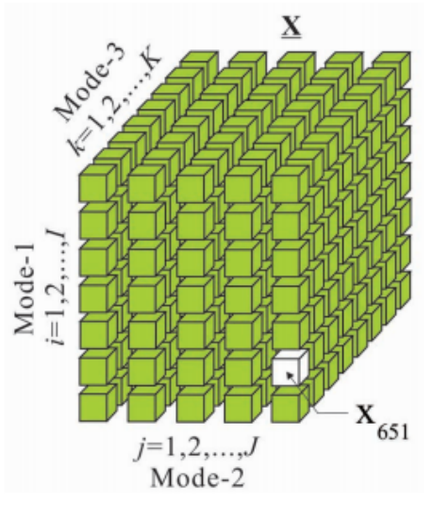
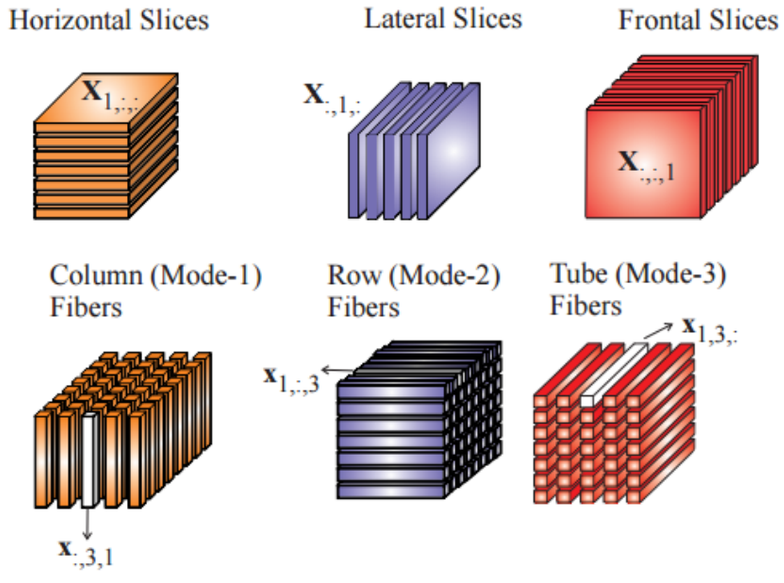
a brief survey of tensors

multilinear algebra, decompositions, method of moments and deep learning

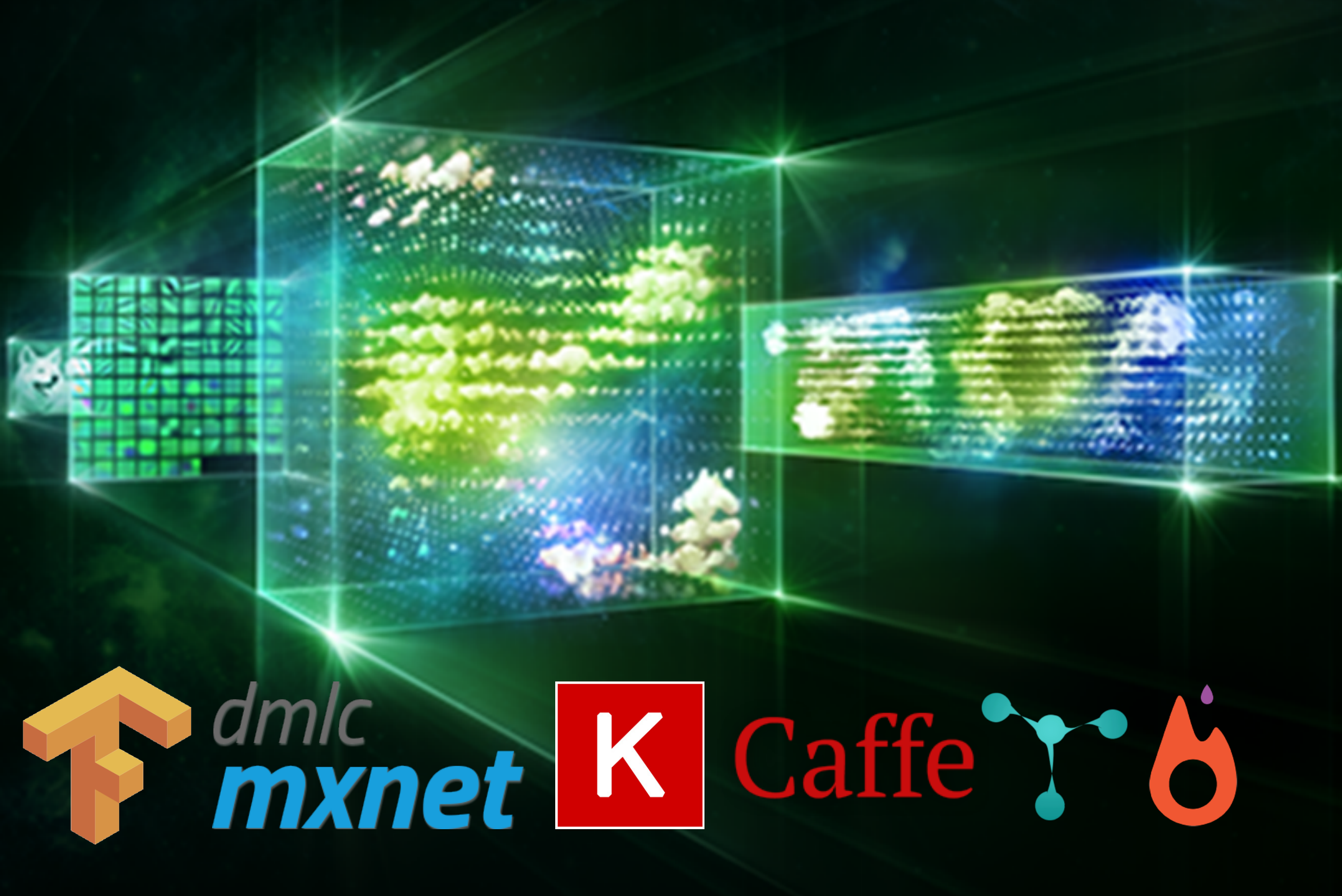
Berton Earnshaw
Senior Scientist, Savvysherpa
bertonearnshaw@savvysherpa.com





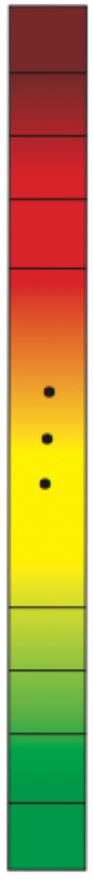
savvysherpa®

tensors, tensors everywhere…



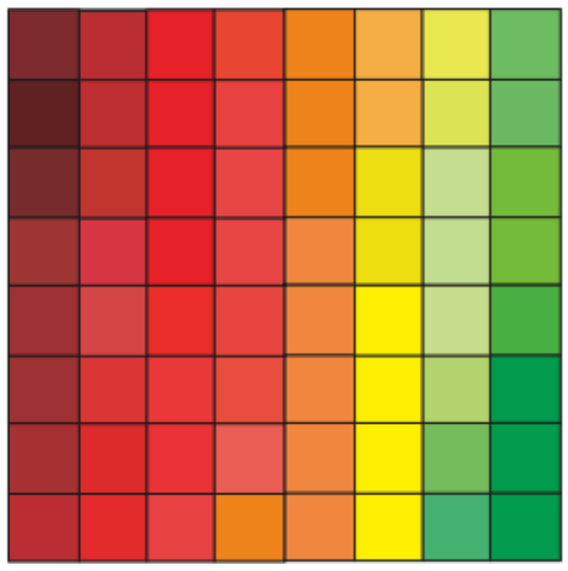
tensor = multidimensional array

vector



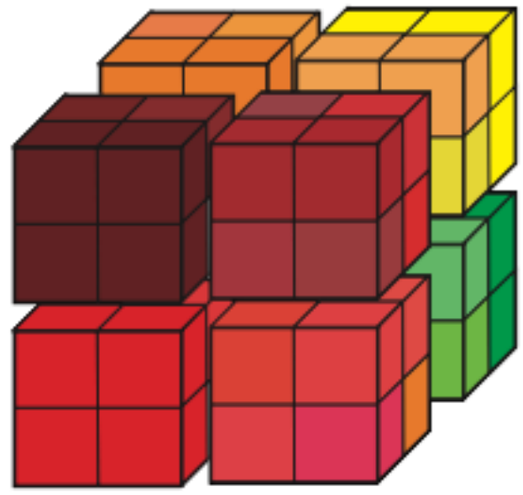
$\mathbf{v} \in \mathbb{R}^{64}$

matrix



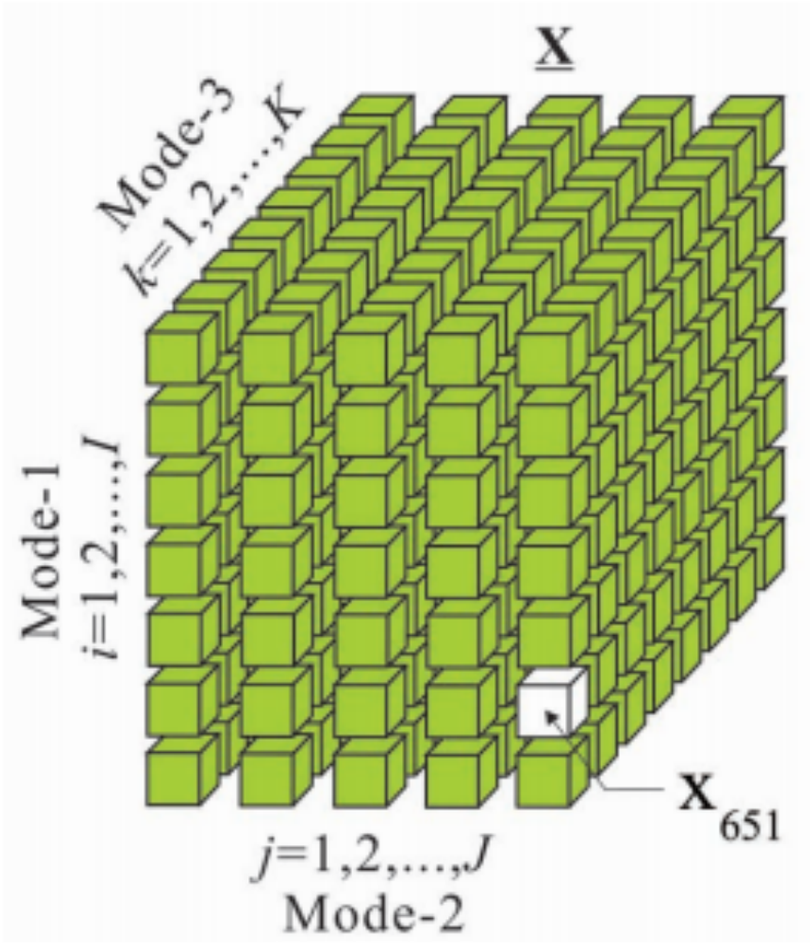
$X \in \mathbb{R}^{8x8}$

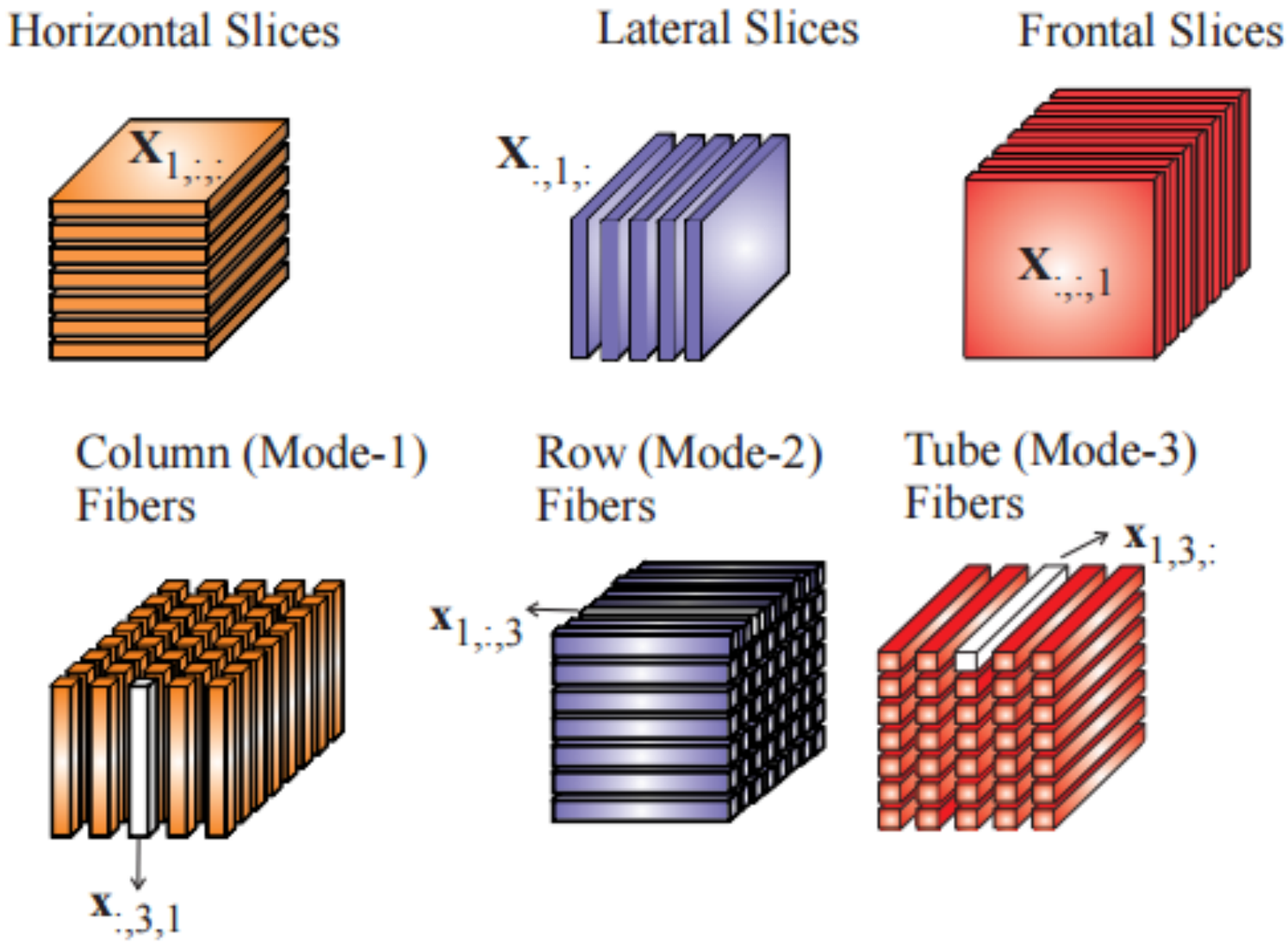
tensor



$\mathcal{X} \in \mathbb{R}^{4x4x4}$

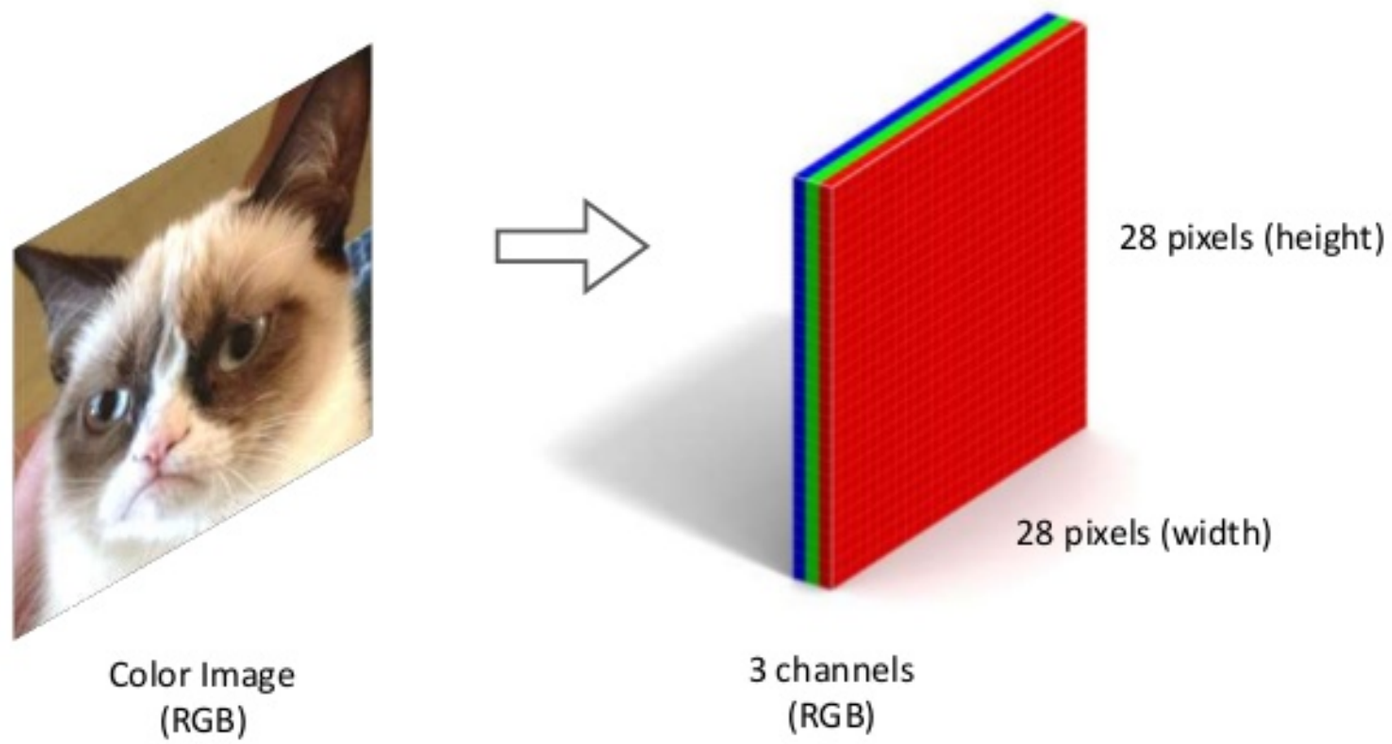
third-order tensors

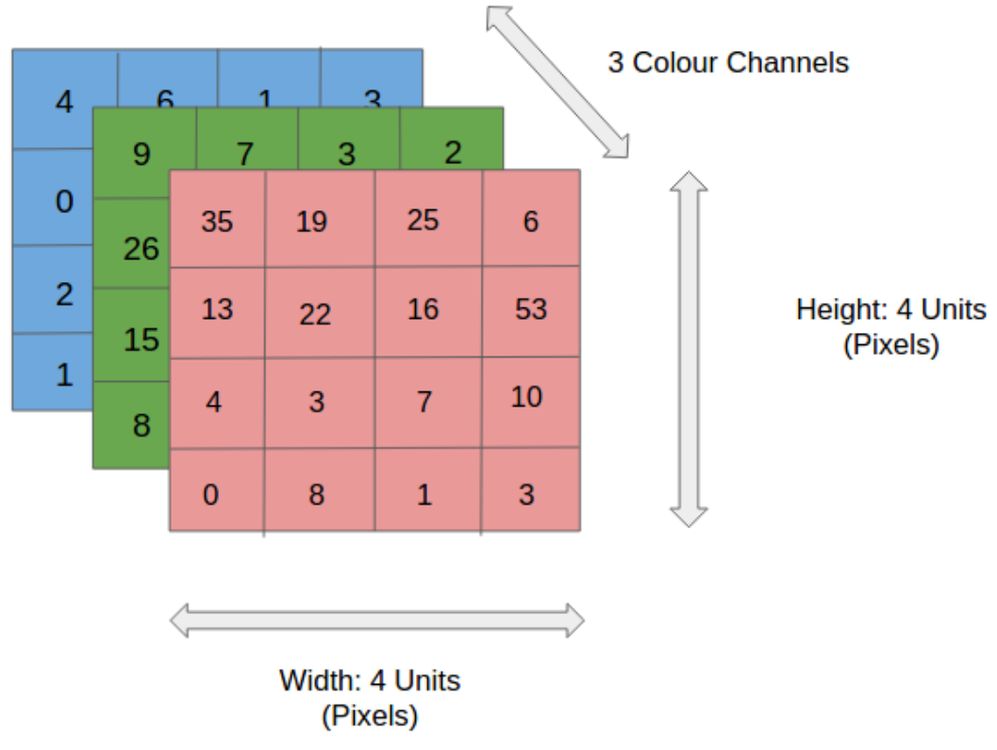




$$\mathcal{X} \in \mathbb{R}^{7x5x8}$$

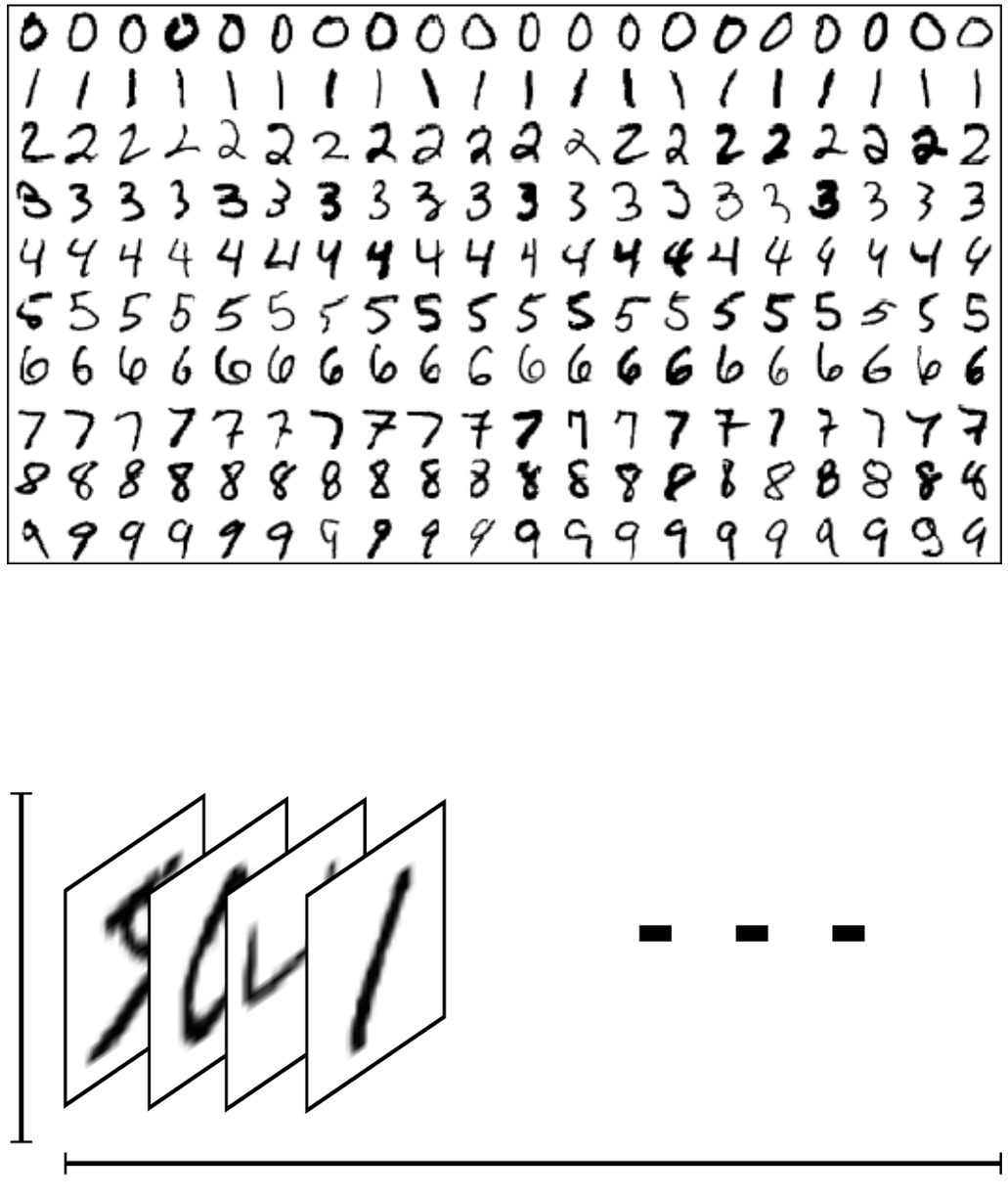
color image is 3rd-order tensor



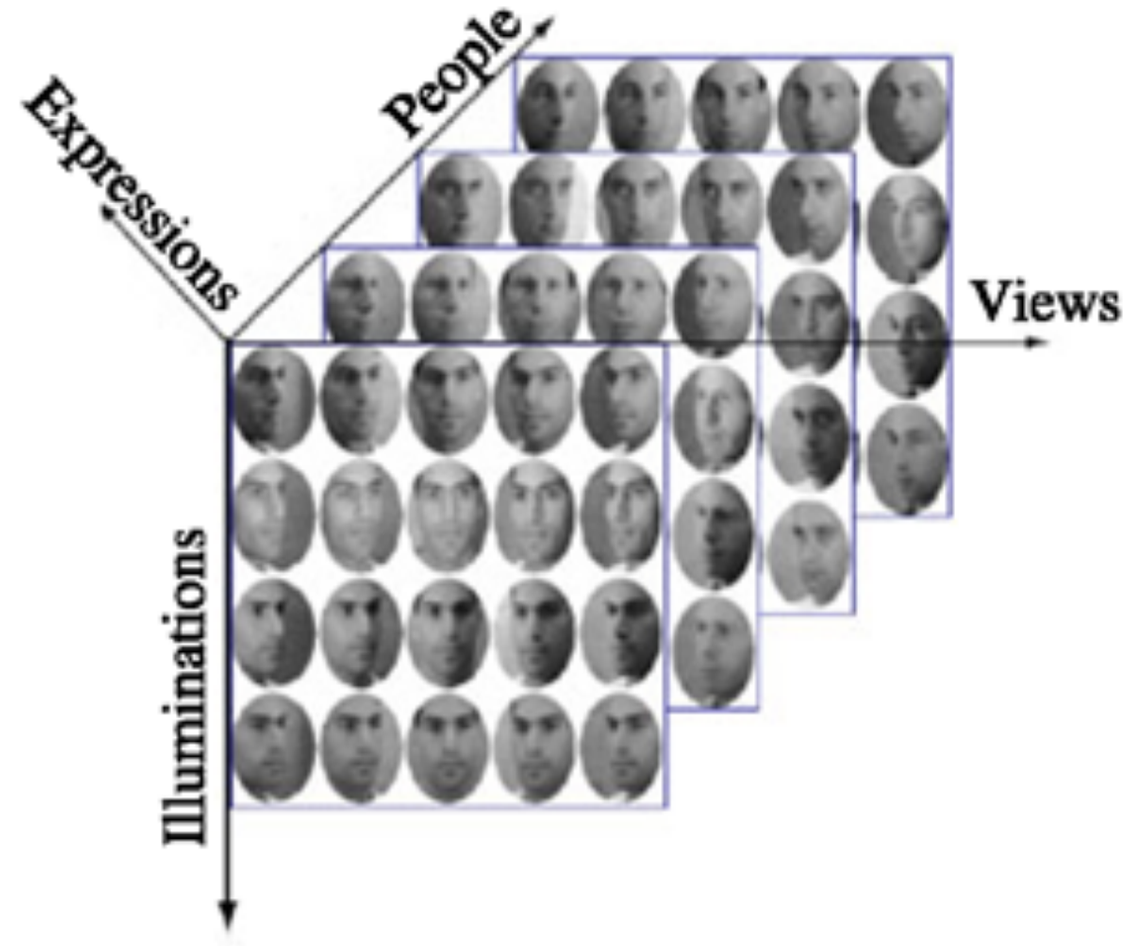


color video is 4th-order tensor

MNIST is third-order tensor



facial images database is 6th-order tensor



tensors are just bookkeeping devices!

easy right?

multilinear algebra

wrong!

multilinear algebra basics

mode-n product

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j \, i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} \; u_{j i_n}$$

$\mathcal{X} \in \mathbb{R}^{3x4x2}$

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

$\mathbf{U} \in \mathbb{R}^{2x3}$

$$\mathbf{U} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U} \in \mathbb{R}^{2x4x2}$

$$\mathbf{Y}_1 = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix}, \quad \mathbf{Y}_2 = \begin{bmatrix} 130 & 157 & 184 & 211 \\ 172 & 208 & 244 & 280 \end{bmatrix}$$

multilinear algebra basics

mode-n matricization (unfolding)

$$\mathcal{X} \in \mathbb{R}^{3x4x2} \qquad \mathbf{X}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

mode-1
$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

mode-2
$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix},$$

mode-3
$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \cdots & 21 & 22 & 23 & 24 \end{bmatrix}$$

multilinear algebra basics

mode-n matricization of multilinear product

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} \quad \Leftrightarrow$$

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)} \mathbf{X}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)} \right)^{\mathsf{T}}$$

Kronecker product

$A \in \mathbb{R}^{I \times J}$, $B \in \mathbb{R}^{K \times L}$

$A \otimes B \in \mathbb{R}^{IK \times JL}$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix}$$

tensor decompositions

decomposition = compression

$$X \approx AB^T$$

$$X \in \mathbb{R}^{I \times J},\ A \in \mathbb{R}^{I \times K},\ B \in \mathbb{R}^{J \times K}\ \ (K \ll I, J)$$

matrix	**# entries**
X	IJ
AB^T	$(I+J)K$

compression = understanding

$$X \approx AB^T$$

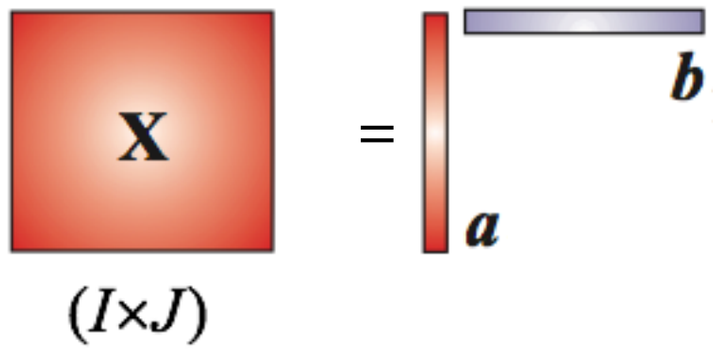
$$X \in \mathbb{R}^{I \times J},\ A \in \mathbb{R}^{I \times K},\ B \in \mathbb{R}^{J \times K}\ \ (K \ll I,J)$$

columns of B = basis of low-dim. subspace

rows of A = low-dim. representation of X

A,B encode X using fewer bits!

rank-1 matrices



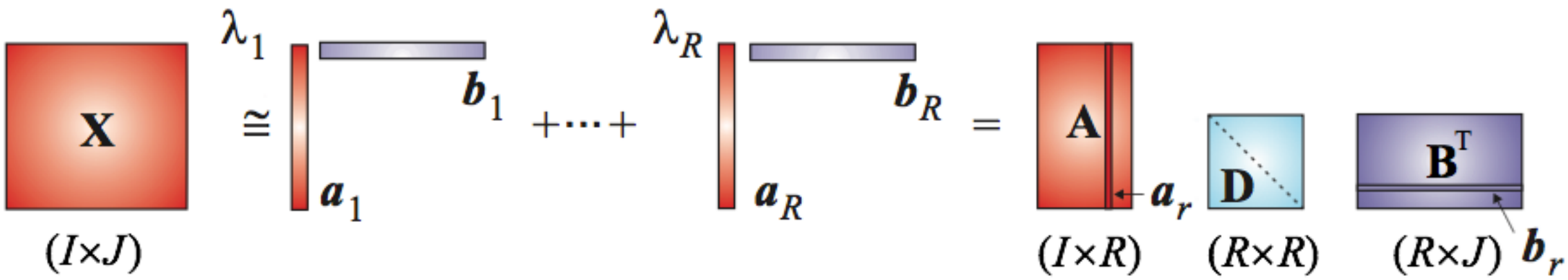
$$\mathbf{X} = \mathbf{a}\mathbf{b}^{\mathsf{T}} = \mathbf{a} \circ \mathbf{b}$$

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^{\mathrm{T}} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 \end{bmatrix}.$$

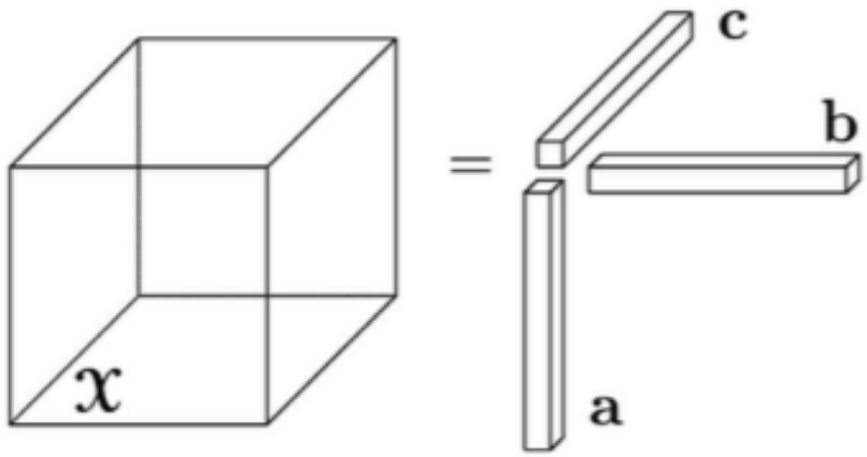
decomposition = sum rank-1 matrices

$$X \approx AB^T$$

$$AB^T = \sum^K_{k=1} \mathbf{a}_k \mathbf{b}_k^T = \sum^K_{k=1} \mathbf{a}_k \circ \mathbf{b}_k$$



rank-1 tensors

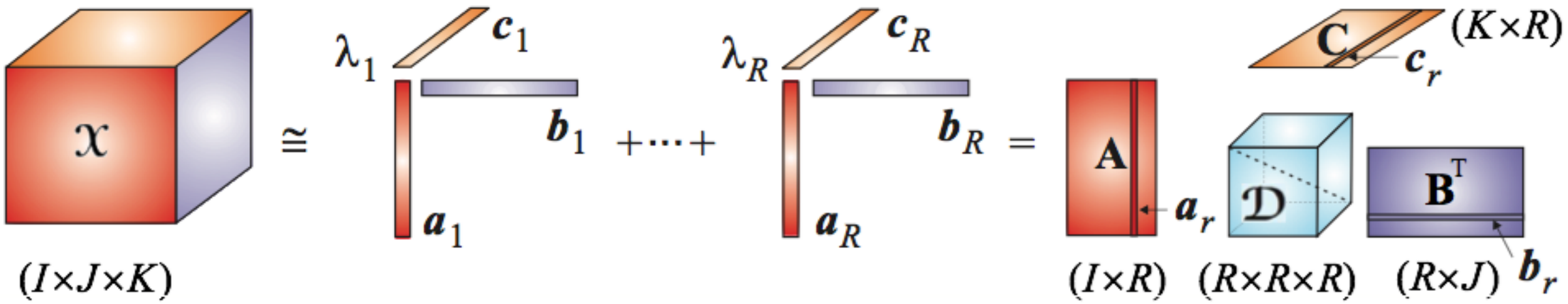


$$\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$$

decomposition = sum rank-1 tensors

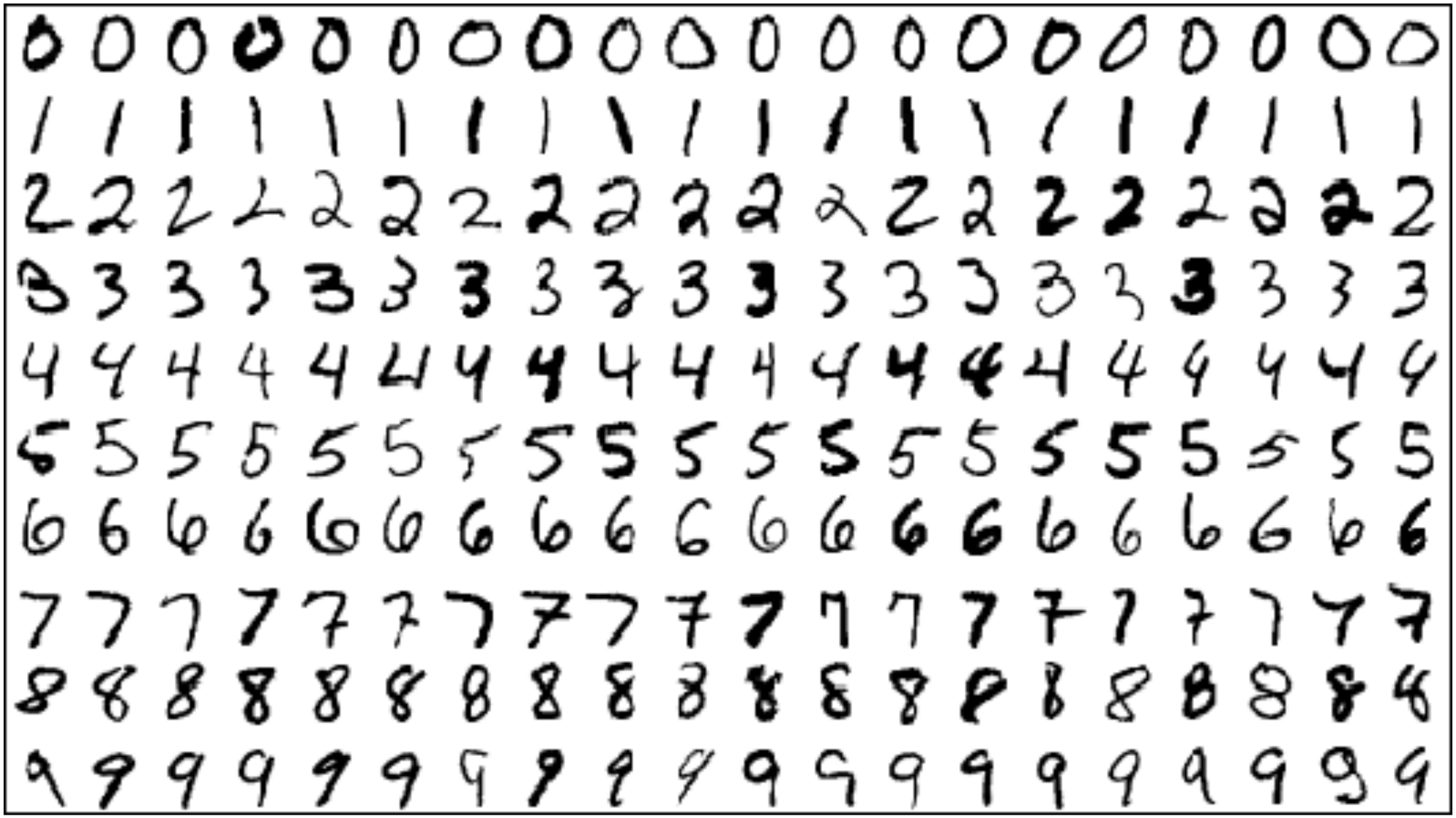
$$\boldsymbol{\mathcal{X}} \approx [[A,B,C]]$$

$$[[A,B,C]] = \sum^{K}_{k=1} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$$

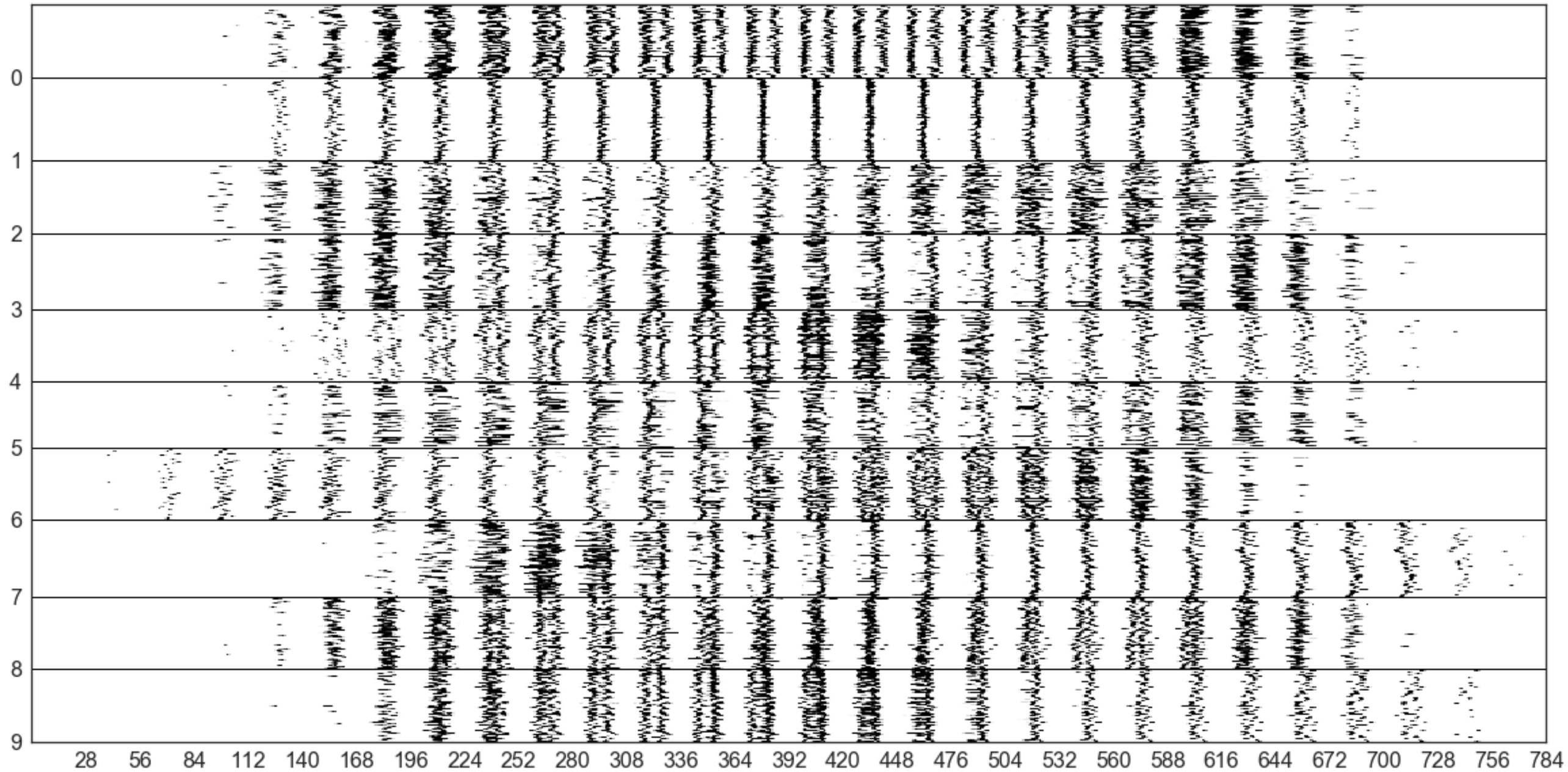


When K is minimal, called the *Canonical Polyadic Decomposition*

MNIST decomposition

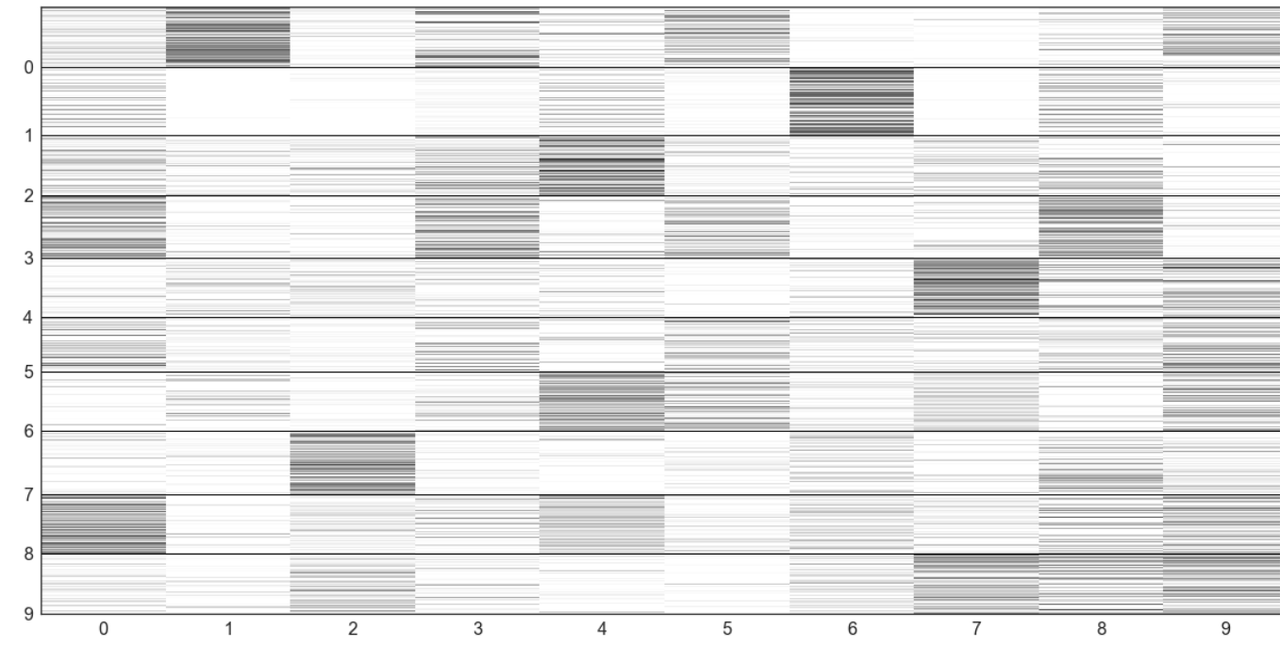


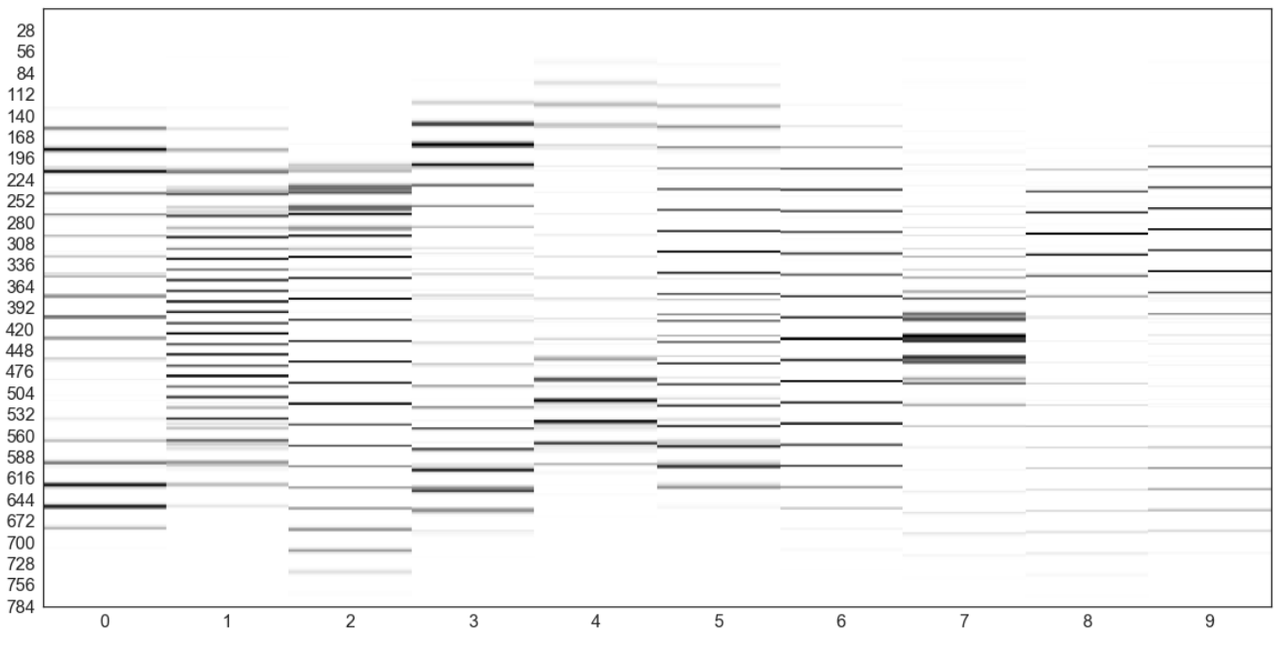
MNIST as matrix

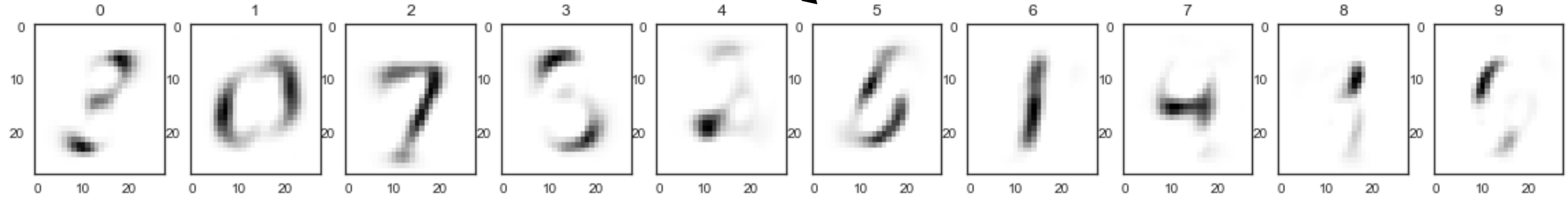


MNIST matrix decomposition

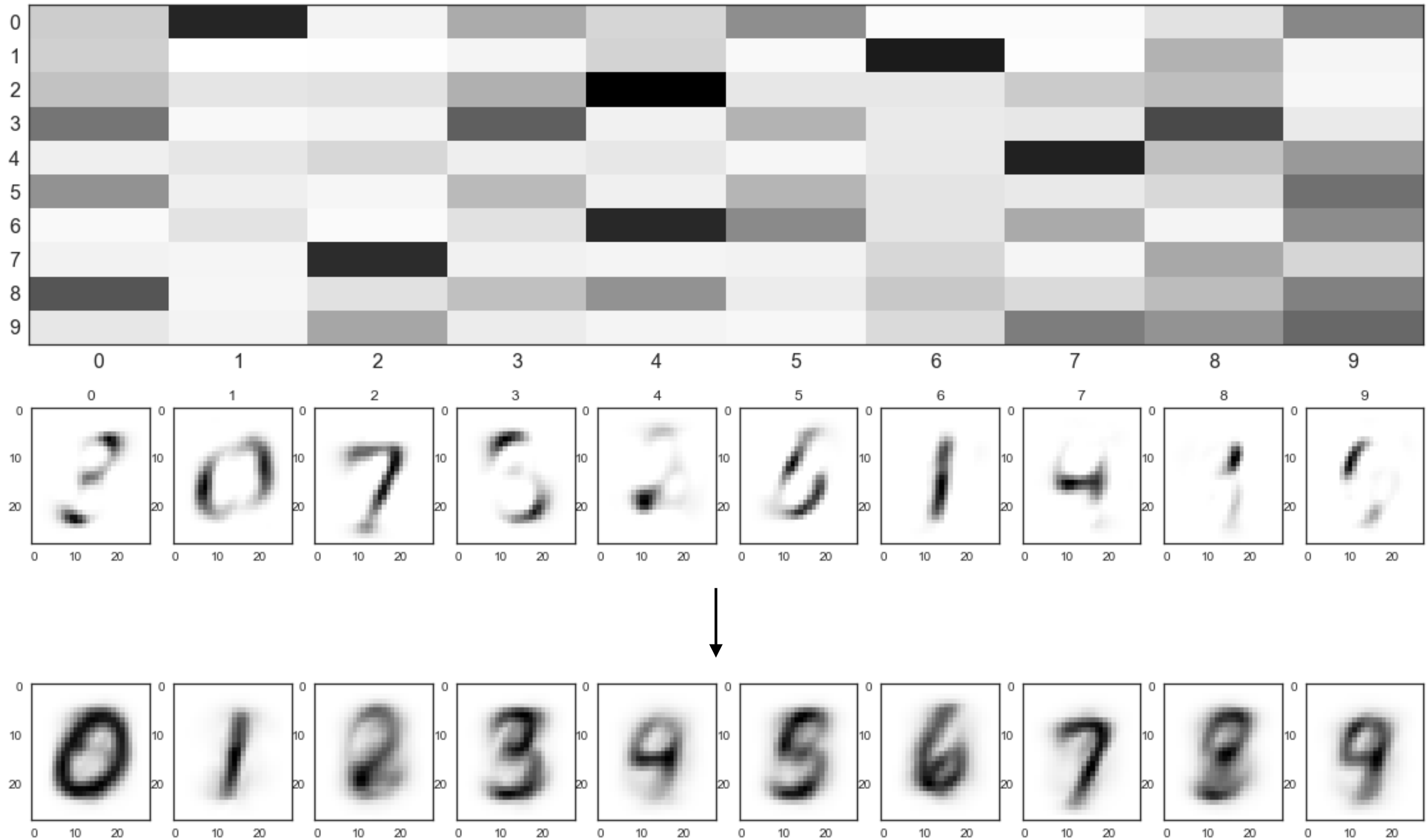
$$X \approx AB^T = \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \qquad (K = 10)$$







MNIST matrix reconstruction

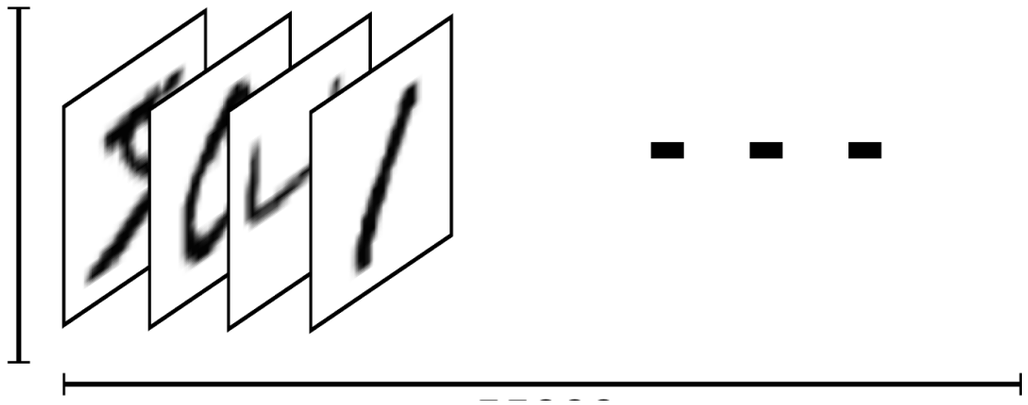


representation size

original	70,000 * (28 * 28) = 54,880,000	
matrix	(70,000 + (28 * 28)) * 10 = 707,840	1.29%

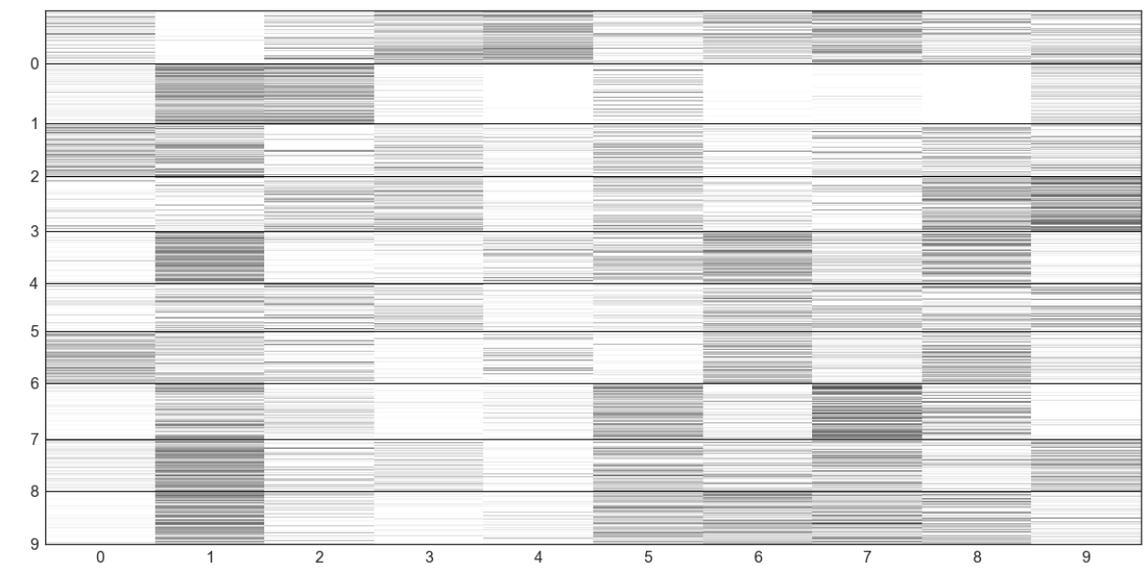
matrix decomposition requires ~1% original memory!

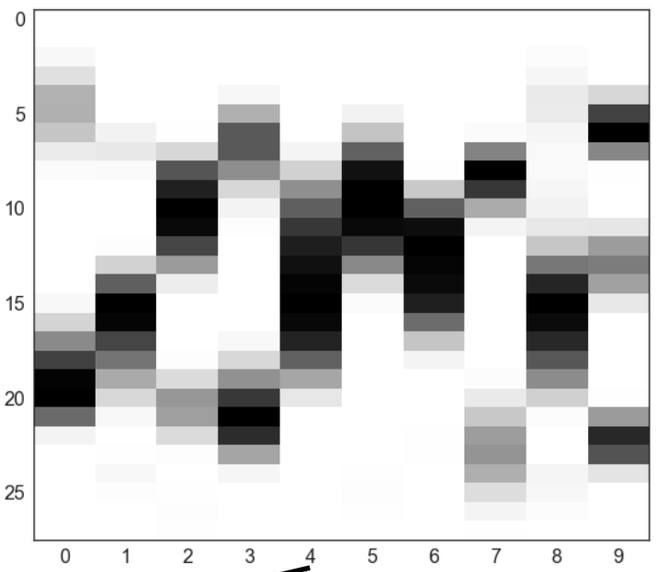
MNIST as tensor

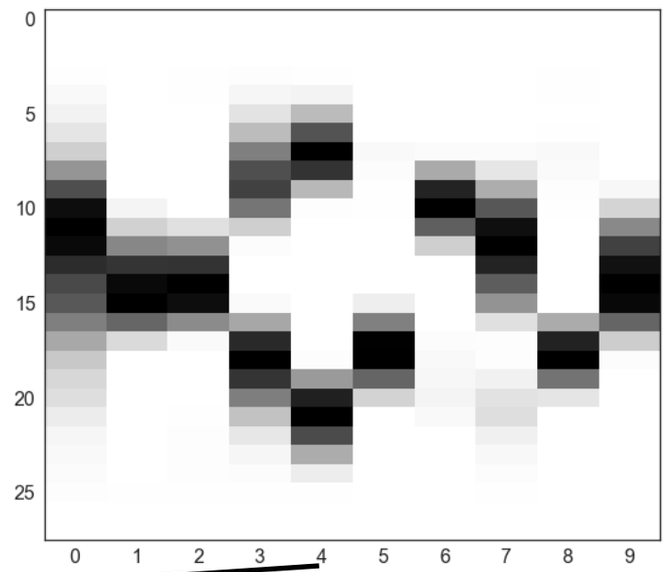


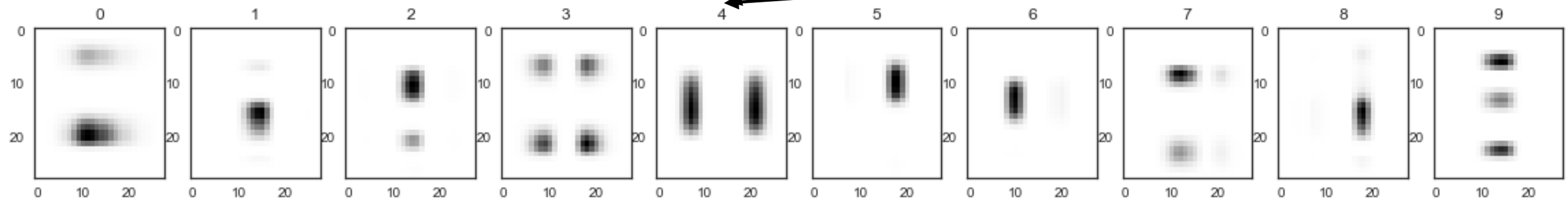
MNIST tensor decomposition

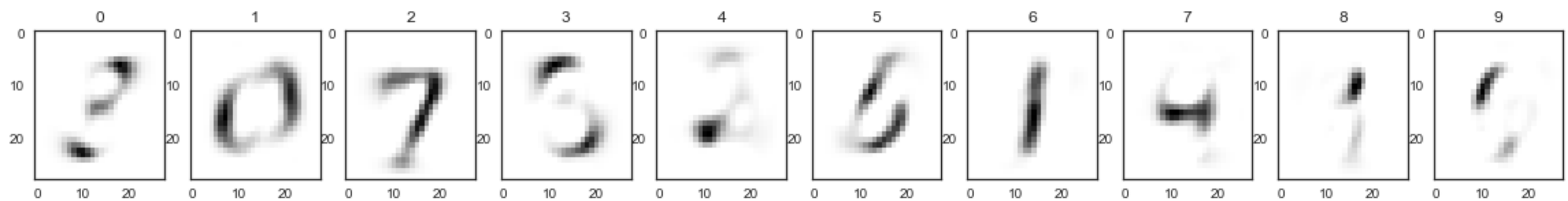
$$\boldsymbol{\mathcal{X}} \approx [[A,B,C]] = \sum^{K}_{k=1} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k \quad (K = 10)$$



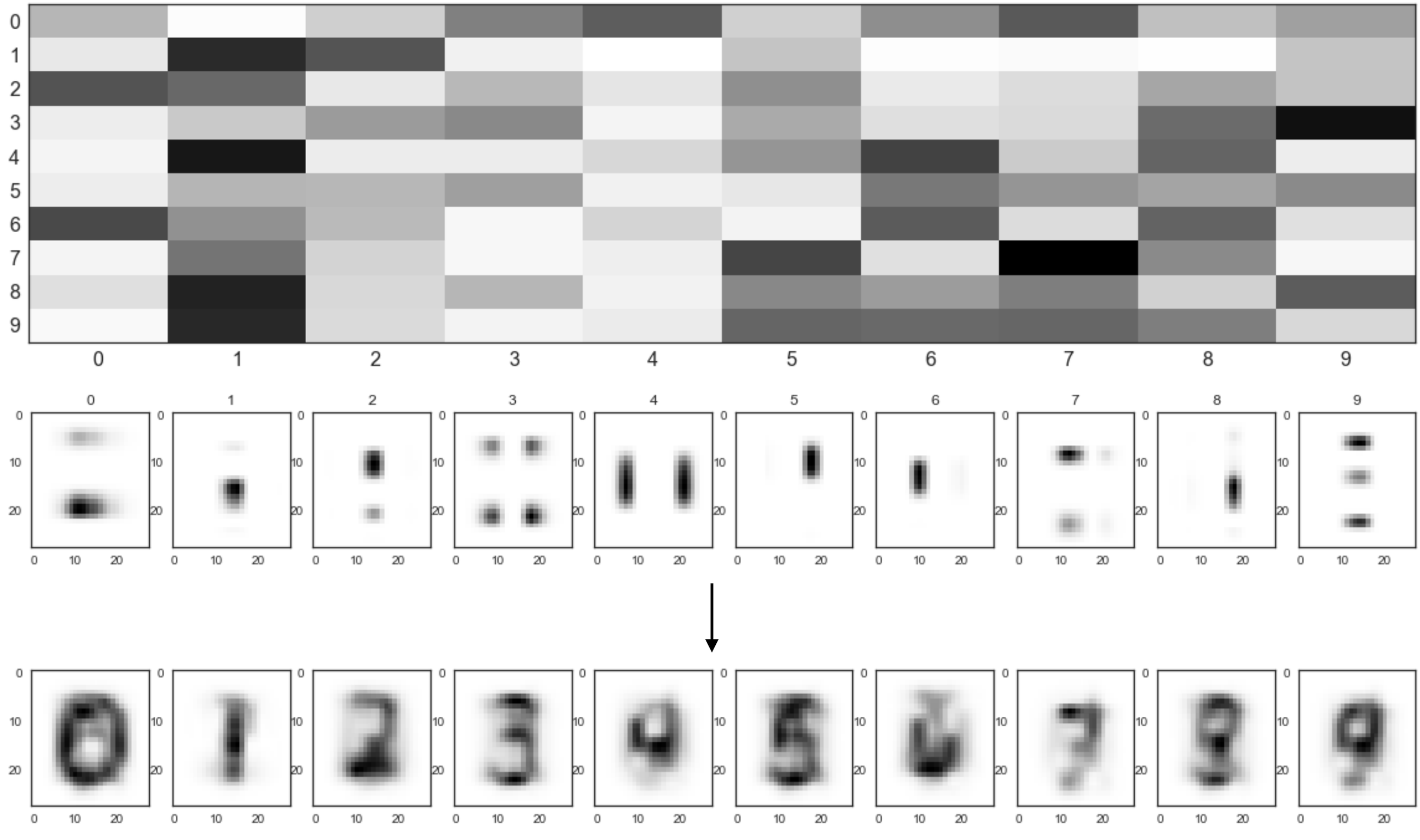








MNIST tensor reconstruction



representation size

original	70,000 * (28 * 28) = 54,880,000	
matrix	(70,000 + (28 * 28)) * 10 = 707,840	1.29%
tensor	(70,000 + 28 + 28) * 10 = 700,560	1.28%

tensor decomposition requires less memory than matrix

uniqueness of decomposition

$$\boldsymbol{\mathcal{X}} = [[A,B,C]] = \sum^{K}_{k=1} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$$

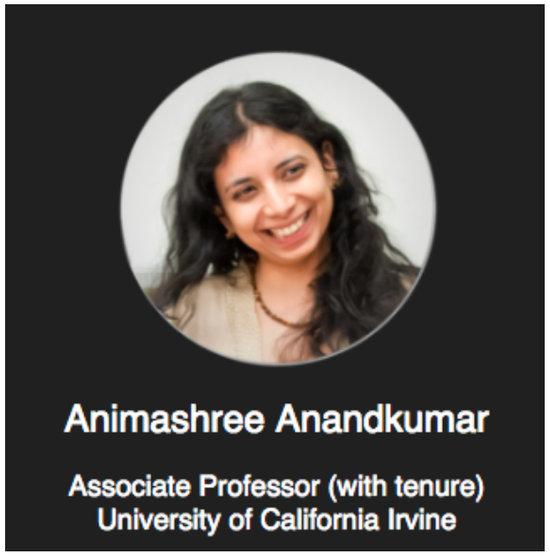
Kruskal's rank theorem: CPD is unique (up to unavoidable scaling and permutation of columns of A, B and C) if

$$r_A + r_B + r_C \geq 2K + 2$$

where r_M is the *Kruskal rank* of the matrix M (the largest number r such that any choice of r columns of M is linearly independent).

Note: rank(M) $\geq r_M$

learning via method of moments



"Tensor Decompositions for Learning Latent Variable Models"
By A. Anandkumar, R. Ge, D. Hsu, S.M. Kakade and M. Telgarsky.Â Journal of Machine Learning Research 15 (2014) 2773-2832.

"Online Tensor Methods for Learning Latent Variable Models"
By F. Huang, U. N. Niranjan, M. U. Hakeem, A. Anandkumar. Accepted to JMLR 2014.

"Beating the Perils of Non-Convexity: Guaranteed Training of Neural Networks using Tensor Methods"
By Majid Janzamin, Hanie Sedghi, Anima Anandkumar. *June 2015.*

NEW **"Provable Tensor Methods for Learning Mixtures of Generalized Linear Models"**
By H. Sedghi, M. Janzamin, A. Anandkumar.
In *the International Conference on Artificial Intelligence and Statistics (AISTATS), Cadiz, Spain, May 2016.*

NEW **"Training Input-Output Recurrent Neural Networks through Spectral Methods"**
By H. Sedghi, A. Anandkumar.

NEW **"Unsupervised Learning of Word-Sequence Representations from Scratch via Convolutional Tensor Decomposition"**
By Furong Huang, A. Anandkumar.

NEW **"Training Input-Output Recurrent Neural Networks through Spectral Methods"**
By H. Sedghi, A. Anandkumar.

learning via method of moments

Tensor Decompositions for Learning Latent Variable Models

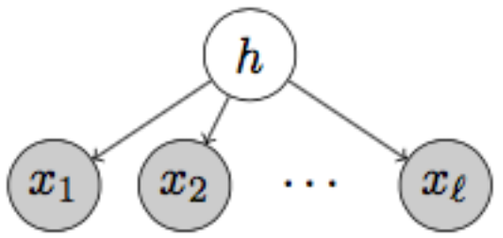
Anima Anandkumar[1], Rong Ge[2], Daniel Hsu*[3], Sham M. Kakade[2], and Matus Telgarsky[4]

Theorem 3.1 ([AHK12]). *If*

$$\begin{aligned} M_2 &:= \mathbb{E}[x_1 \otimes x_2] \\ M_3 &:= \mathbb{E}[x_1 \otimes x_2 \otimes x_3], \end{aligned}$$

then

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^{k} w_i \ \mu_i \otimes \mu_i \otimes \mu_i. \end{aligned}$$

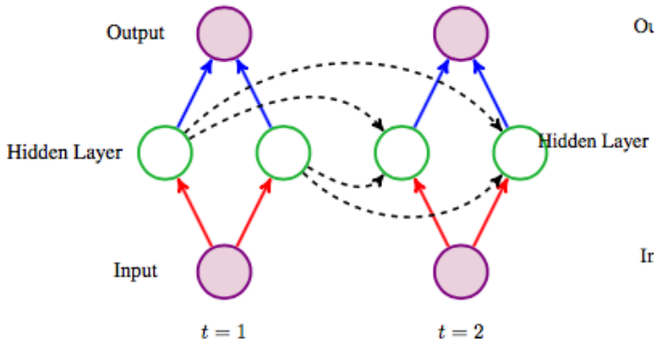


(a) Multi-view models

learning via method of moments

Training Input-Output Recurrent Neural Networks through Spectral Methods

Hanie Sedghi* Anima Anandkumar†



(b) IO-RNN

Theorem 4 (Learning parameters of RNN for quadratic activation function) *We have the following properties for an IO-RNN:*

$$\mathbb{E}\left[y_t \otimes \mathcal{S}_2(x[n], t)\right] = 2 \sum_{i \in d_h} A_2^{(i)} \otimes A_1^{(i)} \otimes A_1^{(i)}. \tag{7}$$

Hence, we can recover A_1, A_2 via tensor decomposition, assuming that they are full row rank.

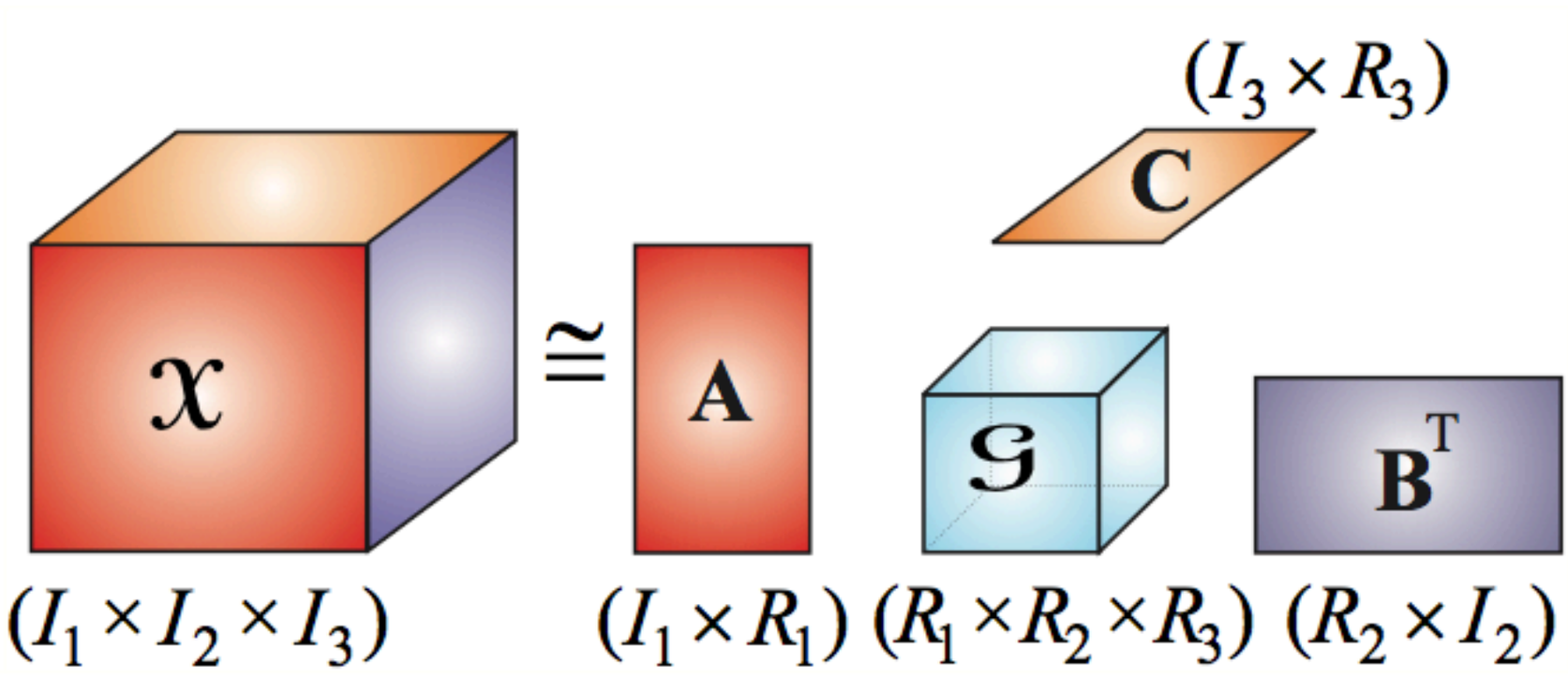
In order to learn U we form the tensor $\mathbb{E}\left[y_t \otimes \mathcal{S}_4(x_{t-1})\right]$, and under quadratic activations, we have

$$\mathbb{E}\left[y_t \otimes \textit{Reshape}(\mathcal{S}_4(x[n], t-1), [1\ 2], [3\ 4])\right] = \sum_{i \in d_h} A_2^{(i)} \otimes [U(A_1 \odot A_1)]^{(i)} \otimes [U(A_1 \odot A_1)]^{(i)}.$$

Hence, we can recover $U(A_1 \odot A_1)$ via tensor decomposition. Since A_1 is previously recovered using (7), and $(A_1 \odot A_1)^\dagger$ exists due to full rank assumption, we can recover U. Thus, Algorithm 1 (GLOREE) consistently recovers the parameters of IO-RNN under quadratic activations.

Tucker decomposition

$$\boldsymbol{\mathcal{X}} = [[\boldsymbol{\mathcal{G}};A,B,C]] = \boldsymbol{\mathcal{G}} \times_1 A \times_2 B \times_3 C$$



classification via convolutional networks

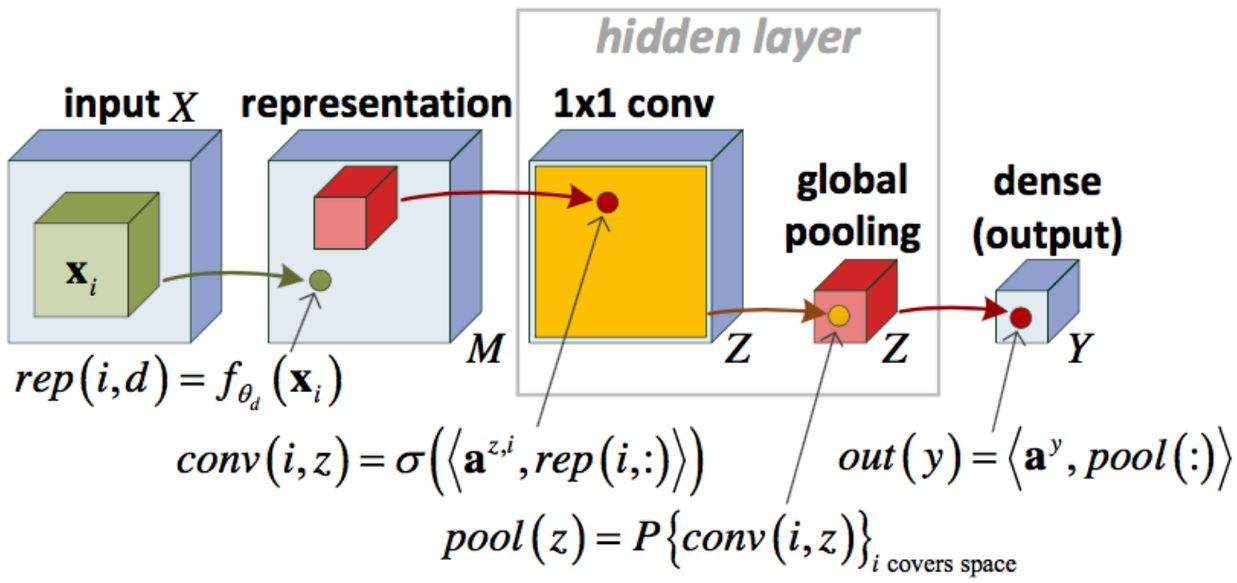
Task: learn Y score functions h_y such that the prediction rule

$$\hat{y} = \underset{y \in [Y]}{\operatorname{argmax}}\, h_y(X)$$

is as accurate as possible.

coefficient tensor $\mathcal{A}(h_y)_{d_1 \ldots d_N} = h_y(\mathbf{x}^{(d_1)}, \ldots, \mathbf{x}^{(d_N)})$

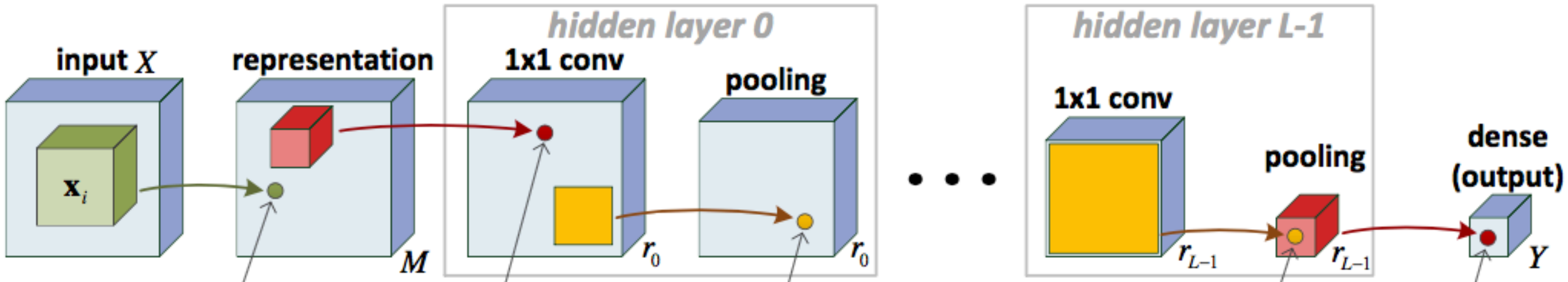
tensor decomposition $\mathcal{A}\left(h_y^S\right) = \sum_{z=1}^{Z} a_z^y \cdot (F\mathbf{a}^{z,1}) \otimes_g \cdots \otimes_g (F\mathbf{a}^{z,N})$



hierarchical Tucker decomposition

$$
\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} (F\mathbf{a}^{0,2j-1,\alpha}) \otimes_g (F\mathbf{a}^{0,2j,\alpha}) \\
&\cdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \underbrace{\phi^{l-1,2j-1,\alpha}}_{\text{order } 2^{l-1}} \otimes_g \underbrace{\phi^{l-1,2j,\alpha}}_{\text{order } 2^{l-1}} \\
&\cdots \\
\phi^{L-1,j,\gamma} &= \sum_{\alpha=1}^{r_{L-2}} a_\alpha^{L-1,j,\gamma} \underbrace{\phi^{L-2,2j-1,\alpha}}_{\text{order } \frac{N}{4}} \otimes_g \underbrace{\phi^{L-2,2j,\alpha}}_{\text{order } \frac{N}{4}}
\end{aligned}
$$

$$
\mathcal{A}(h_y^D) = \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \underbrace{\phi^{L-1,1,\alpha}}_{\text{order } \frac{N}{2}} \otimes_g \underbrace{\phi^{L-1,2,\alpha}}_{\text{order } \frac{N}{2}} \qquad (7)
$$



universality and depth efficiency

Convolutional Rectifier Networks as Generalized Tensor Decompositions

Nadav Cohen
The Hebrew University of Jerusalem
cohennadav@cs.huji.ac.il

Amnon Shashua
The Hebrew University of Jerusalem
shashua@cs.huji.ac.il

Claim 4. *Assuming covering templates exist, with ReLU activation and max pooling the shallow ConvNet is universal (hence so is the deep).*

Claim 5. *With ReLU activation and average pooling, both the shallow and deep ConvNets are not universal.*

Claim 9. *There exist weight settings for a deep ConvNet with ReLU activation and max pooling, giving rise to score functions that cannot be realized by a shallow ConvNet with ReLU activation and max pooling if the number of hidden channels in the latter (Z) is less than* $\min\{r_0, M\}^{N/2} \cdot \frac{2}{M \cdot N}$.

references

- Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51.3 (2009): 455-500.
- Cichocki, Andrzej, et al. "Tensor decompositions for signal processing applications: From two-way to multiway component analysis." IEEE Signal Processing Magazine 32.2 (2015): 145-163.
- Anandkumar, Animashree, et al. "Tensor decompositions for learning latent variable models." Journal of Machine Learning Research 15.1 (2014): 2773-2832.
- Sedghi, Hanie, and Anima Anandkumar. "Training Input-Output Recurrent Neural Networks through Spectral Methods." arXiv preprint arXiv:1603.00954 (2016).
- Cohen, Nadav, Or Sharir, and Amnon Shashua. "On the expressive power of deep learning: A tensor analysis." arXiv preprint arXiv:1509.05009 554 (2015).
- Cohen, Nadav, and Amnon Shashua. "Convolutional rectifier networks as generalized tensor decompositions." International Conference on Machine Learning (ICML). 2016.

https://github.com/bearnshaw/mnist-factorization