Low Rank Decomposition

马恺声

清华大学

Outline

- Why We Need Low Rank Decomposition?
- Mathematical Insights
- Optimization
- Recent Works in Image Problems
- Recent Works in Neural Networks

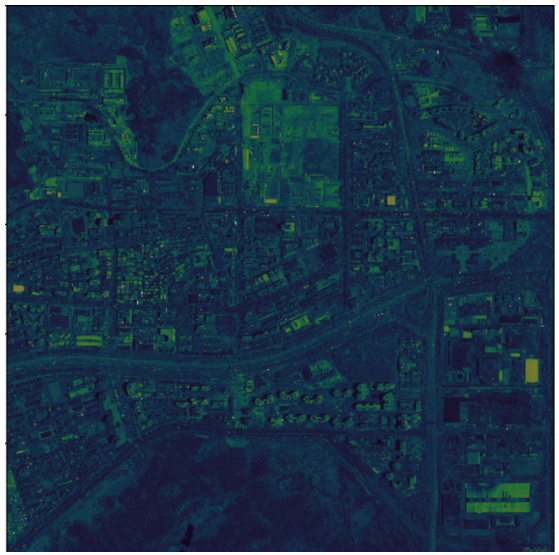
Why we need low rank decomposition

- **Data with higher-dimension and higher-correlation**

MNIST

ImageNet

Scenery



Remote sensing data

Larger and larger

- **We need efficient models.**
- **We have to abandon noisy, useless information.**

Low rank decomposition helps to speedup and saving memory.

Outline

- Why We Need Low Rank Decomposition?
- **Mathematical Insights**
- Optimization
- Recent Works in Image Problems
- Recent Works in Neural Networks

Singularly Valuable Decomposition (SVD)

- The Eigenvalue Decomposition (EVD)

  — If A is a symmetric real n x n matrix, there exist an orthogonal matrix V and a diagonal D such that

$$A = VDV^T$$

- The Single Value Decomposition (SVD)

  — If A is a symmetric real m x n matrix, there exist two orthogonal matrices U and V, and a diagonal Σ such that

$$A = U\Sigma V^T$$

Kalman D. A singularly valuable decomposition: the SVD of a matrix[J], 1996

Weighted Low Rank Approximation (WLRA)

- **Original weighted low rank approximation, ICML'2003**

In the weighted low rank approximation problem, one is given a matrix $M \in \mathbb{R}^{n \times d}$, a weight matrix $W \in \mathbb{R}^{n \times d}$, and an integer parameter k, and would like to find factors $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times d}$ so as to minimize

$$\|W \circ (M - U \cdot V)\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} W_{i,j}^2 (M_{i,j} - \langle U_{i,*}, V_{*,j} \rangle)^2$$

- **Regularized weighted low rank approximation, NIPS'2019**

$$\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|W \circ (UV - M)\|_F^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2$$

- **Theoretical bound**

Theorem 8. *Given matrices $A, W \in \mathbb{R}^{n \times d}$ and $\varepsilon < 0.1$ and suppose the conditions of Theorem 3 hold. Furthermore, let $\sigma = \max_{i,j}\{\sigma_1(V^* D_{W_{i,:}}), \sigma_1(D_{W_{:,j}} U^*)\}$.*

There is an algorithm to find $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times d}$ in time $poly(n) \left(\frac{\sigma^2}{\lambda} \cdot \log\left(\frac{\Delta(\sigma^2+\lambda)n}{\lambda\tau}\right)\right)^l \cdot \log^{O(1)}\left(\frac{\Delta}{\delta}\right)$, where $l = O((s + \log(\frac{1}{\varepsilon}))^2 \frac{r^2}{\varepsilon^2})$, such that $\|W \circ (UV - A)\|_F^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 \leq (1+\varepsilon)\mathrm{OPT} + \tau$.

N Srebro, T Jaakkola, Weighted Low-Rank Approximations, ICML 2003
F Ban, et al, Regularized Weighted Low Rank Approximation, NIPS 2019

Tensor: Canonical Polyadic Decomposition (CPD)

- **The CP decomposition factorizes a tensor into a sum of component rank-one tensors.**

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

Table 3.1 *Some of the many names for the CP decomposition.*

Name	Proposed by
Polyadic form of a tensor	Hitchcock, 1927 [105]
PARAFAC (parallel factors)	Harshman, 1970 [90]
CANDECOMP or CAND (canonical decomposition)	Carroll and Chang, 1970 [38]
Topographic components model	Möcks, 1988 [166]
CP (CANDECOMP/PARAFAC)	Kiers, 2000 [122]

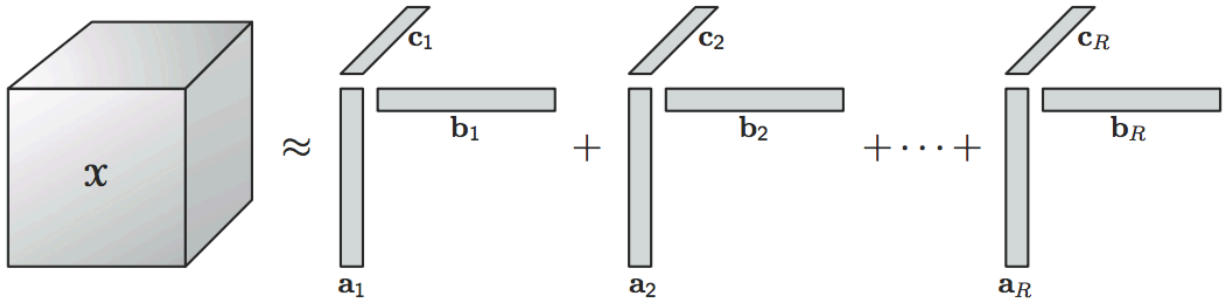


Fig. 3.1 *CP decomposition of a three-way array.*

TG Kolda, BW Bader, Tensor Decompositions and Applications, SIAM Review, 2009

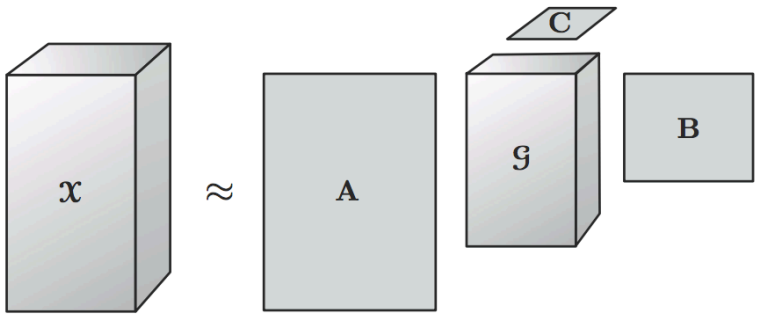
Tensor: Tucker Decomposition (TD)

- TD decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mod

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r$$

Table 4.1 *Names for the Tucker decomposition (some specific to three-way and some for N-way).*

Name	Proposed by
Three-mode factor analysis (3MFA/Tucker3)	Tucker, 1966 [226]
Three-mode PCA (3MPCA)	Kroonenberg and De Leeuw, 1980 [140]
N-mode PCA	Kapteyn et al., 1986 [113]
Higher-order SVD (HOSVD)	De Lathauwer et al., 2000 [63]
N-mode SVD	Vasilescu and Terzopoulos, 2002 [229]



$$\mathcal{X} \in \mathbb{R}^{I \times J \times K} \quad \mathbf{A} \in \mathbb{R}^{I \times P}$$
$$\mathbf{B} \in \mathbb{R}^{J \times Q}; \quad \mathbf{C} \in \mathbb{R}^{K \times R}$$
$$\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$$

Fig. 4.1 *Tucker decomposition of a three-way array.*

TG Kolda, BW Bader, Tensor Decompositions and Applications, SIAM Review, 2009

Other Tensor Decomposition

Table 5.1 *Other tensor decompositions.*

Name	Proposed by
Individual differences in scaling (INDSCAL)	Carroll and Chang, 1970 [38]
Parallel factors for cross products (PARAFAC2)	Harshman, 1972 [92]
CANDECOMP with linear constraints (CANDELINC)	Carroll et al., 1980 [39]
Decomposition into directional components (DEDICOM)	Harshman, 1978 [93]
PARAFAC and Tucker2 (PARATUCK2)	Harshman and Lundy, 1996 [100]

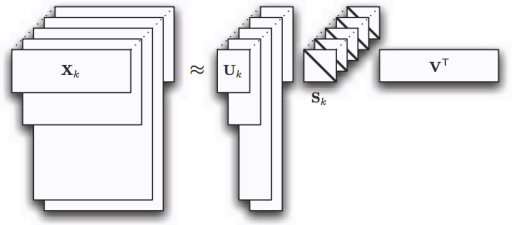


Fig. 5.1 *Illustration of PARAFAC2.*

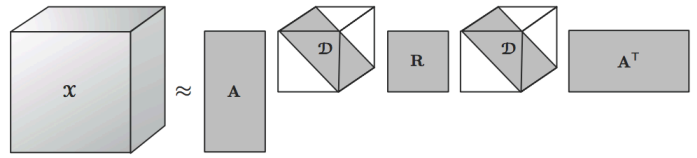


Fig. 5.2 *Three-way DEDICOM model.*

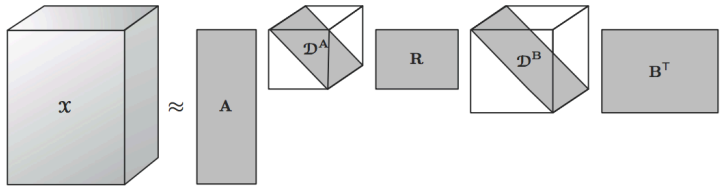


Fig. 5.3 *General PARATUCK2 model.*

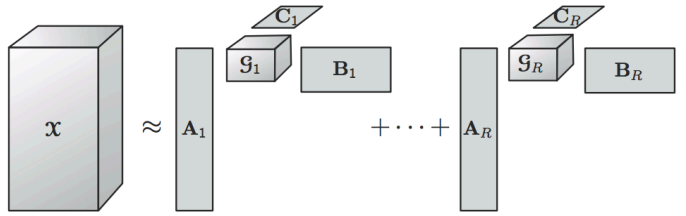


Fig. 5.4 *Block decomposition of a third-order tensor.*

TG Kolda, BW Bader, Tensor Decompositions and Applications, SIAM Review, 2009

Outline

- Why We Need Low Rank Decomposition?
- Mathematical Insights
- **Optimization**
- Recent Works in Image Problems
- Recent Works in Neural Networks

Optimization Approaches

- Fast Randomized Singular Value Thresholding for Low-Rank Optimization. TPAMI'2018
- Optimization-based algorithms for tensor decompositions: canonical polyadic decomposition, decomposition in rank-terms, and a new generalization. SIAM Journal on Optimization, 2013
- Decentralized sketching of low rank matrices, NIPS'2019
- Learning-Based Low-Rank Approximations. NIPS'2019
- Fast Low-rank Metric Learning for Large-scale and High-dimensional Data. NIPS'2019
- Low-Rank Bandit Methods for High-Dimensional Dynamic Pricing. NeurIPS'2019
- Distributed Low-rank Matrix Factorization With Exact Consensus. NeurIPS'2019
- Compressed Factorization: Fast and Accurate Low-Rank Factorization of Compressively-Sensed Data. ICML'2019
- A Fast Frequent Directions Algorithm for Low Rank Approximation, TPAMI'2019
- Large-Scale Low-Rank Matrix Learning with Nonconvex Regularizers, TPAMI'2019

Outline

- Why We Need Low Rank Decomposition?
- Mathematical Insights
- Optimization
- **Recent Works in Image Problems**
- Recent Works in Neural Networks

Image Denoising (CVPR'2017)

- **Multispectral images (MSI): contains abundant information with multiple specific frequencies across the electromagnetic spectrum, but is usually contaminated by the noises.**
- **Optimization model:** L is the low rank tensor, x is the desired image.

$$\left\{\hat{\mathcal{X}}, \hat{\mathcal{L}}_i\right\} = \arg\min_{\mathcal{X},\mathcal{L}_i} \frac{1}{2}||\mathcal{X} - \mathcal{Y}||_F^2 + \mu||\nabla_z \mathcal{X}||_p + \omega \sum_i \left(\frac{1}{\lambda_i^2}||\mathcal{R}_i\mathcal{X} - \mathcal{L}_i||_F^2 + rank_2(\mathcal{L}_i)\right)$$

where $Y \in R^{M\times N\times B}$ is the noisy data, R_iX represents the constructed tensor for each exemplar cubic, ∇_z denotes the first-order forward finite-difference operator along the z-axis (spectral direction), p ($0 \le p \le 1$) is the parameter to control the sparsity of hyper-Laplacian, μ and ω are the regularization parameters.

(a) Original (PSNR:dB) (b) Noisy (16.09) (c) WNNM (26.99) (d) LSCD (27.17) (e) CBM3D (28.03) (f) LLRT (**28.32**)

Hyper-Laplacian Regularized Unidirectional Low-Rank Tensor Recovery for Multispectral Image Denoising. CVPR 2017

Image Restoration (CVPR'2018)

- **Tensor decomposition:**

$$\mathcal{L}_p = \sum_{r \in S_\ell} \lambda_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r$$

- **Compression sensing:**

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$$

- **Our optimization:**

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \eta \sum_p \|\tilde{\mathcal{T}}_p \boldsymbol{x} - \mathcal{L}_p\|_F^2$$

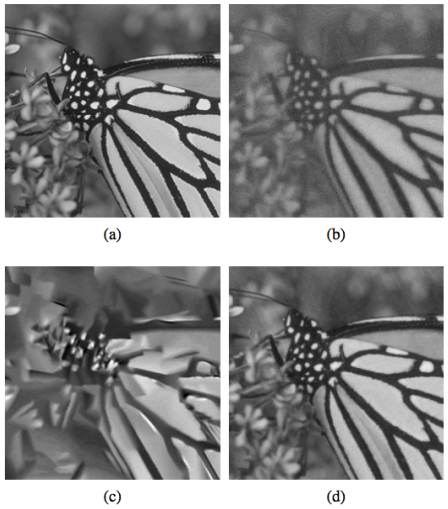


Figure 4: Reconstruction results from noiseless CS measurements for Monarch at CSr= 0.06. (a) Original image; (b) TVAL3 (19.82 dB); (c) NLR-CS (18.51 dB); (d) Ours (28.21 dB).

Table 1: Reconstruction PSNR (dB) and SSIM of 8 images using different CS recovery methods at different CSr.

Image	Method	CSr				
		0.02	0.04	0.06	0.08	0.10
Barbara	BM3D-CS	16.44, 0.389	18.03, 0.422	19.70, 0.470	21.85, 0.561	23.70, 0.635
	TVAL3	21.32, 0.588	22.58, 0.620	23.28, 0.639	23.78, 0.654	24.25, 0.667
	NLR-CS	16.79, 0.400	18.90, 0.433	20.56, 0.501	24.00, 0.659	25.13, 0.725
	D-AMP	15.08, 0.354	17.76, 0.417	20.44, 0.496	22.66, 0.623	24.05, 0.660
	Ours	**26.15, 0.812**	**27.29, 0.835**	**27.40, 0.837**	**28.05, 0.848**	**28.30, 0.852**
Boats	BM3D-CS	16.76, 0.398	17.98, 0.440	20.11, 0.507	23.21, 0.624	24.65, 0.687
	TVAL3	21.58, 0.634	23.66, 0.681	24.57, 0.699	25.22, 0.713	25.75, 0.725
	NLR-CS	17.20, 0.409	19.41, 0.489	21.76, 0.559	23.76, 0.638	25.52, 0.711
	D-AMP	15.52, 0.365	18.41, 0.456	20.76, 0.528	22.85, 0.600	25.22, 0.703
	Ours	**30.79, 0.898**	**31.73, 0.909**	**31.84, 0.910**	**32.35, 0.916**	**32.33, 0.912**
Cameraman	BM3D-CS	16.84, 0.432	18.34, 0.496	20.62, 0.585	22.32, 0.654	23.85, 0.711
	TVAL3	20.33, 0.557	21.53, 0.579	22.02, 0.580	22.37, 0.578	23.19, 0.688
	NLR-CS	17.00, 0.436	19.10, 0.531	21.60, 0.620	23.99, 0.709	26.25, 0.789
	D-AMP	16.13, 0.401	17.51, 0.461	19.84, 0.562	21.74, 0.629	23.61, 0.705
	Ours	**25.88, 0.828**	**26.47, 0.829**	**26.66, 0.823**	**27.16, 0.824**	**27.64, 0.819**
Foreman	BM3D-CS	18.96, 0.610	21.06, 0.680	25.48, 0.726	29.80, 0.831	31.69, 0.866
	TVAL3	23.35, 0.777	26.81, 0.808	28.47, 0.821	29.45, 0.828	30.02, 0.833
	NLR-CS	19.64, 0.645	22.33, 0.695	28.09, 0.809	31.68, 0.865	34.15, 0.900
	D-AMP	19.42, 0.639	22.07, 0.687	26.31, 0.750	29.12, 0.813	31.90, 0.872
	Ours	**34.01, 0.934**	**34.61, 0.936**	**35.18, 0.937**	**35.33, 0.939**	**36.02, 0.939**
House	BM3D-CS	18.12, 0.518	20.35, 0.590	25.79, 0.755	29.78, 0.816	30.50, 0.824
	TVAL3	22.57, 0.706	24.58, 0.724	26.36, 0.740	27.63, 0.766	28.68, 0.814
	NLR-CS	18.80, 0.554	21.77, 0.620	27.16, 0.761	31.23, 0.835	33.66, 0.869
	D-AMP	17.74, 0.492	19.93, 0.577	24.00, 0.693	27.39, 0.758	29.84, 0.818
	Ours	**30.42, 0.873**	**31.76, 0.884**	**33.07, 0.898**	**34.32, 0.910**	**34.80, 0.913**
Lena	BM3D-CS	16.09, 0.427	18.22, 0.511	20.47, 0.603	23.55, 0.694	24.63, 0.729
	TVAL3	21.60, 0.671	23.23, 0.693	23.88, 0.697	24.41, 0.701	24.81, 0.705
	NLR-CS	16.71, 0.461	19.07, 0.545	21.36, 0.649	25.47, 0.746	25.82, 0.771
	D-AMP	15.53, 0.399	17.21, 0.485	19.93, 0.567	22.29, 0.681	24.60, 0.733
	Ours	**28.51, 0.881**	**29.70, 0.894**	**29.85, 0.889**	**30.42, 0.898**	**30.65, 0.896**
Monarch	BM3D-CS	14.55, 0.352	15.24, 0.390	17.65, 0.500	19.75, 0.581	21.64, 0.666
	TVAL3	17.31, 0.537	18.91, 0.577	19.82, 0.595	20.56, 0.611	21.24, 0.624
	NLR-CS	14.69, 0.357	16.03, 0.413	18.53, 0.552	21.84, 0.686	25.12, 0.796
	D-AMP	14.01, 0.334	14.86, 0.366	17.55, 0.497	19.84, 0.587	21.77, 0.675
	Ours	**26.14, 0.884**	**27.28, 0.895**	**28.21, 0.901**	**28.97, 0.904**	**29.25, 0.904**
Parrots	BM3D-CS	16.62, 0.506	19.32, 0.613	23.07 0.693	25.04, 0.761	26.10, 0.787
	TVAL3	21.11, 0.714	22.90, 0.722	23.47, 0.714	24.38, 0.770	25.84, 0.804
	NLR-CS	17.39, 0.550	20.84, 0.681	24.95, 0.782	28.18, 0.835	**30.35**, 0.875
	D-AMP	16.89, 0.520	19.55, 0.619	22.62, 0.684	24.10, 0.711	26.31, 0.792
	Ours	**28.28, 0.899**	**28.88, 0.893**	**29.22, 0.890**	**29.55, 0.887**	29.84, **0.885**
Average	BM3D-CS	16.80, 0.454	18.57, 0.518	21.61, 0.605	24.42, 0.690	25.85, 0.738
	TVAL3	21.15, 0.648	23.02, 0.676	23.98, 0.685	24.72, 0.703	25.47, 0.732
	NLR-CS	17.28, 0.477	19.68, 0.551	23.00, 0.654	26.27, 0.747	28.25, 0.805
	D-AMP	16.29, 0.438	18.41, 0.509	21.43, 0.610	23.75, 0.675	25.91, 0.745
	Ours	**28.77, 0.876**	**29.71, 0.884**	**30.18, 0.886**	**30.77, 0.891**	**31.10, 0.890**

Nonlocal Low-Rank Tensor Factor Analysis for Image Restoration. CVPR 2018

Face recognition (CVPR'2019)

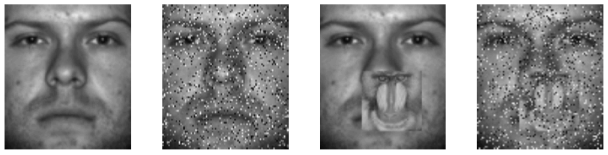
- **Sparse representation classification with Low-Rank Laplacian-Uniform**

$$\min_{\boldsymbol{x}} \ \alpha\|T_M(\boldsymbol{e}_1)\|_{w,*} + (1-\alpha)\|W\boldsymbol{e}_2\|_1 + \beta\|\boldsymbol{x}\|_1$$
$$\text{s.t.} \quad \boldsymbol{y} - D\boldsymbol{x} = \boldsymbol{e}_1 + \boldsymbol{e}_2$$

Two types of noise:1) unpredictable; 2) structured, continuous occlusion.

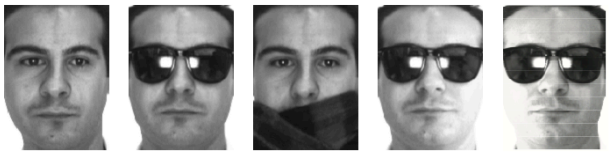
where where $T_M(\cdot)$ transforms a vector back into a 2D image. $0 < \alpha < 1$ is a weight for balancing the reweighted l_1-norm penalty and the low-rank penalty corresponding to the two types of residuals. β is the weight of the sparsity penalty of coding coefficients.

- **Robust face recognition with various noises (EYB database), real disguise (AR database), uncontrolled conditions (LFW database)**



(a) (b) (c) (d)

Figure 4. Face images of EYB under different noises. (a) A clean face image. (b) With 20% pixel corruption. (c) With 20% occlusion. (d) With 20% occlusion and 20% pixel corruption.



(a) (b) (c) (d) (e)

Figure 6. Face images in AR under real disguise. (a) A clean face image. (b) With sunglasses. (c) With scarf. (d) With sunglasses and left side light on. (e) With sunglasses and right side light on.

Methods	Clean	20% OC	40% OC
CESR [12]	66.3	60.3	32.4
S-SRC [35]	77.1	47.6	14.1
R-SRC [35]	79.9	69.3	30.7
RRC-L1 [40]	76.0	74.7	32.6
RRC-L2 [40]	76.7	67.7	24.4
HQ-M [13]	81.4	65.4	23.6
HQ-A [13]	81.3	65.4	24.0
F-IRNNLS [17]	78.9	73.0	45.7
F-LR-IRNNLS [17]	81.0	71.9	45.0
LR-LUM	**83.5**	**77.6**	**60.8**

Table 1. Accuracies (in percentage) under mixed noises on LFW. 20% OC means 20% block occlusion plus 20% pixel corruption.

Low-Rank Laplacian-Uniform Mixed Model for Robust Face Recognition. CVPR'2019

Moving Object Detection (CVPR'2019)

- Our proposed formulation seeks to decompose tensor data D into a low-rank tensor L, an illumination change tensor C, and a sparse foreground tensor S, such that

$$\min_{\mathcal{L},\mathcal{S},\mathcal{C}} \|\mathcal{L}\|_* + \lambda_1\|\mathcal{S}\|_{1,1,2} + \lambda_2(\|\mathcal{C}\|_k^{sp})^2$$
$$s.t.\ \ \mathcal{D} = \mathcal{L} + \mathcal{S} + \mathcal{C}$$

Figure 6: Columns from left to right show sample image, illumination changes, and detected moving objects for (a) cubicle and (b) backdoor sequences

Sequence	Wildlife4	Wildlife5	Wildlife6	Railway1	Railway2	Industrial area1
PCP [3]	0.4150	0.4016	0.3092	0.3634	0.4086	0.2869
DECOLOR [35]	0.3475	0.2010	0.2604	0.2853	0.3021	0.3242
ILISD [23]	0.6493	0.7012	0.6501	0.6376	0.6221	0.6089
TRPCA [18]	0.2934	0.3082	0.2855	0.3447	0.2805	0.2914
TLISD	**0.7508**	**0.8049**	**0.7522**	**0.7241**	**0.7116**	**0.7035**

Table 2: Comparison of F-measure score between our proposed method and other compared methods on EIC dataset

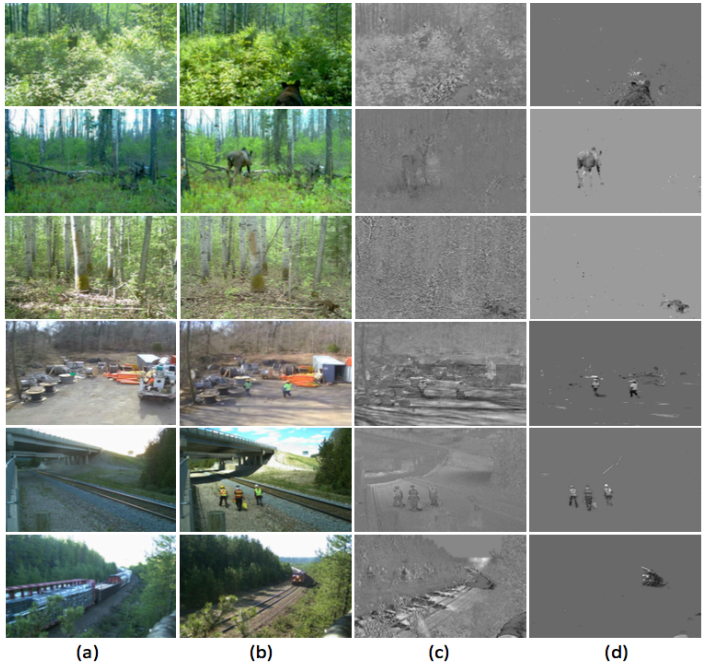


Figure 8: Columns (a) and (b): two sample images of each sequence, (c) and (d): illumination changes captured in $\mathcal{C}$, and detected objects of images in (b), respectively

Moving Object Detection Under Discontinuous Change in Illumination Using Tensor Low-Rank and Invariant Sparse Decomposition. CVPR 2019

Moving Object Detection (CVPR'2019)

- Our proposed formulation seeks to decompose tensor data D into a low-rank tensor L, an illumination change tensor C, and a sparse foreground tensor S, such that

$$\min_{\mathcal{L},\mathcal{S},\mathcal{C}} \|\mathcal{L}\|_* + \lambda_1\|\mathcal{S}\|_{1,1,2} + \lambda_2(\|\mathcal{C}\|_k^{sp})^2$$
$$s.t.\ \ \mathcal{D} = \mathcal{L} + \mathcal{S} + \mathcal{C}$$

Figure 6: Columns from left to right show sample image, illumination changes, and detected moving objects for (a) cubicle and (b) backdoor sequences

Sequence	Wildlife4	Wildlife5	Wildlife6	Railway1	Railway2	Industrial area1
PCP [3]	0.4150	0.4016	0.3092	0.3634	0.4086	0.2869
DECOLOR [35]	0.3475	0.2010	0.2604	0.2853	0.3021	0.3242
ILISD [23]	0.6493	0.7012	0.6501	0.6376	0.6221	0.6089
TRPCA [18]	0.2934	0.3082	0.2855	0.3447	0.2805	0.2914
TLISD	**0.7508**	**0.8049**	**0.7522**	**0.7241**	**0.7116**	**0.7035**

Table 2: Comparison of F-measure score between our proposed method and other compared methods on EIC dataset

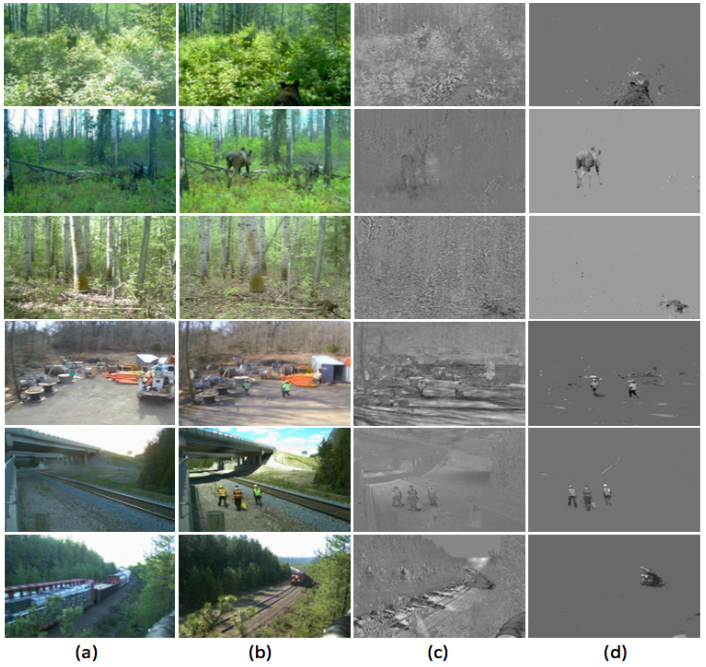


Figure 8: Columns (a) and (b): two sample images of each sequence, (c) and (d): illumination changes captured in $\mathcal{C}$, and detected objects of images in (b), respectively

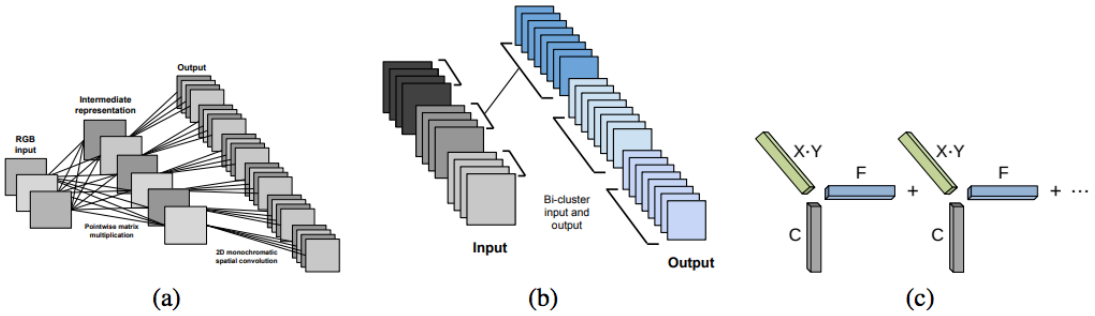
Learning a Low Tensor-Train Rank Representation for Hyperspectral Image Super-Resolution. TNNLS' 2019

Outline

- Why We Need Low Rank Decomposition?
- Mathematical Insights
- Optimization
- Recent Works in Image Problems
- **Recent Works in Neural Networks**

Decompose Tensor to R(1) x R(1) x R(1), NIPS'2014

- **Decomposes weights with Low rank approximation -> CPD**



(a) (b) (c)

$$\tilde{W}_S = \sum_{k=1}^{K} \alpha_k \otimes \beta_k \otimes \gamma_k$$

Figure 1: A visualization of monochromatic and biclustering approximation structures. **(a)** The monochromatic approximation, used for the first layer. Input color channels are projected onto a set of intermediate color channels. After this transformation, output features need only to look at one intermediate color channel. **(b)** The biclustering approximation, used for higher convolution layers. Input and output features are clustered into equal sized groups. The weight tensor corresponding to each pair of input and output clusters is then approximated. **(c)** The weight tensors for each input-output pair in (b) are approximated by a sum of rank 1 tensors using techniques described in

Approximation method	Number of parameters	Approximation hyperparameters	Reduction in weights	Increase in error
Standard colvolution	$CXYF$			
Conv layer 1: Monochromatic	$CC' + XYF$	$C' = 6$	3×	0.43%
Conv layer 2: Biclustering + outer product decomposition	$GHK(\frac{C}{G} + XY + \frac{F}{H})$	$G = 48; H = 2; K = 6$	5.3×	0.68%
Conv layer 2: Biclustering + SVD	$GH(\frac{C}{G}K_1 + K_1XYK_2 + K_2\frac{F}{H})$	$G = 2; H = 2; K_1 = 19; K_2 = 24$	3.9×	0.9%
Standard FC	NM			
FC layer 1: Matrix SVD	$NK + KM$	$K = 250$ $K = 950$	13.4× 3.5×	0.8394% 0.09%
FC layer 2: Matrix SVD	$NK + KM$	$K = 350$ $K = 650$	5.8× 3.14×	0.19% 0.06%
FC layer 3: Matrix SVD	$NK + KM$	$K = 250$ $K = 850$	8.1× 2.4×	0.67% 0.02%

Table 2: Number of parameters expressed as a function of hyperparameters for various approximation methods and empirical reduction in parameters with corresponding network performance.

- ➢ Pros: easy to be implemented
- ➢ Cons: limited speedup times

Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation, NIPS'2014

Decompose Matrix to R(1) x R(1), ICLR'2016

- **Decomposes weights with Low rank approximation**

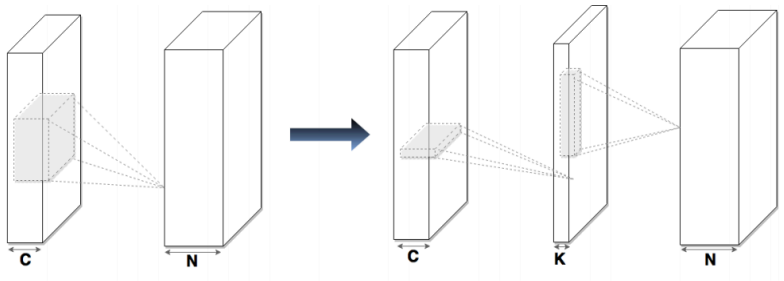


Figure 2: The proposed parametrization for low-rank regularization. Left: The original convolutional layer. Right: low-rank constraint convolutional layer with rank-K.

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

$$E_1(\mathcal{H}, \mathcal{V}) := \sum_{n,c} \|\mathcal{W}_n^c - \sum_{k=1}^{K} \mathcal{H}_n^k (\mathcal{V}_k^c)^T\|_F^2.$$

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

Table 2: CIFAR-10 performance

METHOD	WITHOUT AUG.	WITH AUG.	SPEEDUP
CNN (ours)	15.12%	12.62%	1×
Low-rank CNN (ours)	14.50%	13.10%	2.9×
CNN + Dropout (ours)	13.90%	12.29%	0.96×
Low-rank CNN + Dropout (ours)	13.81%	11.41%	2.8×
NIN (ours)	10.12%	8.19%	1×
Low-rank NIN (ours)	**8.69%**	**6.98%**	1.5×
CNN + Maxout (Goodfellow et al., 2013)	11.68%	9.38%	-
NIN (Lin et al., 2013)	10.41%	8.81%	-
CNN (Srivastava et al., 2014)	12.61%	-	-
NIN + APL units (Agostinelli et al., 2014)	9.59%	7.51%	-

- ➢ Pros: easy to be implemented
- ➢ Cons: still limited speedup times

Convolutional neural networks with low-rank regularization. ICLR'2016

Other Earlier Works

Directly using low rank decomposition (or with usual norm) to represent weights.

- Low-rank matrix factorization for deep neural network training with high-dimensional output targets, 2013
- Predicting parameters in deep learning. NIPS'2013
- Speeding up convolutional neural networks with low rank expansions, arXiv'2014
- (with CPD) Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv'2014
- Training CNNs with low-rank filters for efficient image classification, ICLR'2016
- L_1-norm low rank matrix decomposition by neural networks and mollifiers, TNNLS'2016

Lately, adding new regularization or applied to more complex system…

Coordinating Filters CNN (ICCV'2017)

- Correlation exists among trained filters in DNNs and those filters lie in a low-rank space.
- Figure2 indicates high correlation among filters within a cluster.

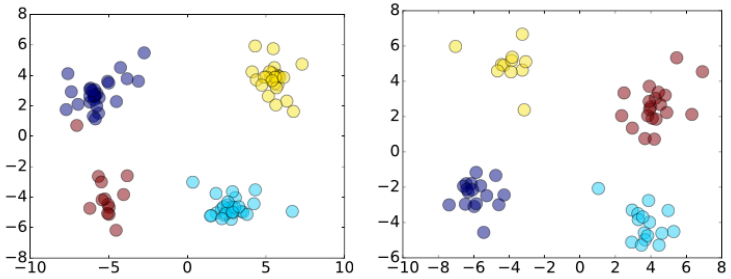


Figure 2. *Linear Discriminant Analysis* (LDA) of filters in the first convolutional layer of *AlexNet* (left) and *GoogLeNet* (right).

- Method: low rank approximate + force regularization

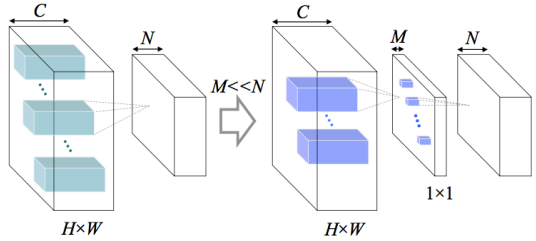


Figure 3. Cross-filter LRA of a convolutional layer.

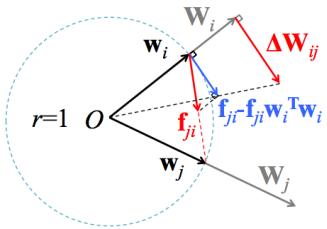


Figure 4. *Force Regularization* to coordinate filters.

$$\Delta \mathbf{W}_i = \sum_{j=1}^{N} \Delta \mathbf{W}_{ij} = ||\mathbf{W}_i|| \sum_{j=1}^{N} \left(\mathbf{f}_{ji} - \mathbf{f}_{ji} \mathbf{w}_i^T \mathbf{w}_i \right),$$

$$\mathbf{W}_i \leftarrow \mathbf{W}_i - \eta \cdot \left(\frac{\partial E(\mathcal{W})}{\partial \mathbf{W}_i} - \lambda_s \cdot \Delta \mathbf{W}_i \right)$$

$$\mathcal{O}_n \approx \left(\sum_{m=1}^{M} b_m^{(n)} \mathcal{B}_m \right) * \mathcal{I} = \sum_{m=1}^{M} \left(b_m^{(n)} \mathcal{F}_m \right).$$

Tensor W_n can be approximated by a linear combination of the basis $B_m \in R^{1\times C\times H\times W}$ ($M \ll N$) of a low-rank space $B \in R^{M\times C\times H\times W}$

Table 6. Comparison of speedup factor on *AlexNet* by state-of-the-art DNN acceleration methods.

Method	Top-5 err.	conv1	conv2	conv3	conv4	conv5	total
AlexNet in Caffe	19.97%	1.00×	1.00×	1.00×	1.00×	1.00×	**1.00**×
cp-decomposition [18]	20.97% (+1.00%)	–	4.00×	–	–	–	**1.27**×
one-shot [14]	21.67% (+1.70%)	1.48×	2.30×	3.84×	3.53×	3.13×	**2.52**×
SSL [28]	19.58% (-0.39%)	1.00×	1.27×	1.64×	1.68×	1.32×	**1.35**×
	21.63% (+1.66%)	1.05×	3.37×	6.27×	9.73×	4.93×	**3.13**×
our lra_2	20.14% (+0.17%)	2.61×	6.06×	2.48×	2.20×	1.58×	**2.69**×
	21.68% (+1.71%)	2.65×	6.22×	4.81×	4.00×	2.92×	**4.05**×

Coordinating Filters for Faster Deep Neural Networks. ICCV'2017

Knowledge Transfer (TPAMI'2018)

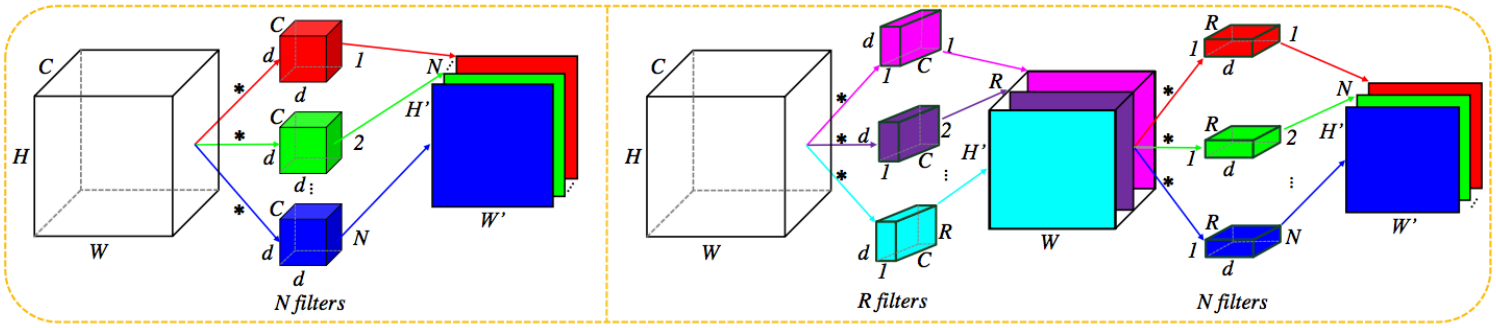
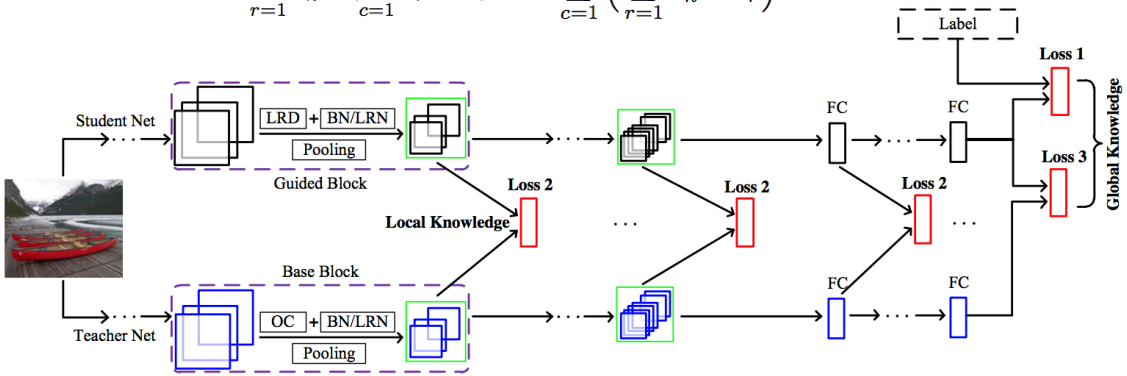


Fig. 2. Low-rank decomposition for compressing a generalized convolutional operation. The R, $H \times W \times C$, $d \times d \times C \times N$, $H' \times W' \times N$ are the rank, input size, filter size and output size, respectively. Left: Original convolution. Right: Low-rank constraint convolution with rank R

$$\mathcal{O}_n \approx \mathcal{T}_n * \mathcal{S} = \sum_{r=1}^{R} \mathcal{T}_n^r * (\sum_{c=1}^{C} \mathcal{V}_r^c * \mathcal{I}^c) = \sum_{c=1}^{C} \left(\sum_{r=1}^{R} \mathcal{T}_n^r * \mathcal{V}_r^c \right) * \mathcal{I}^c$$



LRD: Low-rank Decomposition
BN: Batch Normalization
Loss 1: Softmax Loss For Label Knowledge

OC: Original Convolution
LRN: Local Response Normalization
Loss 2: Euclidean Distance For Local Knowledge

FC: Fully-connected
Pooling: Max Pooling
Loss 3: Cross Entropy For Soften Output Knowledge

Fig. 1. The framework of the proposed LRDKT. A low-rank decomposition (LRD) based compression scheme is first constructed to form a guided block in the student network. Next, the "local" and "global" knowledge is transferred by KT in a unified way, which deals with the performance degradation caused by LRD compression. The based and guided blocks are defined in Sec. 3.3.

Architecture

TABLE 7
Speedup all convolutional layers on VGG-16. LRE* and L1* are the results based on the implementation of [24] and [38], respectively.

Method	PCA energy ratio	CPU speedup	GPU speedup	Top-1 err. ↑	Top-5 err. ↑
LRE [14]	0.3	3.0×	2.90×	15.14%	10.45%
L1 [12]	-	1.99×	2.51×	2.51%	0.96%
TE [37]	-	2.38×	2.89×	-	3.94%
Thinet-C [13]	-	1.99×	2.51×	-1.46%	-1.09%
LRE* [14]	-	3.8×	2.90×	-	9.7%
L1* [12]	-	2×	-	-	0.8%
	-	4×	-	-	8.6%
LRDKT	0.7	2.46×	2.33×	-0.57%	-0.79%
	0.5	2.63×	2.64×	1.93%	0.85%
	0.3	3.00×	2.90×	7.81%	5.89%

TABLE 10
Comparison of different methods on compressing ResNet-50. "Rank" refers to PCA energy ratio.

Method	Rank	#Param.	CPU speedup	GPU speedup	Top-1 Err. ↑	Top-5 Err. ↑
ThiNet [13]	-	16.94M	1.58×	1.42×	0.84%	0.47%
	-	12.38M	1.86×	1.69×	1.87%	1.12%
	-	8.66M	2.21×	2.03×	4.46%	2.84%
LRDFT	0.7	9.8M	2.02×	1.44×	2.36%	1.84%
	0.5	6.3M	2.26×	1.61×	8.52%	4.94%
LRDDK	0.7	9.8M	2.02×	1.44×	1.93%	1.31%
	0.5	6.3M	2.26×	1.61×	7.33%	4.76%
RKT	0.7	9.8M	2.02×	1.44×	5.33%	3.69%
	0.5	6.3M	2.26×	1.61×	14.01%	11.55%
LRDKT	0.7	9.8M	2.02×	1.44×	0.72%	0.38%
	0.5	6.3M	2.26×	1.61×	6.29%	3.74%

Holistic CNN Compression via Low-Rank Decomposition with Knowledge Transfer, TPAMI'2018

Recurrent Neural Network (TIS'2019)

- **Represent weights with tensor decomposition**

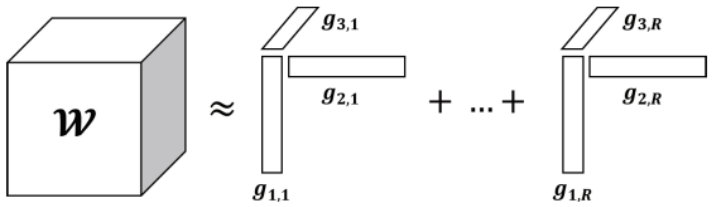


Fig. 4 CP-decomposition for 3rd-order tensor $\mathcal{W}$

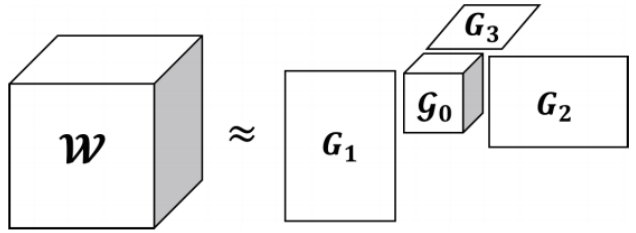


Fig. 5 Tucker decomposition for 3rd-order tensor $\mathcal{W}$

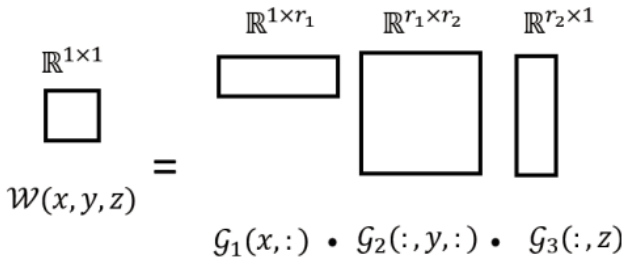


Fig. 6 Tensor Train decomposition for 3rd-order tensor $\mathcal{W}$

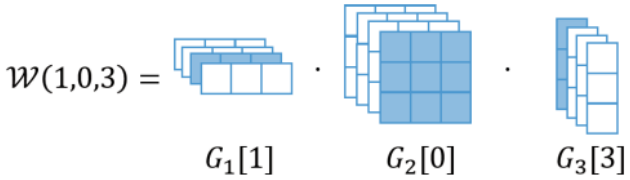


Fig. 7 Representing a tensor $\mathcal{W}$ element at $(1, 0, 3)$ using 3 TT-cores $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$. Blue shaded vectors or matrices are used for chain multiplication

Table 1 Comparison between all models and their configurations based on the number of parameters, negative log-likelihood and accuracy of polyphonic test set

Model	Config	Param	Dataset							
			Nottingham		JSB		PianoMidi		MuseData	
			NLL	ACC	NLL	ACC	NLL	ACC	NLL	ACC
GRU IN:256 OUT:512		1181184	3.369	71.1	8.32	30.24	7.53	27.19	7.12	36.30
CP-GRU IN: 4,4,4,4 OUT: 8,4,4,4	**Rank**									
	10	2456	3.79	67.51	8.60	27.29	8.15	19.03	7.87	27.32
	30	4296	3.48	69.85	8.49	28.33	7.68	25.03	7.27	36.19
	50	6136	3.46	69.56	8.40	28.47	7.66	26.18	7.23	36.34
	80	8896	3.43	69.73	8.41	27.88	7.61	28.28	7.19	36.57
	110	11656	3.34	70.42	8.41	29.45	7.60	27.36	7.18	36.89
TUCKER-GRU IN: 4,4,4,4 OUT: 8,4,4,4	**Cores**									
	2,2,2,2	2232	3.71	68.30	8.57	27.28	7.98	20.79	7.81	29.94
	2,3,2,3	4360	3.64	68.63	8.48	28.10	7.75	24.92	7.38	34.20
	2,3,2,4	6408	3.55	69.10	8.44	28.06	7.73	25.66	7.69	32.50
	2,4,2,4	10008	3.52	69.18	8.41	27.70	7.75	24.46	7.38	35.58
	2,3,3,4	12184	3.41	70.23	8.43	29.03	7.69	25.26	7.43	33.63
TT-GRU IN: 4,4,4,4 OUT: 8,4,4,4	**TT-rank**									
	1,3,3,3,1	2688	3.49	69.49	8.37	28.41	7.60	26.95	7.49	34.99
	1,5,5,5,1	4096	3.45	69.81	8.38	28.86	7.58	27.46	7.50	33.37
	1,7,7,7,1	6016	3.40	70.72	8.37	28.83	7.57	27.58	7.23	36.53
	1,9,9,9,1	8448	3.35	70.82	8.36	29.32	7.58	27.62	7.20	37.81
	1,11,11,11,1	11392	3.38	70.51	8.37	29.55	7.58	28.07	7.16	36.54

Recurrent Neural Network Compression Based on Low-Rank Tensor Representation, TIS'2019

Other references

- Conference:
  - NIPS: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3ANIPS%3A
  - CVPR: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3ACVPR%3A
  - ICCV: https://dblp.uni-trier.de/search?q=low%20rank%20type%3AConference_and_Workshop_Papers%3A%20venue%3AICCV%3A
  - ECCV: https://dblp.uni-trier.de/search?q=low%20rank%20type%3AConference_and_Workshop_Papers%3A%20venue%3AECCV%3A
  - ICLR: https://dblp.uni-trier.de/search?q=low%20rank%20type%3AConference_and_Workshop_Papers%3A%20venue%3AICLR%3A
  - ICML: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3AICML%3A
  - AAAI: https://dblp.uni-trier.de/search?q=low%20rank%20type%3AConference_and_Workshop_Papers%3A%20venue%3AAAAI%3A
- Journal:
  - TIP: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3AIEEE_Trans._Image_Processing%3A
  - TPAMI: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3AIEEE_Trans._Pattern_Anal._Mach._Intell.%3A
  - TNNLS: https://dblp.uni-trier.de/search?q=low%20rank%20venue%3AIEEE_Trans._Neural_Netw._Learning_Syst.%3A