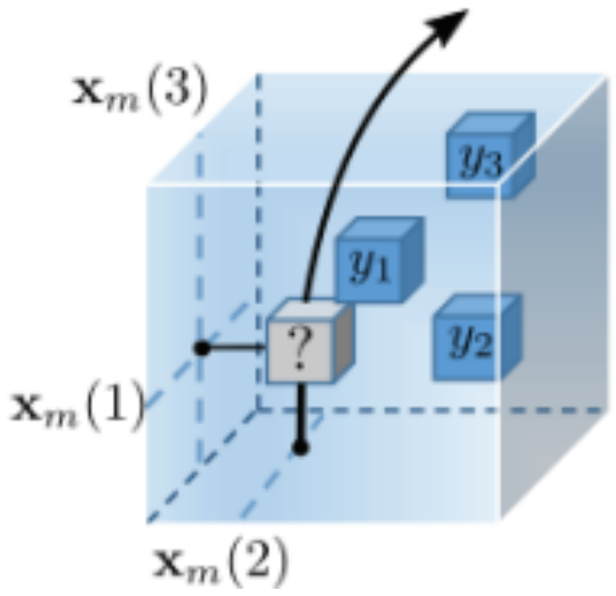
Rank = 1

$$y = f(x_1, x_2) \qquad f(x_1, x_2) = h(x_1)h(x_2)$$

Challenges

- Discretization of features is necessary
- Sample complexity of TNs? Is it comparable with DNNs
- Number of parameters (latent cores in TNs vs. weight parameters in DNNs)

$$y_m = f\left(\mathbf{x}_m(1), \mathbf{x}_m(2), \mathbf{x}_m(3)\right)$$

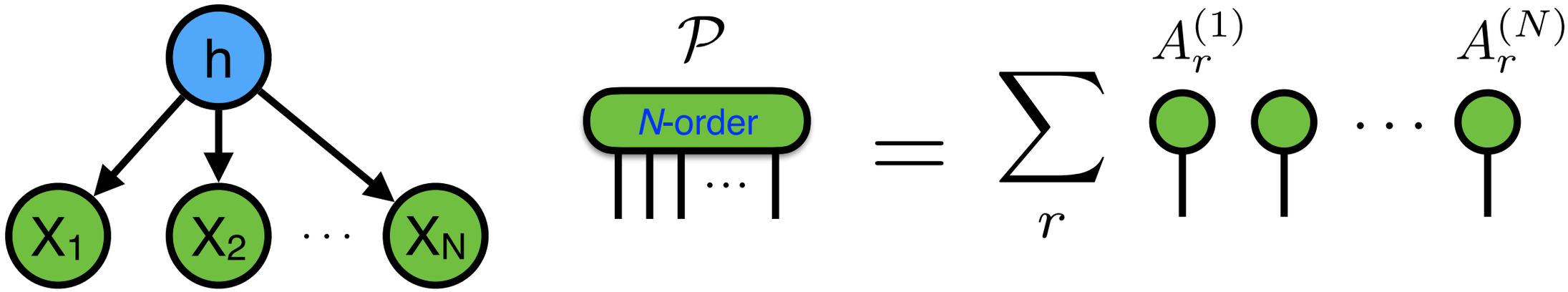


TNs for Probability Graphical Model

- Probability mass function = Non-negative Tensor

$$P(X_1, X_2, \ldots, X_N) \qquad X_n \in [1:d]$$

- Tensor decomposition (CPD)



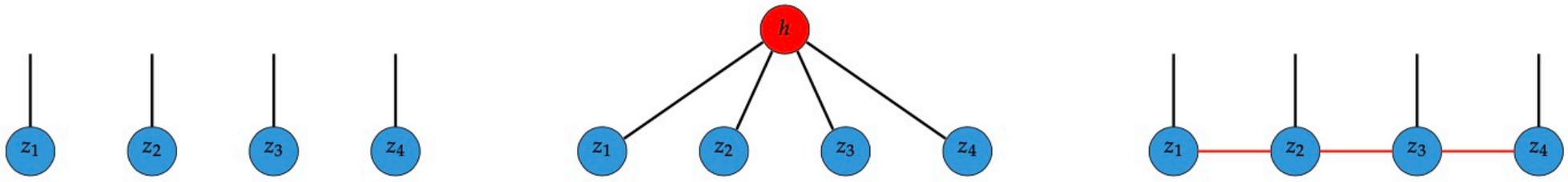
Joint Probability: $P(X_1, X_2, \ldots, X_N) = \sum_h P(h)P(X_1|h)P(X_2|h)\cdots P(X_N|h)$

CPD: $P_{x_1,x_2,\ldots,x_N} = \sum_r \lambda_r A^{(1)}_{x_1,r} A^{(2)}_{x_2,r} \cdots A^{(N)}_{x_N,r}$

- Conditional dependency controlled by tensor rank

Expressive power of tensor-network factorizations for probabilistic modeling (Glasser et al.NuerIPS, 2019)

Rank vs. Dependency



(a) Mean-field (b) Naive Bayes (c) Markov models

Figure: (a) and (b) are (conditionally) independent structures, while (c) induces dependencies.

Approximations

- Mean-field: $p(\boldsymbol{z}) = \prod_{i=1}^{N} p(z_i)$. $\longleftarrow$ Rank-1 CPD
- Naive Bayes: $p(\boldsymbol{z}) = \sum_{h=1}^{r} p(h) \prod_{i=1}^{N} p(z_i \mid h)$. $\longleftarrow$ Rank-r CPD
- Graphical models: $p(\boldsymbol{z}) = \prod_{i=1}^{N} p(z_i \mid Pa(z_i))$. $\longleftarrow$ TNs

Graphical models can be translated into TN language (Robeva and Seigal, 2019; Ran, 2019).

Why use TN for Graphical Models

- Efficient computation in:
  - Marginalization, normalization (partition function)
  - Maximum log-likelihood, MAP estimate
  - Conditional distributions and sampling
  - …

Example

- Factorize probability tensor in MPS/TR format

$$p(\boldsymbol{z}) = \prod_{k=1}^{N} \mathrm{tr}(Q^{(k)}[z_k]),$$

  - Unconstrained $p(\boldsymbol{z})$: I^N.
  - MPS format $p(\boldsymbol{z})$: INr^2.

  where $Q^{(k)}[z_k] \in \mathbb{R}_{+}^{r_k \times r_{k+1}}$, r_k are MPS-ranks (or, boundary conditions).

- The marginal distribution is

$$p(z_{1:k-1}, z_{k+1:N}) = \sum_{i_k=1}^{I_k} p(z_1, \ldots, z_k = i_k, \ldots, z_N)$$

$$= \sum_{i_k=1}^{I_k} \mathrm{tr}\left(\prod_{d=1}^{k-1} Q^{(d)}[z_d] \cdot Q^{(k)}[i_k] \cdot \prod_{d=k+1}^{N} Q^{(d)}[z_d] \right)$$

$$= \mathrm{tr}\left(\prod_{d=1}^{k-1} Q^{(d)}[z_d] \cdot \tilde{Q}_k \cdot \prod_{d=k+1}^{D} Q^{(d)}[z_d] \right),$$

$$\tilde{Q}_k = \sum_{i_k=1}^{I_k} Q^{(d)}[i_k]$$

Summation simply performed on core tensor

Example

- Normalization

$$Z = \mathrm{tr}\left(\prod_{k=1}^{N} \tilde{Q}_k\right)$$

Summation over all core tensors

- Conditional distribution

$$p(z_{1:k-1}, z_{k+1:N} \mid z_k) = \frac{p(z_{1:N})}{p(z_k)},$$

$$p(z_k) = \mathrm{tr}\left(\prod_{d=1}^{k-1} \tilde{Q}_d \cdot Q^{(k)}[z_k] \cdot \prod_{d=k+1}^{N} \tilde{Q}_d\right)$$

Summation over all core tensors except k-th

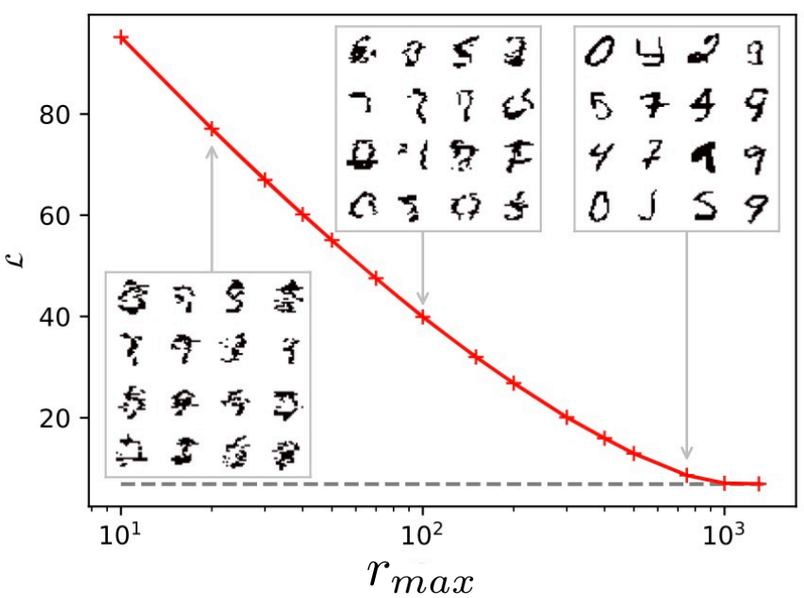
Unsupervised Generative Modeling

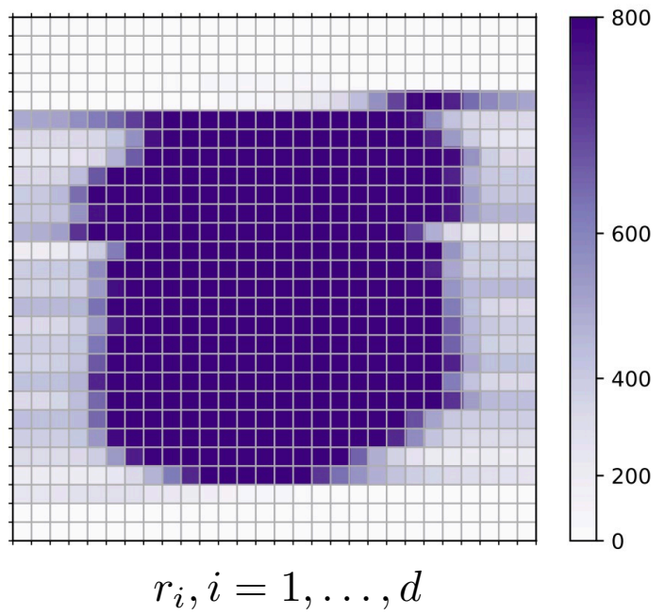
Han et al., Phys. Rev. X 2018

- Modeling joint probability distribution of binary variables

$$p(\mathbf{v}) = \left| \mathcal{G}_1 \; r \; r \; \cdots \; \mathcal{G}_d \;\; v_1 \; \cdots \; v_d \right|^2 / Z$$

- Minimizing negative log-likelihood to learn parameters $\{\mathcal{G}_i\}_{i=1}^d$
- Efficient direct sampling method to generate a sample bit by bit





Continuous Distribution by TNs

- Gaussian mixture distribution

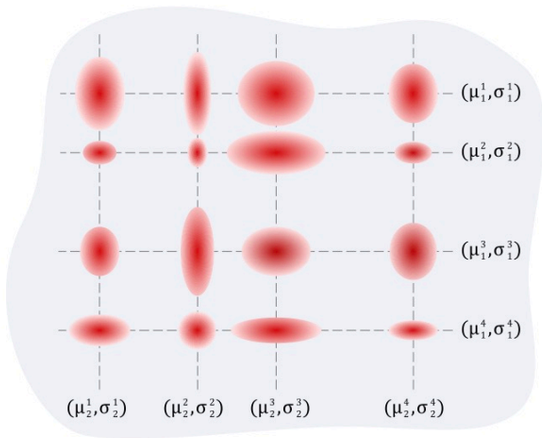
$$p(\boldsymbol{x}) = \sum_{z} p(z) \prod_{k=1}^{N} p(x_k \mid z),$$

Mixture coefficient
discrete variable

- Multi-dimensional Gaussian mixture distribution

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) \prod_{k=1}^{N} p(x_k \mid z_k),$$

Mixture coefficient is a discrete vector
Probability is N-dimensional tensor



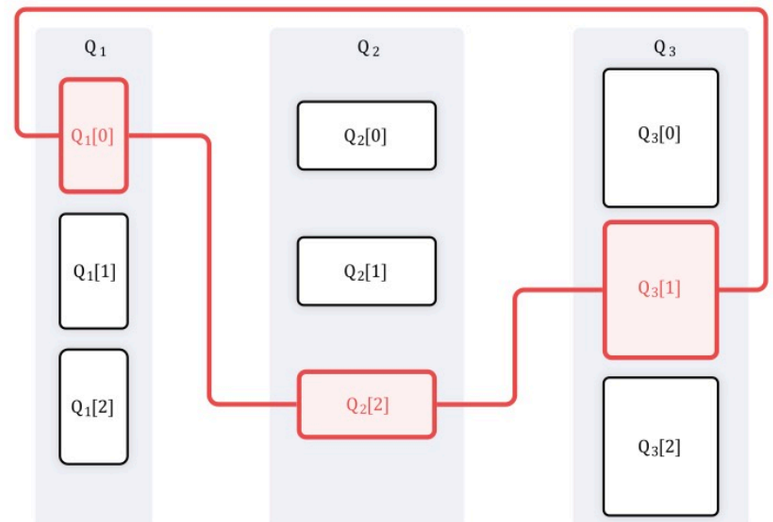
A prior of a googol gaussians: a tensor ring induced prior for generative models (Kuznetsov et al., NeurIPS 2019)

Prior Distribution Modeled by TN

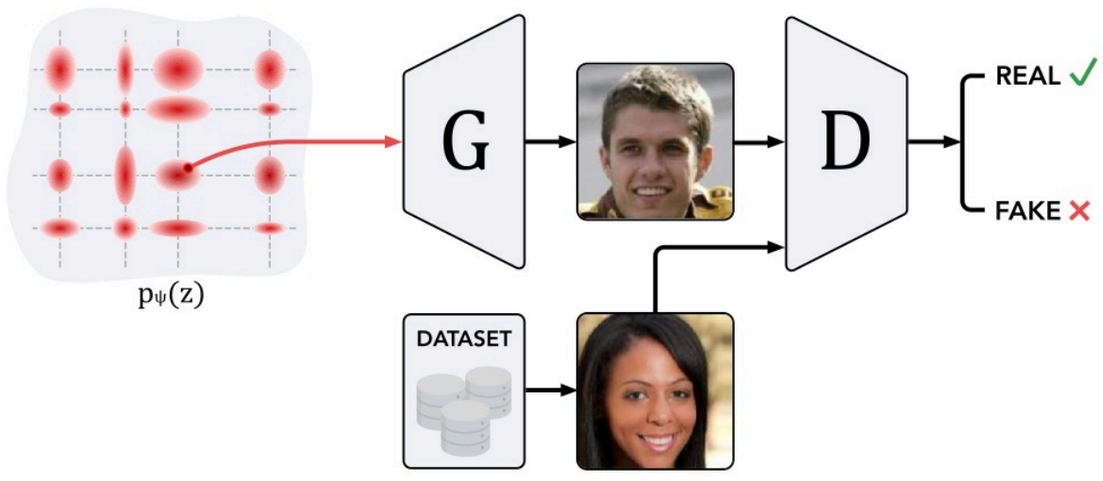
- Probability of mixture coefficient p(z) is TR representation

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) \prod_{k=1}^{N} \mathcal{N}(x_k \mid \mu_{i_k}^{(k)}, \sigma_{i_k}^{(k)})$$

$$= \sum_{\boldsymbol{z}} \mathrm{tr}(\prod_{j=1}^{D} Q^{(j)}[z_j]) \prod_{k=1}^{N} \mathcal{N}(x_k \mid \mu_{i_k}^{(k)}, \sigma_{i_k}^{(k)}).$$



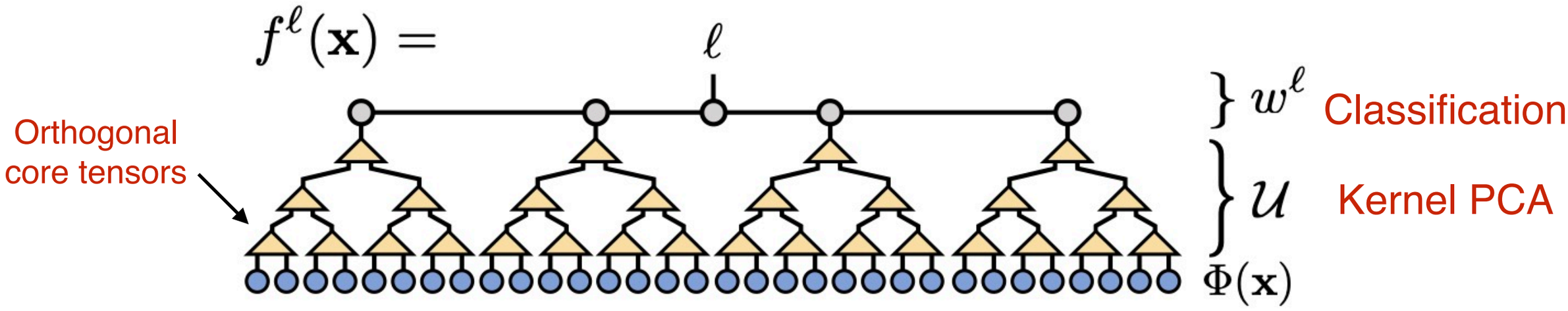
- Tensor Ring Induced Prior



A prior of a googol gaussians: a tensor ring induced prior for generative models (Kuznetsov et al., NeurIPS 2019)

Learning Relevant Features of Data

- Unsupervised learning with reduced order of TN representation
- Supervised learning for the top classification layer



TN Representation: $\Phi(\mathbf{x}) = \phi(x_1) \otimes \cdots \otimes \phi(x_d)$

Local feature mapping: $\phi(x_i) = [1, x_i]^T, \; i \in [1, d]$

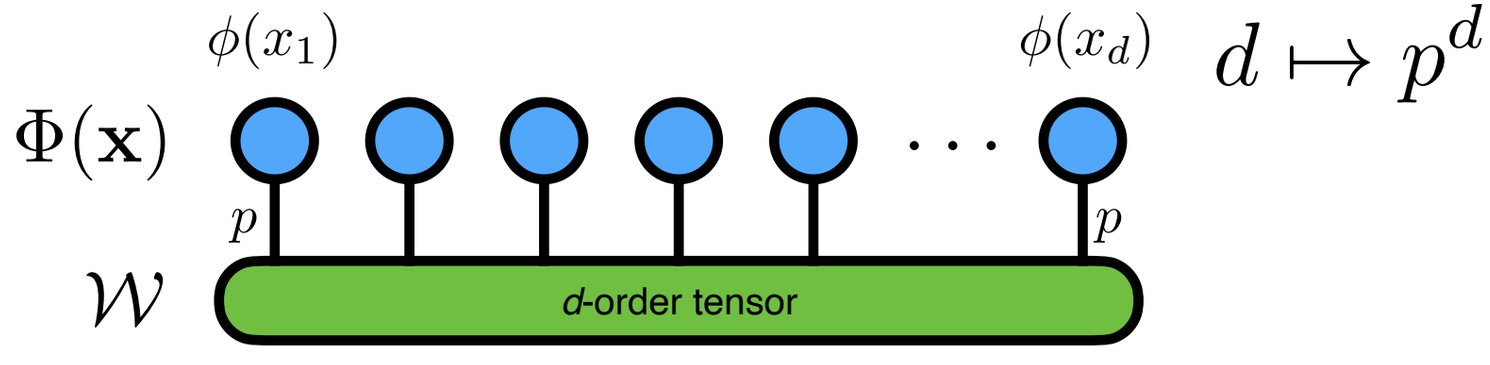
Input data: $\mathbf{x} = [x_1, \ldots, x_d]^T$

89% accuracy on Fashion MNIST data set

Learning Relevant Features of Data with Multi-scale Tensor Networks (Stoudenmire et al. 2018)

TN for Image Classification

$$f(\mathbf{x}) = \langle \mathcal{W}, \Phi(\mathbf{x}) \rangle$$

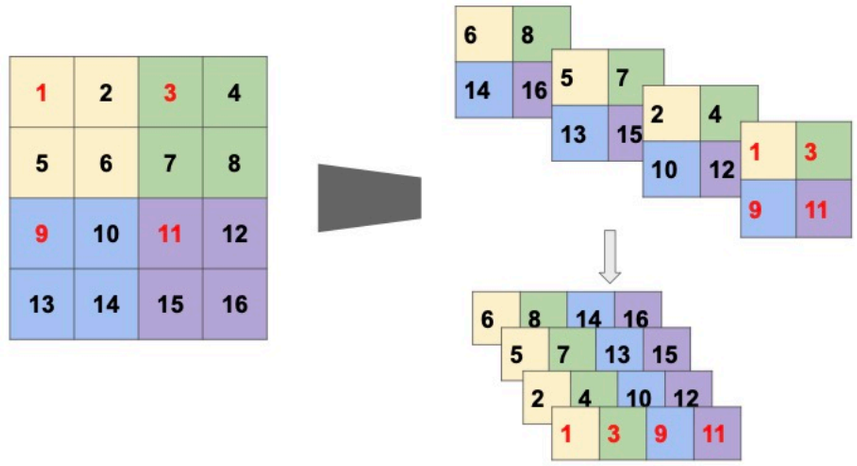


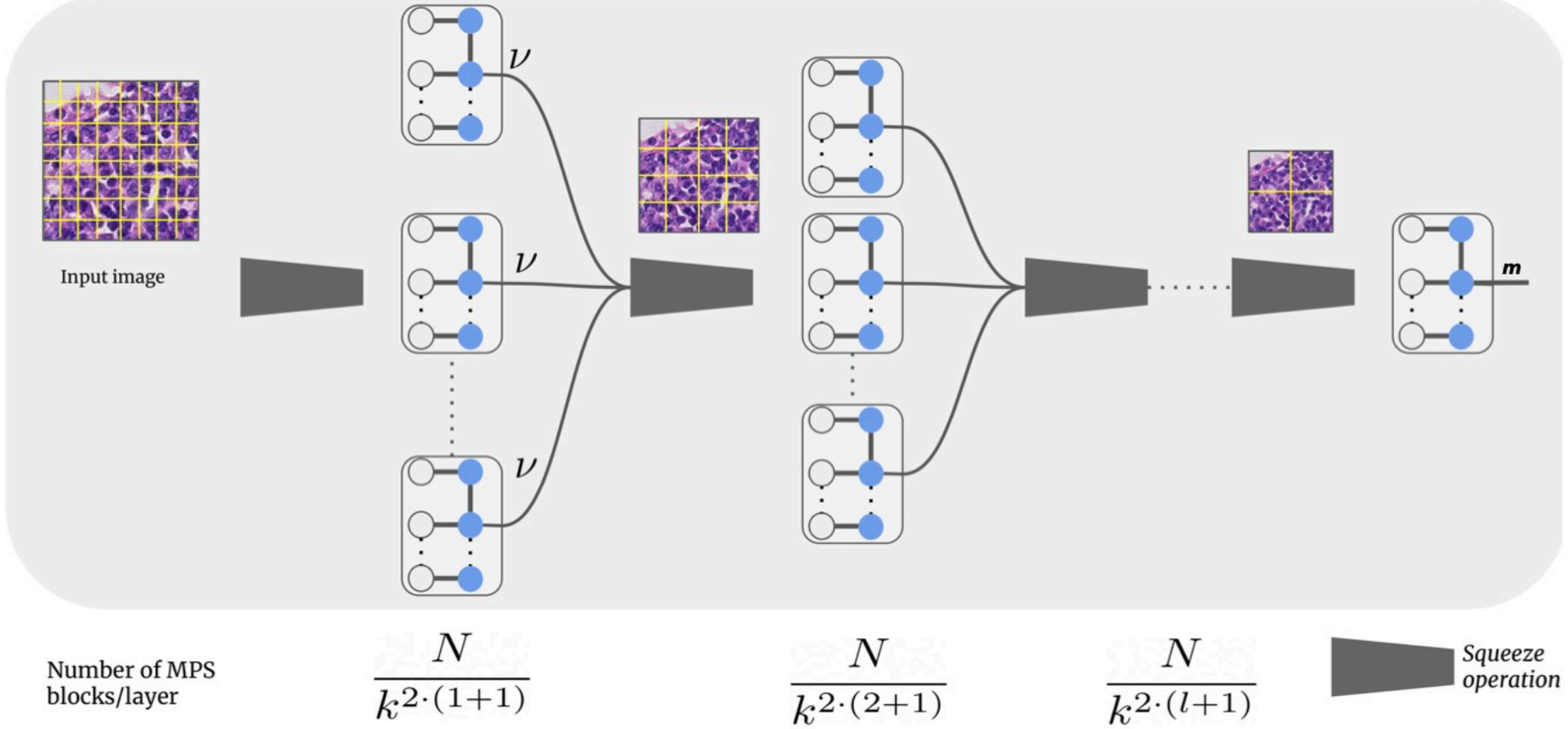
- Global structure information lost by flattening image
- How to apply TN to large image task?
  - Only flatten local small patches
  - Hierarchical TN model

Tensor Networks for Medical Image Classification (Selvan et al., MIDL 2020)

TN for Image Classification

- Capture global structure by squeeze operation
- Single layer to multi-layer TNs
- Comparable performance with DenseNet but fewer parameters

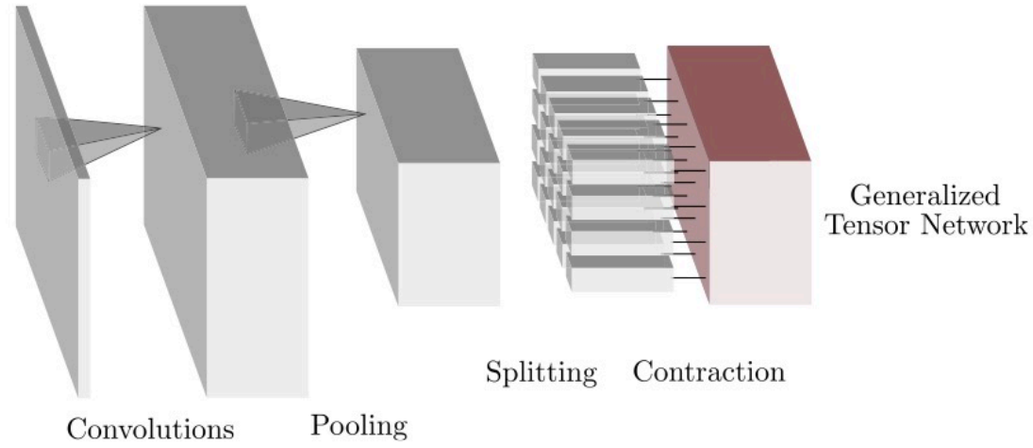


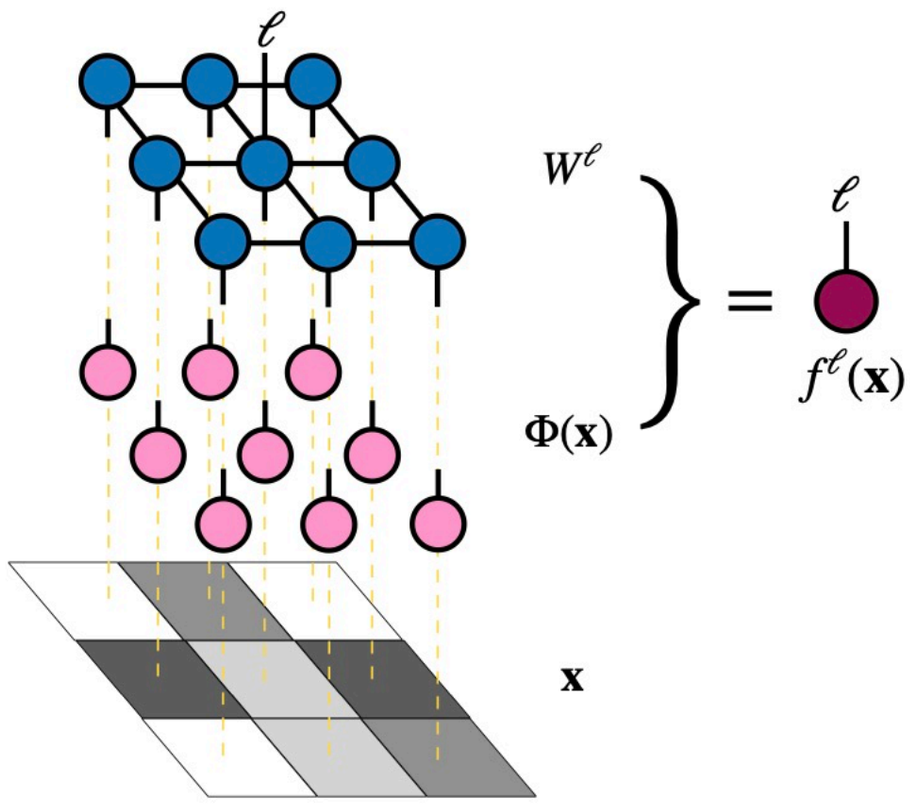


Tensor Networks for Medical Image Classification (Selvan et al., MIDL 2020)

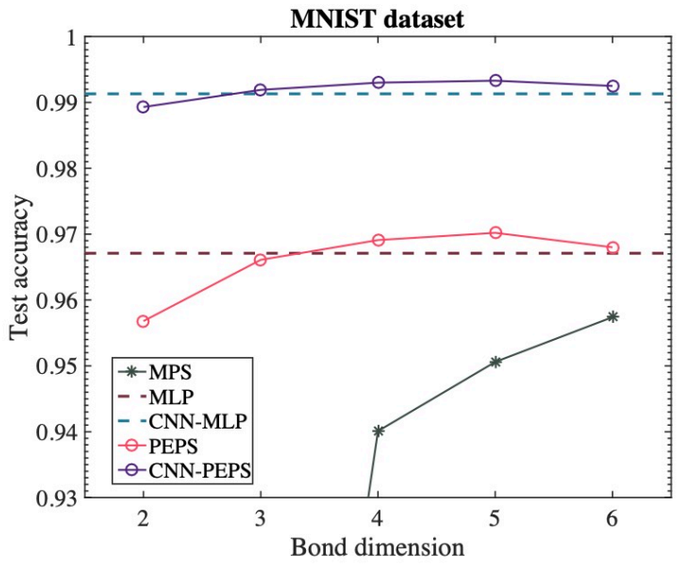
Supervised Learning with Projected Entangled Pair States

- One layer PEPS for supervised learning
- CNN + PEPS as learning model





7% parameters of CNN+MLP

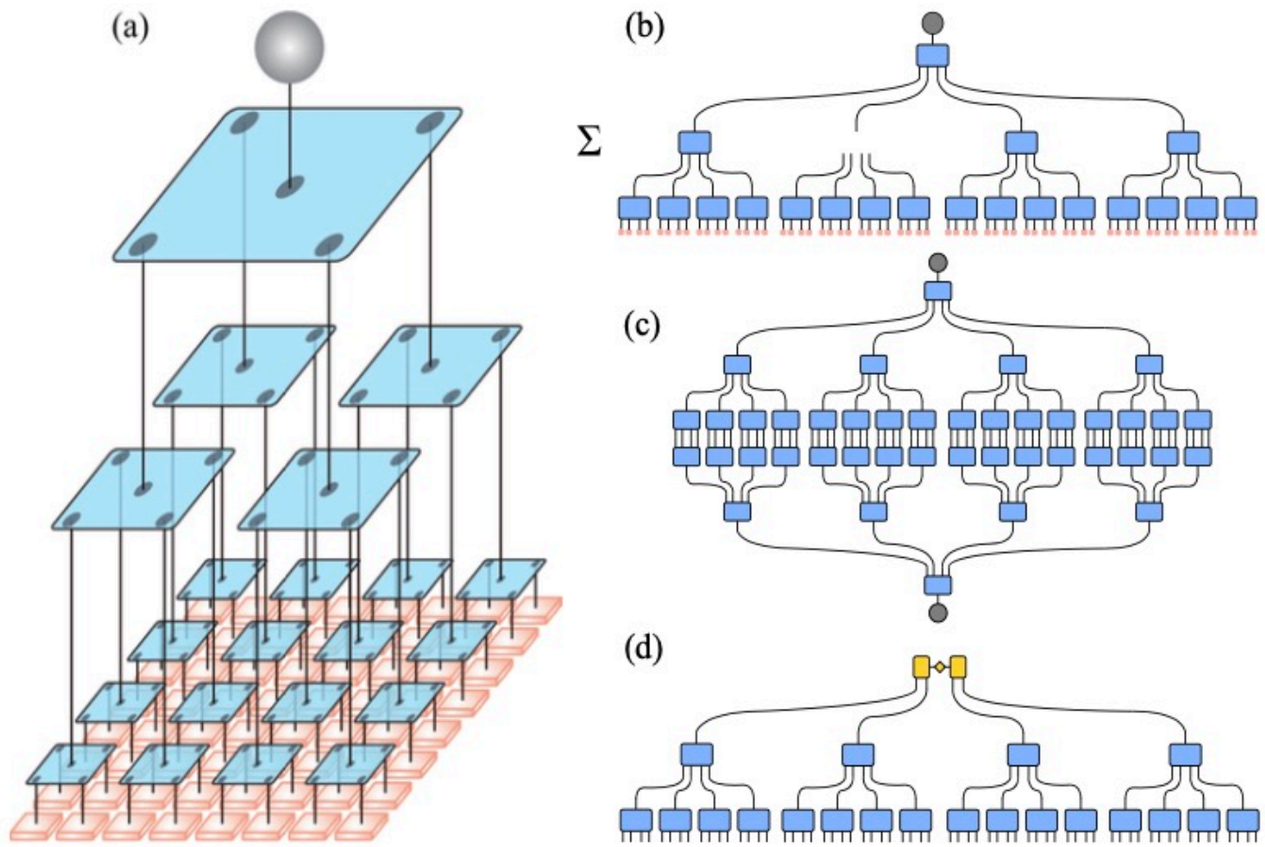


From Probabilistic Graphical Models to Generalized Tensor Networks for Supervised Learning (Glasser, arXiv 2019)

Supervised Learning with Projected Entangled Pair States (Chen et al., arXiv 2020)

Machine Learning by Hierarchical Tensor Networks

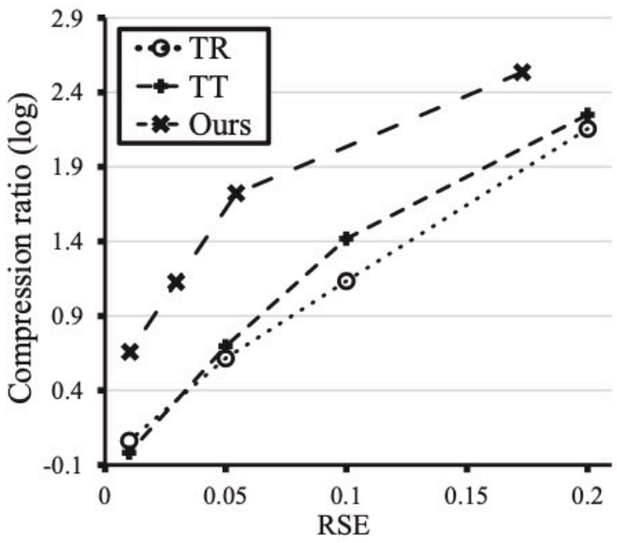
- Supervised learning with tree tensor networks



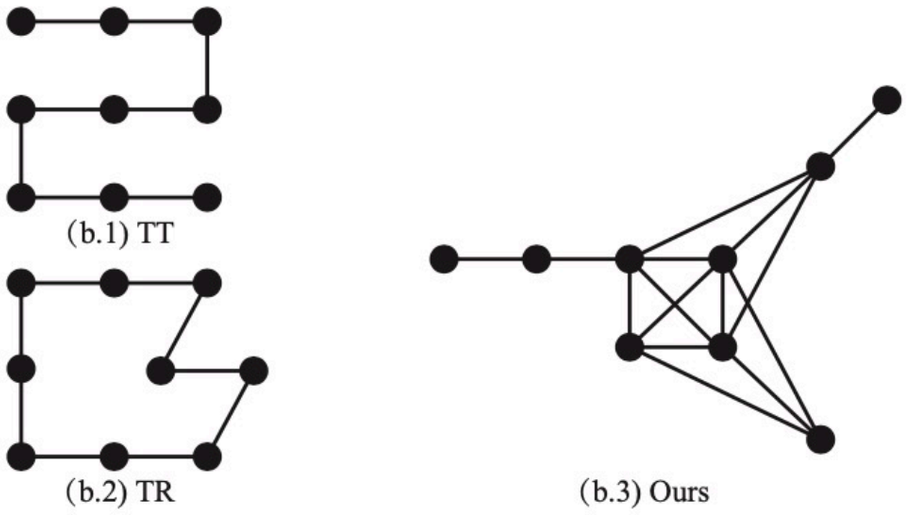
Model Architecture

Machine Learning by Unitary Tensor Network of Hierarchical Tree Structure (Liu et al., 2019)

Learning Tensor Network Structure



(a) RSE *vs.* CR



(b) Graphical strctures of TN

Standard models may not be the most compressive one

- How to choose an optimal tensor network structure is necessary
- Can we learn an optimal TN structure by data?
- Challenge: given an 9-order tensor, there are more than 68 BILLION candidates

Optimization Problem

- The TN structures can be fully described by its adjacency matrice

$$\mathcal{X} = TN(\mathbb{V}; \mathbf{A})$$

Collection of core tensors　　Adjacency matrix

Adjacency Matrix

$$\begin{pmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

Order-4 tensor with TT-format

- Find an optimal A such that

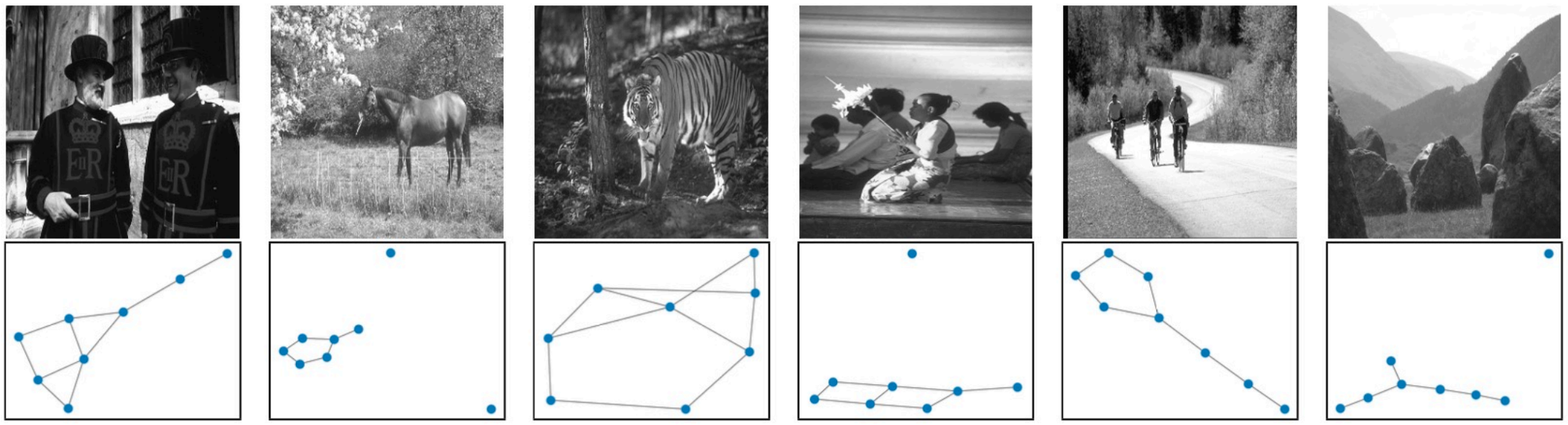
$$\min_{\mathbf{A}\in\mathbb{A}} \frac{1}{\epsilon(\mathbf{A})}, \quad s.t. \exists \widehat{\mathbb{V}} \text{ which satisfies } \left\| \mathcal{X} - TN(\widehat{\mathbb{V}}; \mathbf{A}) \right\|_F^2 \leq \delta,$$

$$\epsilon(\mathbf{A}) = \frac{\text{Uncompressed size of } \mathcal{X}}{\text{Parameter size of } \mathbb{V} \text{ under } \mathbf{A}}.$$

- Maximize compression ratio such that approximation error is upper-bounded

Experiment and Discussion

- 10 natural images are random selected from LIVE dataset [Sheikh et al., 2006], and reshaped as order-8 tensors
- The learned topology have more complex structures than simple ones like line, tree or cycles.



- Optimal structures could significantly boost the expressive power of TN decomposition
- Near-optimal structures are sufficient to outperform the simple ones

Evolutionary Topology Search for Tensor Network Decomposition (Li et al., ICML 2020)

Discussions

- TNs are tools for data modeling, representing model parameters and function approximation
- Theory shows TNs have expressive power similar to DNNs
- Practically, performance of TNs for ML tasks is less attractive
- If TNs can be developed as a general ML model?
- Are there any advantages of TNs from perspective of robustness and interpretability?

Acknowledgements

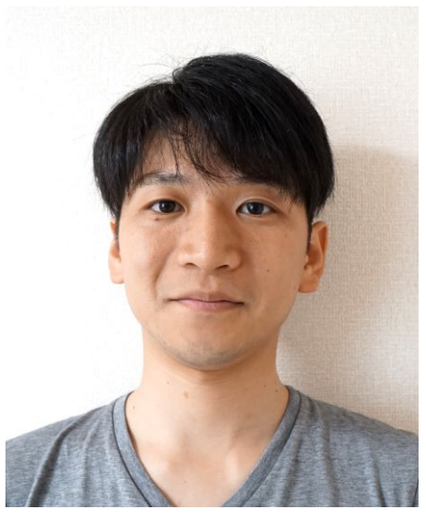
Ming Hou
(RIKEN AIP)

Andrzej Cichocki
(Skoltech)

Chao Li
(RIKEN AIP)

Xiao-yang Liu
(Columbia University)

Cesar F. Caiafa
(CONICET)



Tatsuya Yokota
(NIT)

Andong Wang
(GDUT)