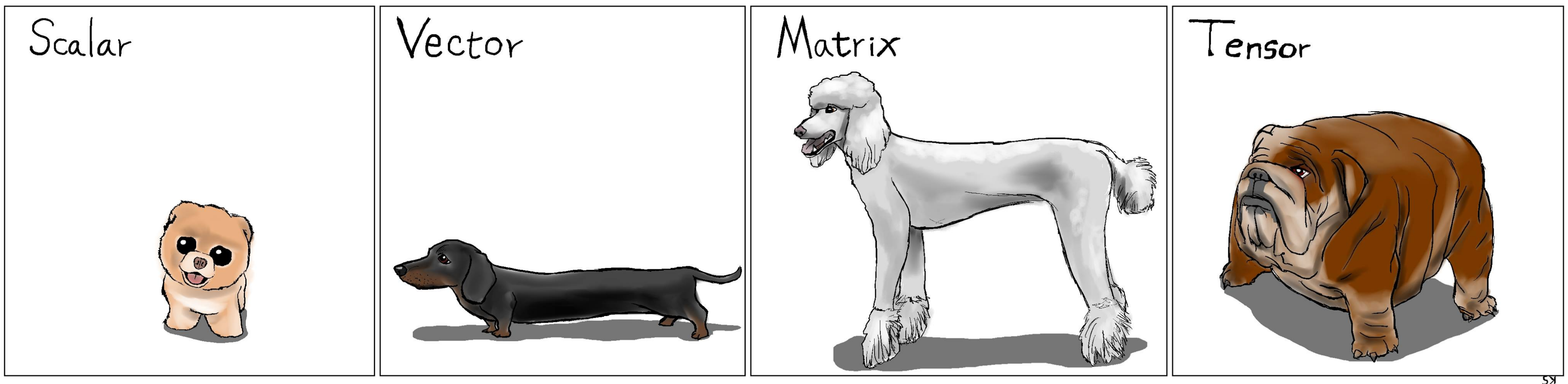
Tensor Methods: Introduction

@JeanKossaifi
jean.kossaifi@gmail.com

Outline

- Linear algebra refresher
- From linear to multi-linear algebra
- Tensor decomposition
- Low-rank tensor regression
- Combining tensor methods and deep learning

Tensors



- Most of the data we use has rich multi-dimensional structure

- Traditional methods do not leverage that structure

Linear Algebra refresher

$$\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m,p}, \quad \alpha \in \mathbb{R}$$

- Transposition: $\mathbf{C} = \mathbf{A}^\top \in \mathbb{R}^{p,m}$ → $c_{i,j} = a_{j,i}$
- Addition: $\mathbf{C} = \mathbf{A} + \mathbf{B} \in \mathbb{R}^{m,p}$ → $c_{i,j} = a_{i,j} + b_{i,j}$
- Scalar multiplication: $\mathbf{C} = \alpha\mathbf{A} \in \mathbb{R}^{m,p}$ → $c_{i,j} = \alpha a_{i,j}$

Linear Algebra refresher

Let $\mathbf{A} \in \mathbb{R}^{m,p}, \mathbf{B} \in \mathbb{R}^{p,n}$ $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m,n}$

$$c_{i,j} = ?$$

Linear Algebra refresher

$\mathbf{A} \in \mathbb{R}^{m,p}, \mathbf{B} \in \mathbb{R}^{p,n}$

$$\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m,n}$$

$$c_{i,j} = \sum_{k=0}^{R} \mathbf{a}_{i,k}\mathbf{b}_{k,j}$$

$$\overbrace{\begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n} \\ b_{1,0} & b_{1,1} & \cdots & b_{0,p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p,0} & b_{p,1} & \cdots & b_{p,n} \end{pmatrix}}^{\mathbf{B}}$$

$$\underbrace{\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,p} \\ a_{1,0} & a_{1,1} & \cdots & a_{0,p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,0} & a_{m,1} & \cdots & a_{m,p} \end{pmatrix}}_{\mathbf{A}} \quad \underbrace{\begin{pmatrix} \boxed{c_{0,0}} & c_{0,1} & \cdots & c_{0,n} \\ c_{1,0} & c_{1,1} & \cdots & c_{0,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,0} & b_{m,1} & \cdots & b_{m,n} \end{pmatrix}}_{\mathbf{C}}$$

Linear Algebra refresher

$\mathbf{A} \in \mathbb{R}^{m,p}, \mathbf{B} \in \mathbb{R}^{p,n}$ $\mathbf{C} = \mathbf{A}\mathbf{B} \in \mathbb{R}^{m,n}$

$$c_{i,j} = \sum_{k=0}^{R} \mathbf{a}_{i,k}\mathbf{b}_{k,j}$$

- Equivalent formulation:

$$\mathbf{A}\mathbf{B} = \sum_{k=0}^{R} \mathbf{a}_{:,k}\mathbf{b}_{k,:}^{\top}$$

Linear Algebra refresher

$\mathbf{A} \in \mathbb{R}^{m,p}, \mathbf{B} \in \mathbb{R}^{p,n}$ $\qquad$ $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m,n}$

$$c_{i,j} = \sum_{k=0}^{R} \mathbf{a}_{i,k}\mathbf{b}_{k,j}$$

- Equivalent formulation:

$$\mathbf{AB} = \sum_{k=0}^{R} \mathbf{a}_{:,k}\mathbf{b}_{k,:}^{\top} = \mathbf{a}_{:,k} \circ \mathbf{b}_{k,:}$$

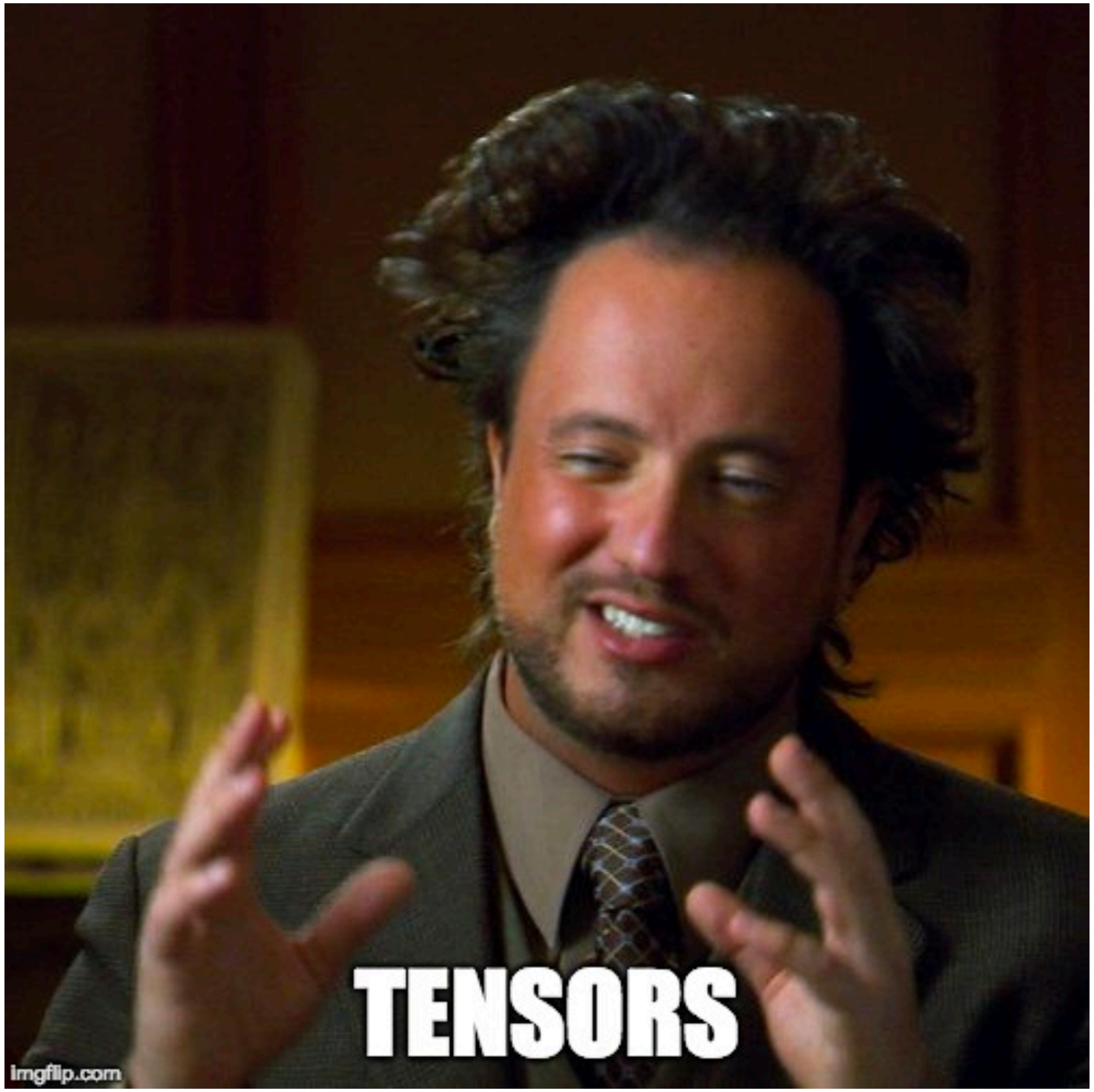
Linear Algebra refresher

$$\mathbf{A} \in \mathbb{R}^{m,p}, \mathbf{B} \in \mathbb{R}^{p,n} \qquad \mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m,n}$$

$$\mathbf{AB} = \sum_{k=0}^{R} \mathbf{a}_{:,k}\mathbf{b}_{k,:}^{\top} = \sum_{k=0}^{R} \mathbf{a}_{:,k} \circ \mathbf{b}_{k,:}$$

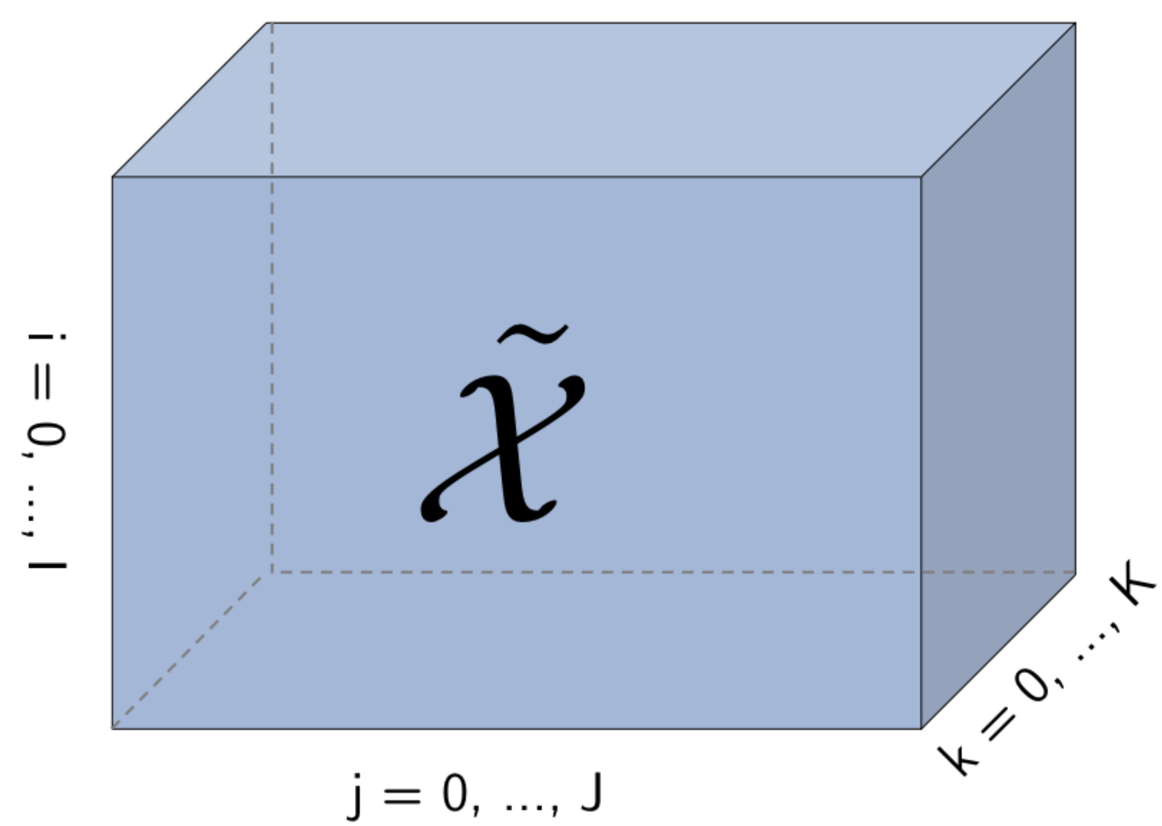
$$\begin{pmatrix} b_{0,0}, & b_{0,1}, & \cdots, & b_{0,n} \end{pmatrix} \qquad\qquad \begin{pmatrix} b_{p,0}, & b_{p,1}, & \cdots, & b_{p,n} \end{pmatrix}$$

$$\mathbf{C} = \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ \vdots \\ a_{m,0} \end{pmatrix} \begin{pmatrix} a_{0,0}b_{0,0} & a_{0,0}b_{0,1} & \cdots & a_{0,0}b_{0,n} \\ a_{1,0}b_{0,0} & a_{1,0}b_{0,1} & \cdots & a_{1,0}b_{0,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,0}b_{0,0} & a_{m,0}b_{0,1} & \cdots & a_{m,0}b_{0,n} \end{pmatrix} + \cdots + \begin{pmatrix} a_{0,p} \\ a_{1,p} \\ \vdots \\ a_{m,p} \end{pmatrix} \begin{pmatrix} a_{0,p}b_{p,0} & a_{0,p}b_{p,1} & \cdots & a_{0,p}b_{p,n} \\ a_{1,p}b_{p,0} & a_{1,p}b_{p,1} & \cdots & a_{1,p}b_{p,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,p}b_{p,0} & a_{m,p}b_{p,1} & \cdots & a_{m,p}b_{p,n} \end{pmatrix}$$


TENSORS
imgflip.com

Tensors

- Tensors can be thought of as multi-dimensional arrays, generalising the concept of matrices



Tensors

- Tensors can be thought of as multi-dimensional arrays, generalising the concept of matrices

- Order of a tensor = number of dimensions

- First order: vector $\mathbf{v} \in \mathbb{R}^{I_0}$

- Second order: matrice $\mathbf{M} \in \mathbb{R}^{I_0,I_1}$

- $\mathrm{N^{th}}$ order, N > 2: higher order tensor $\hat{\mathscr{X}} \in \mathbb{R}^{I_0,I_1,I_2,\cdots,I_N}$

- Mode = dimension (0 to N, e.g. rows, columns, …)

Indexing a tensor

$$\hat{\mathscr{X}} \in \mathbb{R}^{I_0, I_1, I_2, \cdots, I_N}$$

- element $(i_0, i_1, \cdots, i_N)$

$$\hat{\mathscr{X}}_{i_0, i_1, \cdots, i_N} \textbf{ or } \hat{\mathscr{X}}(i_0, i_1, \cdots, i_N)$$

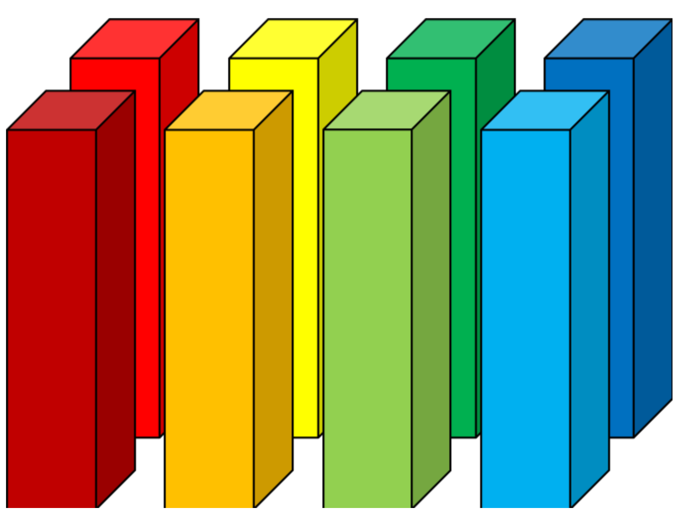
- Corresponds to viewing tensor as an array in $\mathbb{R}^{I_0, I_1, I_2, \cdots, I_N}$

  or a function $\mathbb{R}^{I_0, I_1, I_2, \cdots, I_N} \rightarrow \mathbb{R}$

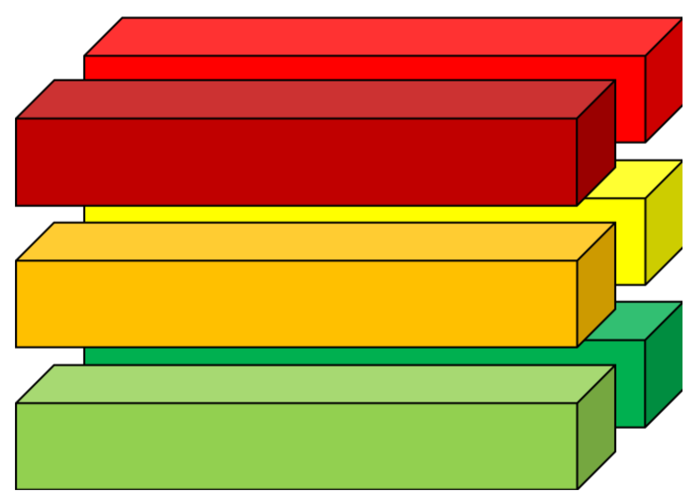
Fibers

- Fibers = generalisation of the concept of rows and columns for matrices

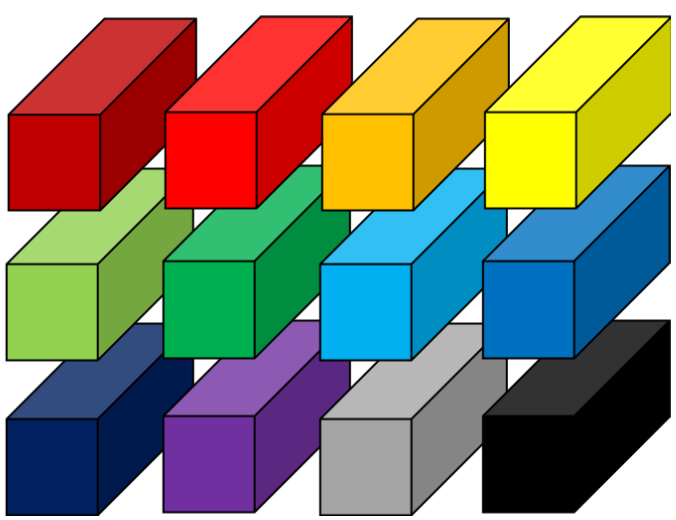
- Obtained by fixing all indices but one



Mode-0 fibers (columns)



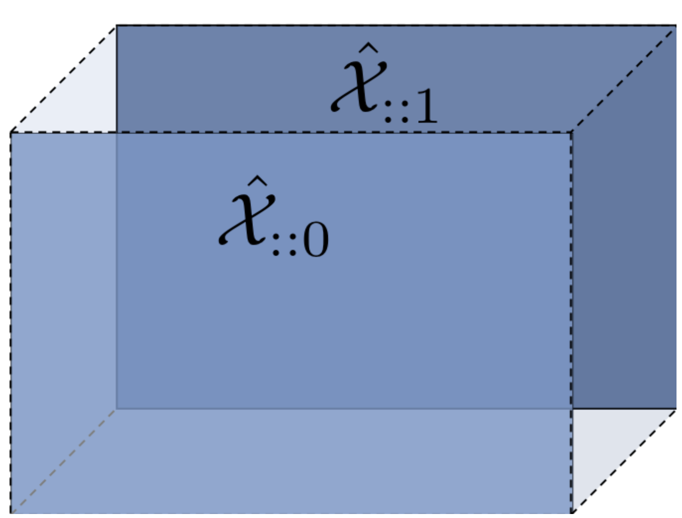
Mode-1 fibers (rows)



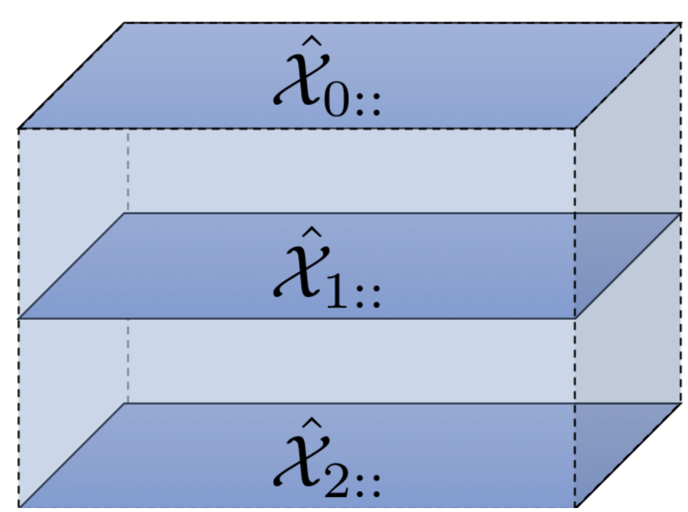
Mode-2 fibers (tubes)

Slices

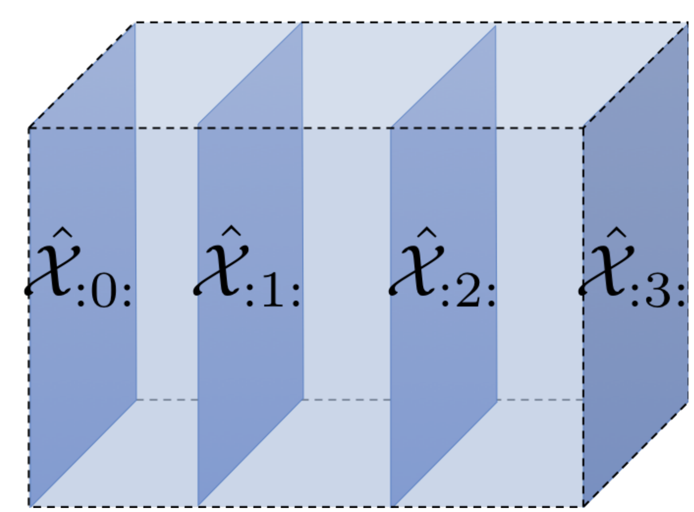
- Slices are obtained by fixing all indices but 2
- Useful to make examples by stacking matrices



Frontal slices



Horizontal slices



Lateral slices

Slices

- A tensor can be represented in multiple ways. The simplest is the slice representation through multiple matrices.

- Let's take for this example the tensor $\hat{\mathscr{X}}$ defined by its frontal slices:

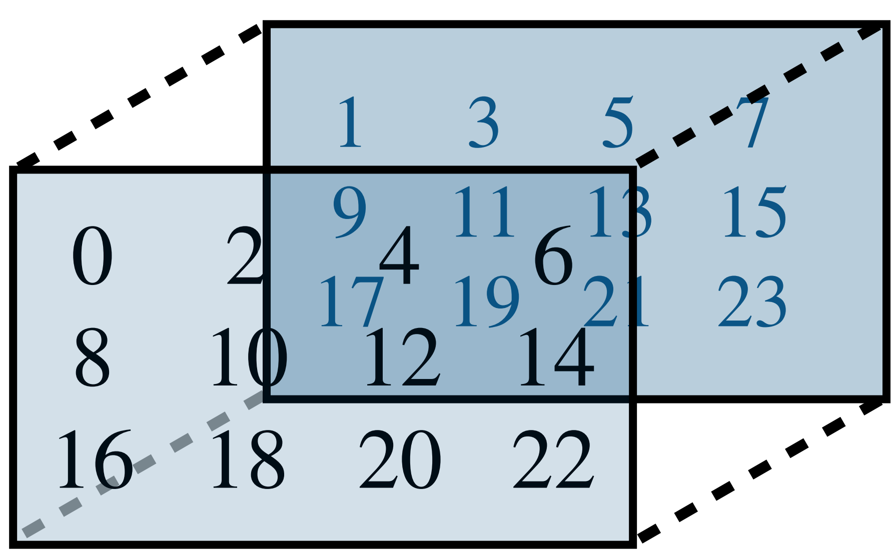
$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

Slices

- Let's take for this example the tensor $\hat{\mathcal{X}}$ defined by its frontal slices:

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$\hat{\mathcal{X}} =$



Vectorisation

- Linear transformation (isomorphism) that maps the elements of a tensor to a vector:

$$vec : \mathbb{R}^{I_0, \cdots, I_N} \to \left(I_0 \times \cdots \times I_N\right)$$
$$\hat{\mathscr{X}} \mapsto vec(\hat{\mathscr{X}})$$

- Maps element $(i_0, i_1, \cdots, i_N)$ of $\hat{\mathscr{X}}$ to element j of $vec(\hat{\mathscr{X}})$ with

$$j = \sum_{k=0}^{N} i_k \times \prod_{m=k+1}^{N} I_m$$

Vectorisation: say what?

- Maps element $(i_0, i_1, \cdots, i_N)$ of $\hat{\mathscr{X}}$ to element j of $vec(\hat{\mathscr{X}})$ with

$$j = \sum_{k=0}^{N} i_k \times \prod_{m=k+1}^{N} I_m$$

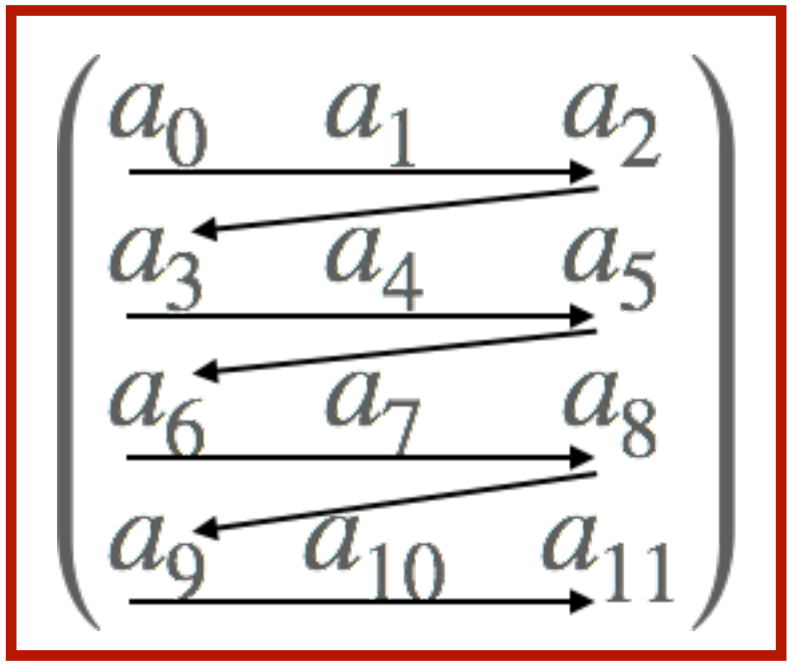
$$\mathbf{A} = \begin{array}{cc} & I_1 \\ I_0 & \begin{pmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} \end{pmatrix} \end{array}$$

$$\mathbf{A}_{1,2} = vec(\mathbf{A})_{1 \times I_0 + 2}$$
$$= vec(\mathbf{A})_5$$

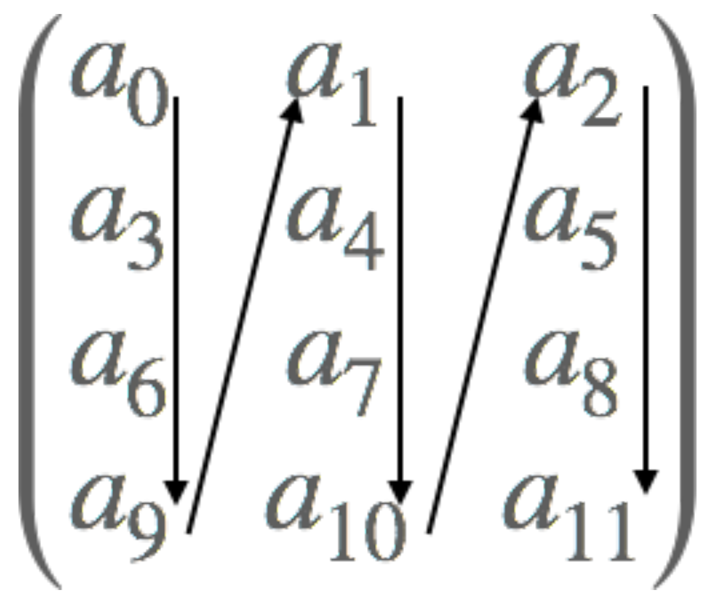
$$vec(\mathbf{A}) = \left(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\right)^\top$$

Vectorisation

- There are several definitions of vectorization:



C-ordering
(default for NumPy,
PyTorch, etc in Python)



Fortran-ordering
Matlab's default

- Just be consistent (and adapt your formulas!)

Kronecker product

$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q}$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

Kronecker product

$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q}$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$$

$$\mathbf{A} \otimes \mathbf{B} = ?$$

Kronecker product

$$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} 2\mathbf{B} & 1\mathbf{B} \\ 3\mathbf{B} & 4\mathbf{B} \end{pmatrix}$$

Kronecker product

$$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} 2\mathbf{B} & 1\mathbf{B} \\ 3\mathbf{B} & 4\mathbf{B} \end{pmatrix} = \begin{pmatrix} 1 & 2 & \frac{1}{2} & 1 \\ 4 & 0 & 2 & 0 \\ \frac{3}{2} & 3 & 2 & 4 \\ 6 & 0 & 8 & 0 \end{pmatrix}$$

Kronecker product

$$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$$

$$\mathbf{B} \otimes \mathbf{A} = ?$$

Kronecker product

$$A \in \mathbb{R}^{m,n}, B \in \mathbb{R}^{p,q} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp,nq}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \frac{1}{2} & 1 \\ 2 & 0 \end{pmatrix}$$

$$\mathbf{B} \otimes \mathbf{A} = \begin{pmatrix} \frac{1}{2}\mathbf{A} & 1\mathbf{A} \\ 2\mathbf{A} & 0\mathbf{A} \end{pmatrix} = \begin{pmatrix} 1 & \frac{1}{2} & 2 & 1 \\ \frac{3}{2} & 2 & 3 & 4 \\ 4 & 2 & 0 & 0 \\ 6 & 8 & 0 & 0 \end{pmatrix}$$

Useful properties

$$\mathbf{X} \in \mathbb{R}^{m,n}, \mathbf{A} \in \mathbb{R}^{p,n}, \mathbf{B} \in \mathbb{R}^{m,k}$$

$$vec(\mathbf{XB}) = \left(\mathbf{I}_n \otimes \mathbf{B}^\top\right) vec(\mathbf{X})$$

$$vec(\mathbf{AX}) = \left(\mathbf{A} \otimes \mathbf{I}_m\right) vec(\mathbf{X})$$

$$vec(\mathbf{AXB}) = \left(\mathbf{A} \otimes \mathbf{B}^\top\right) vec(\mathbf{X})$$

Mode-n unfolding

- Read the tensor as a matrix by re-arranging the fibers:

$$\mathbb{R}^{I_0,\cdots,I_N} \to \left(I_n, M\right)$$
$$\hat{\mathcal{X}} \mapsto \hat{\mathcal{X}}_{[n]}$$

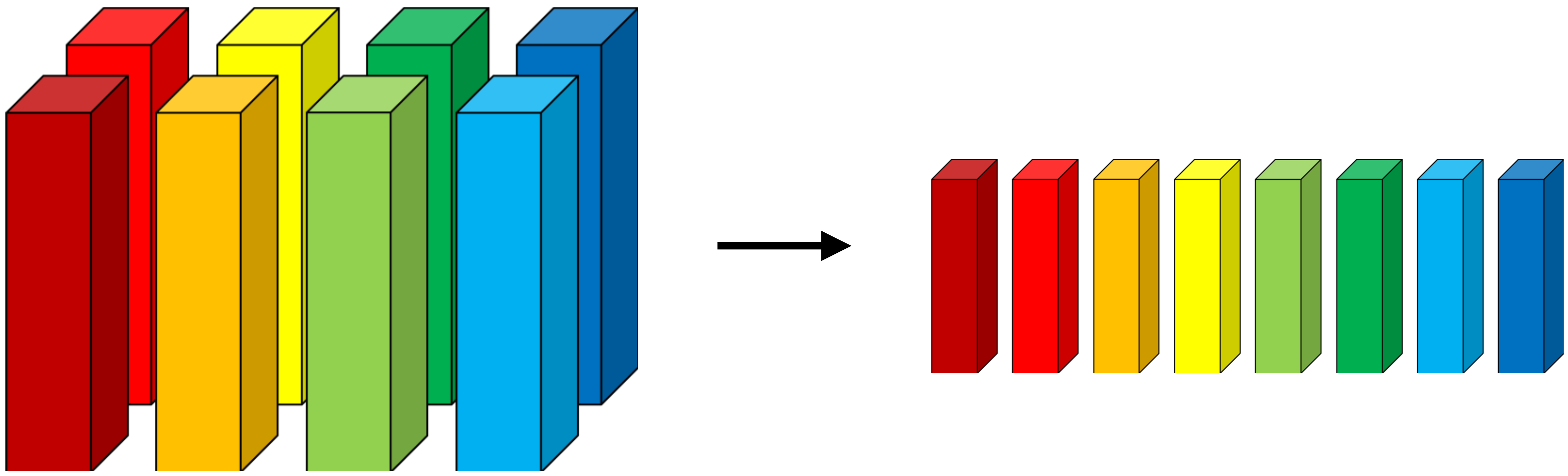
$$M = \prod_{\substack{k=0,\\ k\neq n}}^{N} I_k$$

- Maps element $(i_0, i_1, \cdots, i_N)$ of $\hat{\mathcal{X}}$ to element j of $\hat{\mathcal{X}}_{[n]}$ with

$$j = \sum_{\substack{k=0,\\ k\neq n}}^{N} i_k \times \prod_{\substack{m=k+1,\\ m\neq n}}^{N} I_m$$

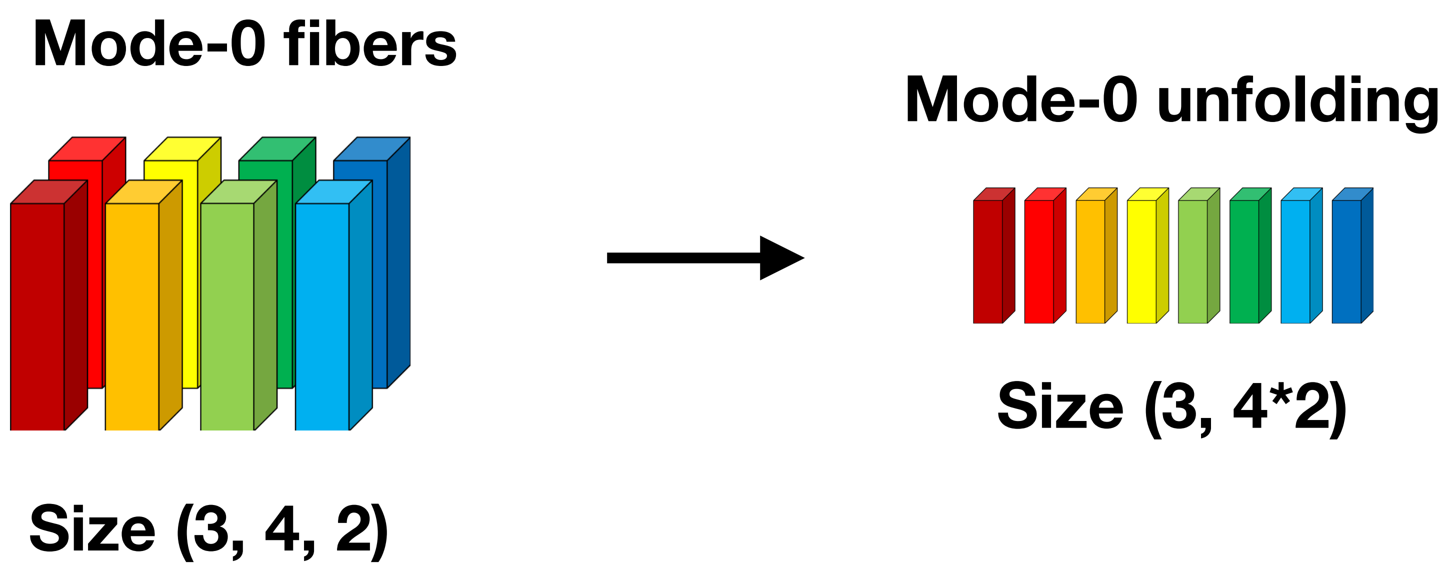
Example: mode-0 unfolding

Mode-0 fibers **Mode-0 unfolding**



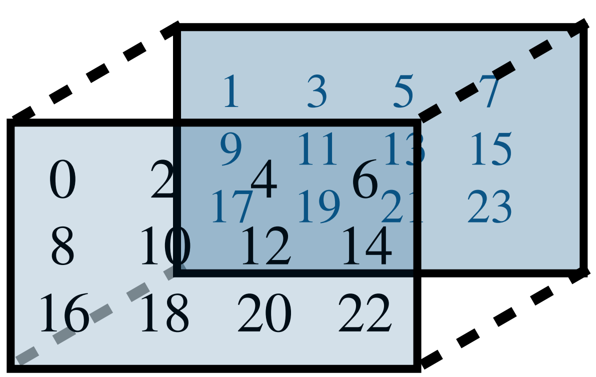
Size (3, 4, 2) **Size (3, 4*2)**

Example: mode-0 unfolding



$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$\hat{\mathscr{X}} =$

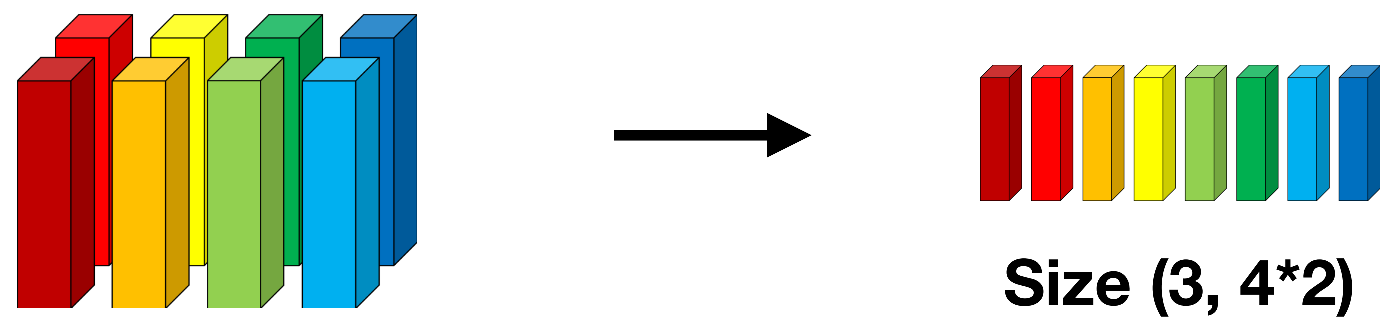


$\longrightarrow \quad \tilde{X}_{[0]} = ?$

Example: mode-0 unfolding

Mode-0 fibers

Mode-0 unfolding

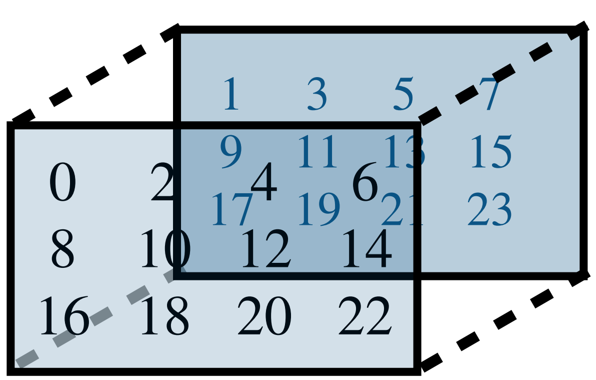


Size (3, 4*2)

Size (3, 4, 2)

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$\hat{\mathscr{X}} =$

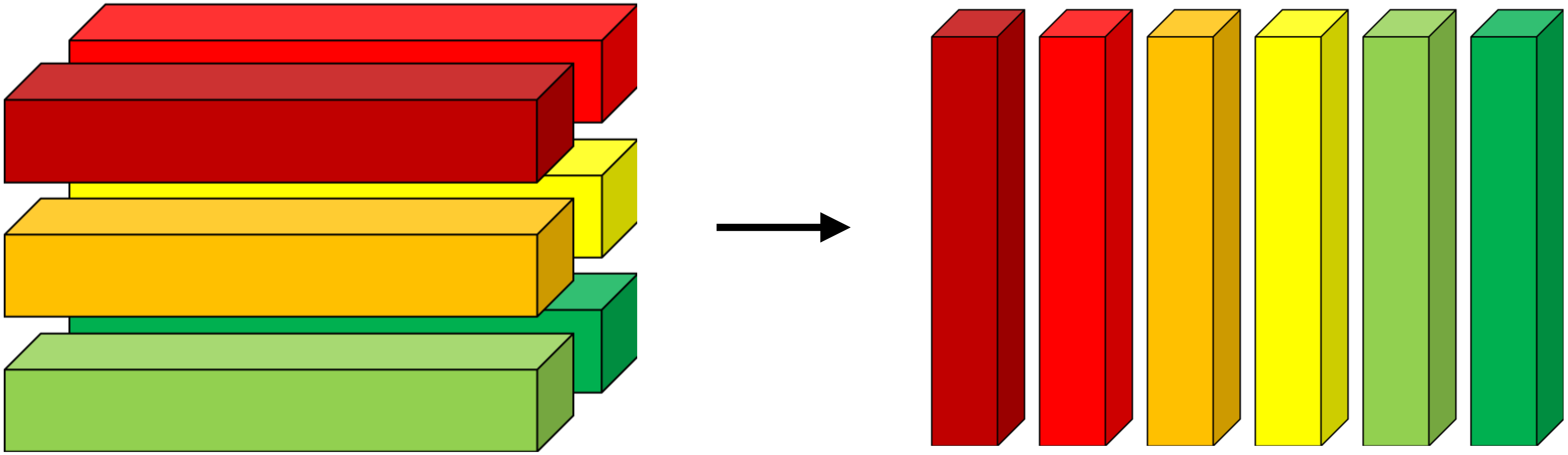


$$\tilde{X}_{[0]} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \end{bmatrix}$$

Example: mode-1 unfolding

Mode-1 fibers

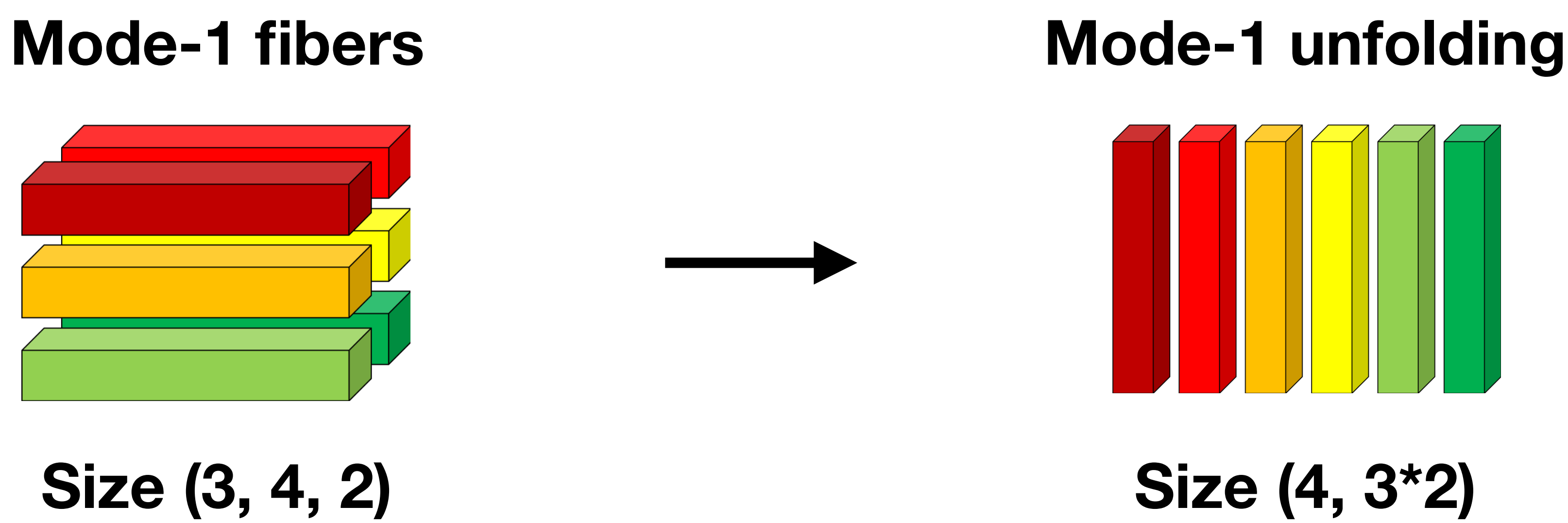
Mode-1 unfolding



Size (3, 4, 2)

Size (4, 3*2)

Example: mode-1 unfolding

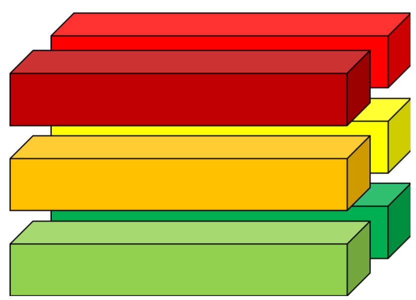


$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$\tilde{X}_{[1]} = ?$$

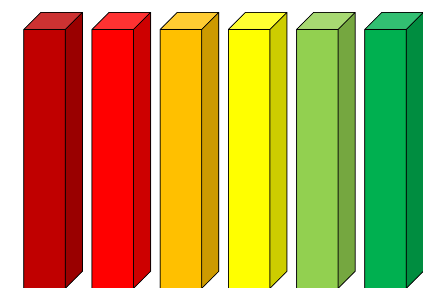
Example: mode-1 unfolding

Mode-1 fibers



Size (3, 4, 2)

Mode-1 unfolding



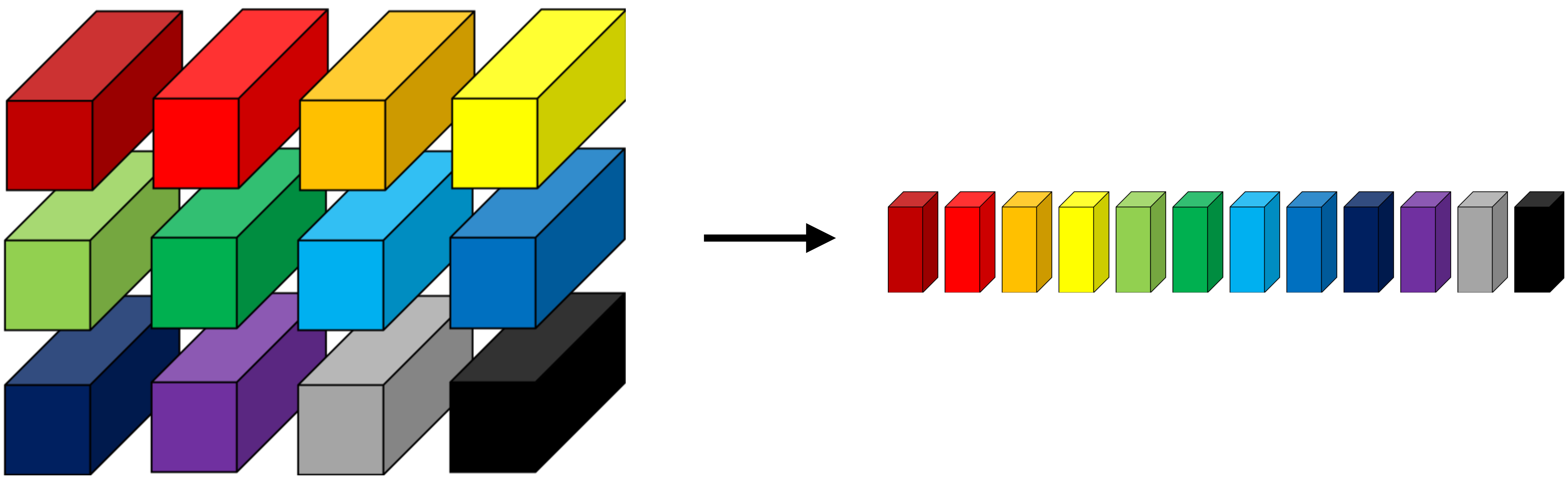
Size (4, 3*2)

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$\tilde{X}_{[1]} = \begin{bmatrix} 0 & 1 & 8 & 9 & 16 & 17 \\ 2 & 3 & 10 & 11 & 18 & 19 \\ 4 & 5 & 12 & 13 & 20 & 21 \\ 6 & 7 & 14 & 15 & 22 & 23 \end{bmatrix}$$

Example: mode-2 unfolding

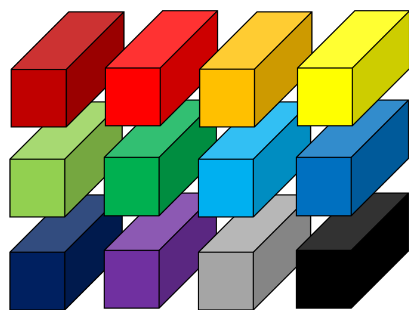
Mode-2 fibers **Mode-2 unfolding**



Size (3, 4, 2) **Size (2, 3*4)**

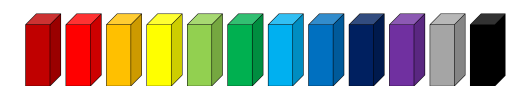
Example: mode-2 unfolding

Mode-2 fibers



Size (3, 4, 2)

Mode-2 unfolding



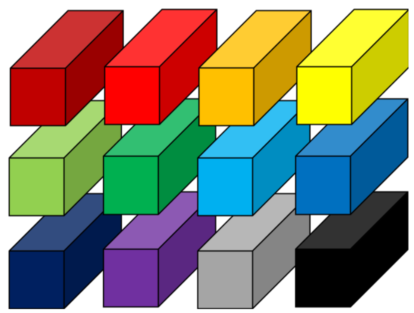
Size (2, 3*4)

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$\tilde{X}_{[2]} = ?$$

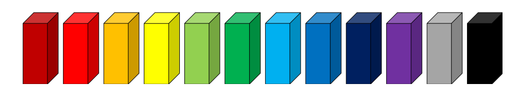
Example: mode-2 unfolding

Mode-2 fibers



Size (3, 4, 2)

Mode-2 unfolding



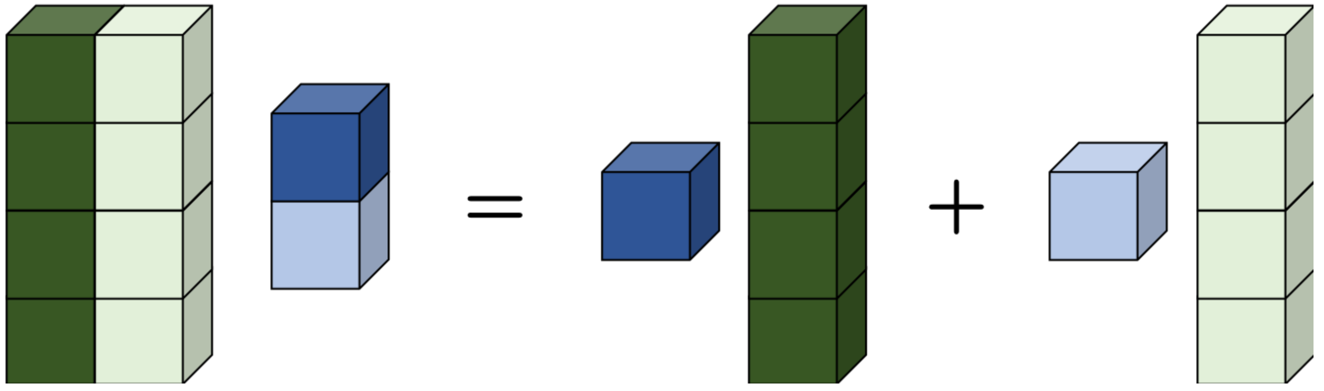
Size (2, 3*4)

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$\tilde{X}_{[2]} = \begin{bmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 20 & 22 \\ 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \end{bmatrix}$$

Tensor contraction: n-mode product

- Natural generalisation of matrix-vector and matrix-matrix product

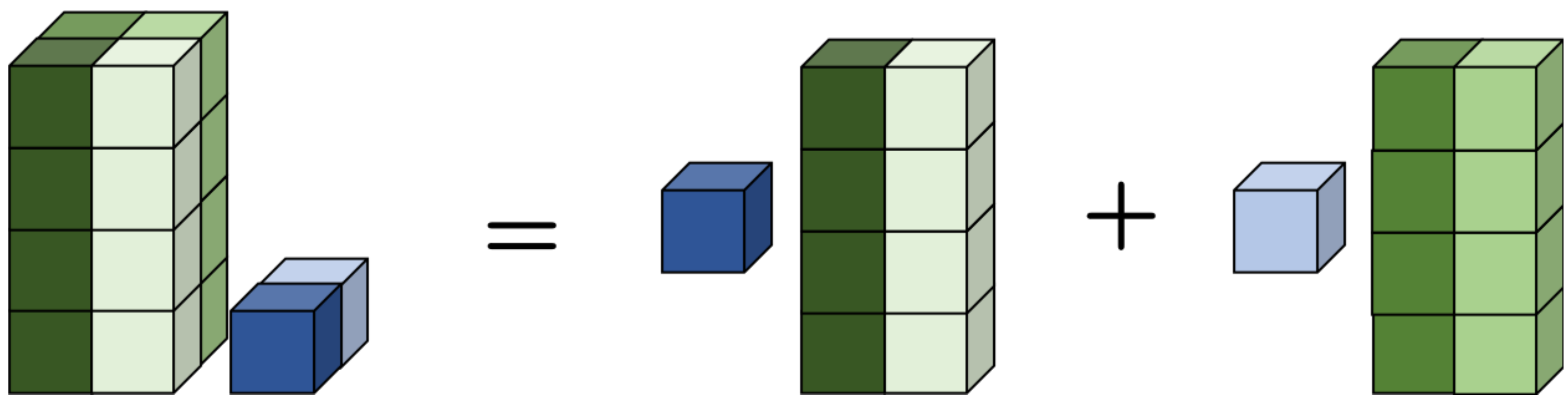


$$\mathbf{M}\mathbf{u} = \sum_k u_k \mathbf{M}_{:,k}$$

Tensor contraction: n-mode product

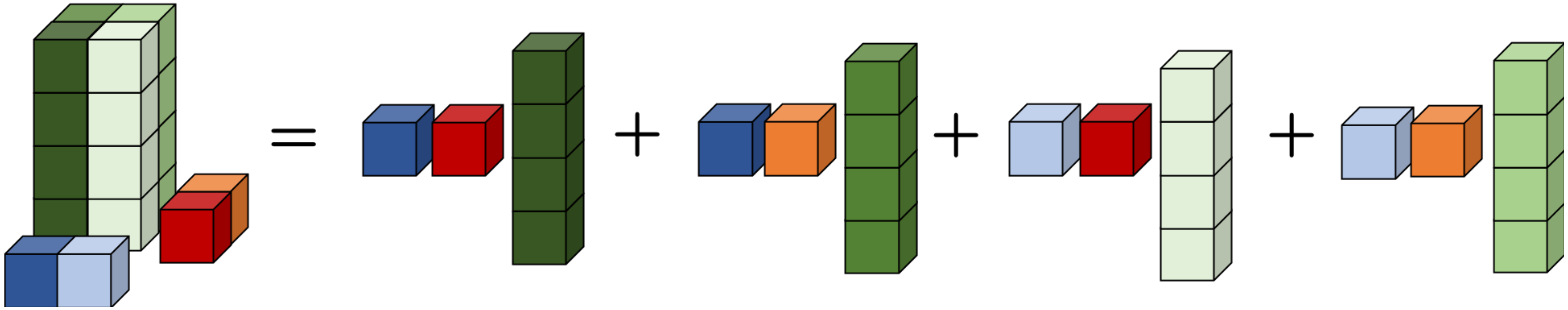
- Natural generalisation of matrix-vector and matrix-matrix product
- When multiplying a tensor by a matrix or a vector, we now have to specify the mode **n** along which to take the product: n-mode product
- E.g $\hat{\mathscr{X}} \times_1 \mathbf{u}$

Tensor contraction: n-mode product



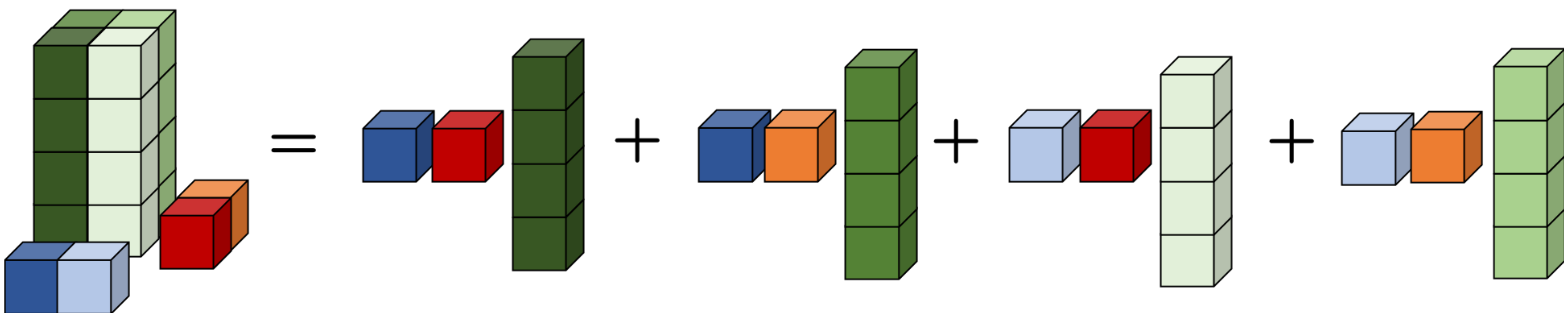
$$\hat{\mathscr{X}} \times_1 \mathbf{u} = \sum_k u_k \hat{\mathscr{X}}_{:,k,:}$$

Tensor contraction: n-mode product



$$\hat{\mathscr{X}} \times_1 \mathbf{u} \times_2 \mathbf{v} = \sum_{i,j} u_i v_j \hat{\mathscr{X}}_{:,i,j}$$

Tensor contraction: n-mode product



$$\hat{\mathscr{X}} \times_1 \mathbf{u} \times_2 \mathbf{v} = \sum_{i,j} u_i v_j \hat{\mathscr{X}}_{:,i,j}$$

- Alternative notation: $\hat{\mathscr{X}} \times_1 \mathbf{u} \times_2 \mathbf{v} = \hat{\mathscr{X}} \times_0 \mathbf{I} \times_1 \mathbf{u} \times_2 \mathbf{v} = \hat{\mathscr{X}}\,(\mathbf{I}, \mathbf{u}, \mathbf{v})$

N-mode product: Useful properties

- N-mode product can be with vectors or matrices

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \sum_i M_{:,i} \hat{\mathscr{X}}_{:,i,:}$$

N-mode product: Useful properties

- N-mode product can be with vectors or matrices

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \sum_i M_{:,i} \hat{\mathscr{X}}_{:,i,:}$$

- Equivalent formulation using unfolding:

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \mathbf{M} \hat{\mathscr{X}}_{[1]}$$

N-mode product: Useful properties

- N-mode product can be with vectors or matrices

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \sum_i M_{:,i} \hat{\mathscr{X}}_{:,i,:}$$

- Equivalent formulation using unfolding:

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \mathbf{M}\hat{\mathscr{X}}_{[1]}$$

- Unfolding on mode-product on all modes:

$$\left(\hat{\mathscr{X}} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_N \mathbf{U}^{(N)}\right)_{[n]} = \mathbf{U}^{(n)} \hat{\mathscr{X}}_{[n]} \left(\mathbf{U}^{(0)} \otimes \cdots \mathbf{U}^{(n-1)} \otimes \mathbf{U}^{(n+1)} \otimes \cdots \otimes \mathbf{U}^{(N)}\right)^{\top}$$

N-mode product: Useful properties

- N-mode product can be with vectors or matrices

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \sum_i M_{:,i} \hat{\mathscr{X}}_{:,i,:}$$

- Equivalent formulation using unfolding:

$$\hat{\mathscr{X}} \times_1 \mathbf{M} = \mathbf{M} \hat{\mathscr{X}}_{[1]}$$

- Unfolding on mode-product on all modes:

$$\left(\hat{\mathscr{X}} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_N \mathbf{U}^{(N)} \right)_{[n]} = \mathbf{U}^{(n)} \hat{\mathscr{X}}_{[n]} \left(\mathbf{U}^{(0)} \otimes \cdots \mathbf{U}^{(n-1)} \otimes \mathbf{U}^{(n+1)} \otimes \cdots \otimes \mathbf{U}^{(N)} \right)^\top$$

- Equivalent formulation using vec:

$$vec(\hat{\mathscr{X}} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_N \mathbf{U}^{(N)}) = \left(\mathbf{U}^{(0)} \otimes \cdots \otimes \mathbf{U}^{(N)} \right) vec(\hat{\mathscr{X}})$$

Tensor diagrams

- Explicitly writing tensor contraction can be (very) cumbersome and hard to read..

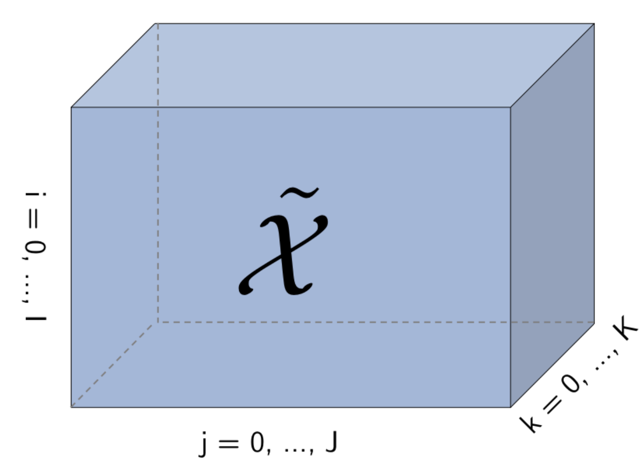
$$\hat{\mathscr{T}}_{\alpha,\beta,\gamma,\delta} = \underbrace{\sum_{i,j,k,l,m,n} \hat{\mathscr{X}}_{i,j,\alpha,\beta,m,n} \hat{\mathscr{Y}}_{i,j,k,l\gamma} \hat{\mathscr{Z}}_{k,l,m,n\delta}}_{!!???}$$

Tensor diagrams

- Explicitly writing tensor contraction can be (very) cumbersome and hard to read..

$$\hat{\mathscr{T}}_{\alpha,\beta,\gamma,\delta} = \underbrace{\sum_{i,j,k,l,m,n} \hat{\mathscr{X}}_{i,j,\alpha,\beta,m,n} \hat{\mathscr{Y}}_{i,j,k,l\gamma} \hat{\mathscr{Z}}_{k,l,m,n\delta}}_{!!???}$$

- Hard to represent higher order tensors

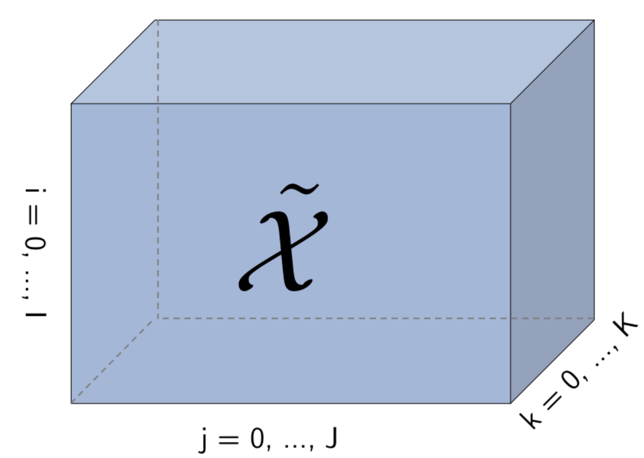


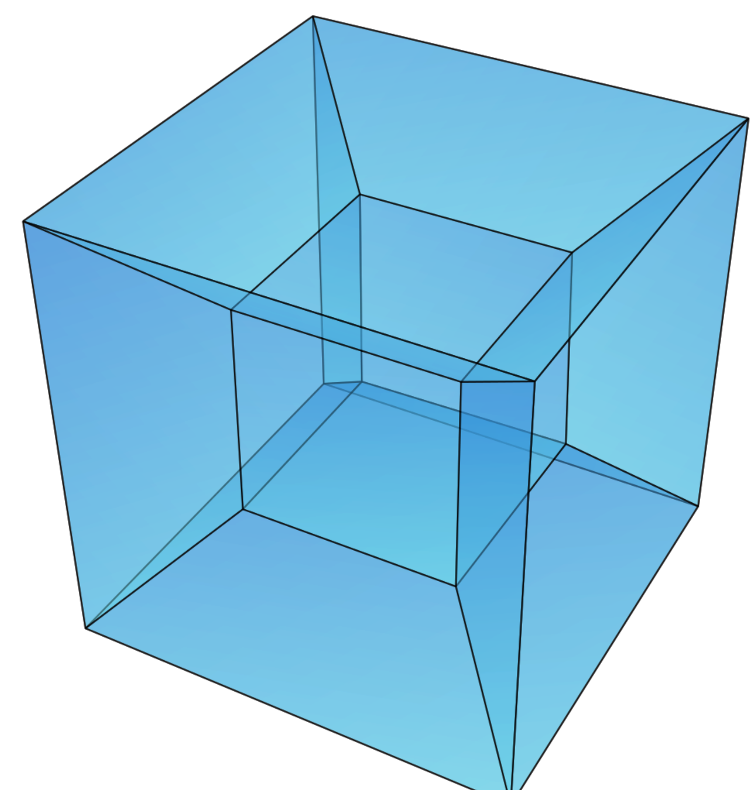
Tensor diagrams

- Explicitly writing tensor contraction can be (very) cumbersome and hard to read..

$$\hat{\mathscr{T}}_{\alpha,\beta,\gamma,\delta} = \underbrace{\sum_{i,j,k,l,m,n} \hat{\mathscr{X}}_{i,j,\alpha,\beta,m,n}\hat{\mathscr{Y}}_{i,j,k,l\gamma}\hat{\mathscr{Z}}_{k,l,m,n\delta}}_{!!???}$$

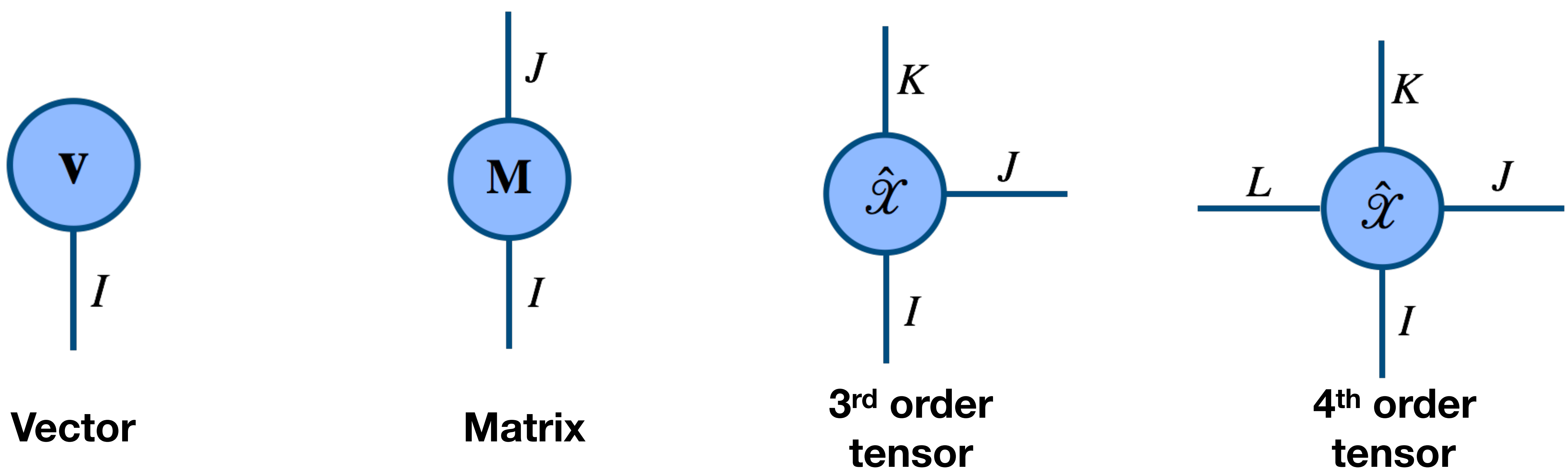
- Hard to represent higher order tensors





Tensor diagrams

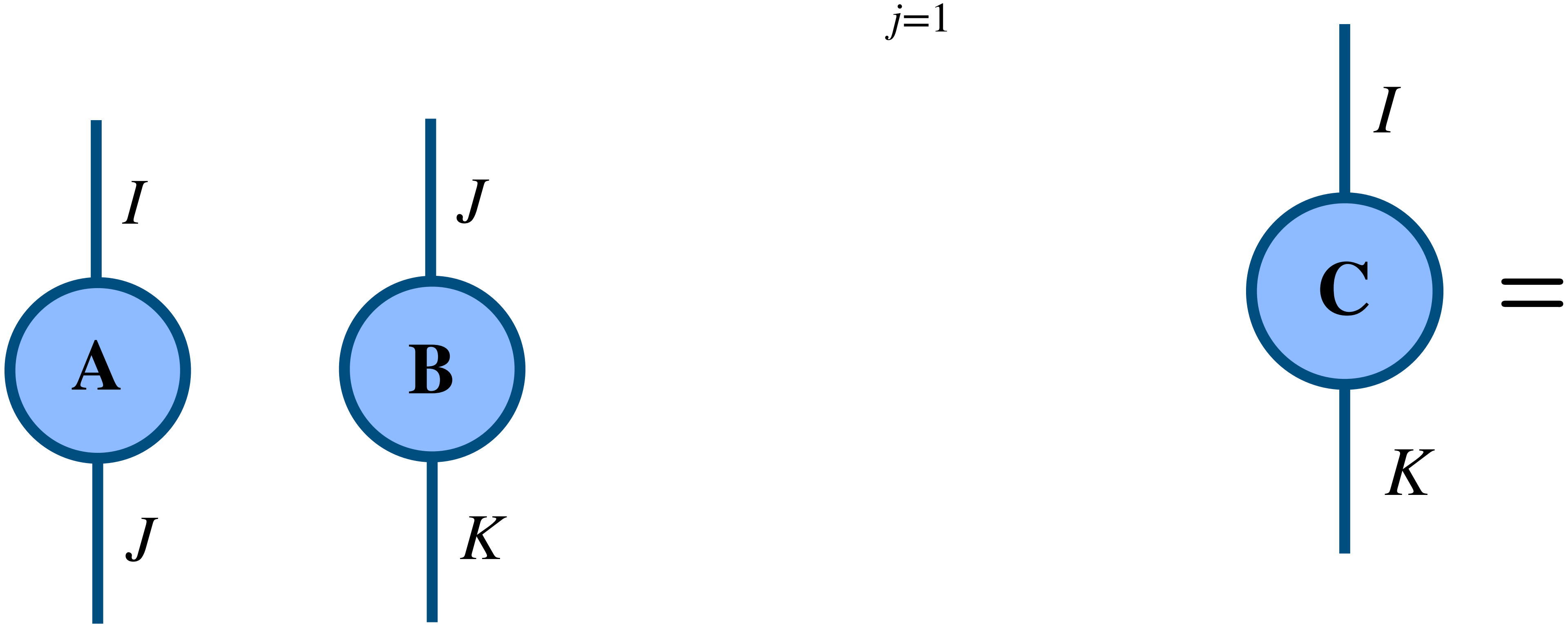
- Represent only variables and indices (dimensions)
- Tensors = vertices, mode = edge, order = degree



Tensor diagrams

- Contraction on a given dimension: simply link the indices over which to contract together!

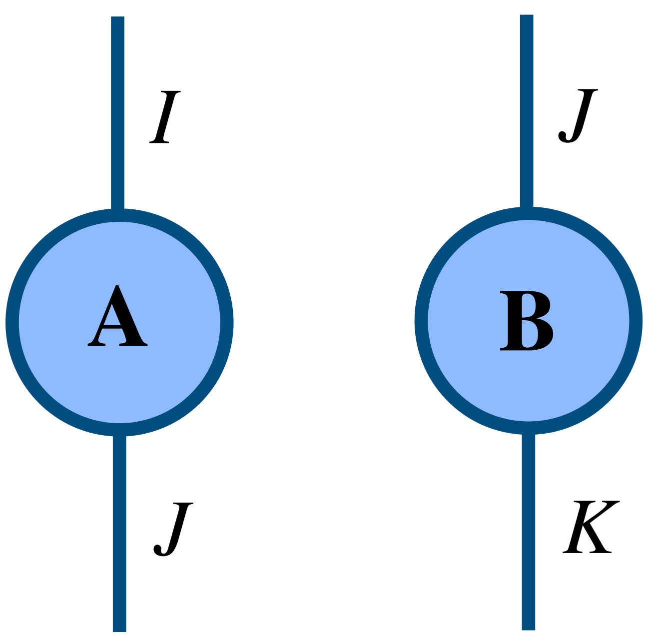
$$\mathbf{C} = \mathbf{A}\mathbf{B} = \sum_{j=1}^{J} \mathbf{a}_{:,j} \mathbf{b}_{j,:}^{\top}$$

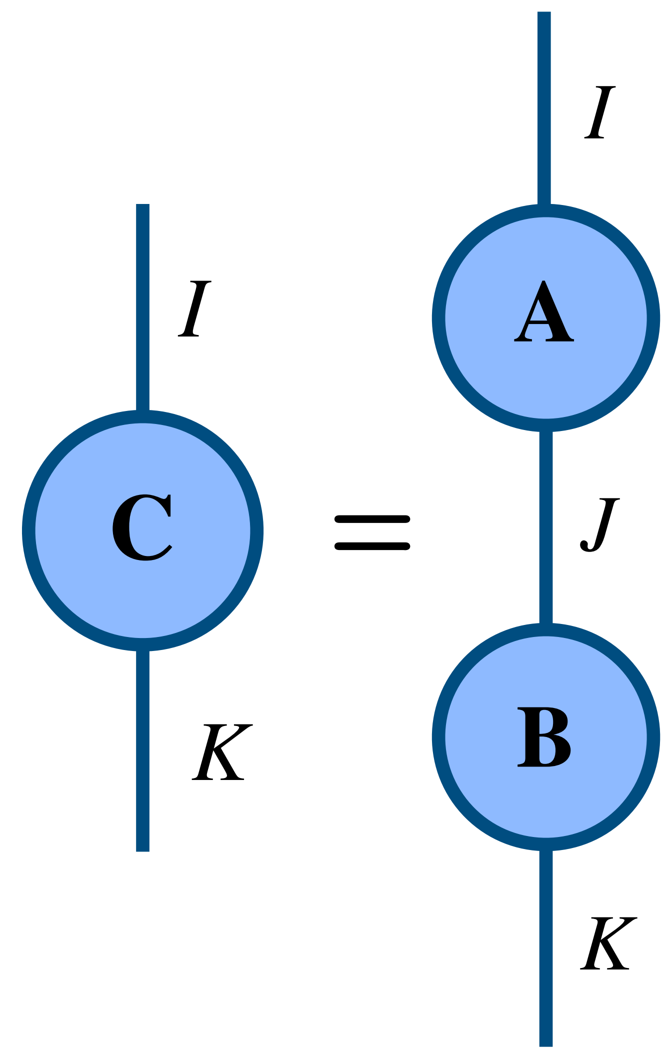


Tensor diagrams

- Contraction on a given dimension: simply link the indices over which to contract together!

$$\mathbf{C} = \mathbf{A}\mathbf{B} = \sum_{j=1}^{J} \mathbf{a}_{:,j}\mathbf{b}_{j,:}^{\top}$$

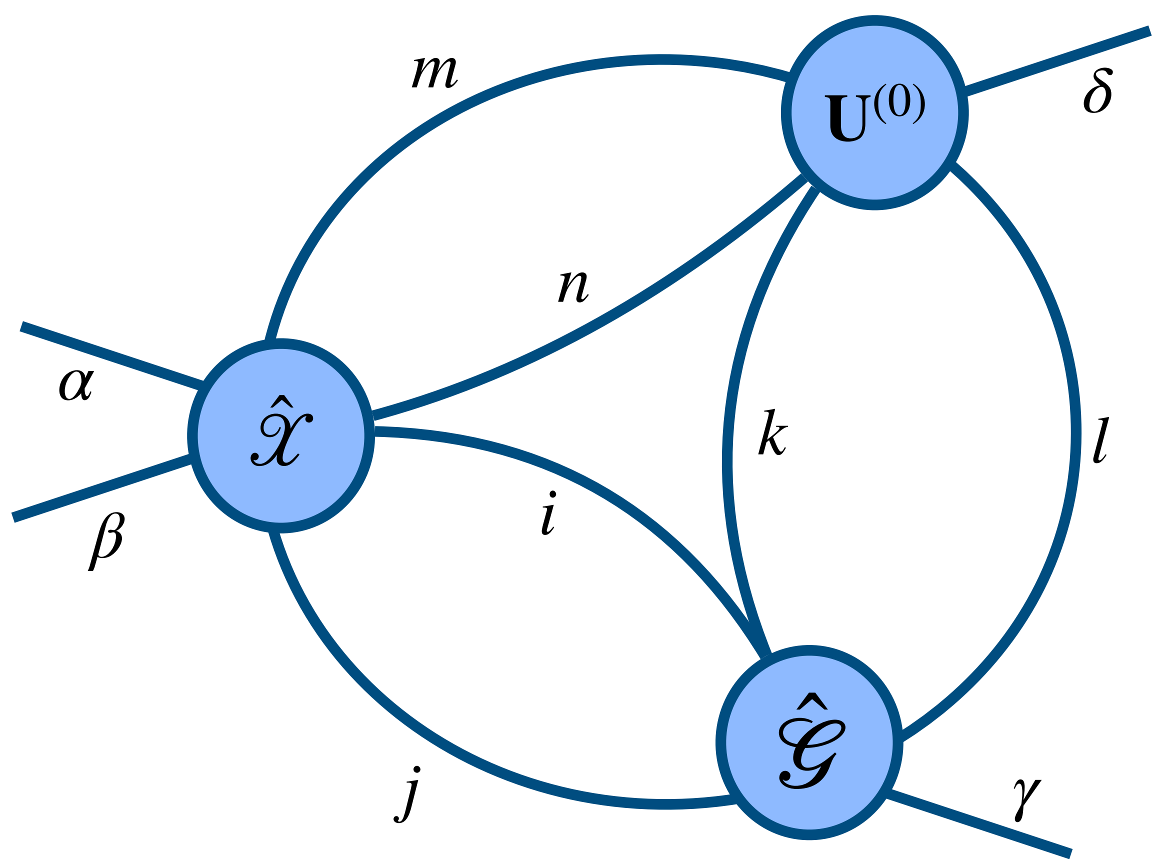




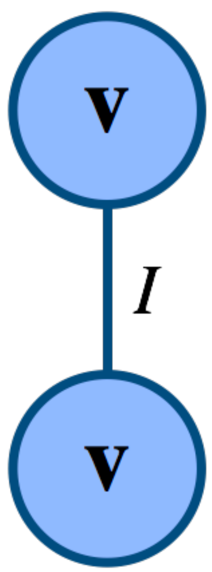
Tensor diagrams

- Contraction on a given dimension: simply link the indices over which to contract together!

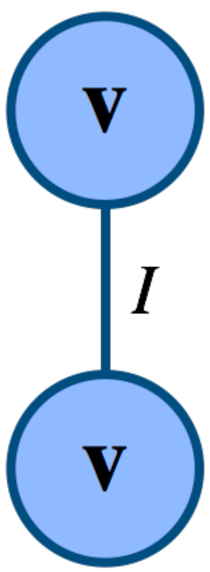
$$\hat{\mathcal{T}}_{\alpha,\beta,\gamma,\delta} = \underbrace{\sum_{i,j,k,l,m,n} \hat{\mathcal{X}}_{i,j,\alpha,\beta,m,n} \hat{\mathcal{Y}}_{i,j,k,l\gamma} \hat{\mathcal{Z}}_{k,l,m,n\delta}}$$



Tensor diagrams



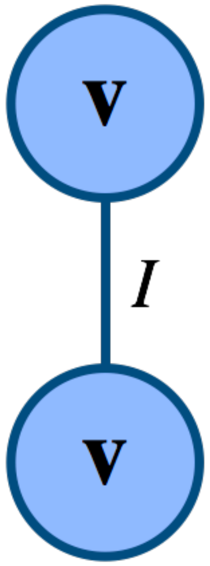
Tensor diagrams



Inner-product

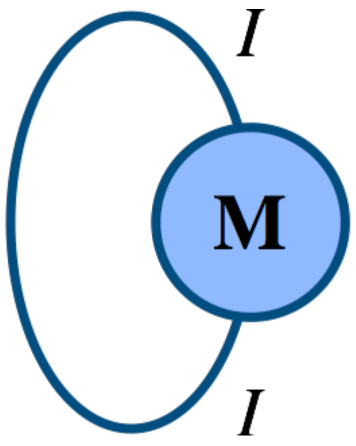
$$\sum_{i=0}^{I-1} v_i \times v_i = \sum_i v_i^2$$

Tensor diagrams

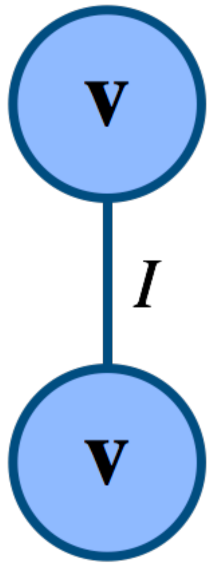


Inner-product

$$\sum_{i=0}^{I-1} v_i \times v_i = \sum_i v_i^2$$

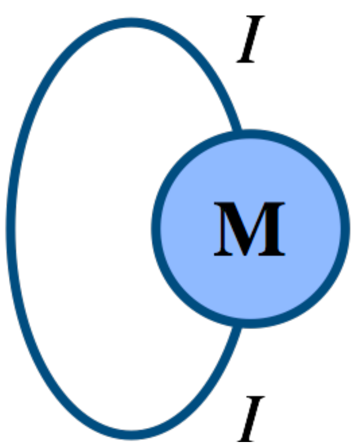


Tensor diagrams



Inner-product

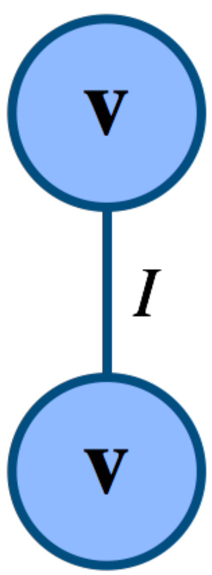
$$\sum_{i=0}^{I-1} v_i \times v_i = \sum_i v_i^2$$

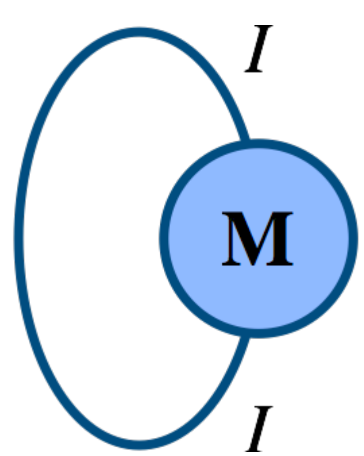


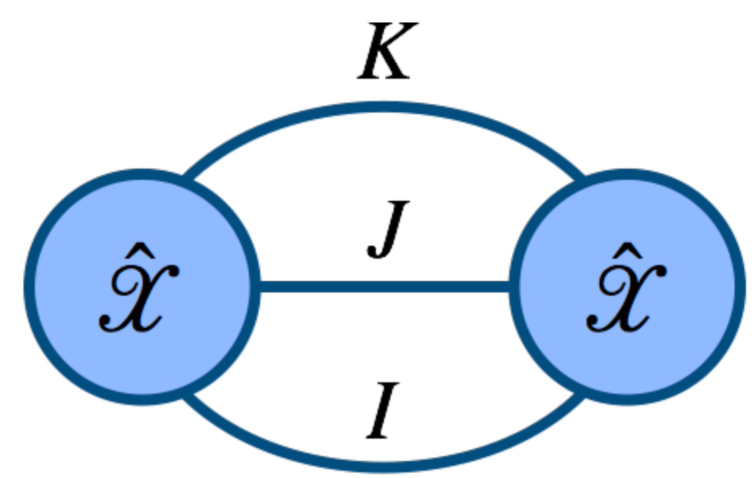
Matrix-trace

$$\sum_{i=0}^{I-1} M_{ii}$$

Tensor diagrams







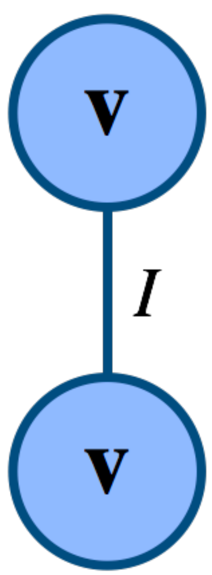
Inner-product

$$\sum_{i=0}^{I-1} v_i \times v_i = \sum_i v_i^2$$

Matrix-trace

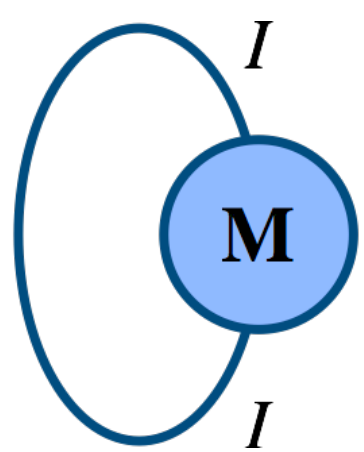
$$\sum_{i=0}^{I-1} M_{ii}$$

Tensor diagrams



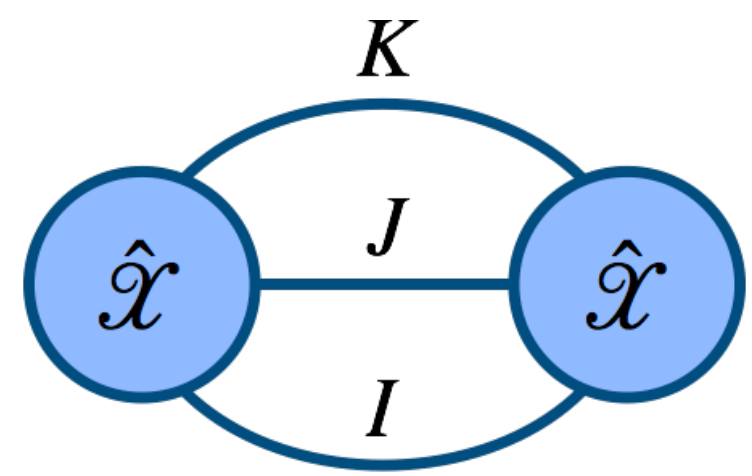
Inner-product

$$\sum_{i=0}^{I-1} v_i \times v_i = \sum_i v_i^2$$



Matrix-trace

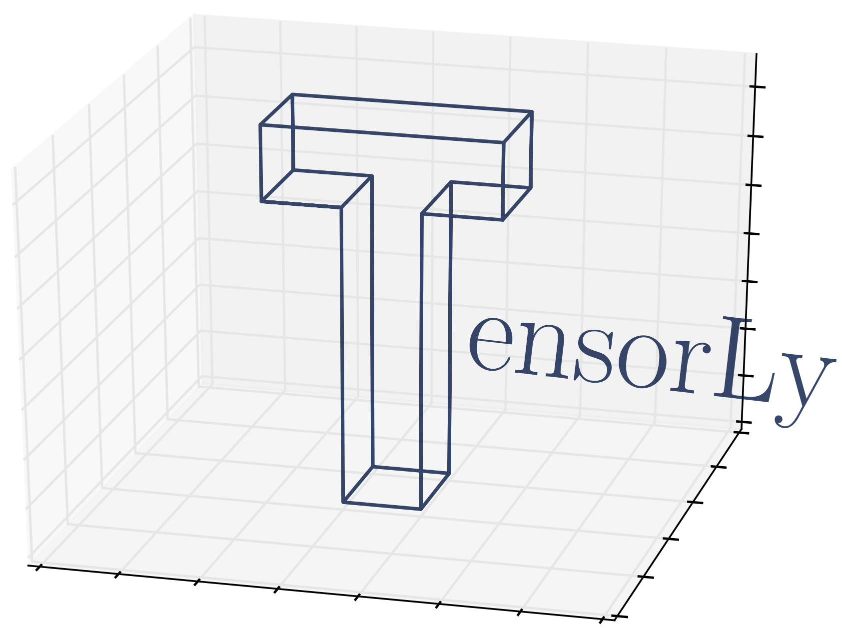
$$\sum_{i=0}^{I-1} M_{ii}$$



Inner-product

$$\sum_{i,j,k} \hat{\mathscr{X}}^2_{i,j,k}$$

Tensor Methods in Python with TensorLy



https://tensorly.org/dev/
https://github.com/tensorly/tensorly

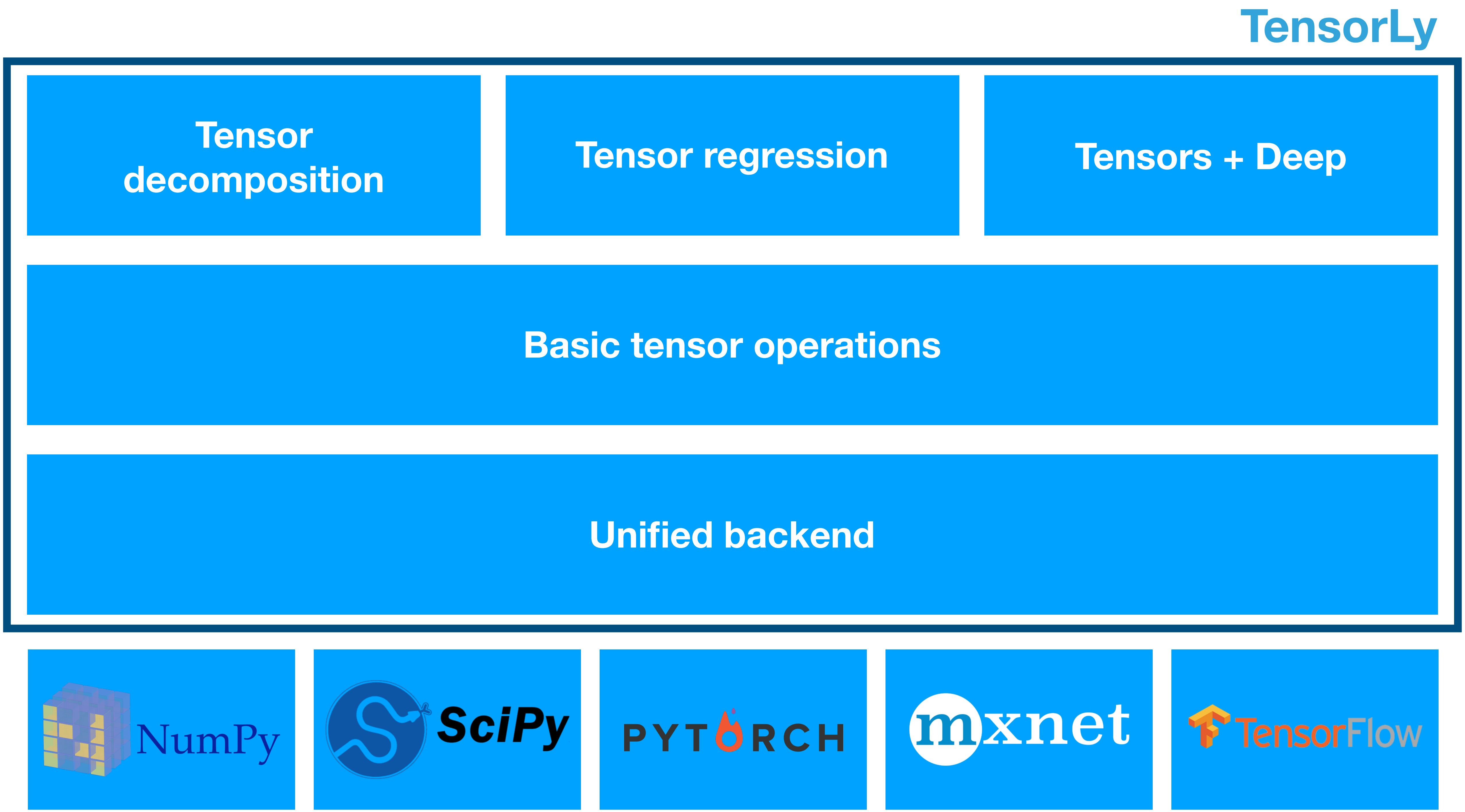
What is TensorLy?

- High-level API for tensor methods and deep tensorized neural networks in Python
- Backend system allows users to perform computations with NumPy, MXNet, PyTorch, TensorFlow and CuPy
- Operations and algorithms can be scaled on multiple CPU or GPU machines

Open-Source

- Open source, on Github
  - https://github.com/tensorly/tensorly
- BSD licensed:
  -> suitable for academic / industrial applications
- Minimal dependencies
- Contributions welcome!

User-Friendly API



Any questions?

@JeanKossaifi
jean.kossaifi@gmail.com

Demonstration: notebook

- Unify the syntax by abstracting away the backend
- tl.set_backend('numpy') # or 'mxnet', 'pytorch', 'cupy'…

```
import tensorly as tl

T = tl.tensor([[1, 2, 3], [4, 5, 6]])    ← NumPy ndarray
tl.tenalg.kronecker([T, T])
tl.clip(T, a_min=2, a_max=5)

tl.set_backend('mxnet')
T = tl.tensor([[1, 2, 3], [4, 5, 6]])    ← MXNet NDArray

tl.set_backend('pytorch')
T = tl.tensor([[1, 2, 3], [4, 5, 6]])    ← PyTorch FloatTensor
```