Learning with Structured Tensor Decompositions

Anand D. Sarwate, Rutgers University

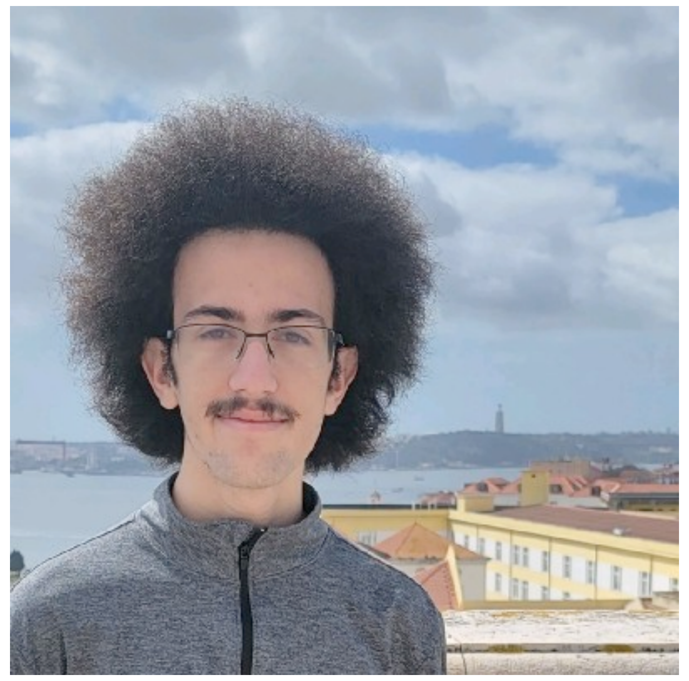
18 July 2024

Center for Advanced Mathematical Sciences
American University of Beirut

Waheed U. Bajwa

Batoul Taki

Zahra Shakeri



Jose Hoyos Sanchez

Mohsen Ghassemi

Tensors in the real world

The history of the word “tensor”

Let’s meet some 19th century physicists

All images: Wikipedia

The history of the word “tensor”

Let’s meet some 19th century physicists

- 1848: William Rowan Hamilton used the word “tensor” to mean the absolute value (norm) of a quaternion. His “tensor” is actually a scalar (!)

All images: Wikipedia

The history of the word “tensor”

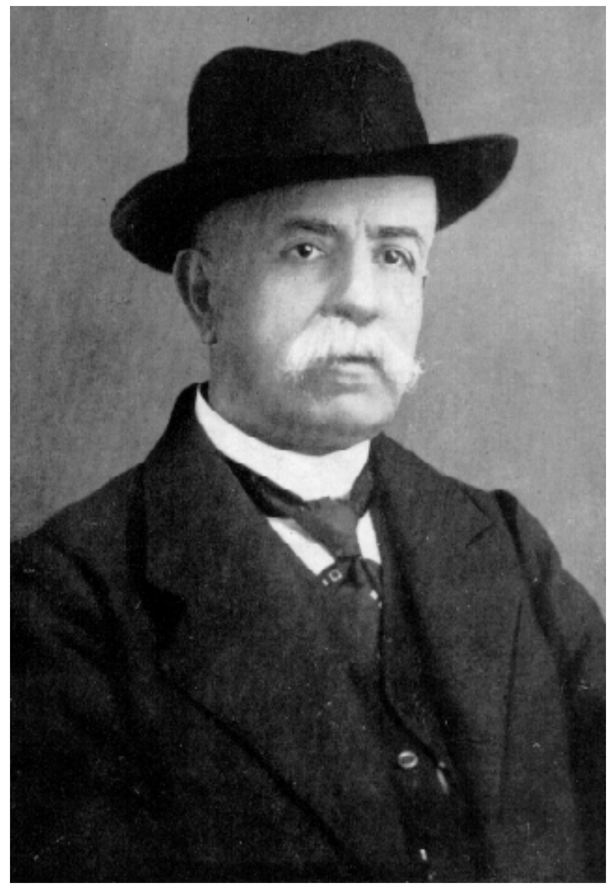
Let’s meet some 19th century physicists

- 1848: William Rowan Hamilton used the word “tensor” to mean the absolute value (norm) of a quaternion. His “tensor” is actually a scalar (!)
- 1898: Woldemar Voigt used “tensor” in his paper *Die fundamentalen physikalischen Eigenschaften der Krystalle in elementarer Darstellung*

All images: Wikipedia

The history of the word “tensor”

Let’s meet some 19th century physicists

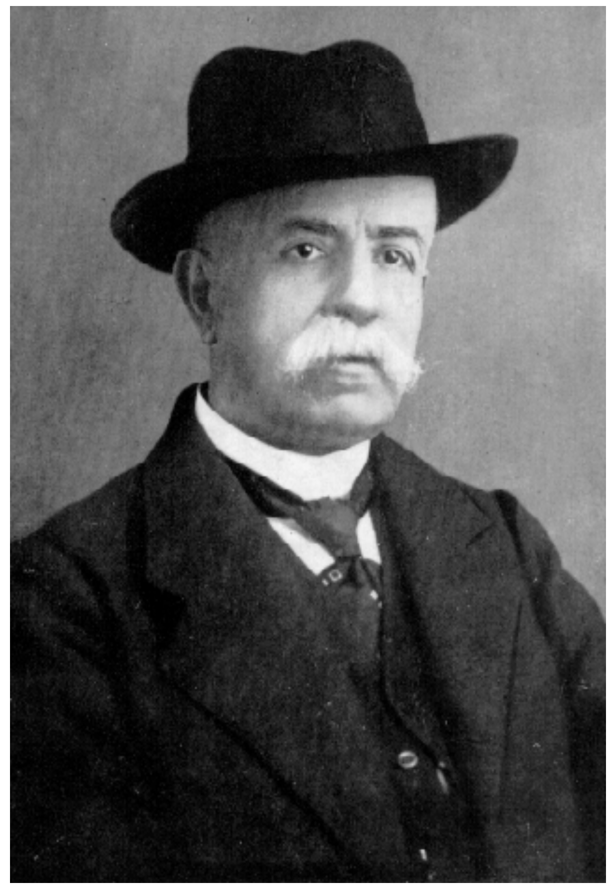


All images: Wikipedia

- 1848: William Rowan Hamilton used the word “tensor” to mean the absolute value (norm) of a quaternion. His “tensor” is actually a scalar (!)
- 1898: Woldemar Voigt used “tensor” in his paper *Die fundamentalen physikalischen Eigenschaften der Krystalle in elementarer Darstellung*
- 1892: Gregorio Ricci-Curbastro developed the theory of tensors. In 1900 he and his student Tullio Levi-Civita write a book on it called *Méthodes de calcul différentiel absolu et leurs applications*

The history of the word “tensor”

Let’s meet some 19th century physicists



All images: Wikipedia

- 1848: William Rowan Hamilton used the word “tensor” to mean the absolute value (norm) of a quaternion. His “tensor” is actually a scalar (!)
- 1898: Woldemar Voigt used “tensor” in his paper *Die fundamentalen physikalischen Eigenschaften der Krystalle in elementarer Darstellung*
- 1892: Gregorio Ricci-Curbastro developed the theory of tensors. In 1900 he and his student Tullio Levi-Civita write a book on it called *Méthodes de calcul différentiel absolu et leurs applications*

From 1900 to the present

A relatively general timeline

All images: Wikipedia

From 1900 to the present

A relatively general timeline

- 1913: Albert Einstein and Marcel Grossman used tensor calculus extensively in their work on general relativity: *Entwurf einer verallgemeinerten Relativitätstheorie und einer Theorie der Gravitation*

All images: Wikipedia

From 1900 to the present

A relatively general timeline

- 1913: Albert Einstein and Marcel Grossman used tensor calculus extensively in their work on general relativity: *Entwurf einer verallgemeinerten Relativitätstheorie und einer Theorie der Gravitation*

All images: Wikipedia

From 1900 to the present

A relatively general timeline

All images: Wikipedia

- 1913: Albert Einstein and Marcel Grossman used tensor calculus extensively in their work on general relativity: *Entwurf einer verallgemeinerten Relativitätstheorie und einer Theorie der Gravitation*
- 1915–17: Levi-Civita and Einstein have a correspondence where the former helped fix the mistakes Einstein made in using tensor analysis.

From 1900 to the present

A relatively general timeline




All images: Wikipedia

- 1913: Albert Einstein and Marcel Grossman used tensor calculus extensively in their work on general relativity: *Entwurf einer verallgemeinerten Relativitätstheorie und einer Theorie der Gravitation*
- 1915–17: Levi-Civita and Einstein have a correspondence where the former helped fix the mistakes Einstein made in using tensor analysis.
- 1922: H. L. Brose's English translation of Weyl's book *Raum, Zeit, Materie* (*Space-Time-Matter*) uses "tensor analysis."

So what is a “tensor” anyway?

Tensors are many different things to many different people

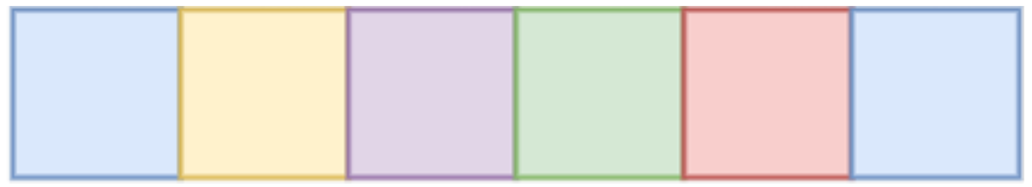
So what is a “tensor” anyway?

Tensors are many different things to many different people

For today, we treat tensors “mechanically” as **multidimensional arrays**.

So what is a “tensor” anyway?

Tensors are many different things to many different people



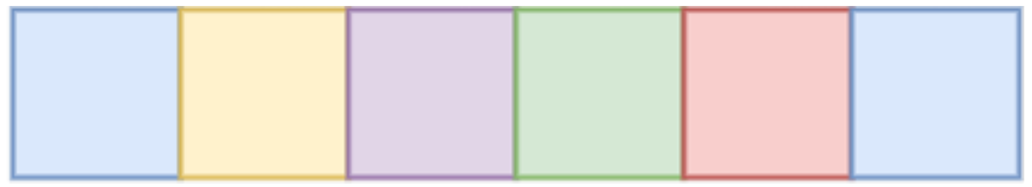
$\mathbf{x} \in \mathrm{R}^m$

First-Order Tensor (Vector)

For today, we treat tensors “mechanically” as **multidimensional arrays**.

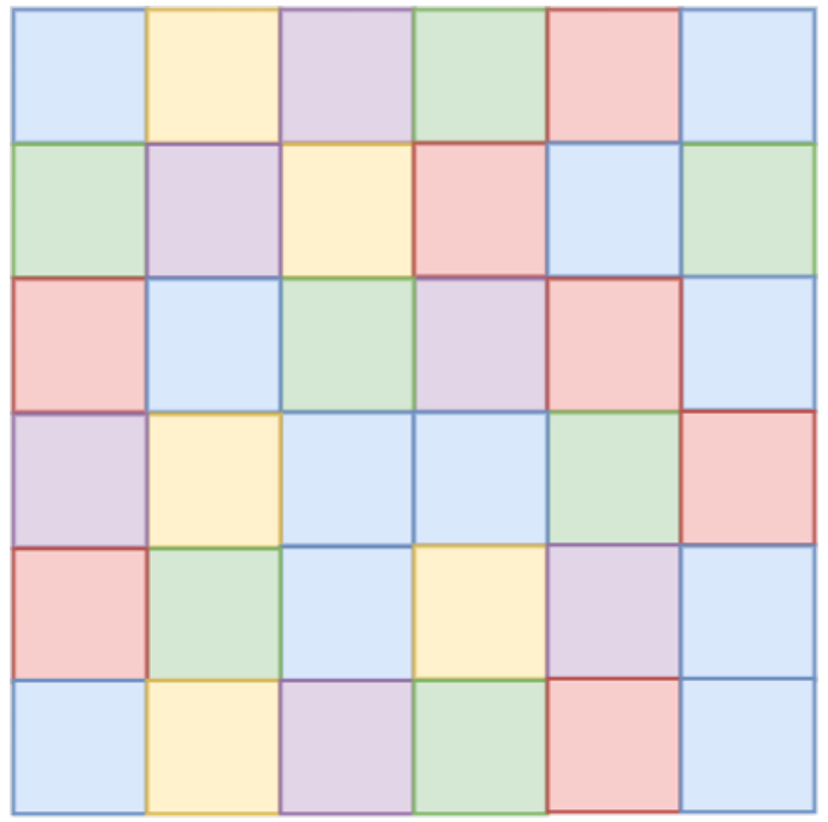
So what is a “tensor” anyway?

Tensors are many different things to many different people



$\mathbf{x} \in \mathrm{R}^m$

First-Order Tensor (Vector)



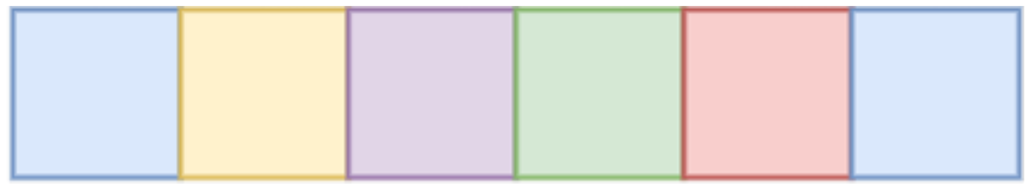
$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)

For today, we treat tensors “mechanically” as **multidimensional arrays**.

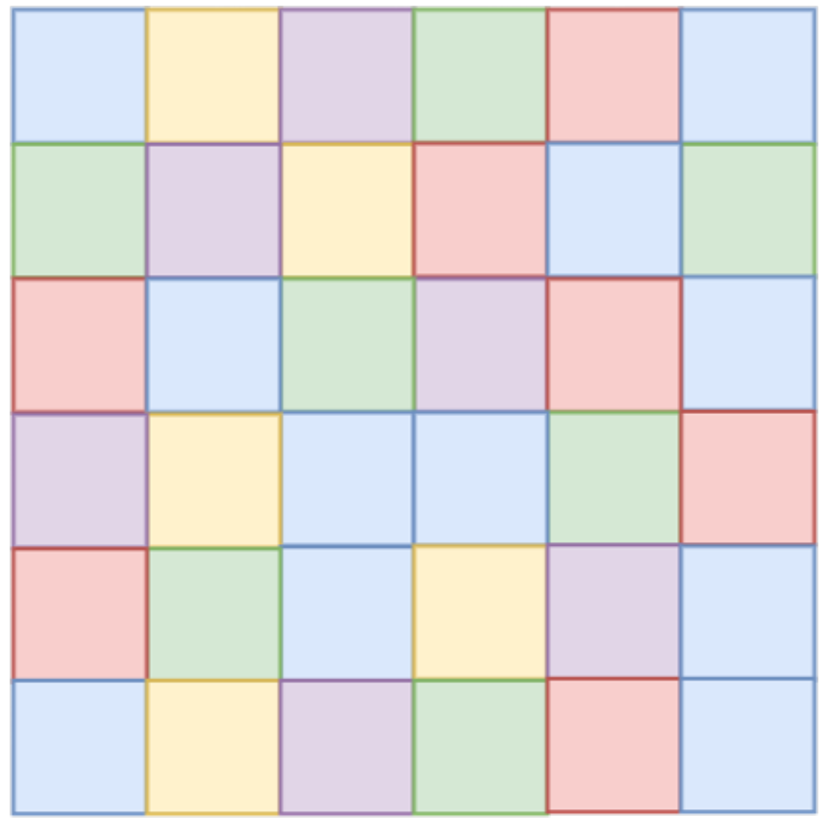
So what is a “tensor” anyway?

Tensors are many different things to many different people



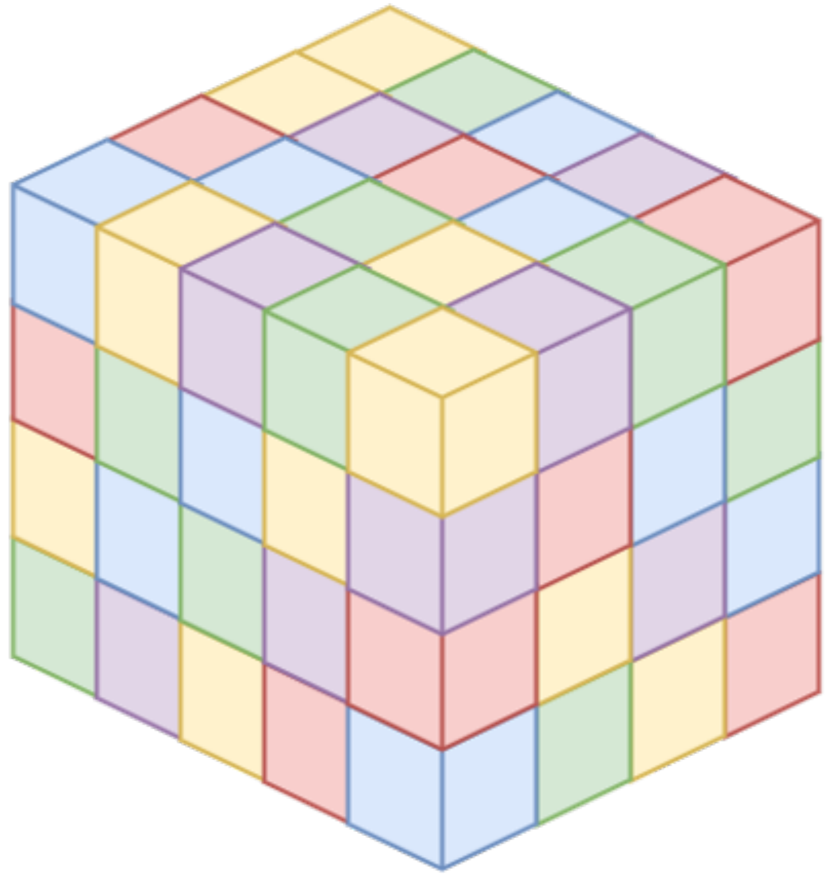
$\mathbf{x} \in \mathrm{R}^m$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



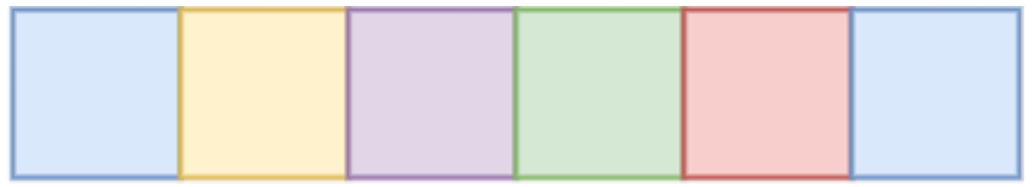
$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

Third-Order Tensor

For today, we treat tensors “mechanically” as **multidimensional arrays**.

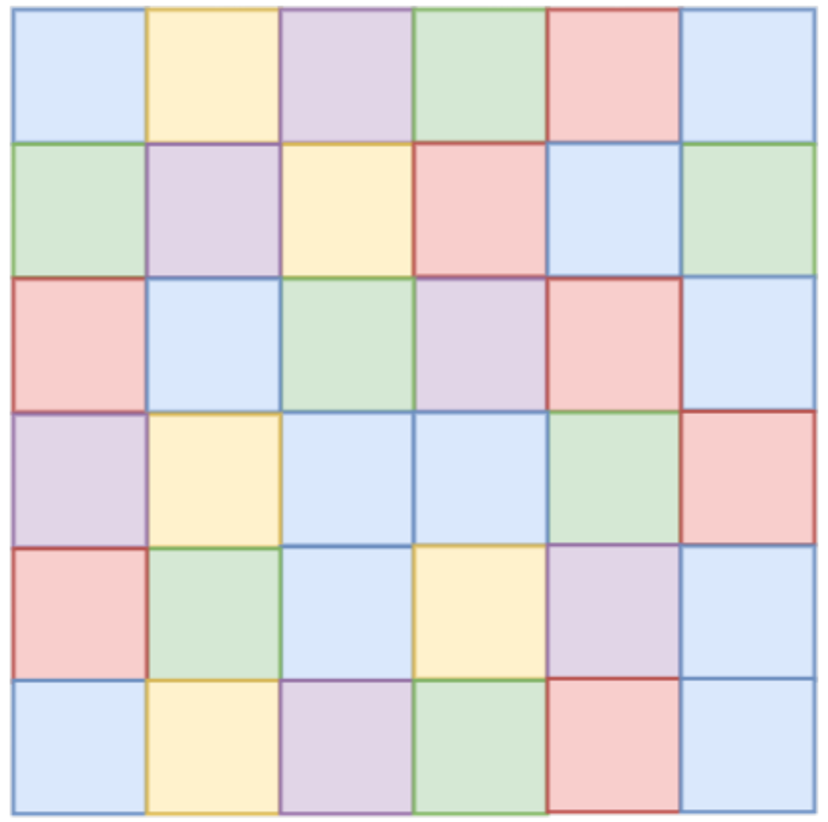
So what is a “tensor” anyway?

Tensors are many different things to many different people



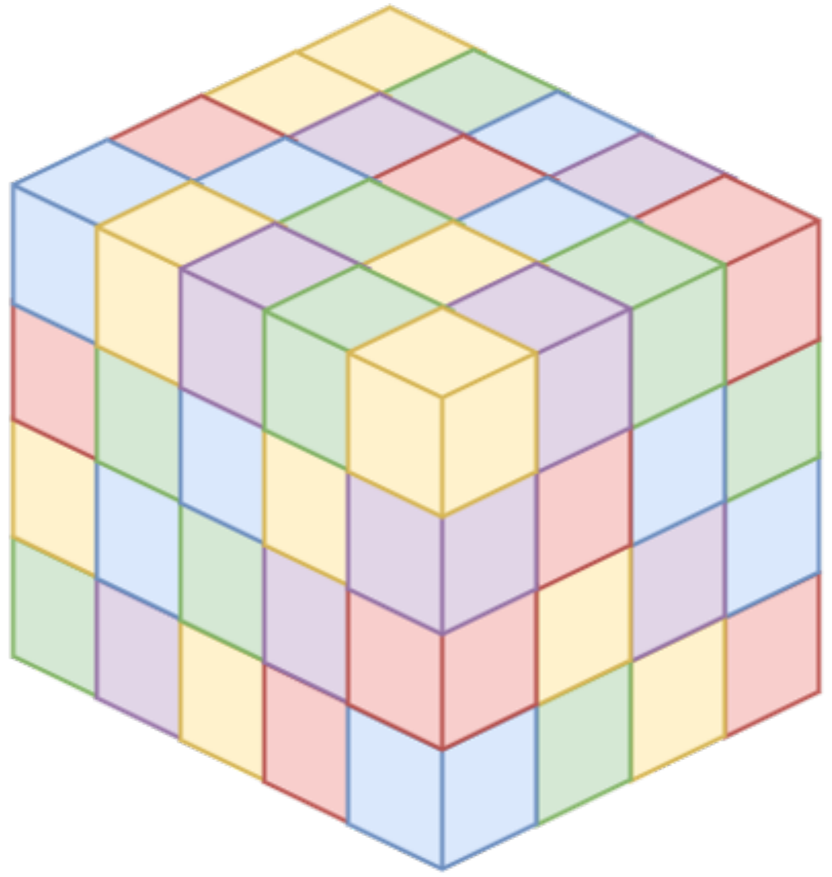
$\mathbf{x} \in \mathrm{R}^{m}$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

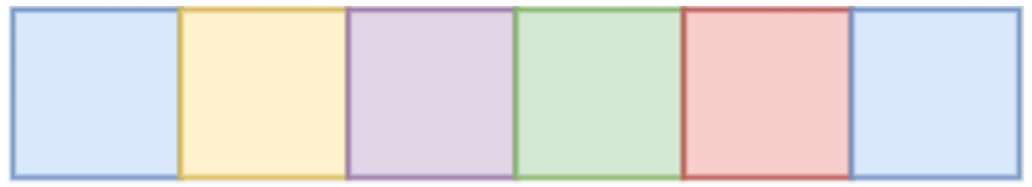
Third-Order Tensor

For today, we treat tensors “mechanically” as **multidimensional arrays**.

$$\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$$

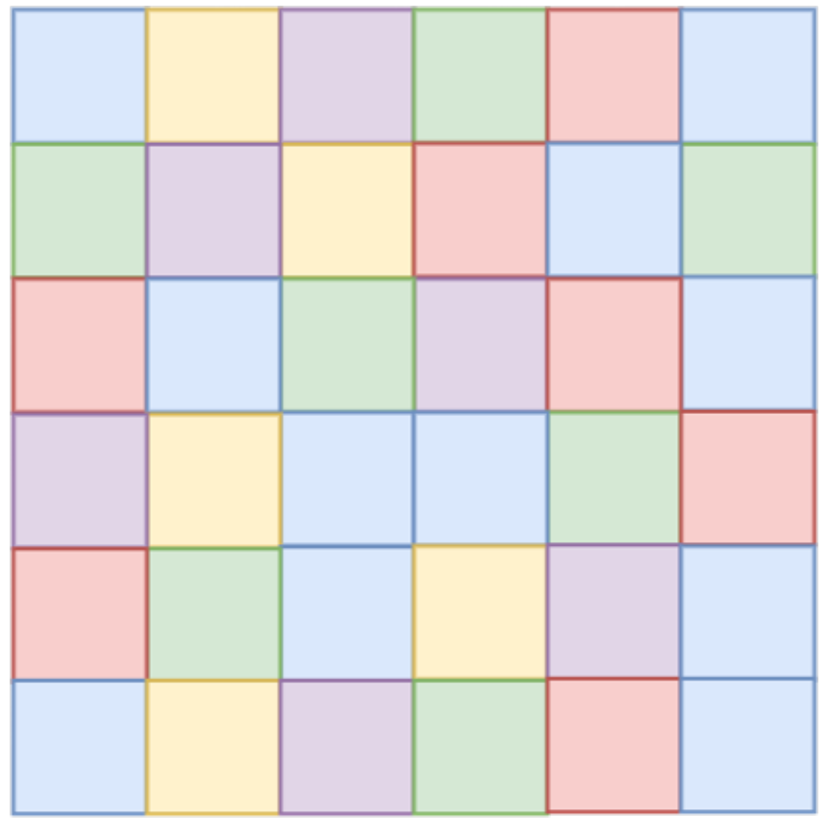
So what is a "tensor" anyway?

Tensors are many different things to many different people



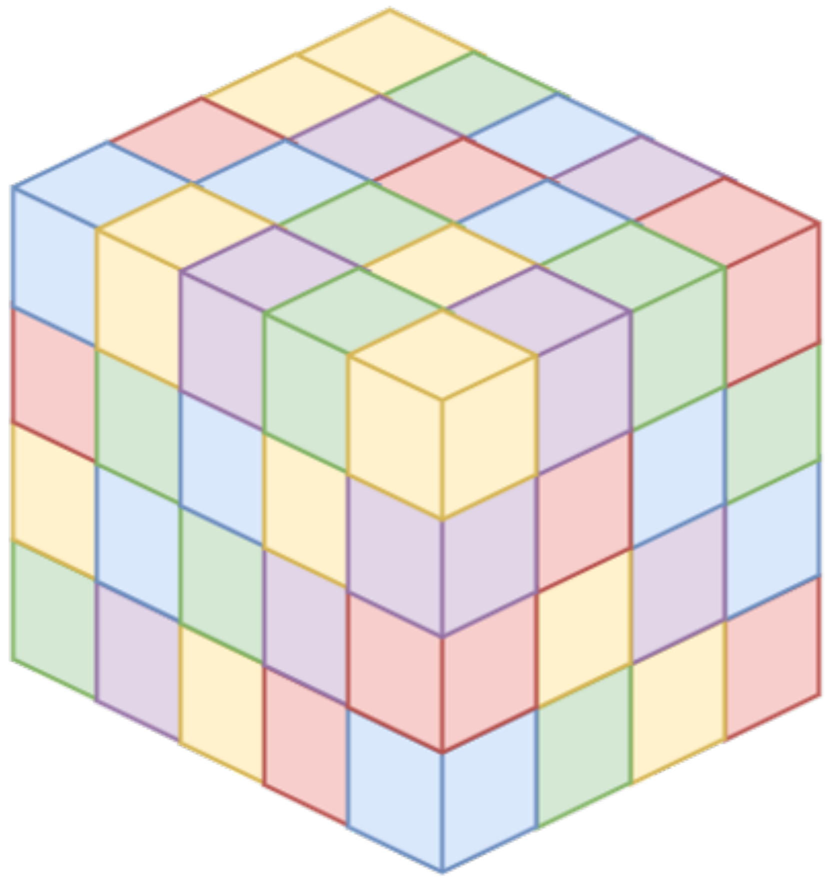
$\mathbf{x} \in \mathrm{R}^{m}$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

Third-Order Tensor

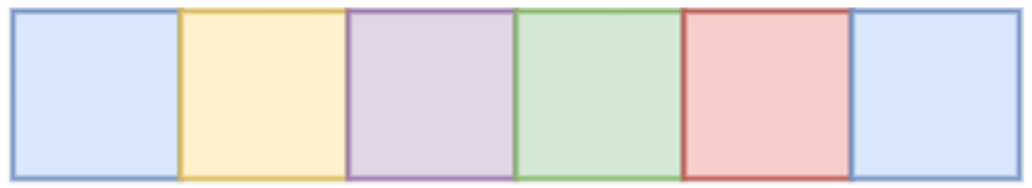
For today, we treat tensors "mechanically" as **multidimensional arrays**.

$$\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$$

Several other (richer?) perspectives:

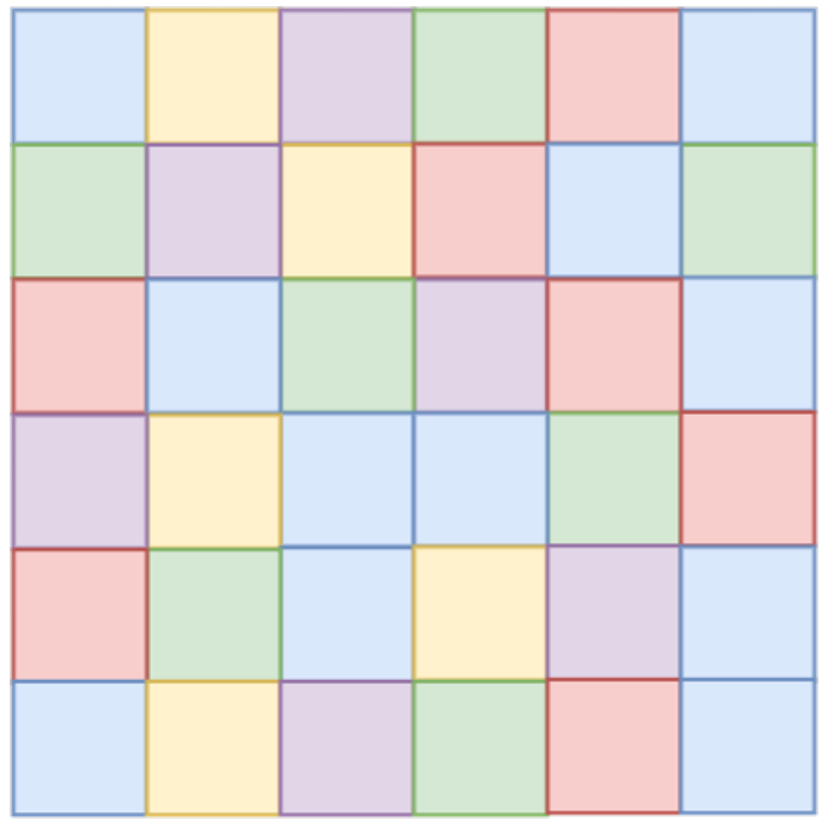
So what is a “tensor” anyway?

Tensors are many different things to many different people



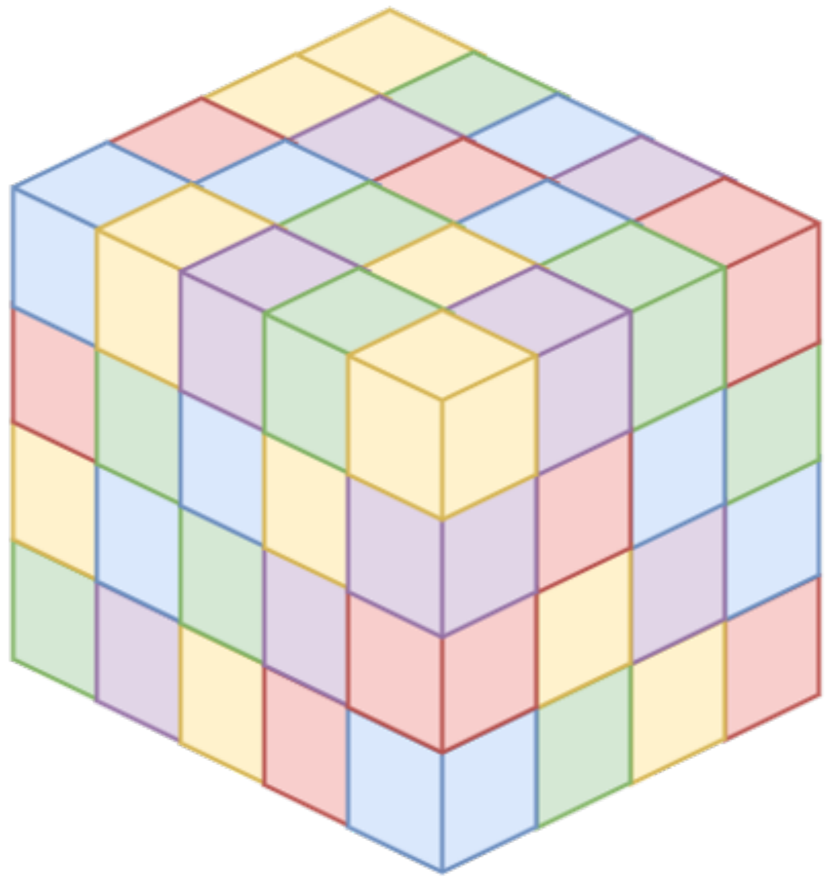
$\mathbf{x} \in \mathrm{R}^m$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

Third-Order Tensor

For today, we treat tensors “mechanically” as **multidimensional arrays**.

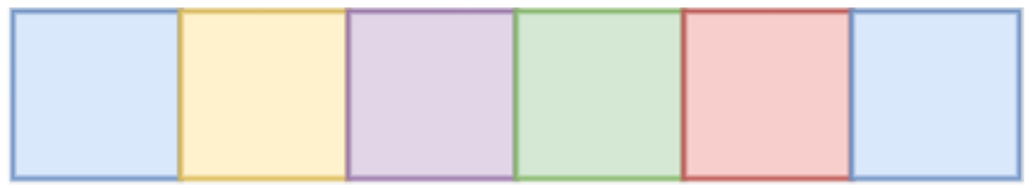
$$\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$$

Several other (richer?) perspectives:

- Point in the tensor product of vector spaces

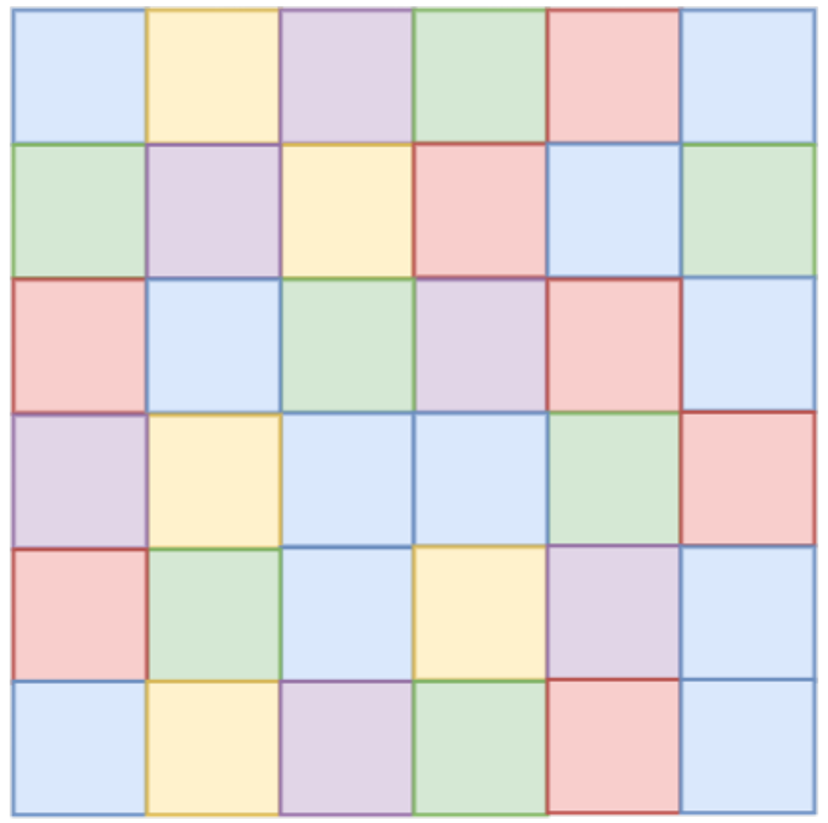
So what is a “tensor” anyway?

Tensors are many different things to many different people



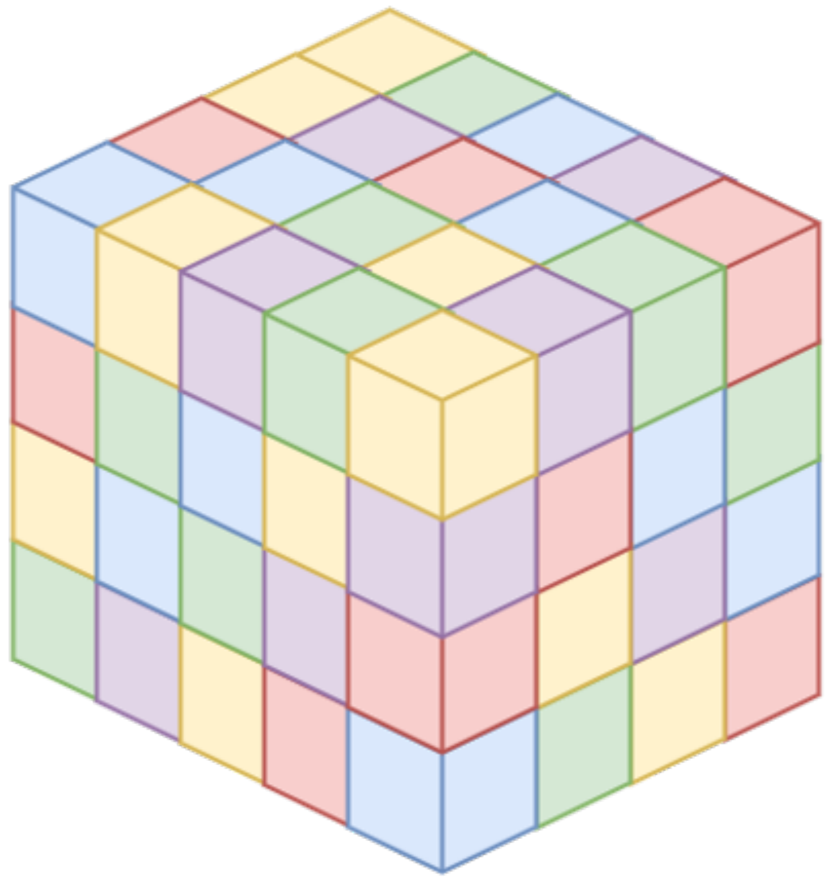
$\mathbf{x} \in \mathrm{R}^{m}$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

Third-Order Tensor

For today, we treat tensors “mechanically” as **multidimensional arrays**.

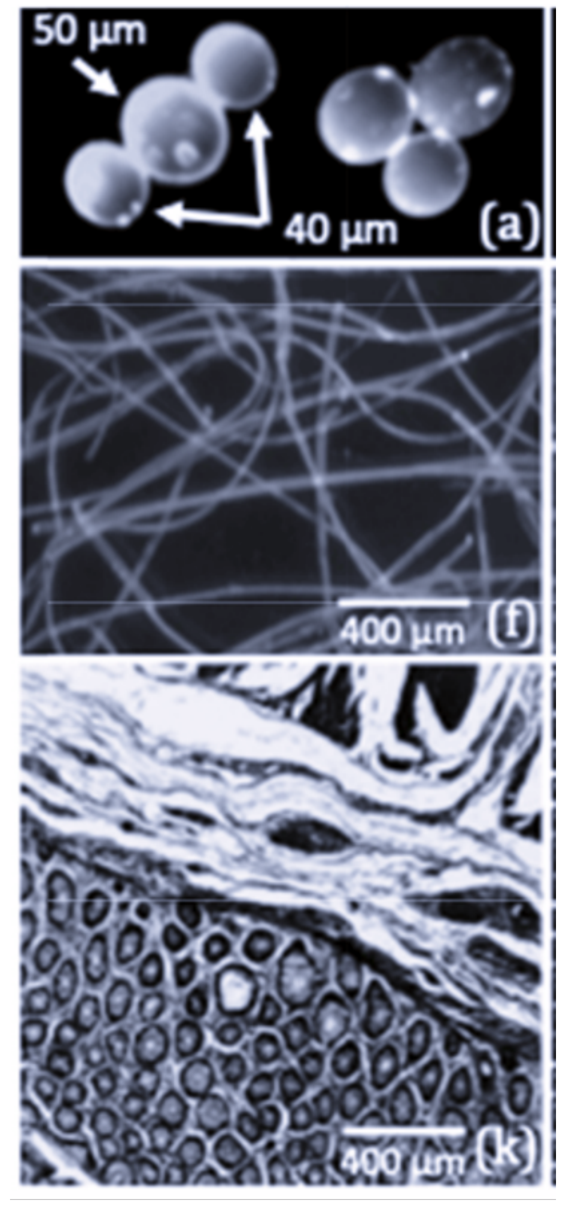
$$\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$$

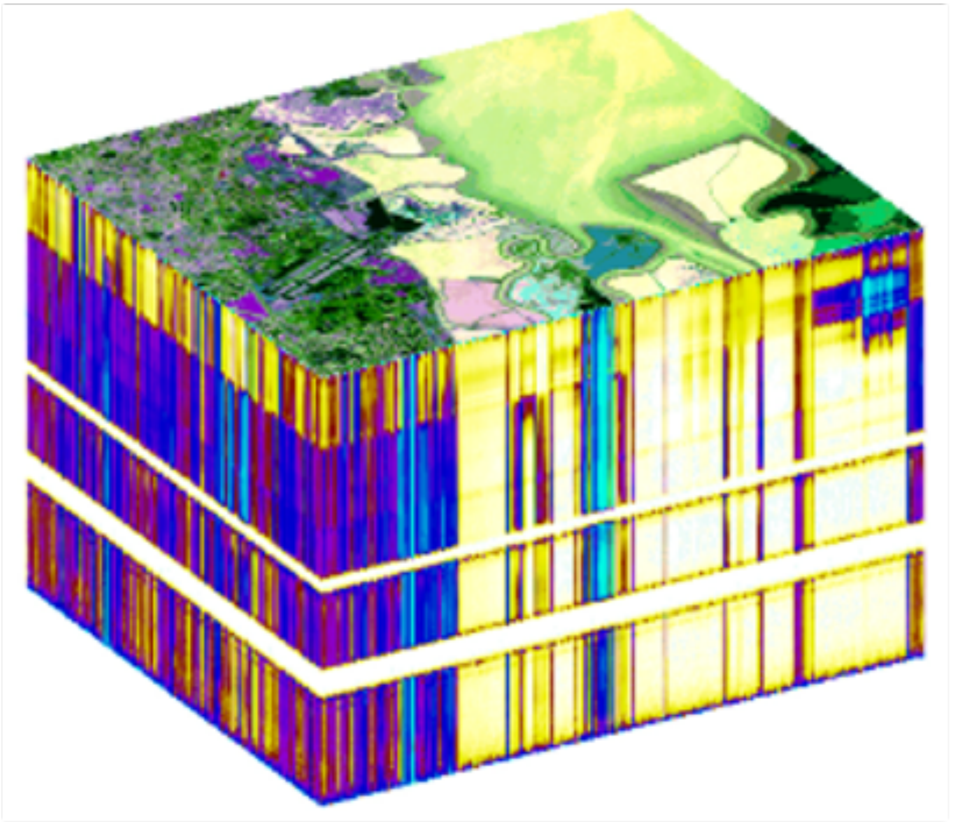
Several other (richer?) perspectives:

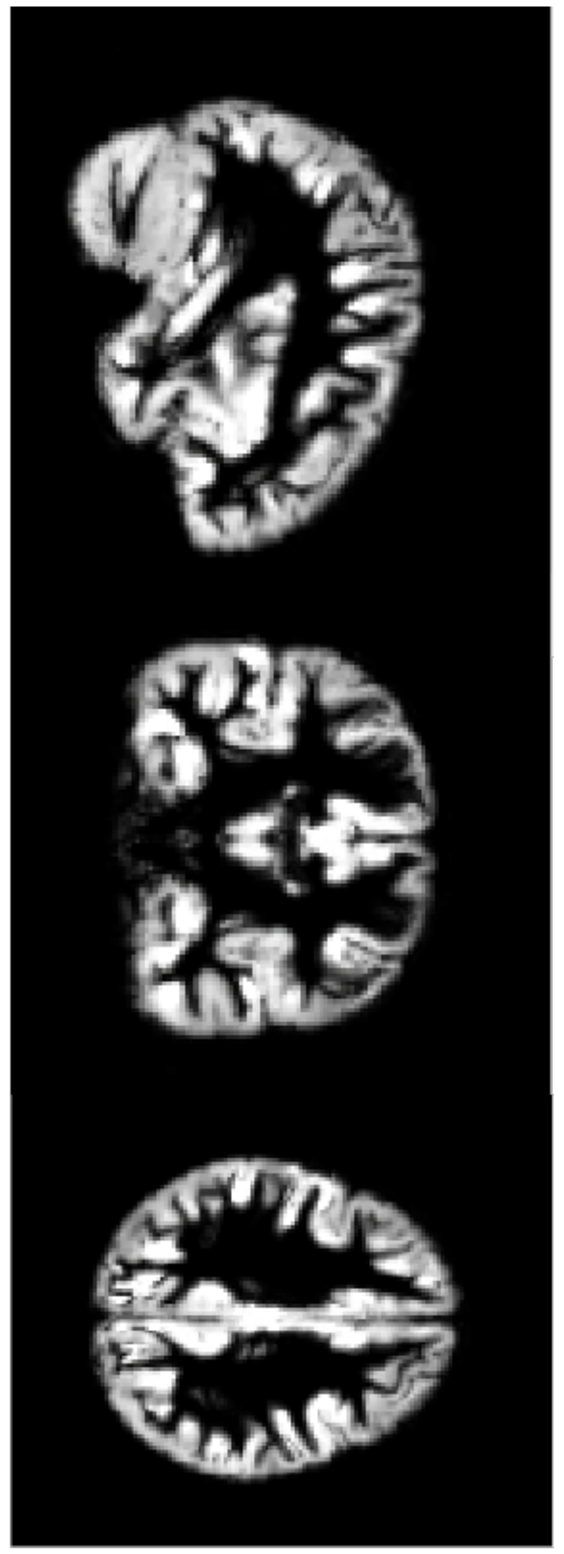
- Point in the tensor product of vector spaces
- Multilinear operator (or a tensor representation of $GL(n)$)

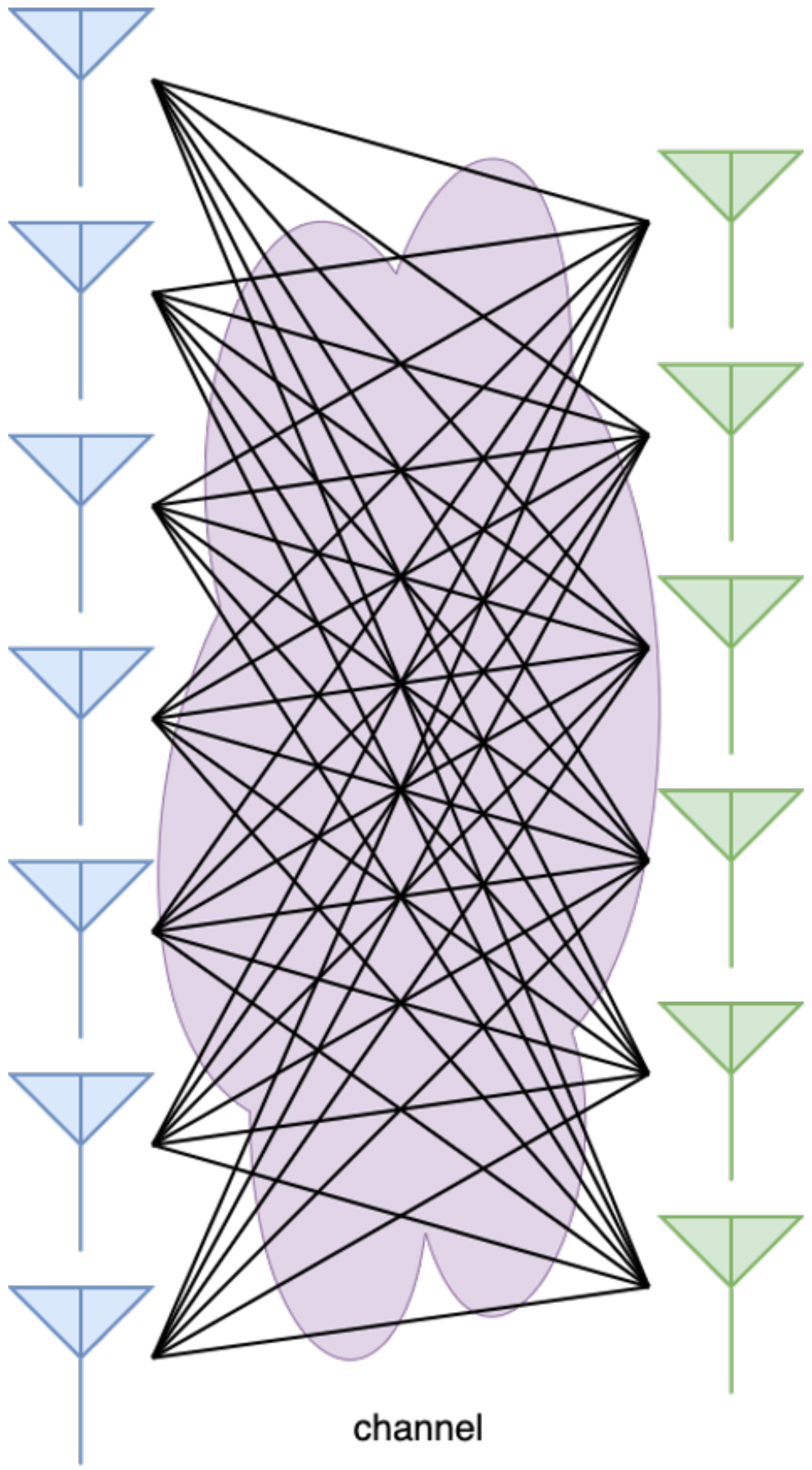
Where do we see tensors?

Multidimensional arrays are everywhere!





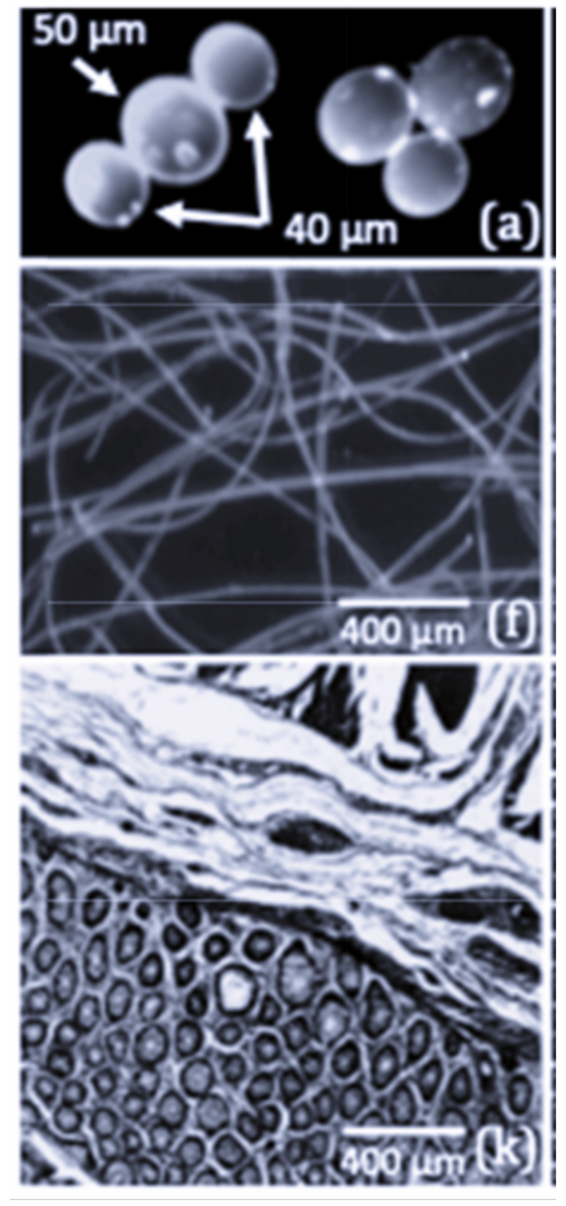


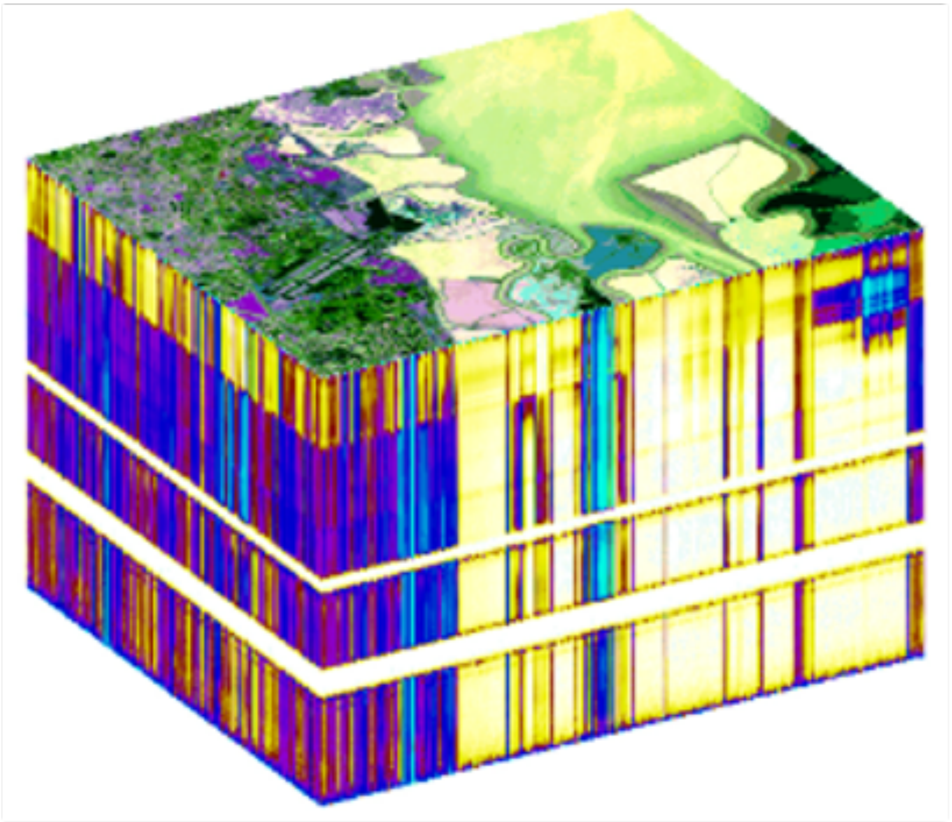


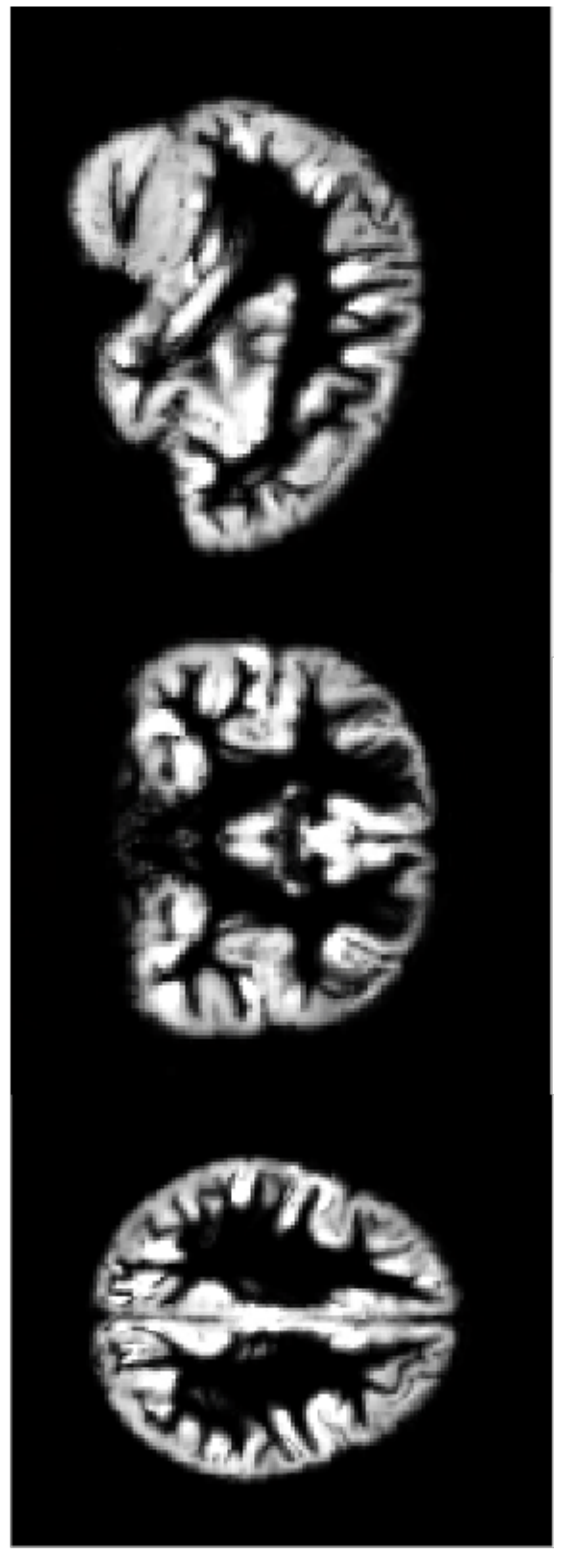
Where do we see tensors?

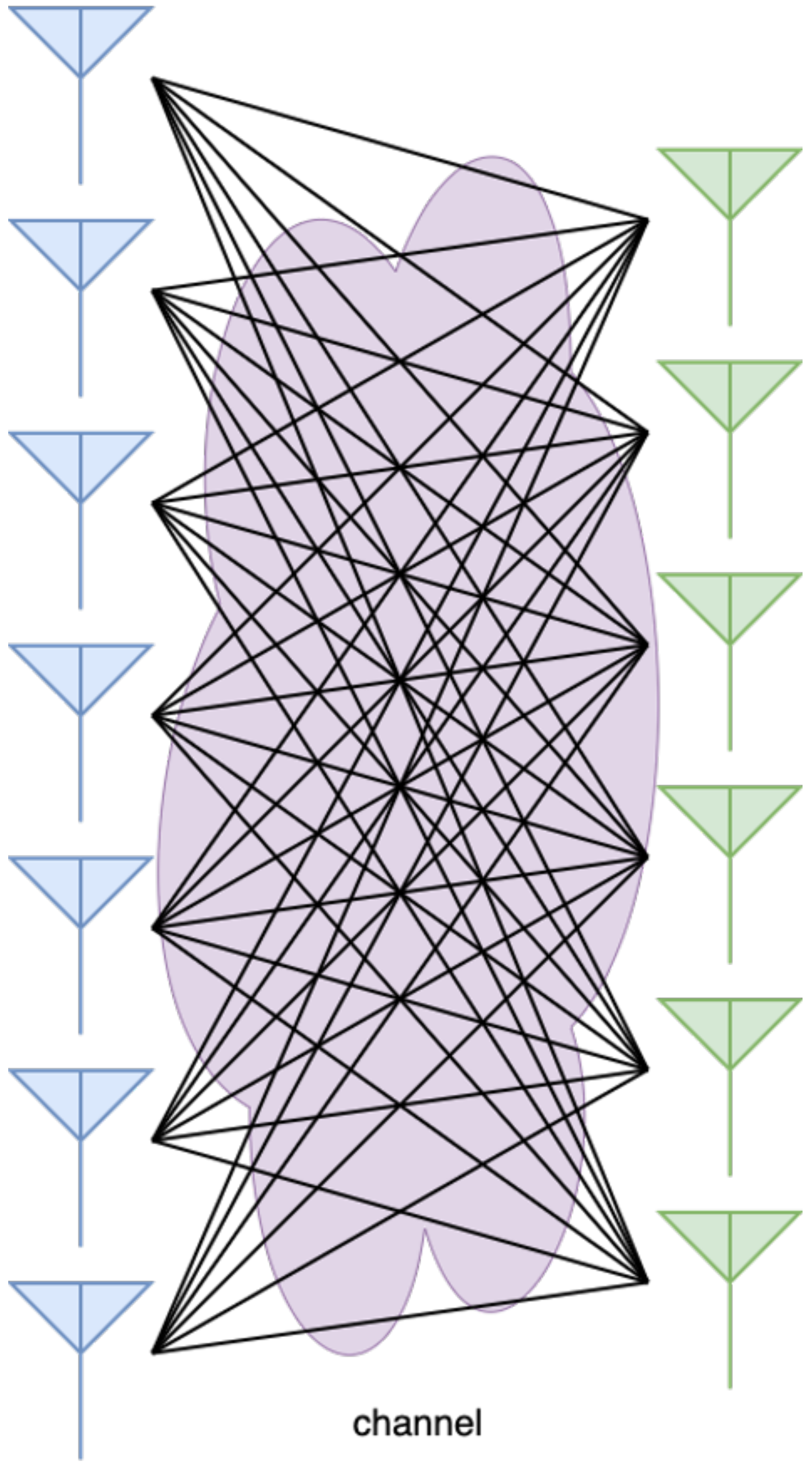
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging





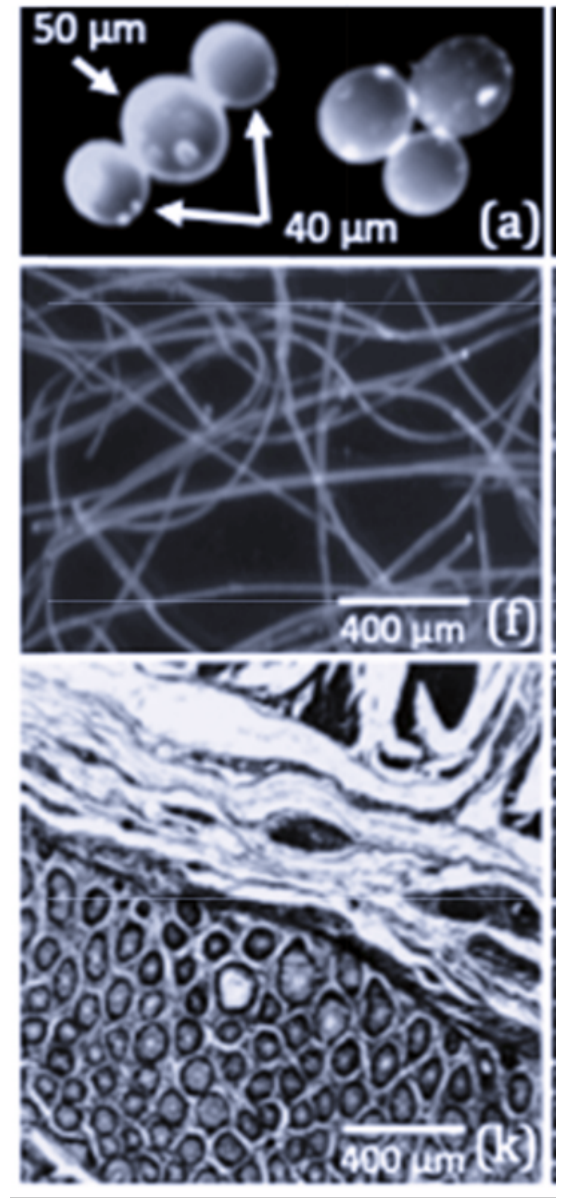


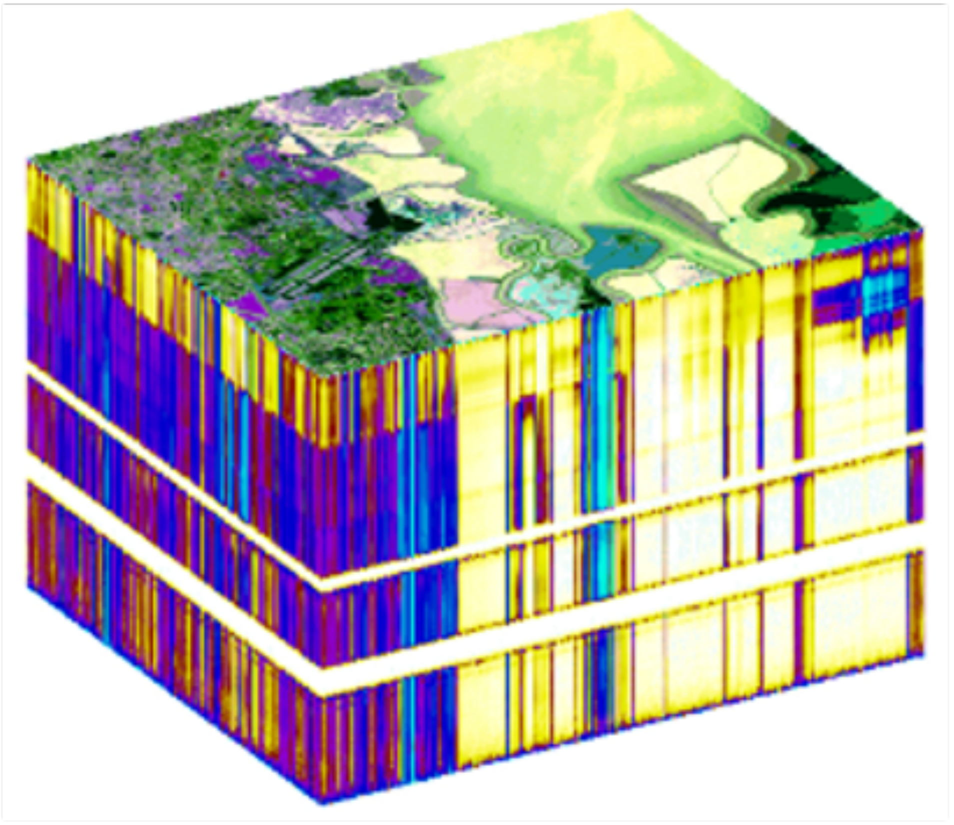


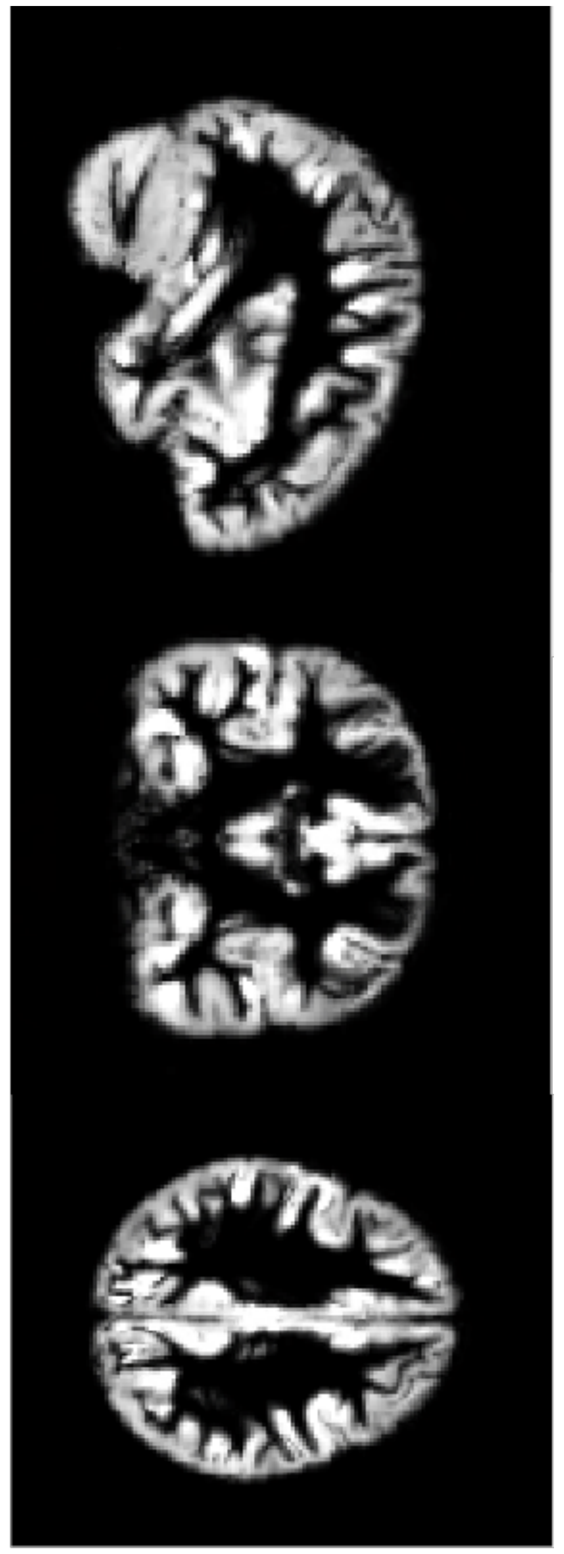
Where do we see tensors?

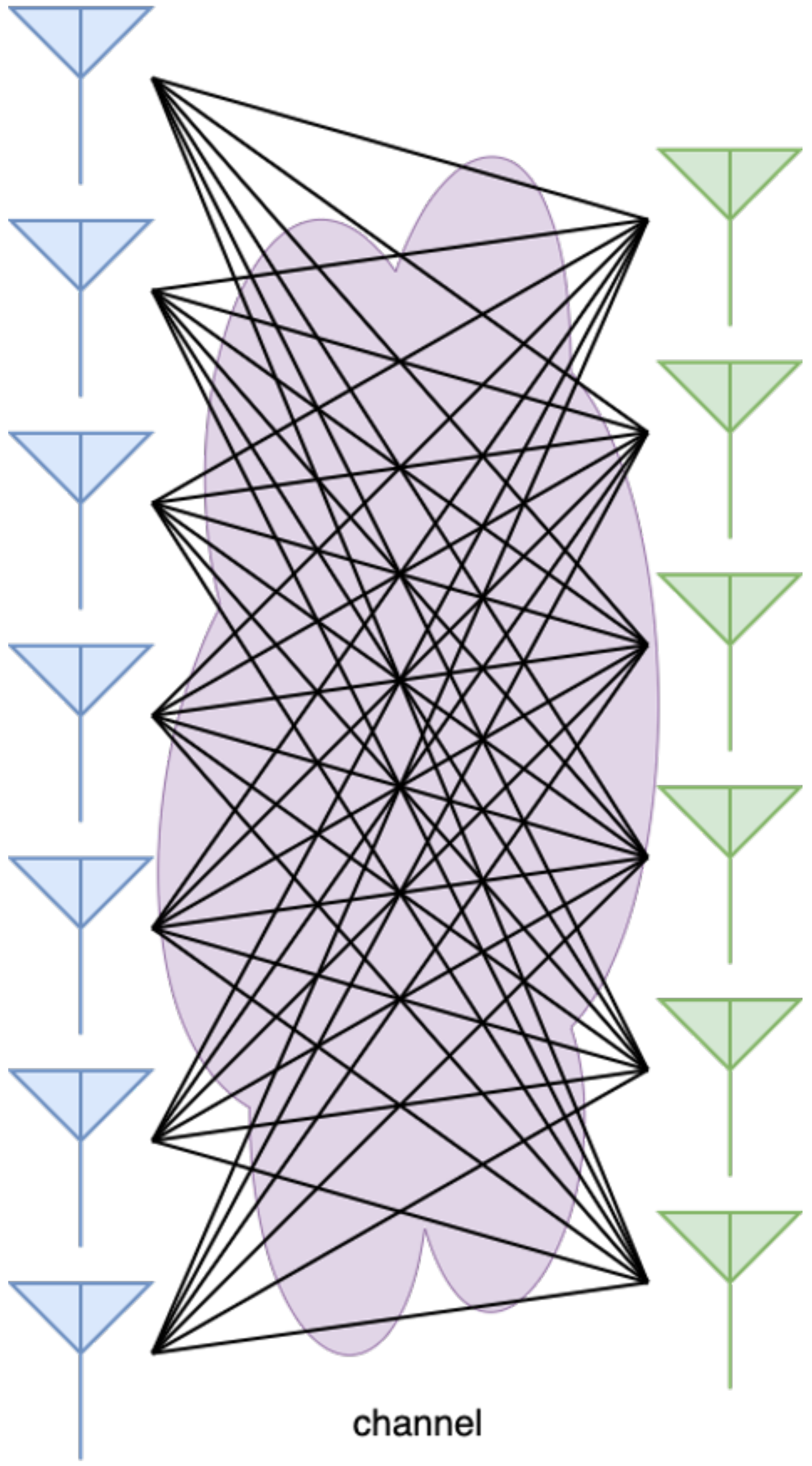
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging





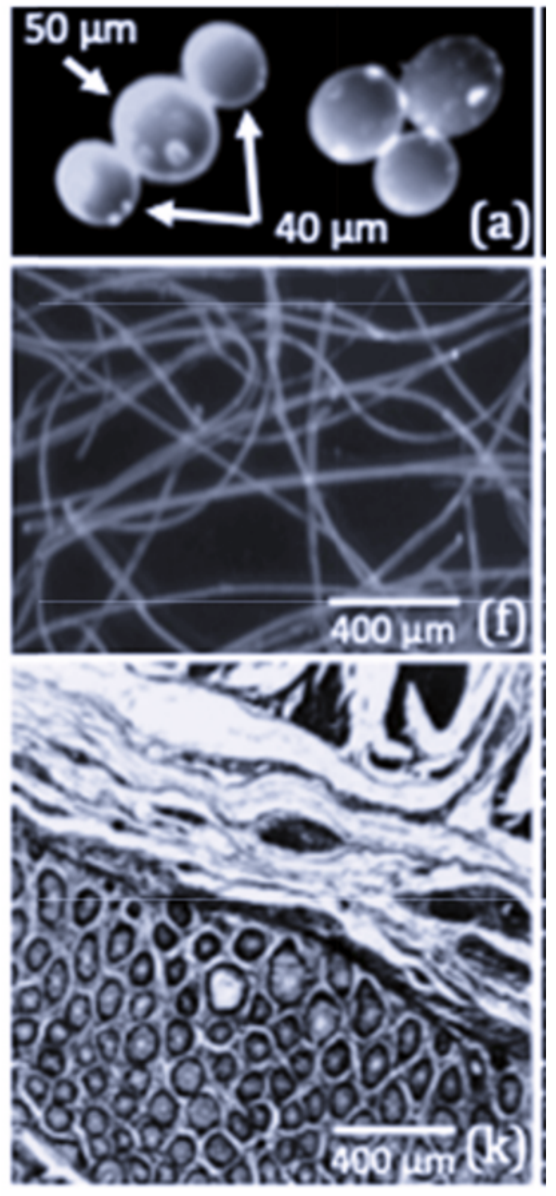


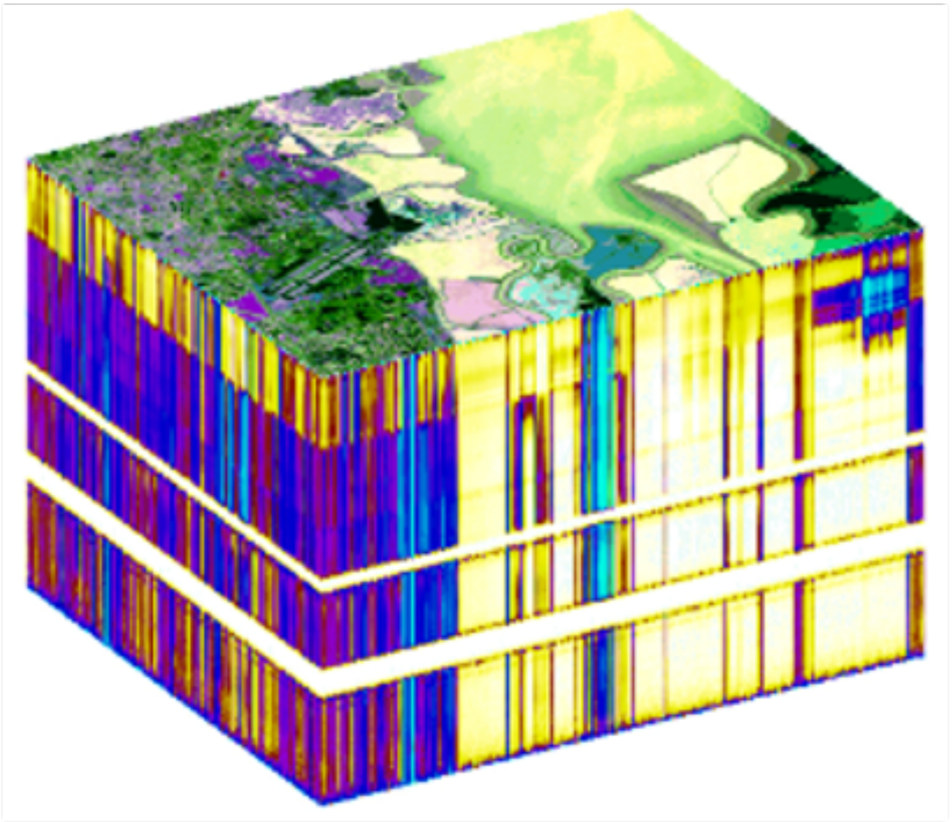


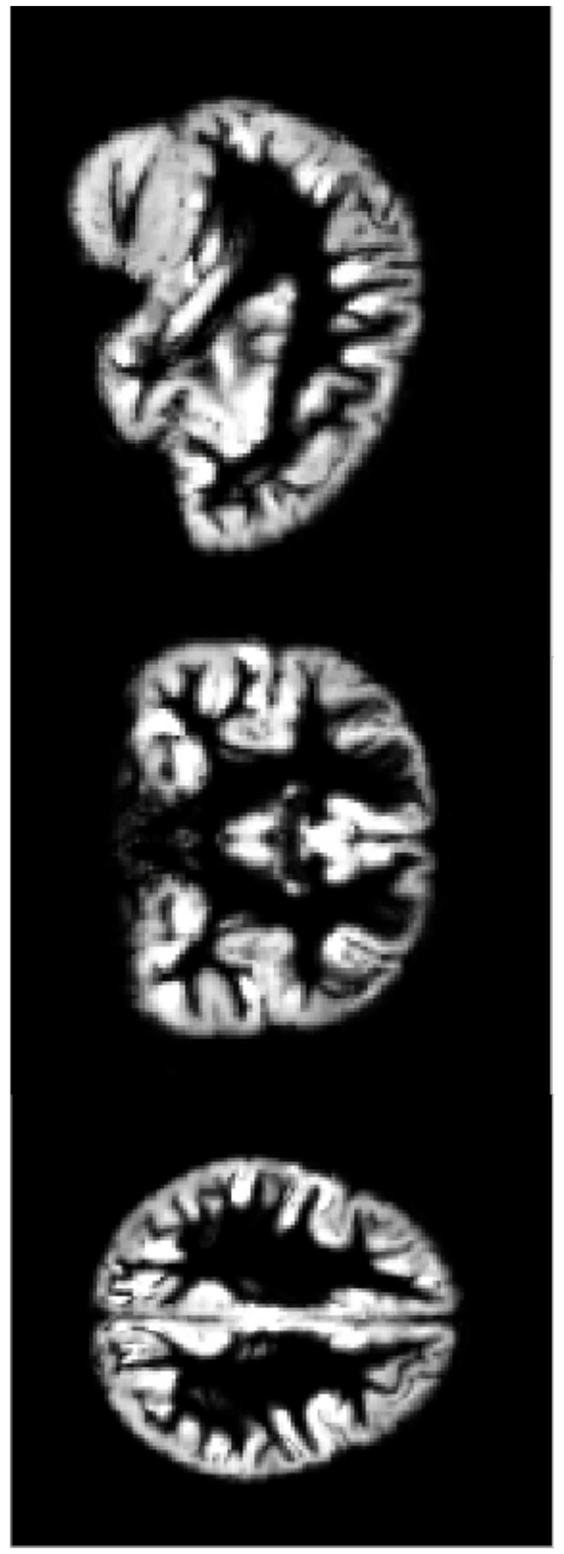
Where do we see tensors?

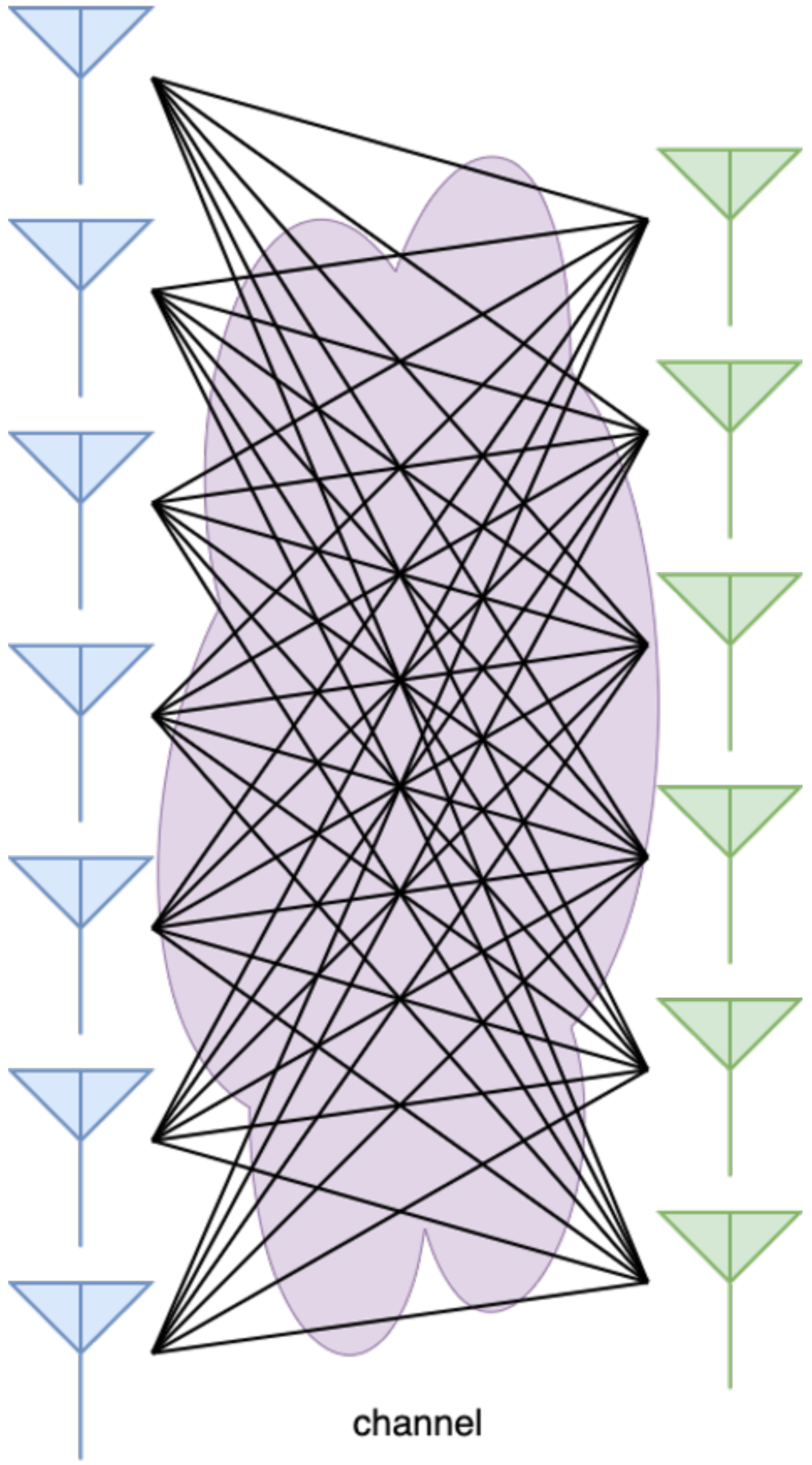
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging
- **Communications:** Massive MIMO





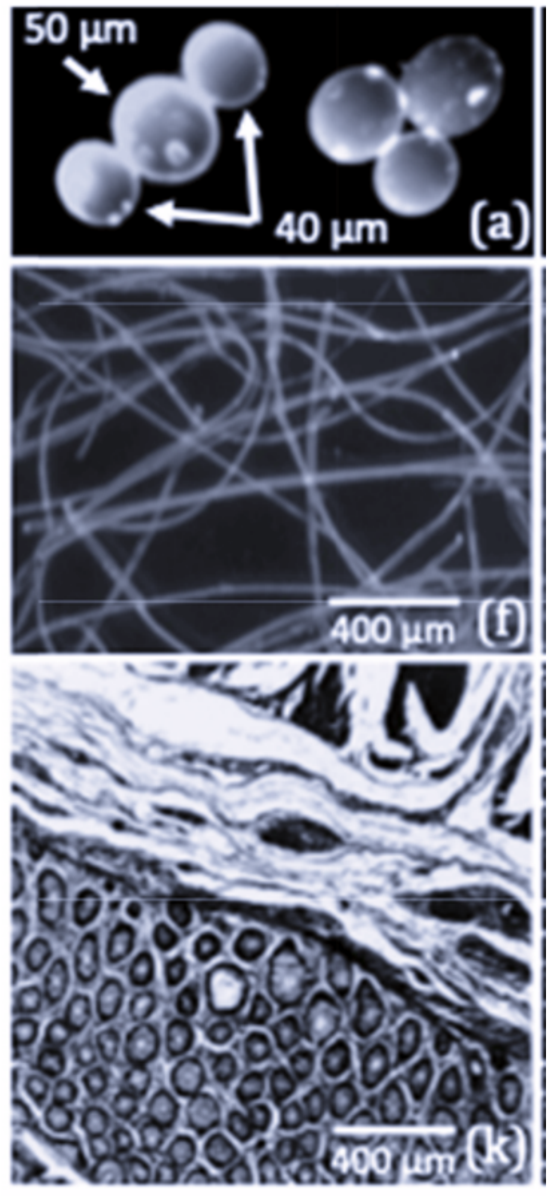


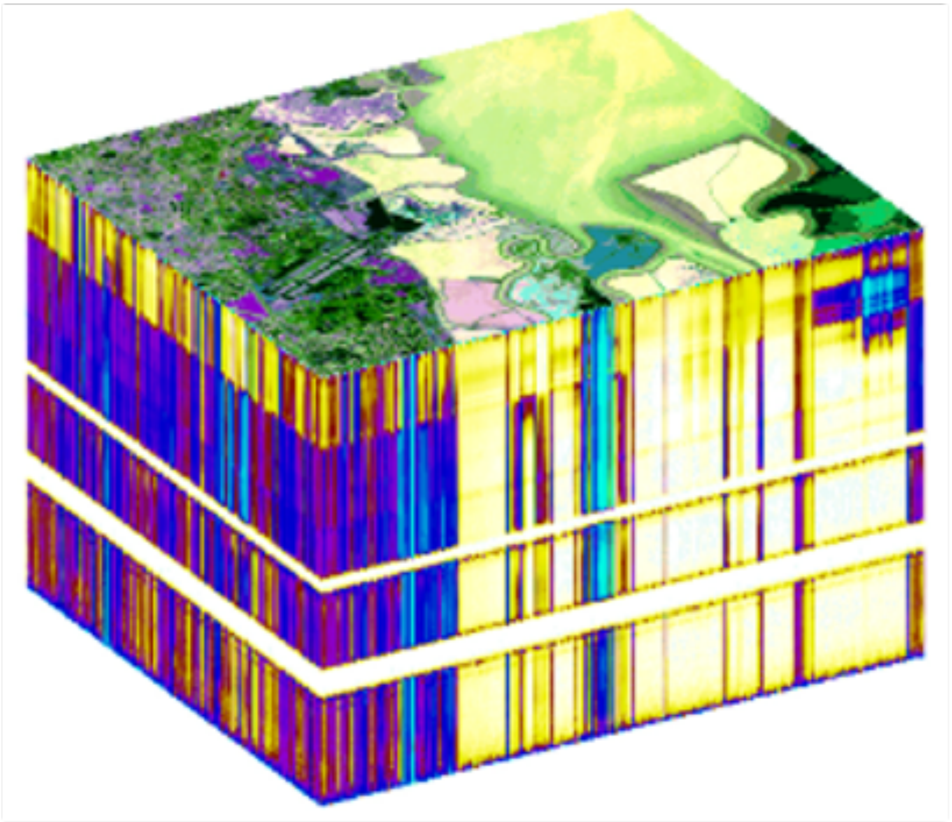


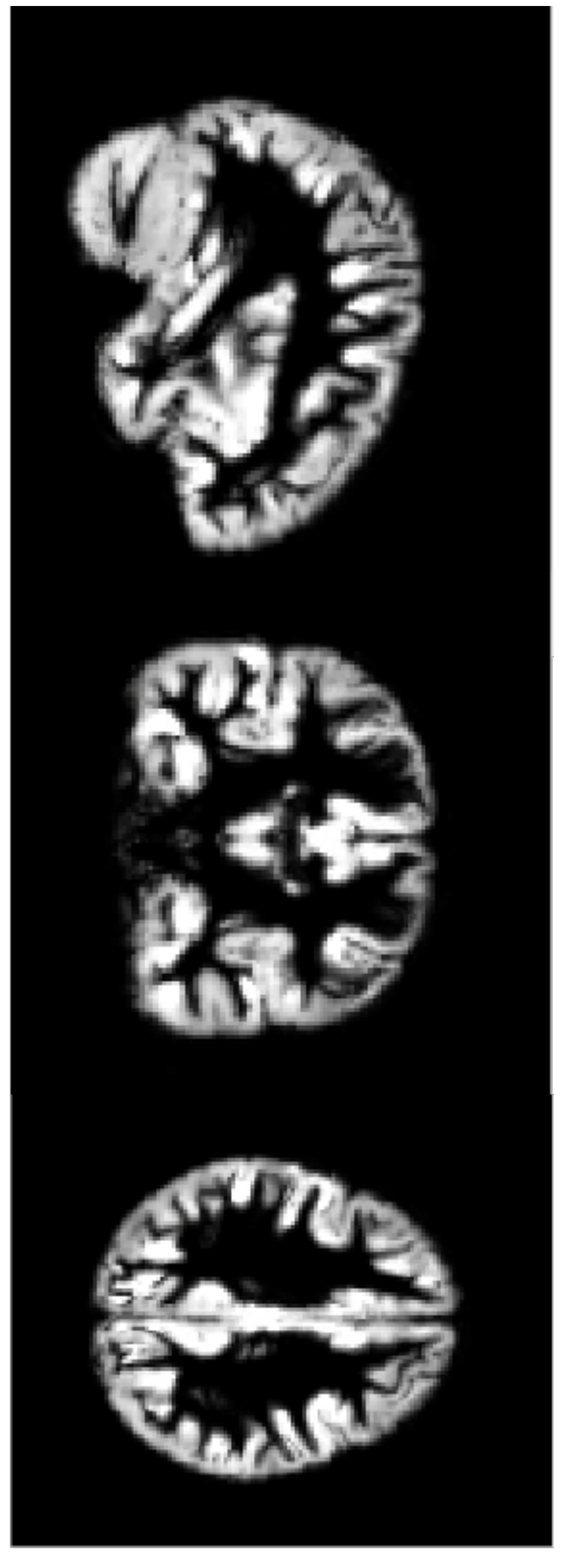
Where do we see tensors?

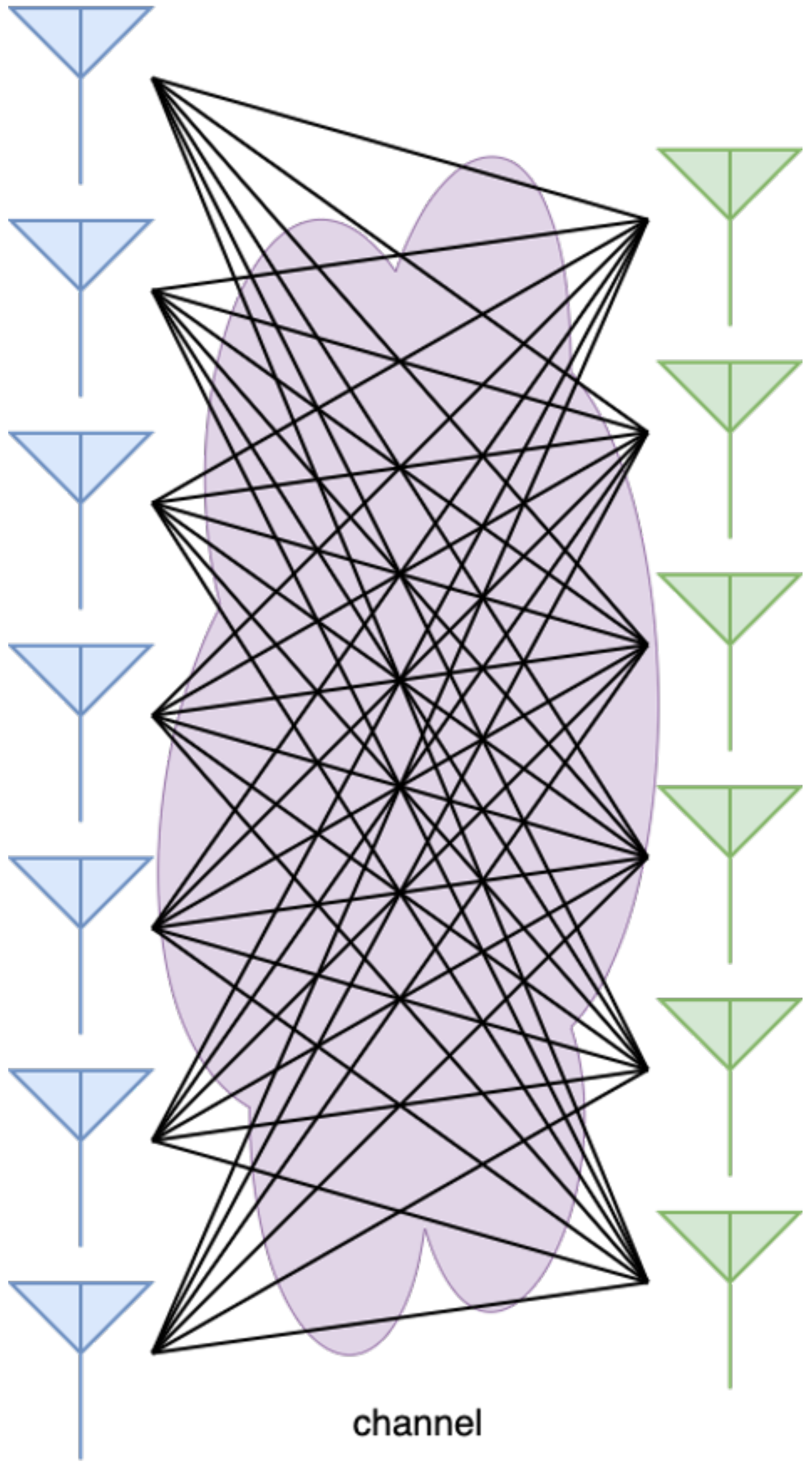
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging
- **Communications:** Massive MIMO
- **Probability:** Joint PMFs on multiple variables





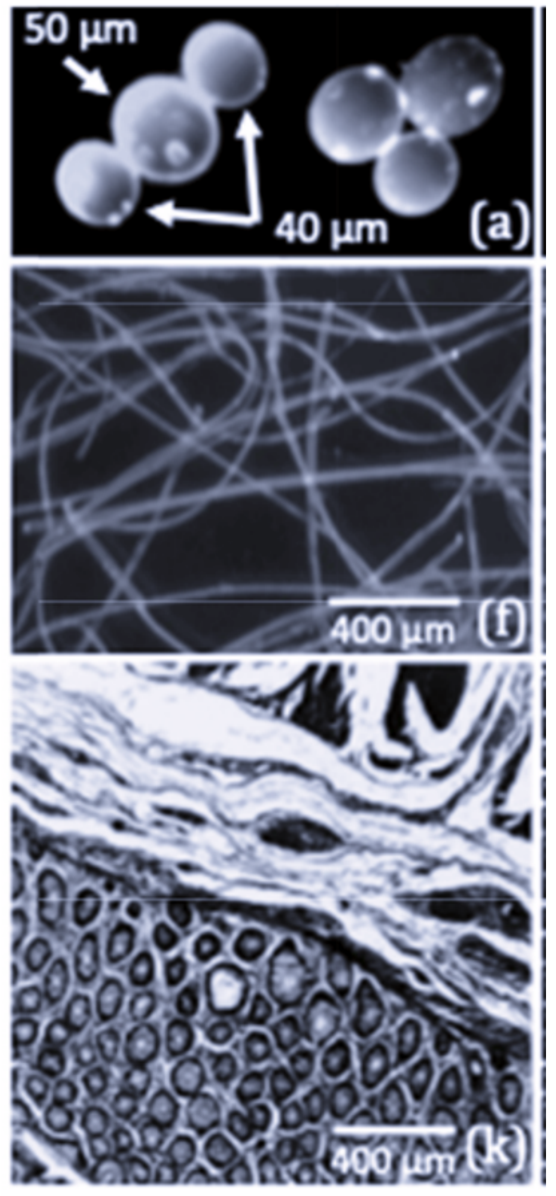


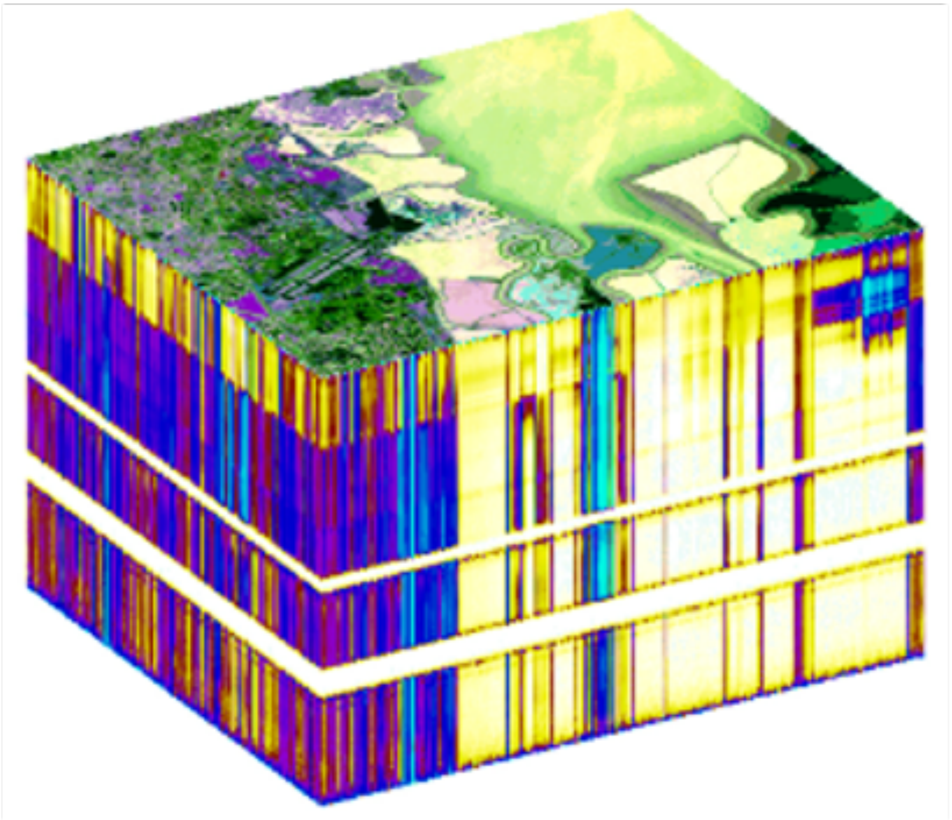


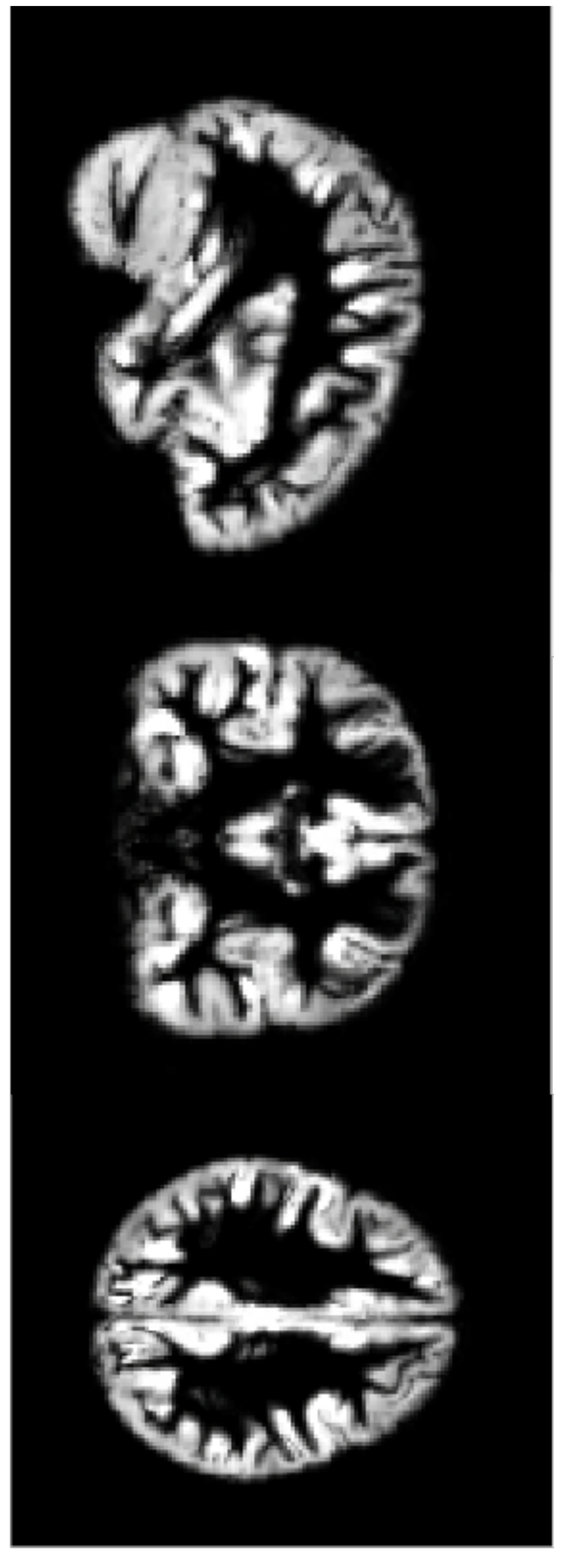
Where do we see tensors?

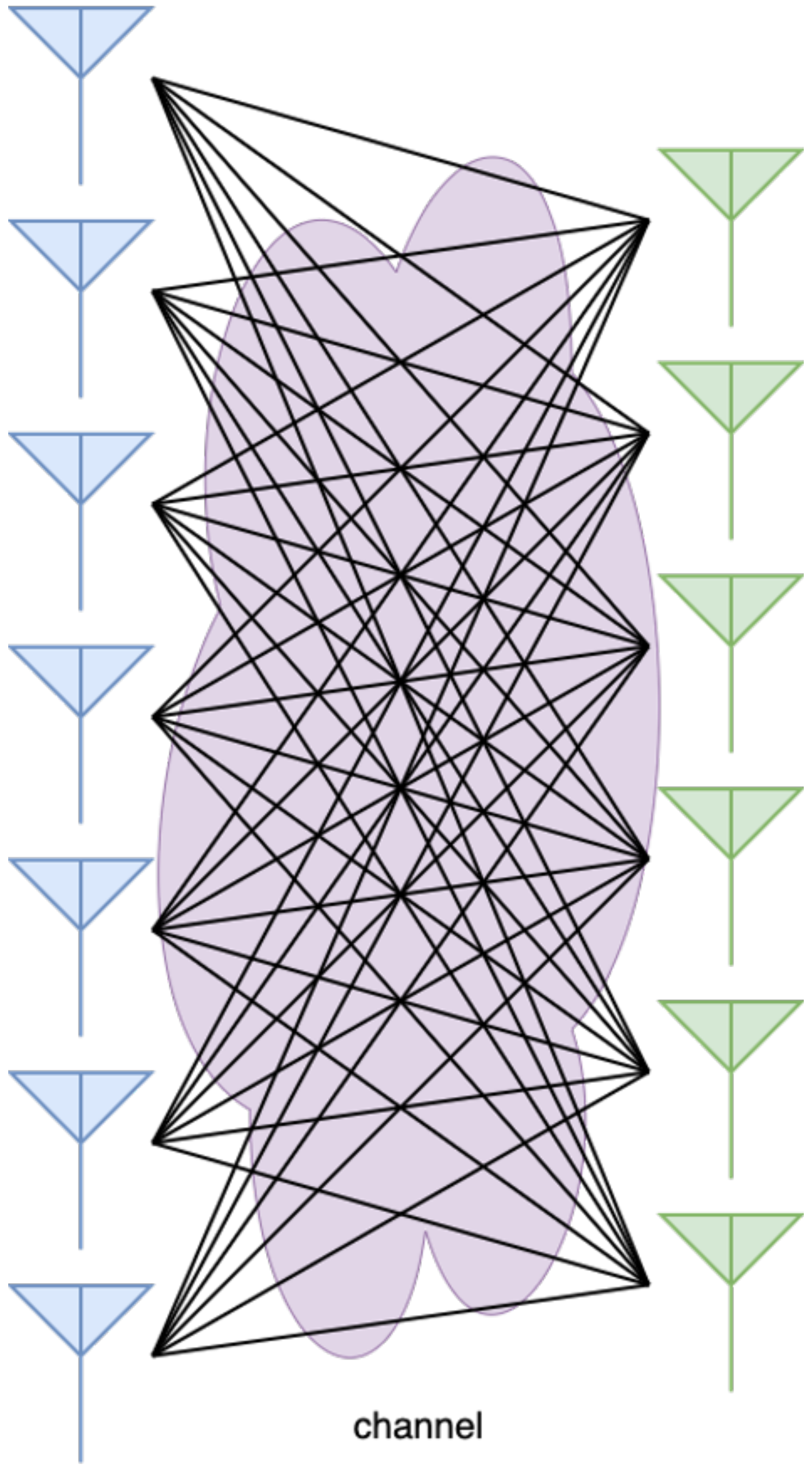
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging
- **Communications:** Massive MIMO
- **Probability:** Joint PMFs on multiple variables
- **Network science:** Time-varying graphs





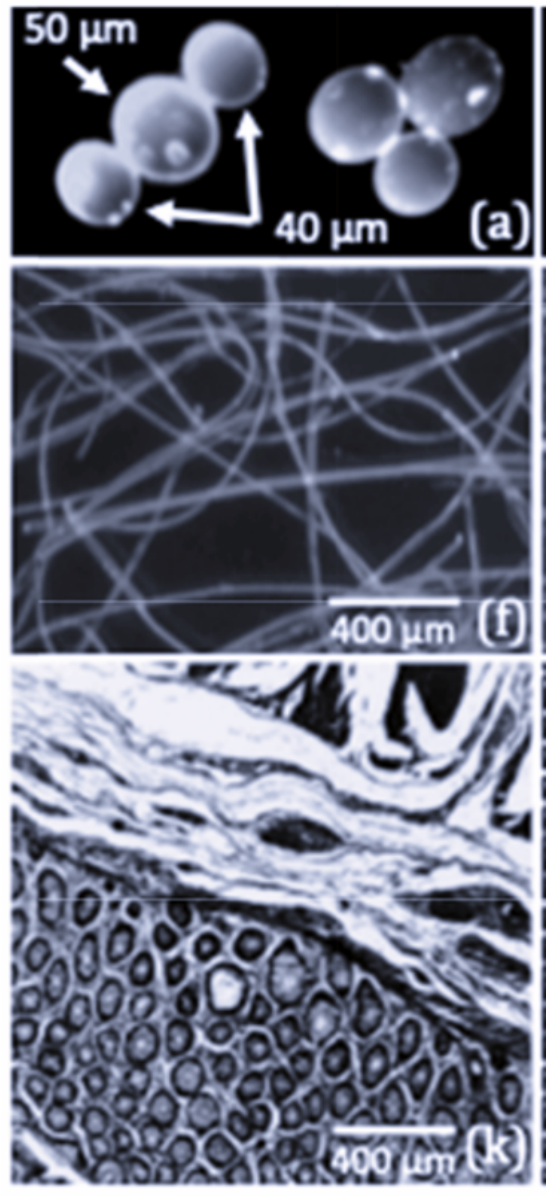


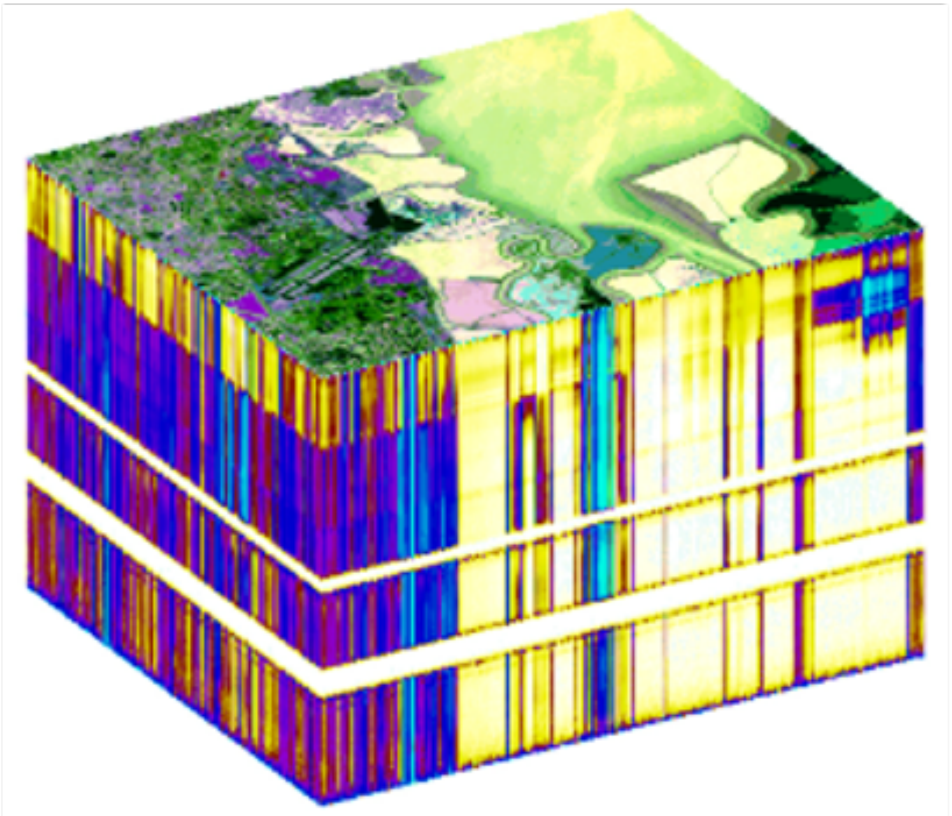


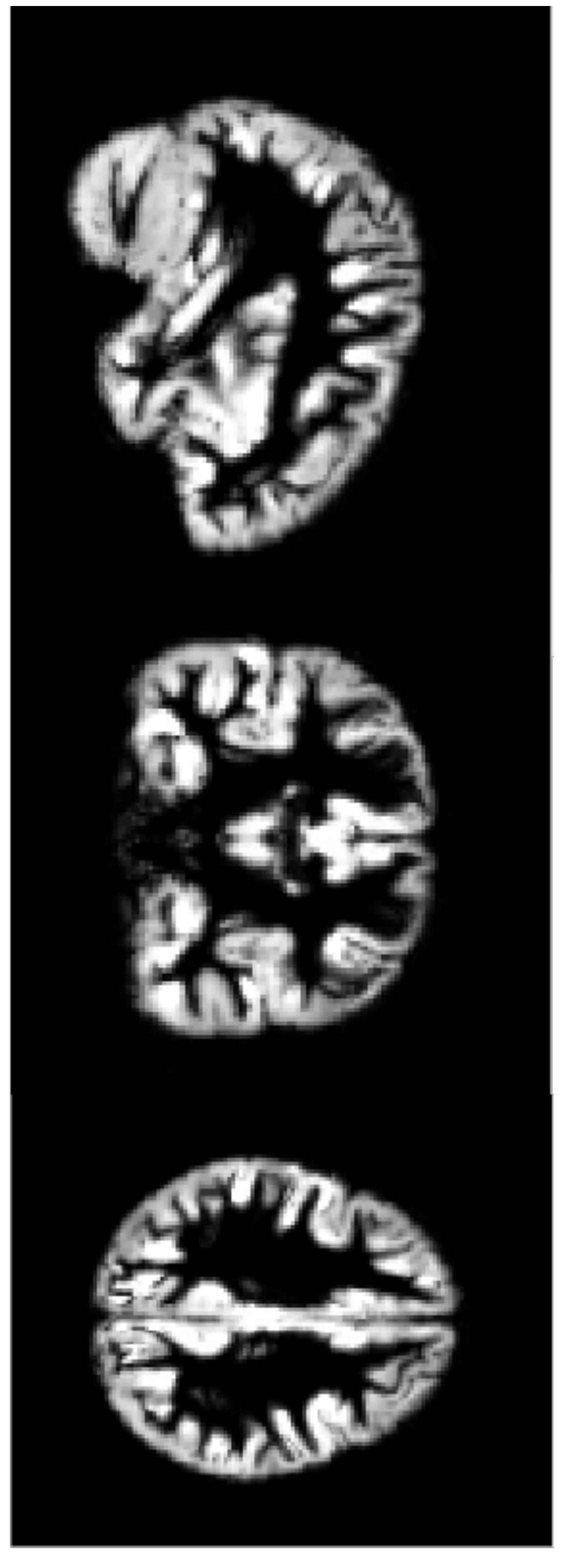
Where do we see tensors?

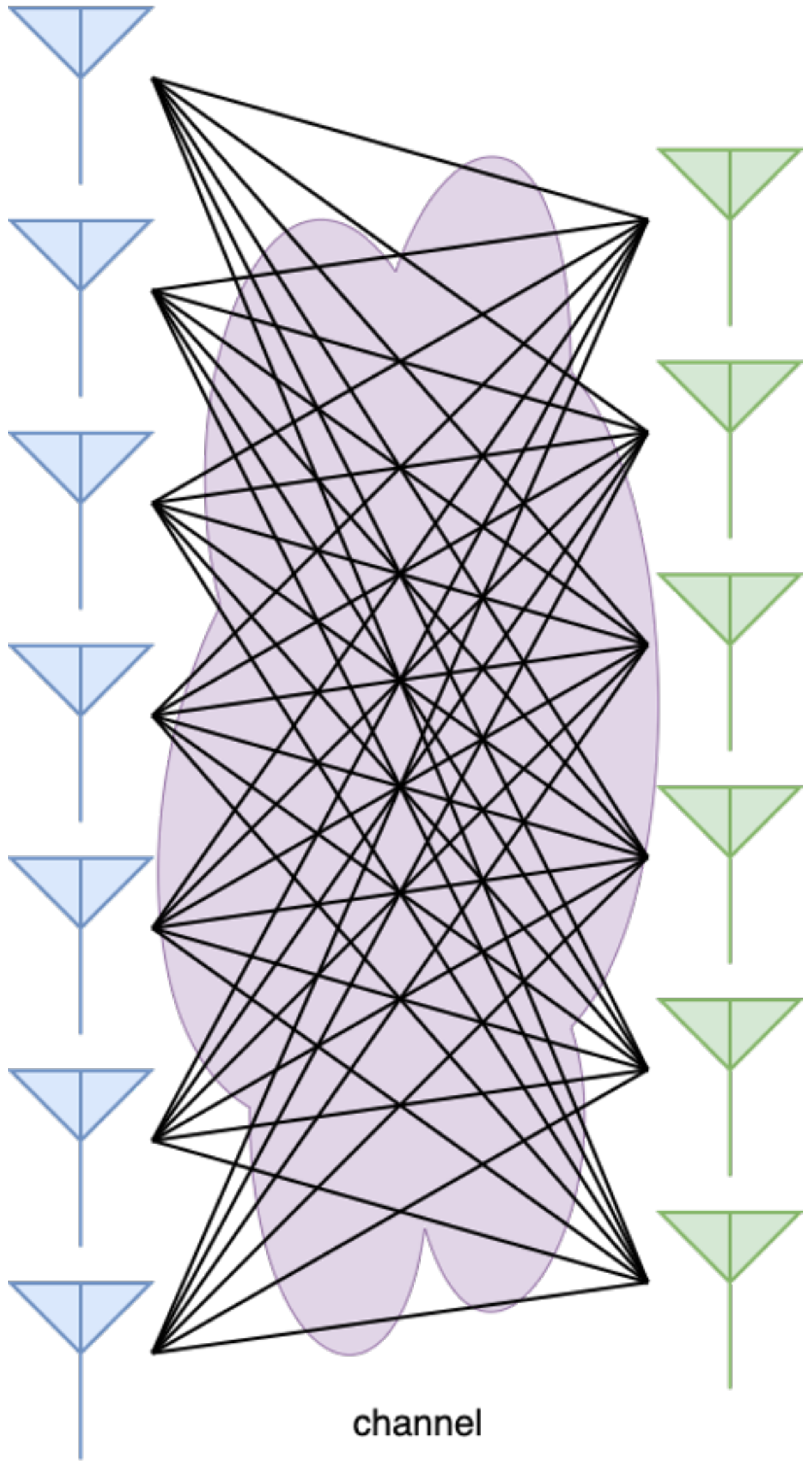
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging
- **Communications:** Massive MIMO
- **Probability:** Joint PMFs on multiple variables
- **Network science:** Time-varying graphs
- Also chemometrics, numerical linear algebra, psychometrics, theoretical computer science…









What do we want to do with tensor data?

All the regular things we do with data…

What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery

What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery

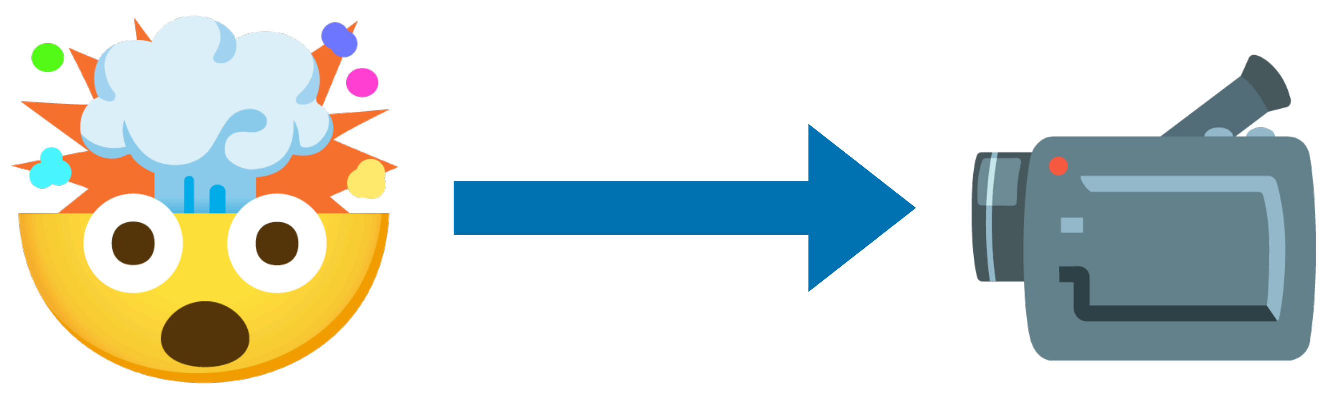
What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery

What do we want to do with tensor data?

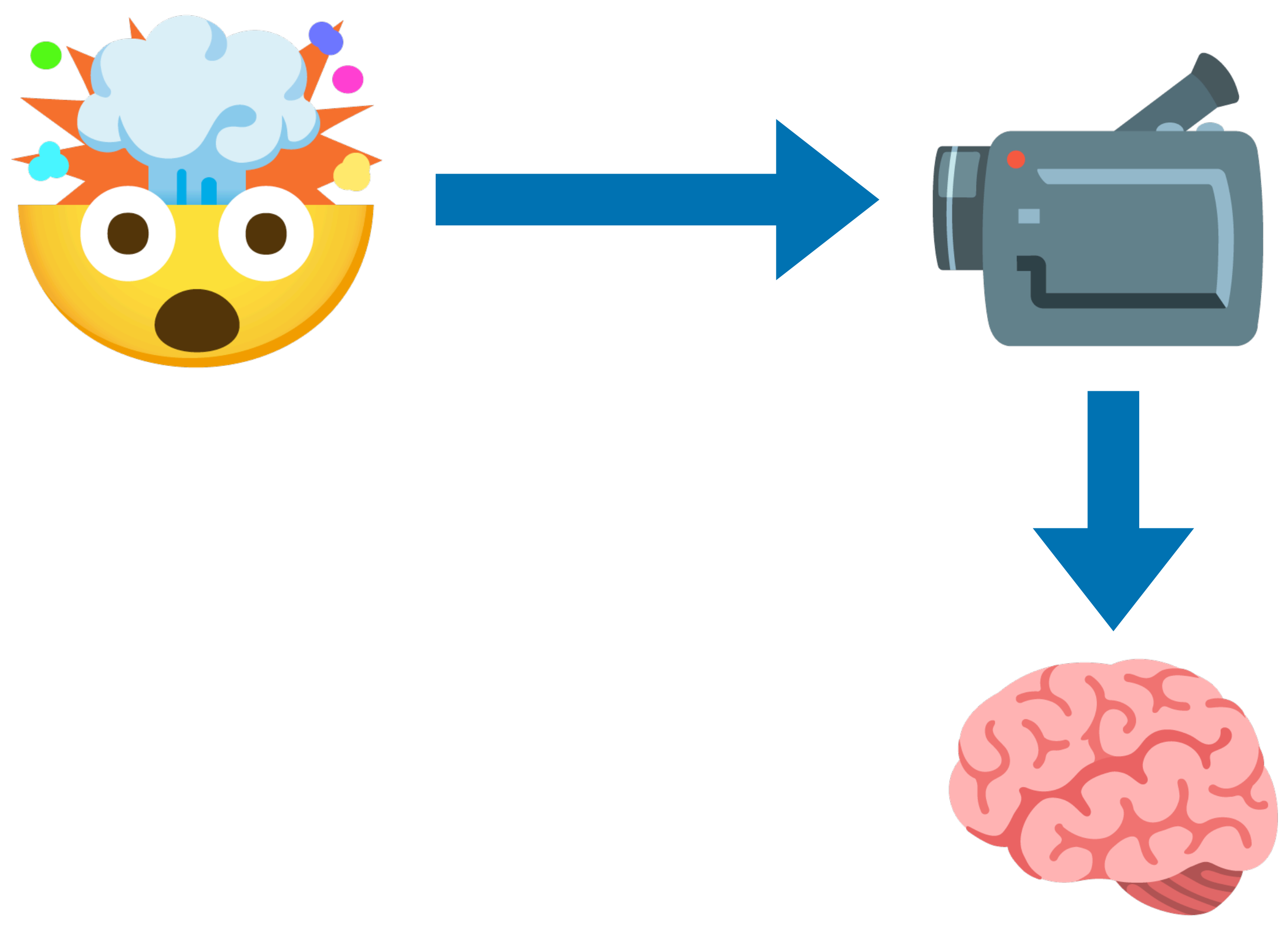
All the regular things we do with data…



- Signal recovery

What do we want to do with tensor data?

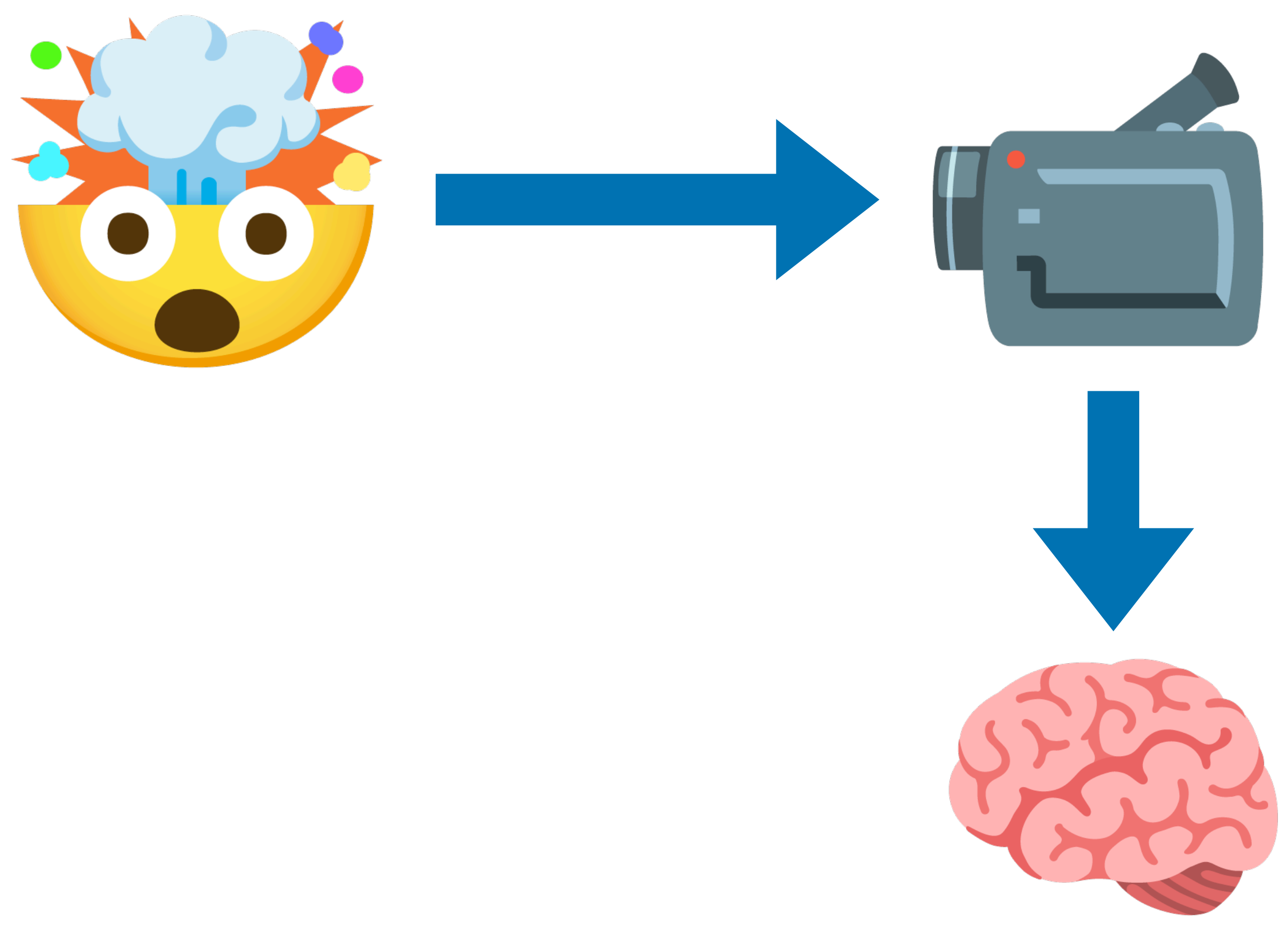
All the regular things we do with data…



- Signal recovery

What do we want to do with tensor data?

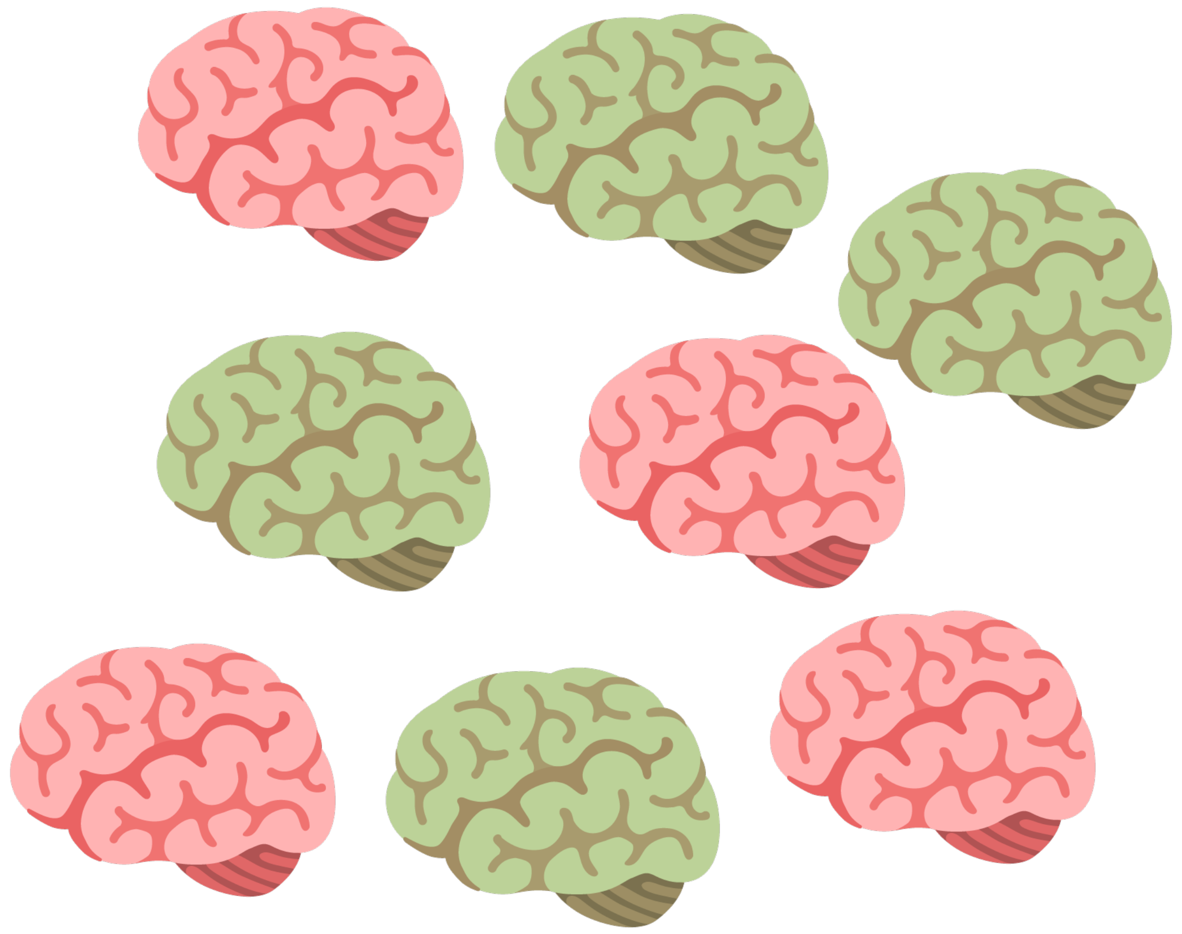
All the regular things we do with data...

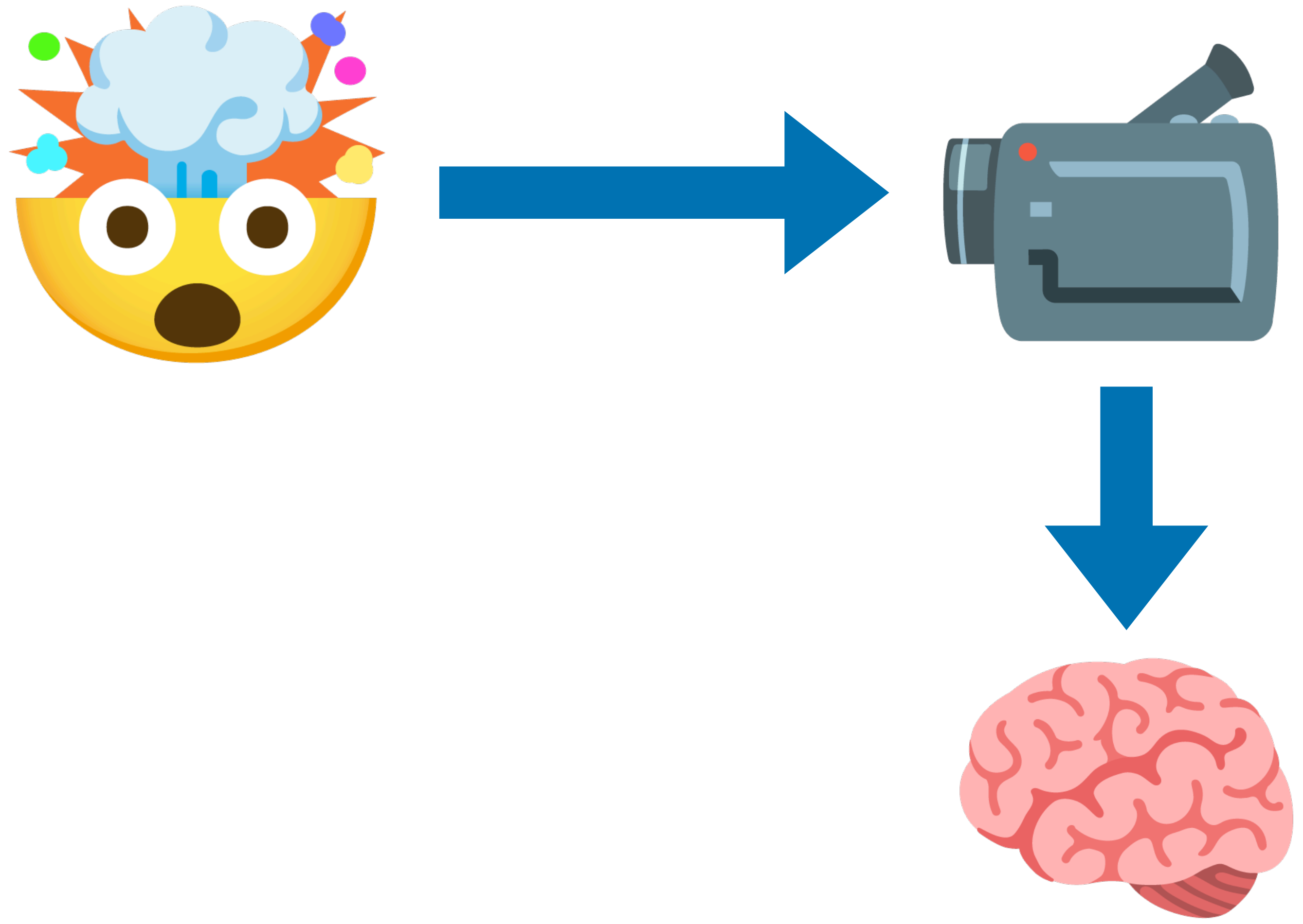


- Signal recovery
- Unsupervised learning (representation learning)

What do we want to do with tensor data?

All the regular things we do with data…

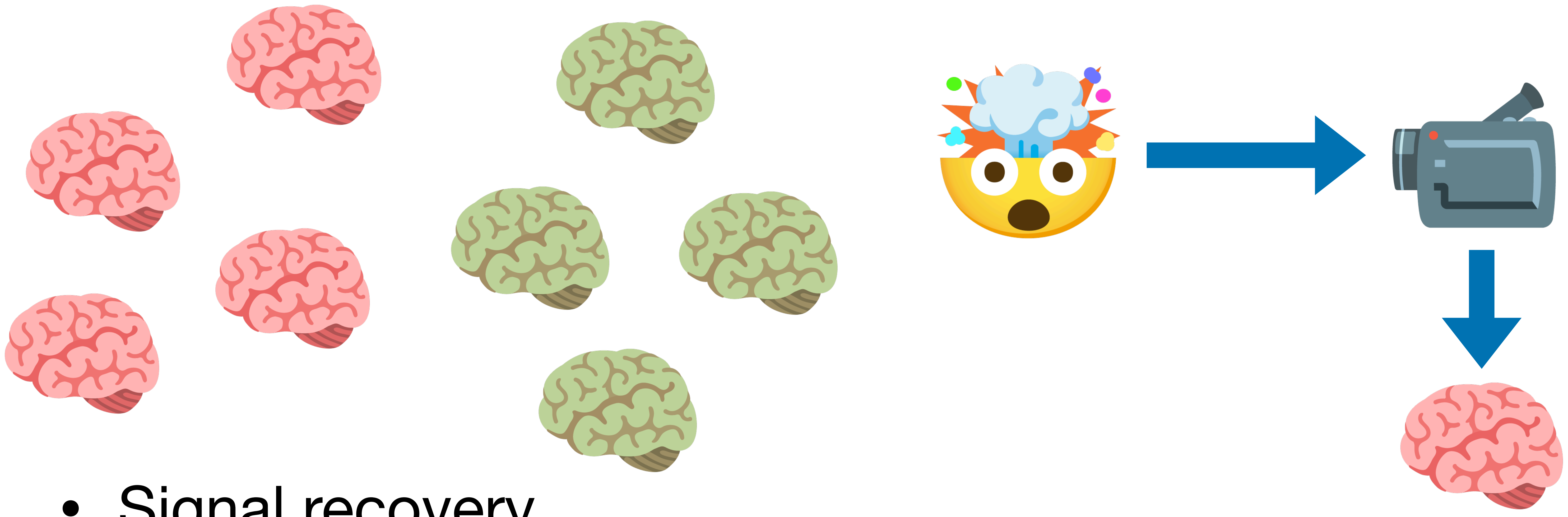




- Signal recovery
- Unsupervised learning (representation learning)

What do we want to do with tensor data?

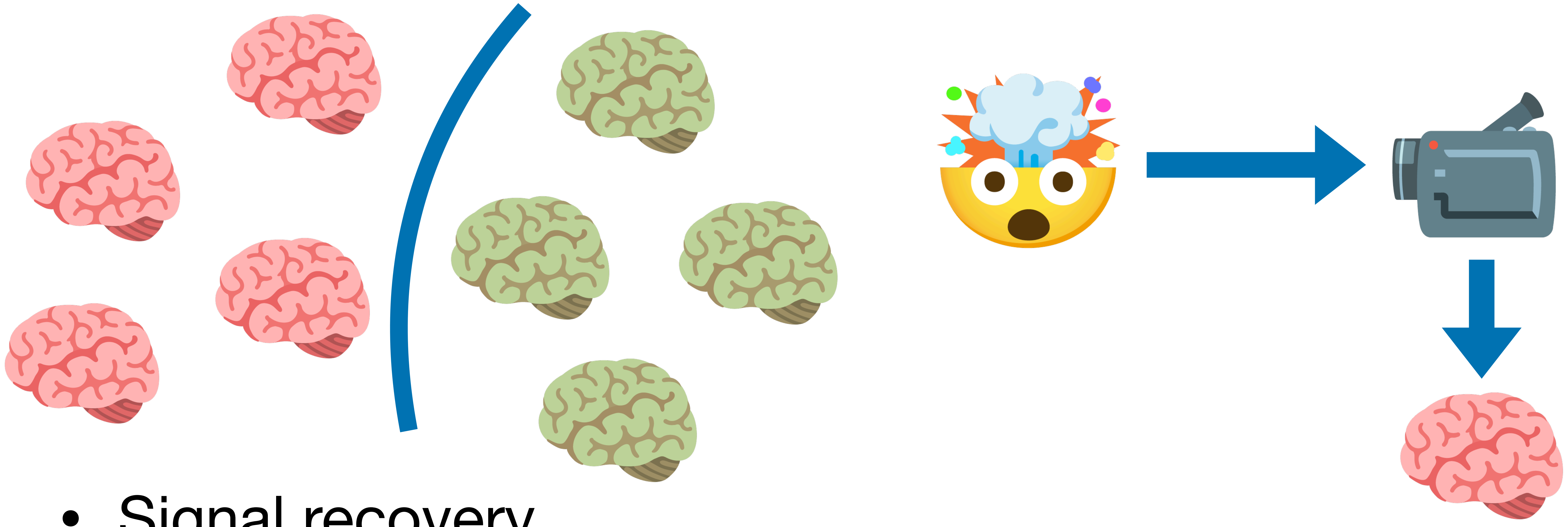
All the regular things we do with data…



- Signal recovery
- Unsupervised learning (representation learning)

What do we want to do with tensor data?

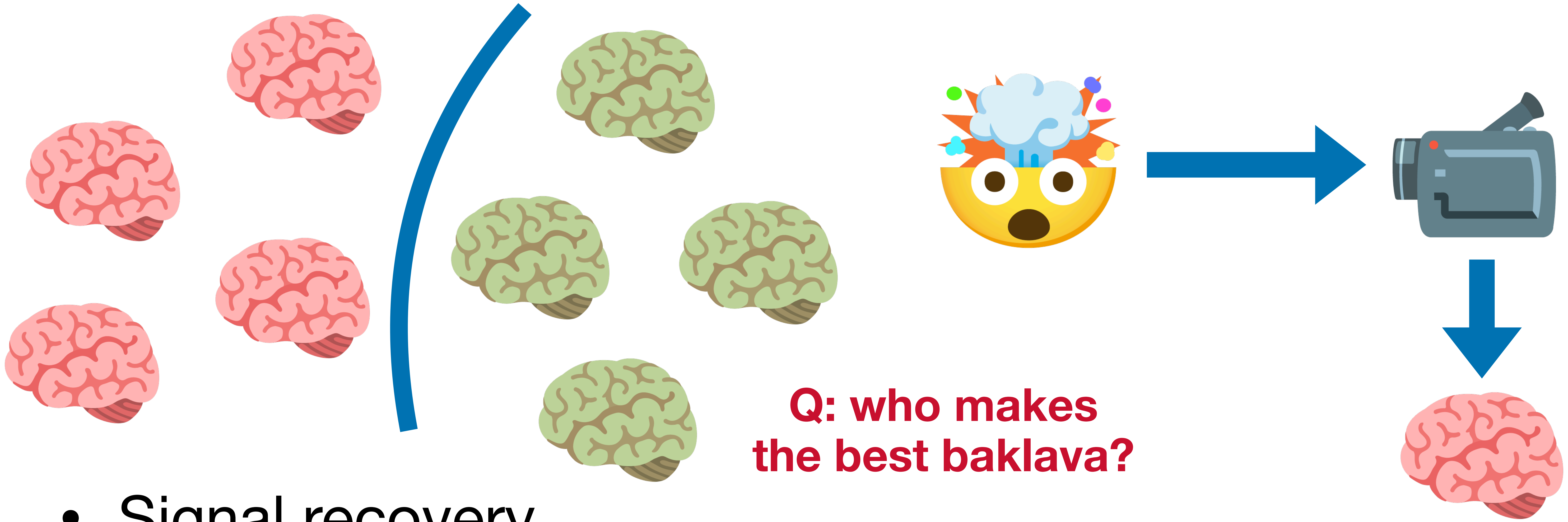
All the regular things we do with data…



- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)

What do we want to do with tensor data?

All the regular things we do with data...



- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)

What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)

What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)

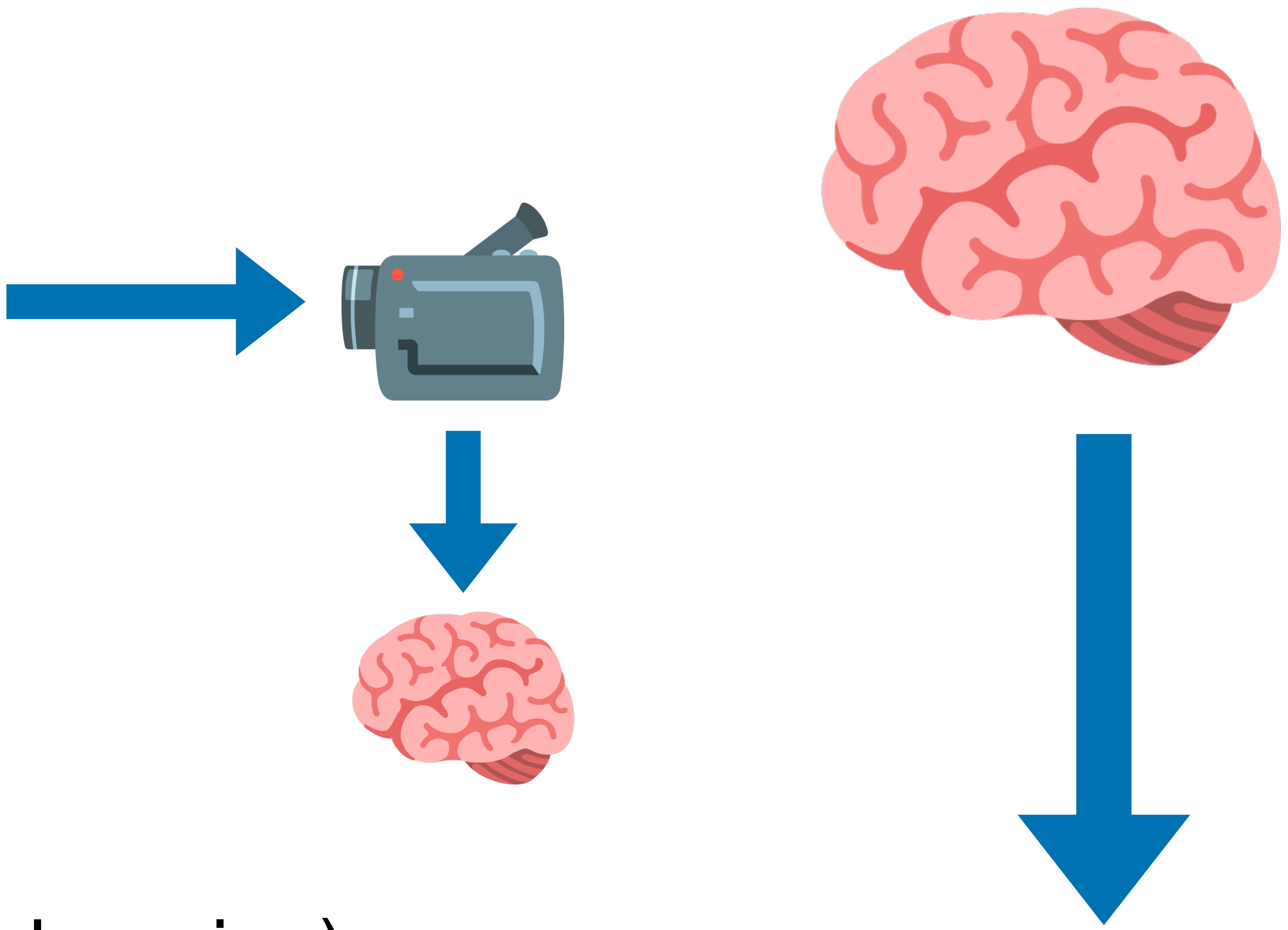
What do we want to do with tensor data?

All the regular things we do with data…

- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)
- Compression, etc…

What do we want to do with tensor data?

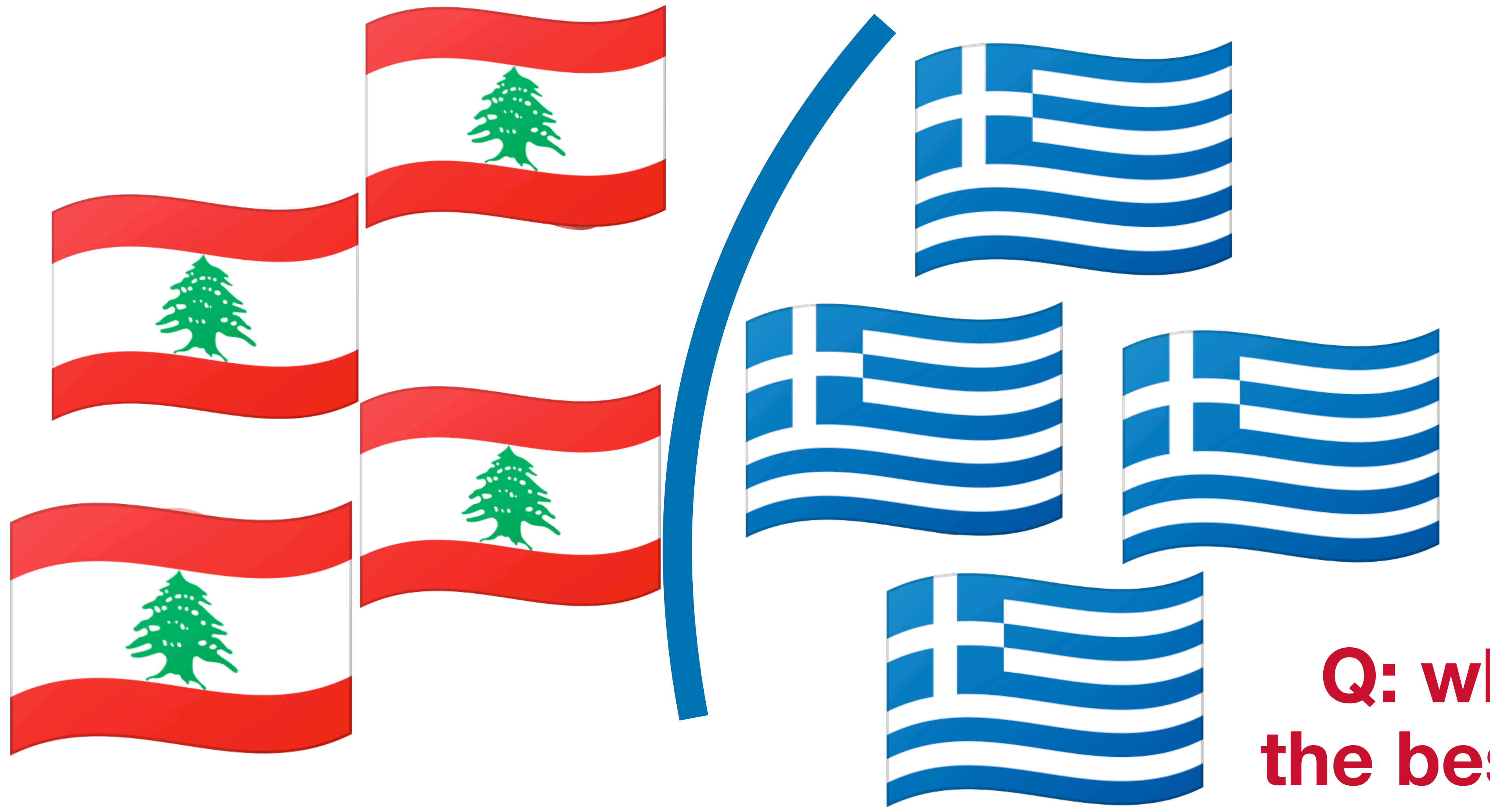
All the regular things we do with data…

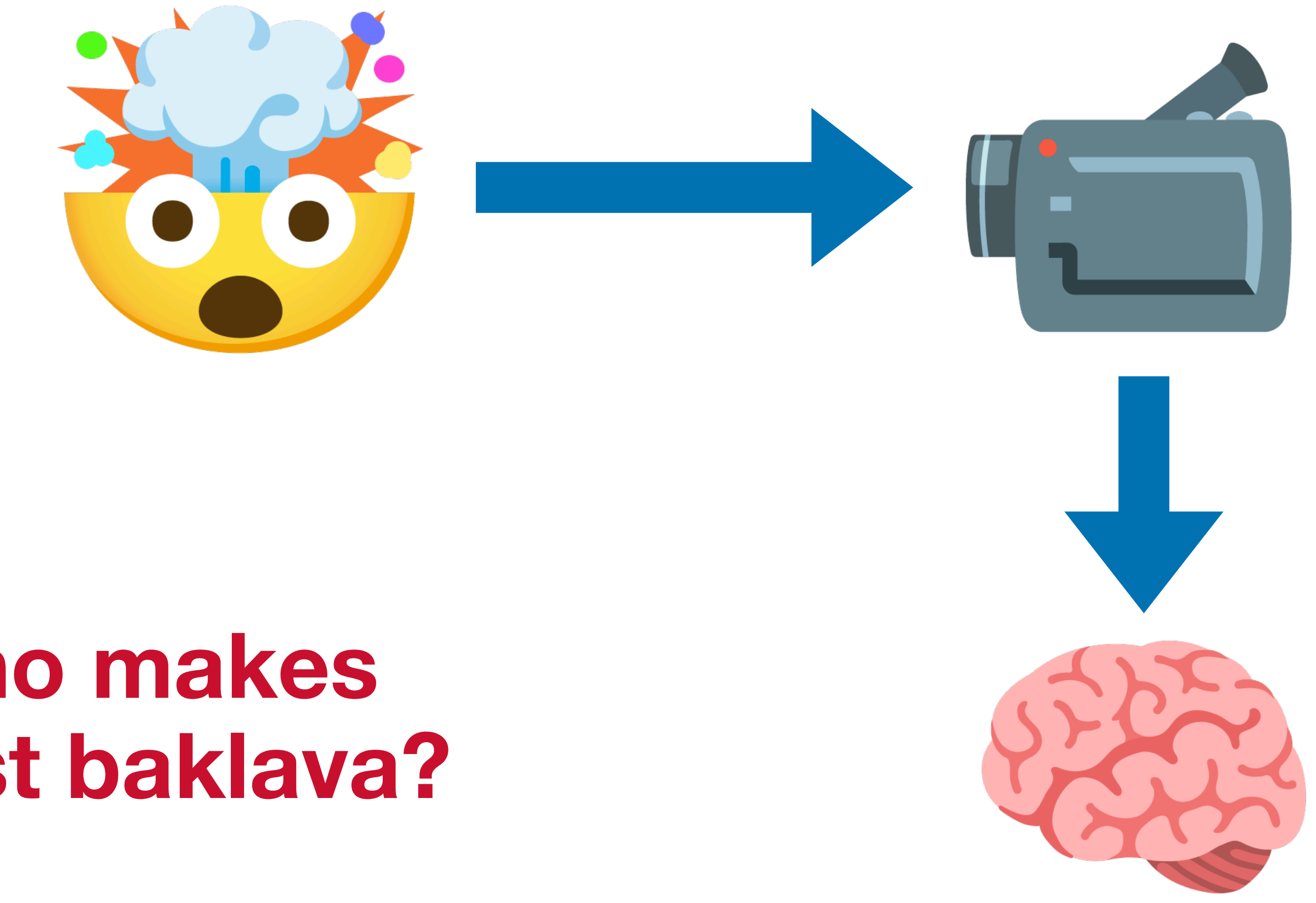


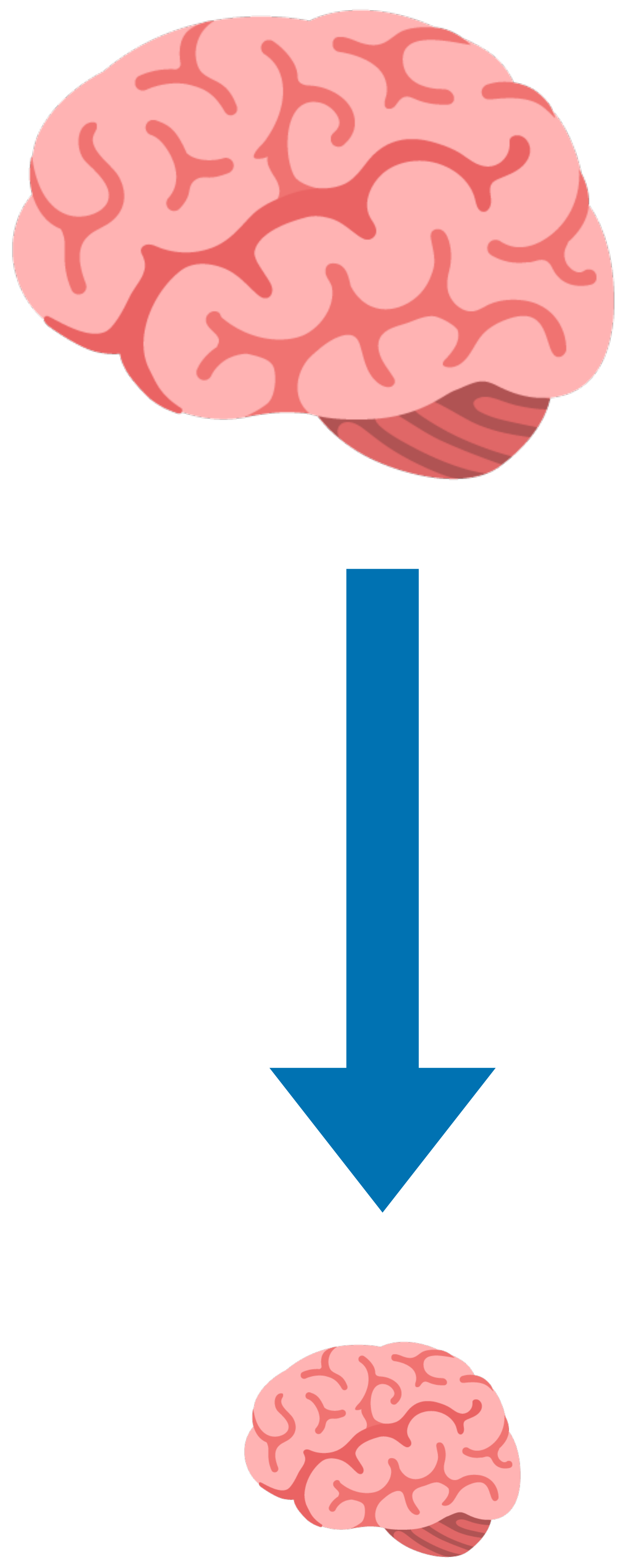
- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)
- Compression, etc…

What do we want to do with tensor data?

All the regular things we do with data…







Q: who makes
the best baklava?

- Signal recovery
- Unsupervised learning (representation learning)
- Supervised learning (prediction)
- Compression, etc…

Unsupervised learning with tensors

Using dictionary learning for sparse representation

Unsupervised learning with tensors

Using dictionary learning for sparse representation

Task: given a collection of tensors $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_n \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$, find a *dictionary* $\underline{\mathbf{d}}_1, \underline{\mathbf{d}}_2, \ldots, \underline{\mathbf{d}}_p$ such that

Unsupervised learning with tensors

Using dictionary learning for sparse representation

Task: given a collection of tensors $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_n \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$, find a *dictionary* $\underline{\mathbf{d}}_1, \underline{\mathbf{d}}_2, \ldots, \underline{\mathbf{d}}_p$ such that

$$\underline{\mathbf{Y}}_i \approx \sum_{j=1}^{p} x_{ij} \underline{\mathbf{d}}_j,$$

Unsupervised learning with tensors

Using dictionary learning for sparse representation

Task: given a collection of tensors $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_n \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$, find a *dictionary* $\underline{\mathbf{d}}_1, \underline{\mathbf{d}}_2, \ldots, \underline{\mathbf{d}}_p$ such that

$$\underline{\mathbf{Y}}_i \approx \sum_{j=1}^{p} x_{ij} \underline{\mathbf{d}}_j,$$

where each vector of coefficients $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$ is s-sparse.

Unsupervised learning with tensors

Using dictionary learning for sparse representation

Task: given a collection of tensors $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_n \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$, find a *dictionary* $\underline{\mathbf{d}}_1, \underline{\mathbf{d}}_2, \ldots, \underline{\mathbf{d}}_p$ such that

$$\underline{\mathbf{Y}}_i \approx \sum_{j=1}^{p} x_{ij} \underline{\mathbf{d}}_j,$$

where each vector of coefficients $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$ is s-sparse.

Application: processing or storing hyperspectral images acquired from a drone.

Supervised learning with tensors

Regression with tensor-valued covariates

Supervised learning with tensors

Regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

Supervised learning with tensors

Regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

$$y_i \approx \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + \text{noise},$$

Supervised learning with tensors

Regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

$$y_i \approx \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + \text{noise},$$

where $\langle \cdot , \cdot \rangle$ is the element-wise inner product.

Supervised learning with tensors

Regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

$$y_i \approx \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + \text{noise},$$

where $\langle \cdot , \cdot \rangle$ is the element-wise inner product.

Application: predicting a brain health condition from an MRI scan.

Supervised learning with tensors

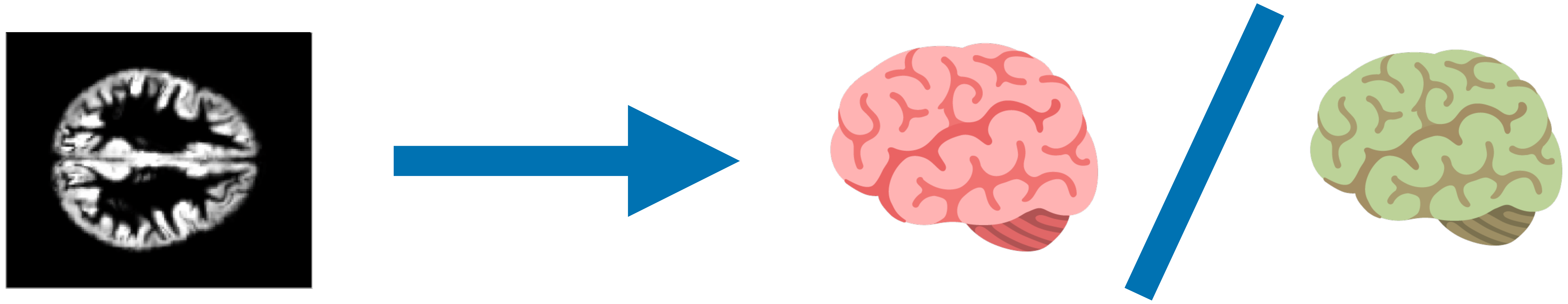
Regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

$$y_i \approx \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + \text{noise},$$

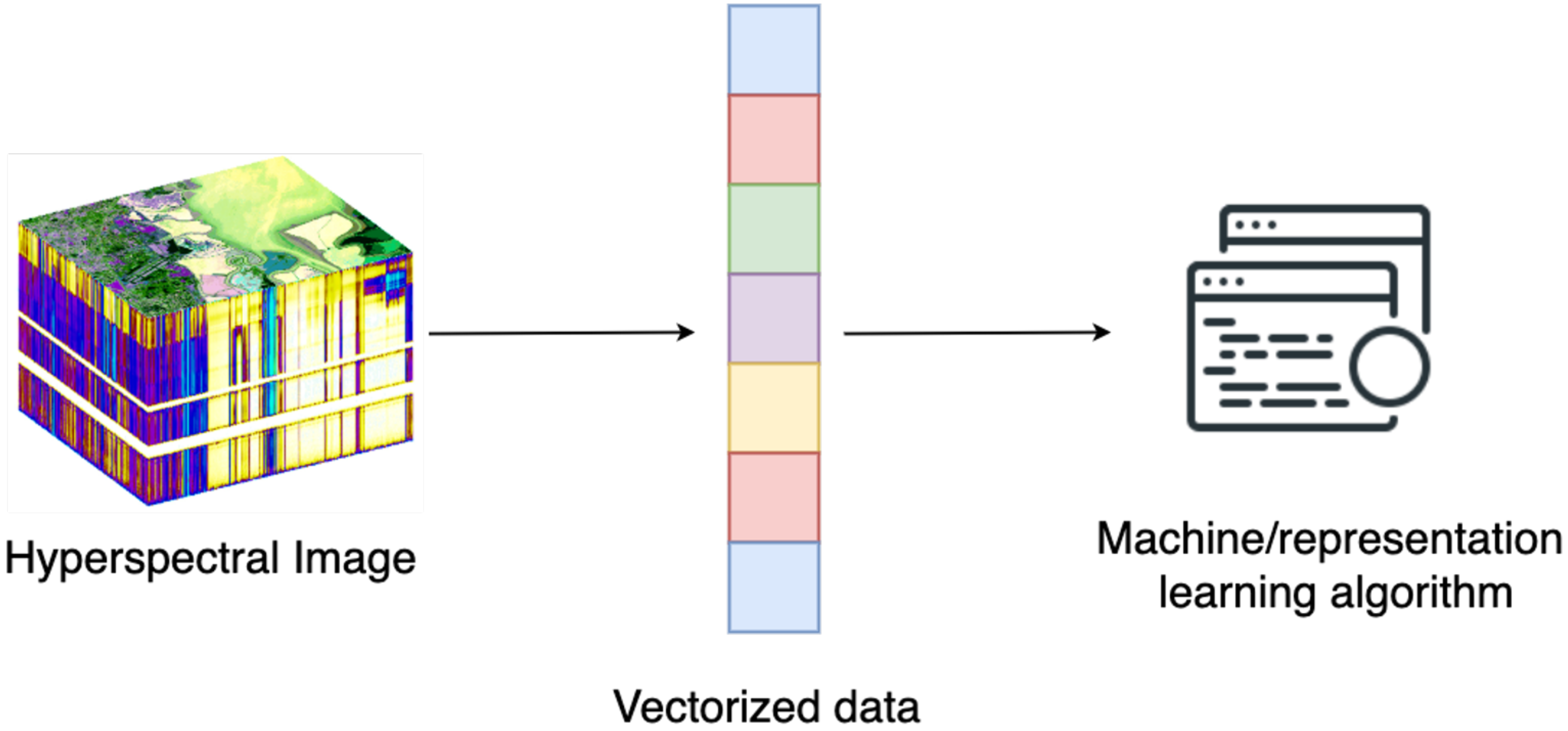
where $\langle \cdot\, , \cdot \rangle$ is the element-wise inner product.

Application: predicting a brain health condition from an MRI scan.



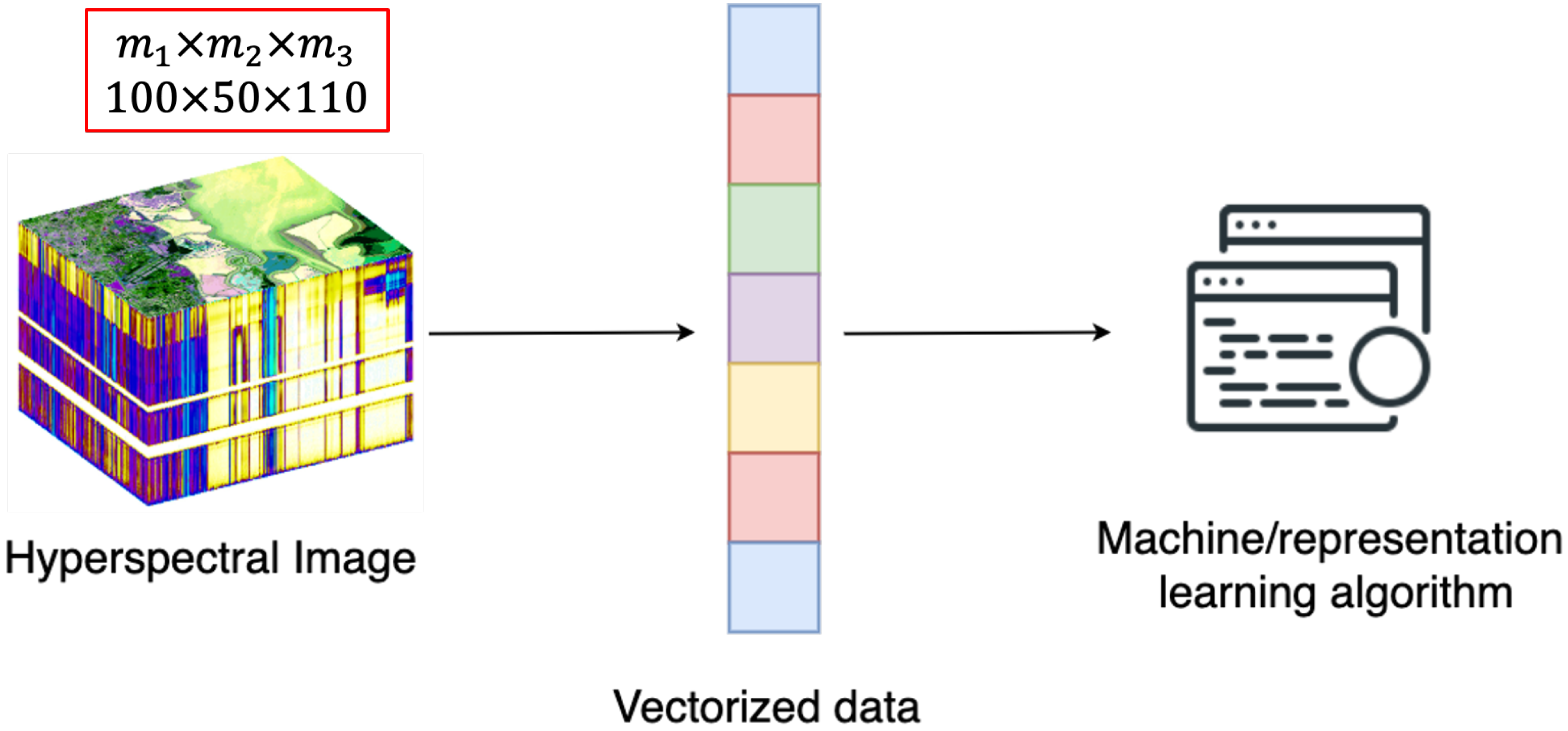
A baseline approach: vectorization

We can always throw away the structure



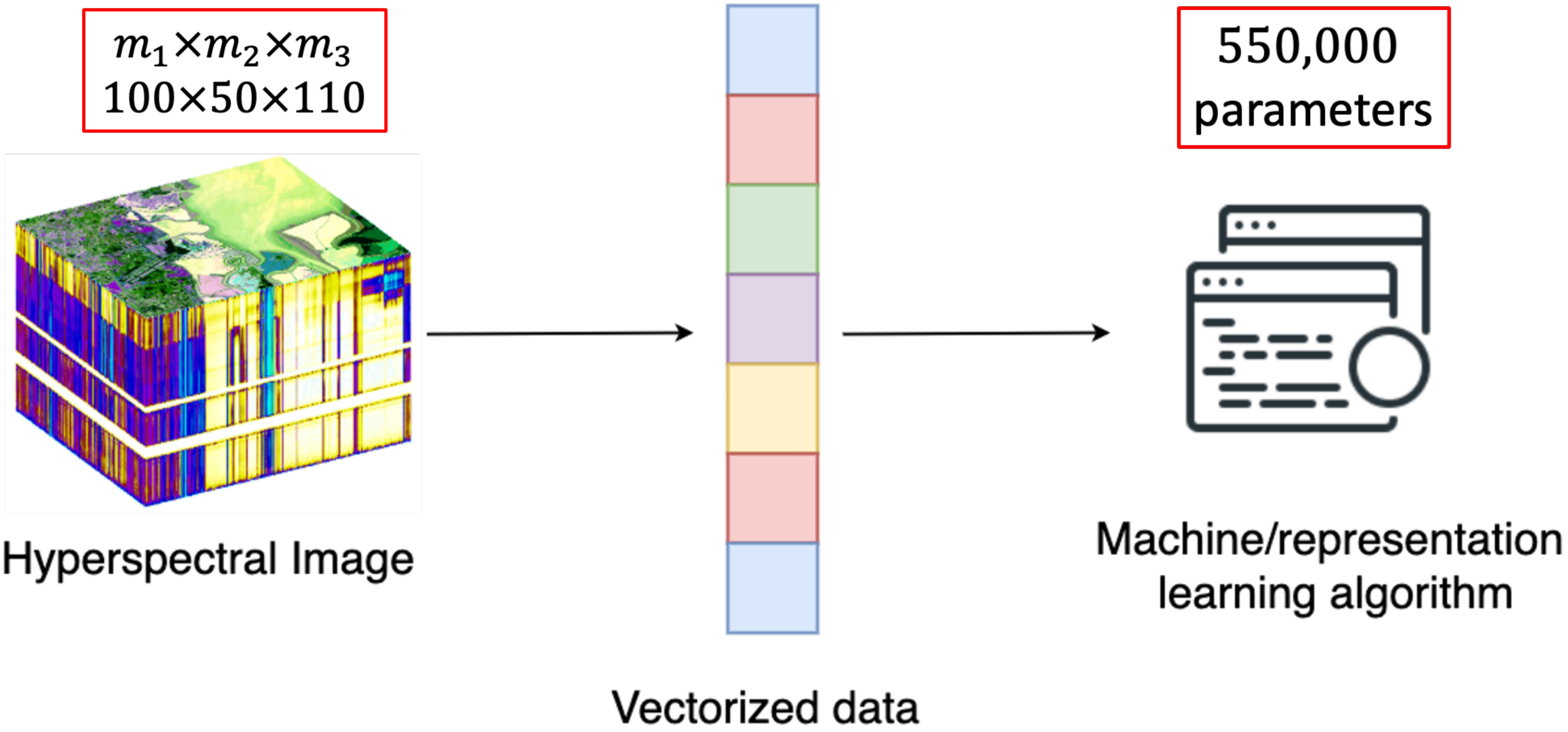
A baseline approach: vectorization

We can always throw away the structure



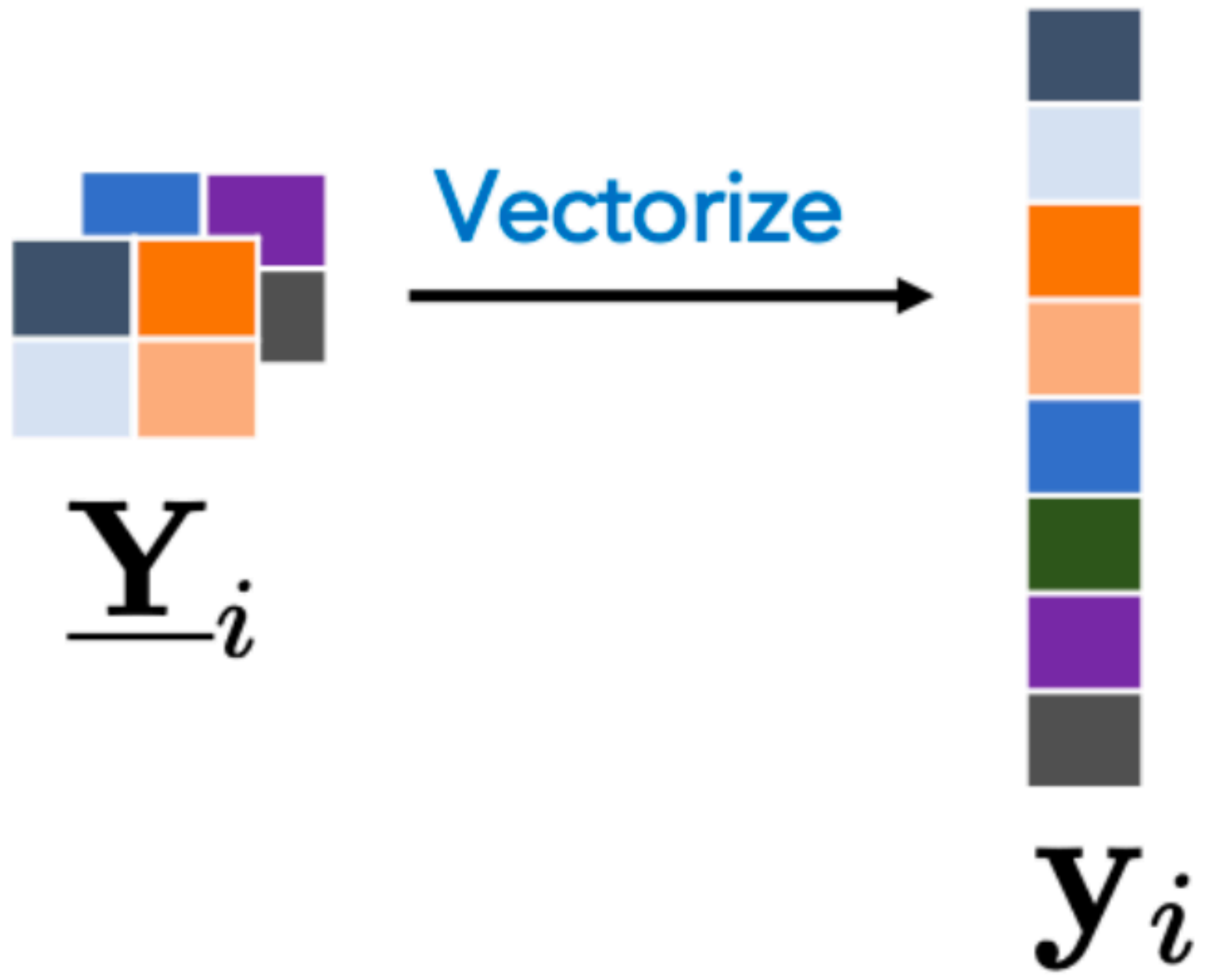
A baseline approach: vectorization

We can always throw away the structure



Why (and why not) vectorize?

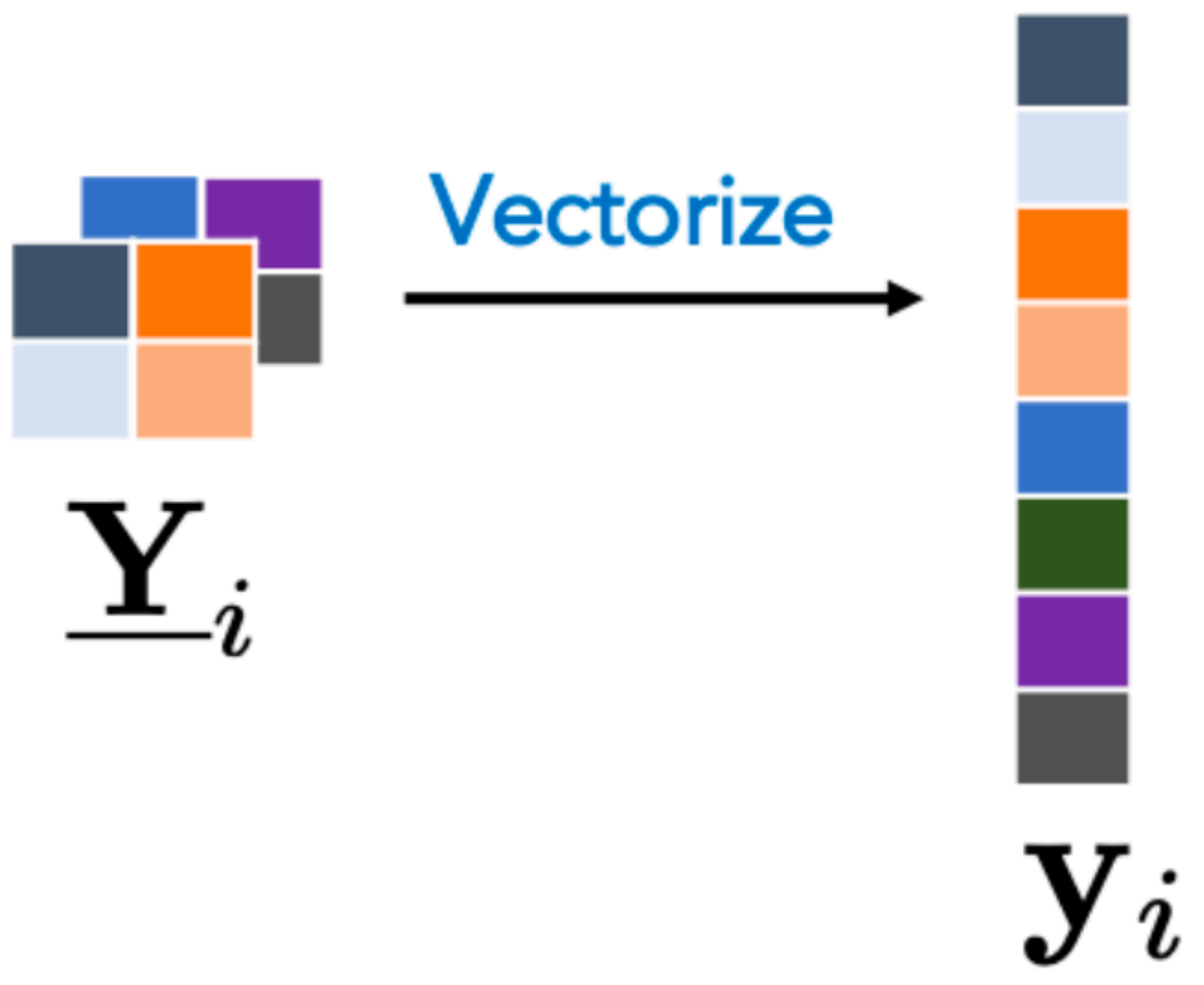
The problems with vectorization



$\underline{\mathbf{Y}}_i$ $\mathbf{y}_i$

Why (and why not) vectorize?

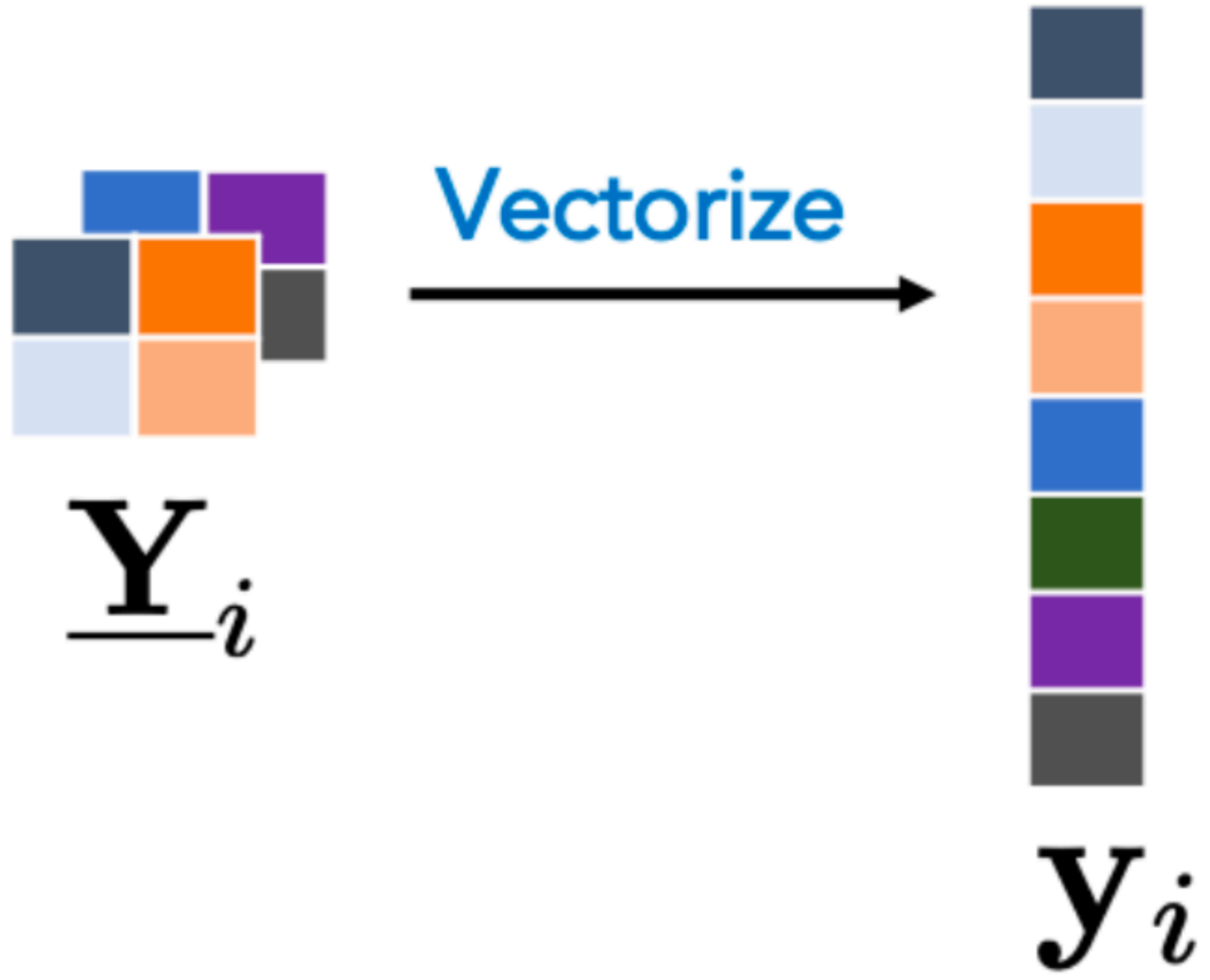
The problems with vectorization



1. Vectorization *ignores the tensor structure*.

Why (and why not) vectorize?

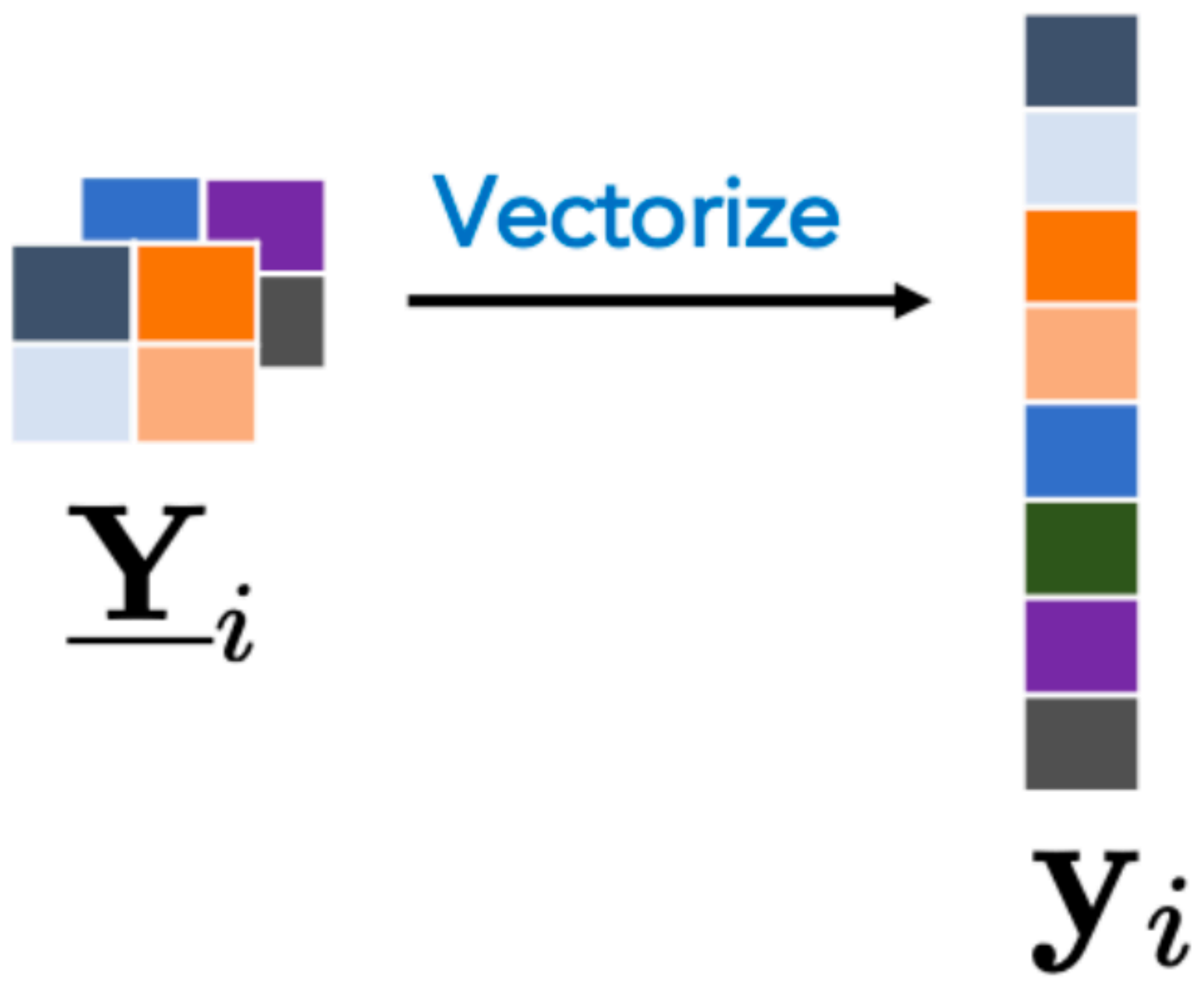
The problems with vectorization



1. Vectorization *ignores the tensor structure*.
2. Resulting problems have very high dimension.

Why (and why not) vectorize?

The problems with vectorization



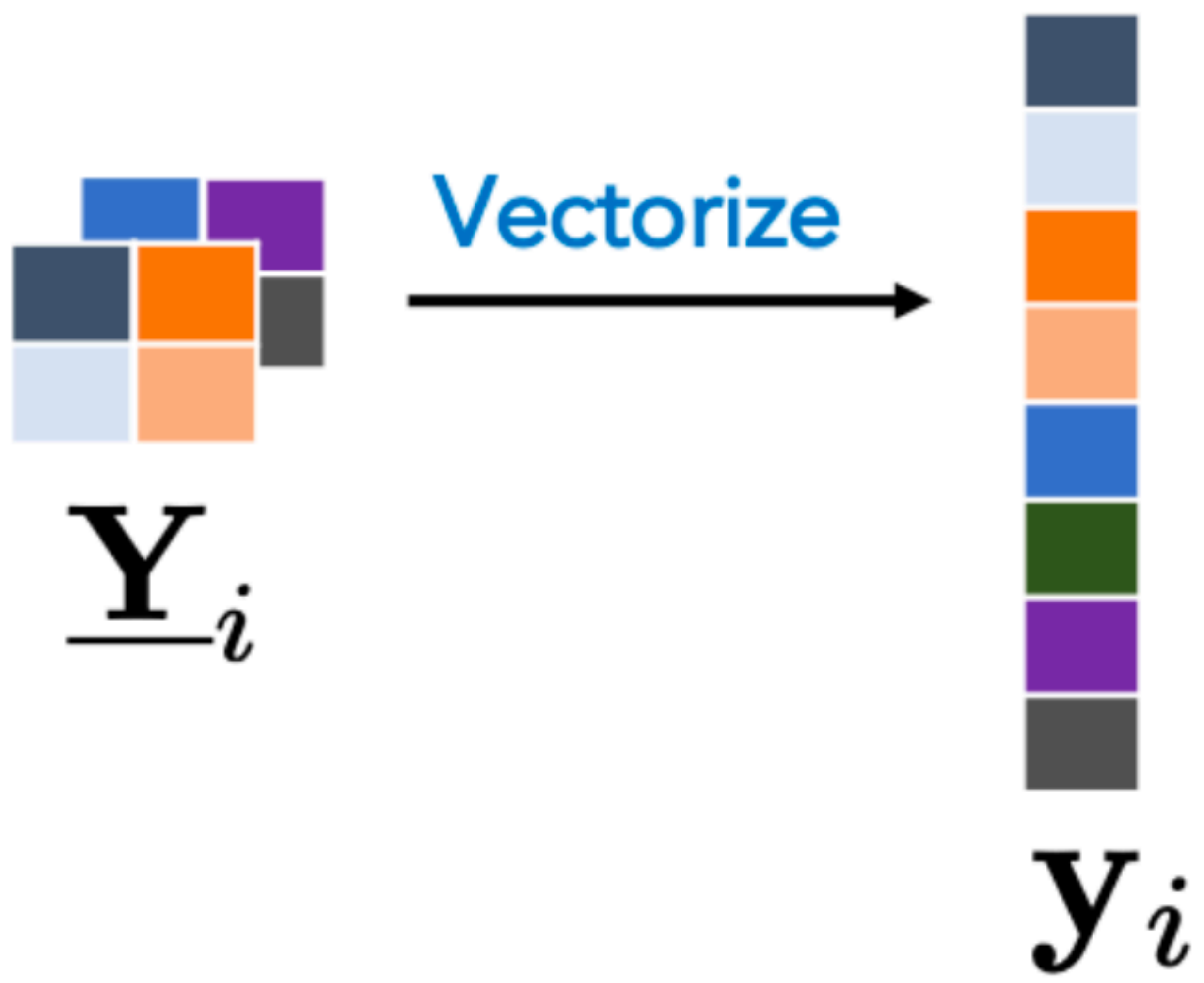
1. Vectorization *ignores the tensor structure*.

2. Resulting problems have very high dimension.

Example: ADHD200 data set has fMRI images of children's brains.

Why (and why not) vectorize?

The problems with vectorization



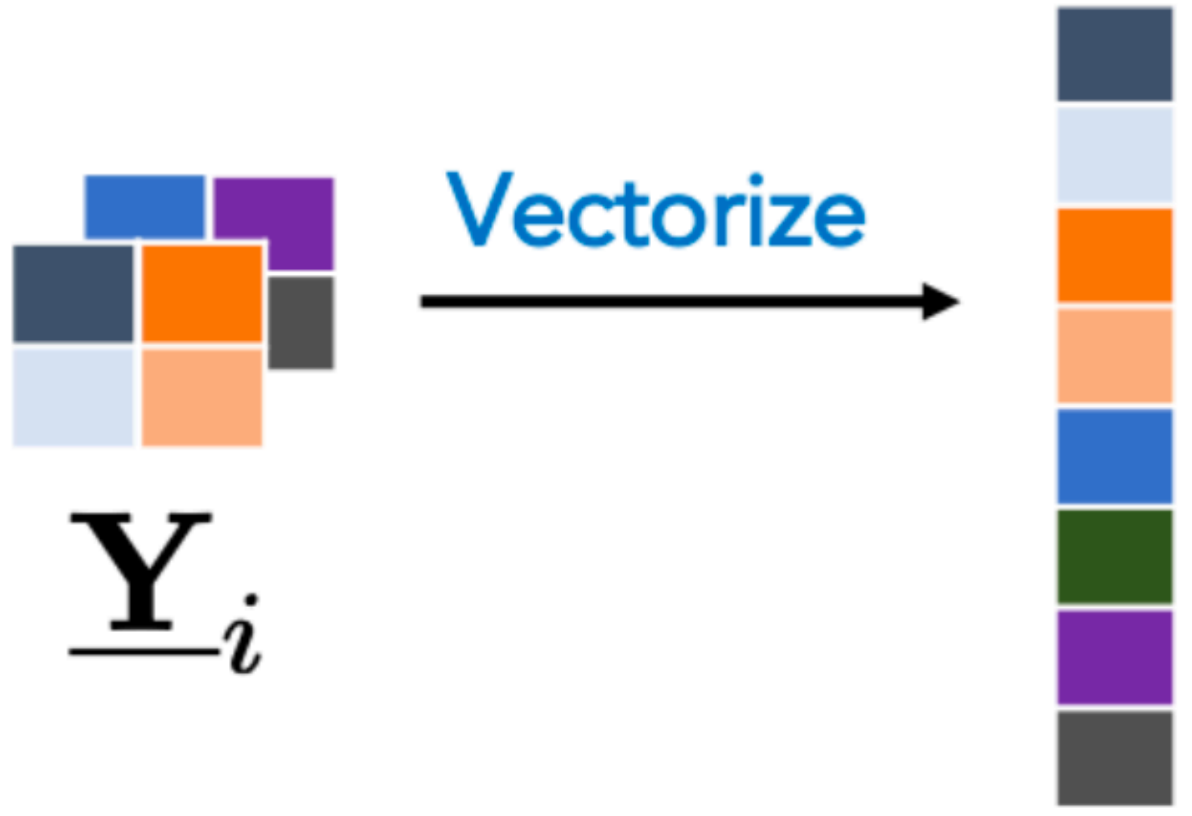
1. Vectorization *ignores the tensor structure*.
2. Resulting problems have very high dimension.

Example: ADHD200 data set has fMRI images of children's brains.

- fMRI data: 121 x 141 x 121 tensor

Why (and why not) vectorize?

The problems with vectorization



$\underline{\mathbf{Y}}_i$ $\mathbf{y}_i$

1. Vectorization *ignores the tensor structure*.

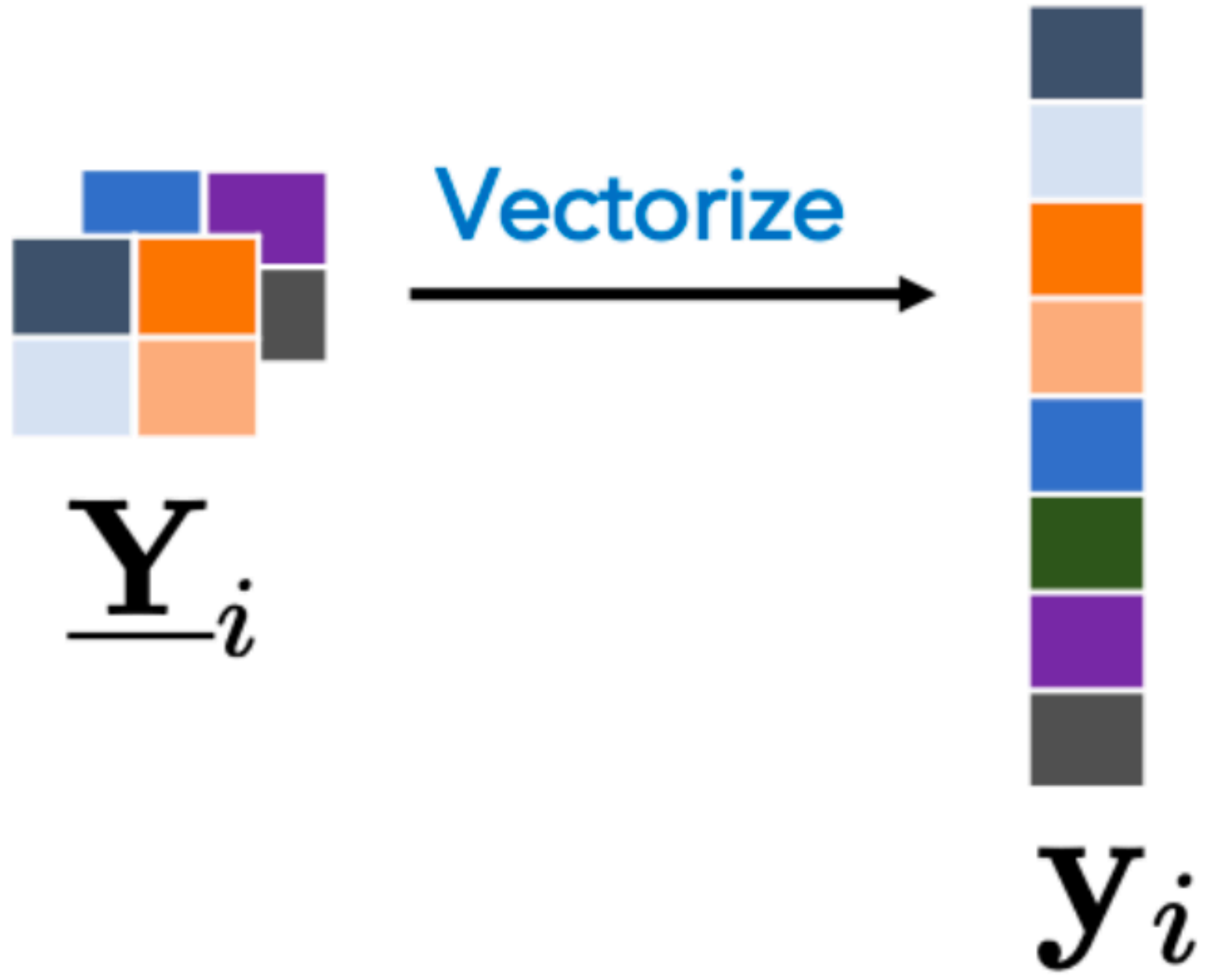
2. Resulting problems have very high dimension.

Example: ADHD200 data set has fMRI images of children's brains.

- fMRI data: 121 x 141 x 121 tensor
- After vectorizing: 2,122,945 dimensional vector

Why (and why not) vectorize?

The problems with vectorization



1. Vectorization *ignores the tensor structure*.
2. Resulting problems have very high dimension.

Example: ADHD200 data set has fMRI images of children's brains.

- fMRI data: 121 x 141 x 121 tensor
- After vectorizing: 2,122,945 dimensional vector
- Sample size: 959 total images

Dealing with overparameterization

This is not just a problem with tensors!

Dealing with overparameterization

This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:

Dealing with overparameterization

This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:

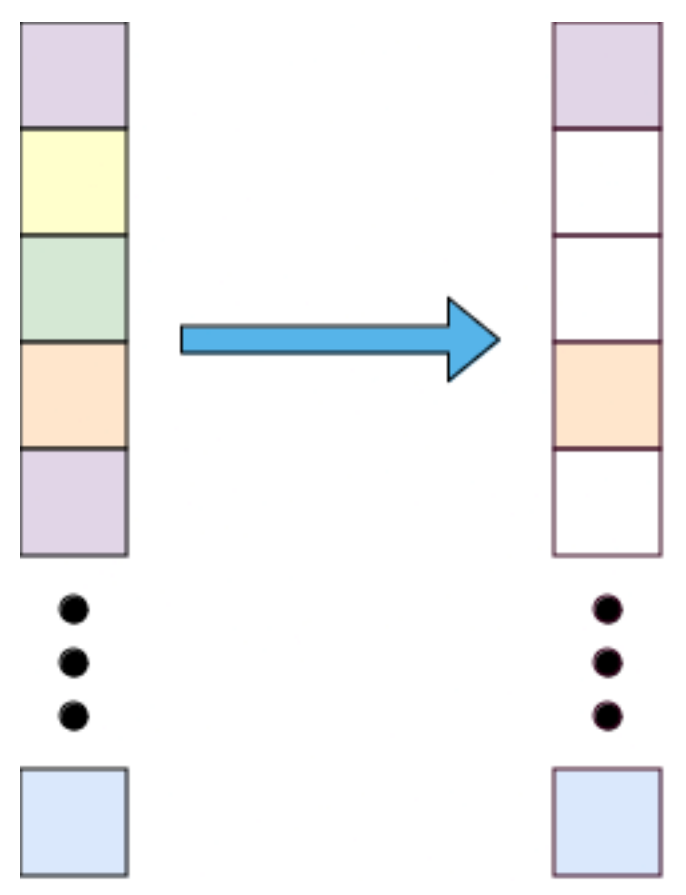
$$y_i = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + z_i$$

Dealing with overparameterization

This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:

$$y_i = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + z_i$$



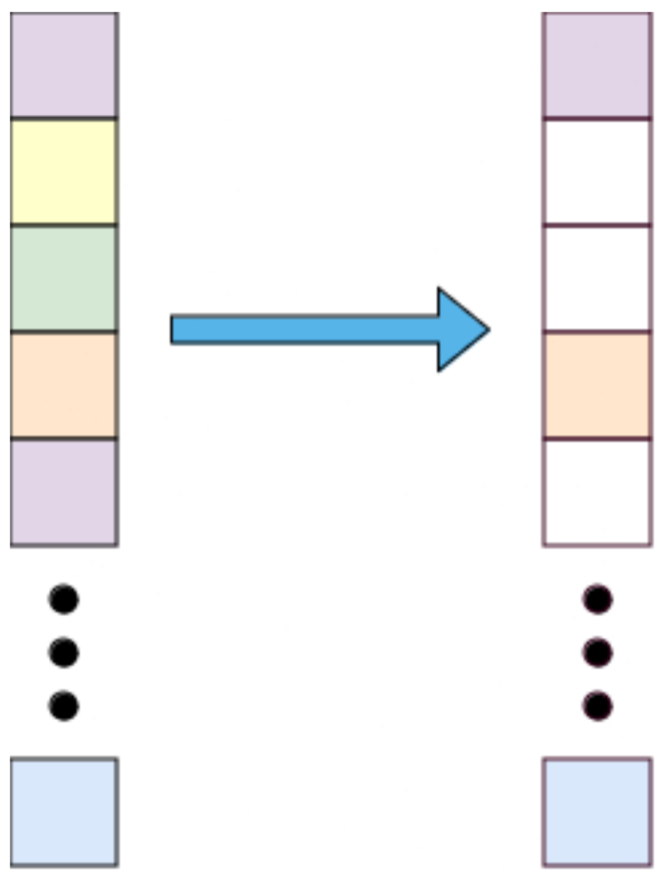
- **Vectors:** model $\underline{\mathbf{B}}$ as *sparse.*

Dealing with overparameterization

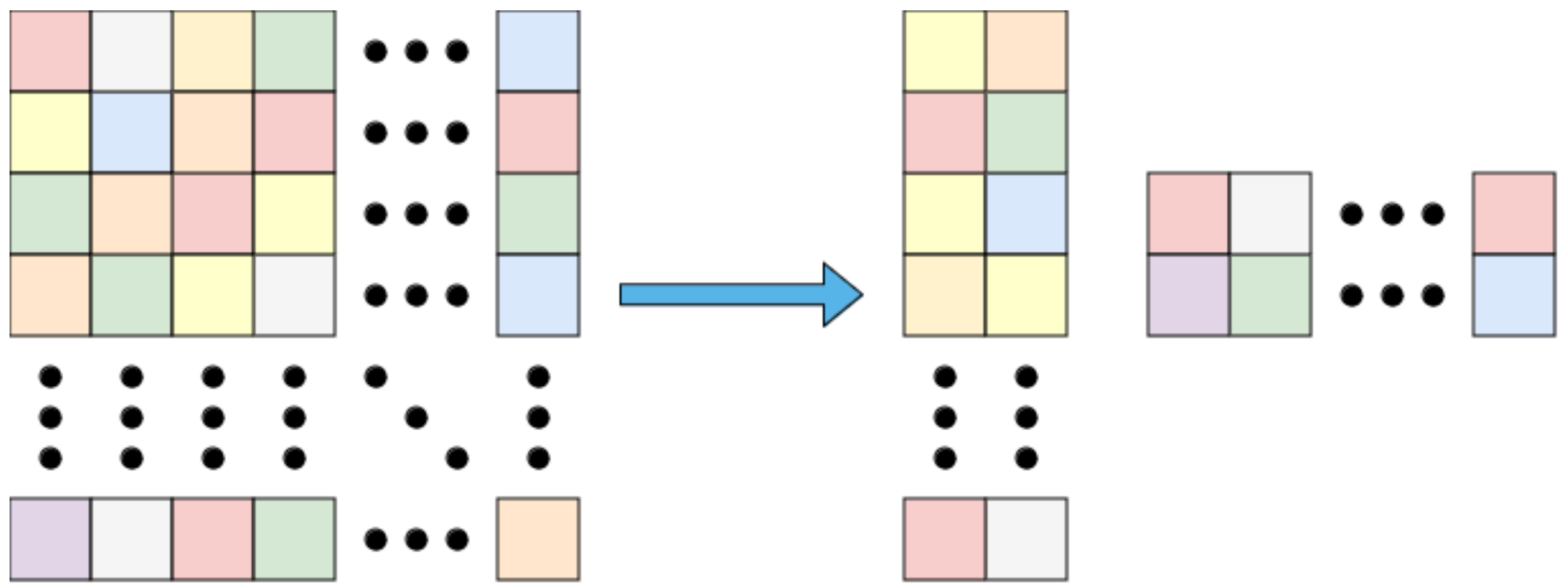
This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:

$$y_i = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + z_i$$



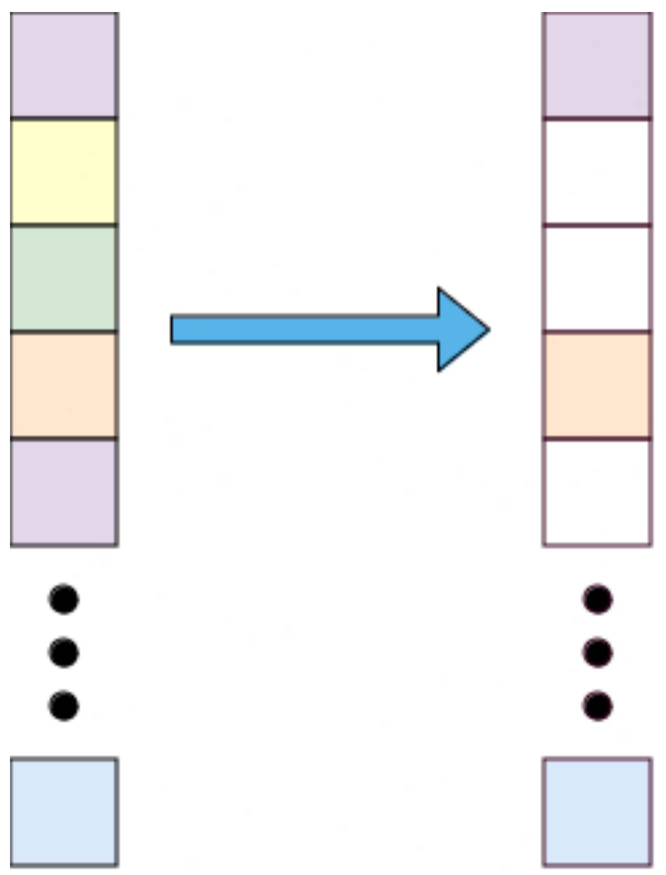
- **Vectors:** model $\underline{\mathbf{B}}$ as *sparse.*
- **Matrices:** model $\underline{\mathbf{B}}$ as *low rank.*



Dealing with overparameterization

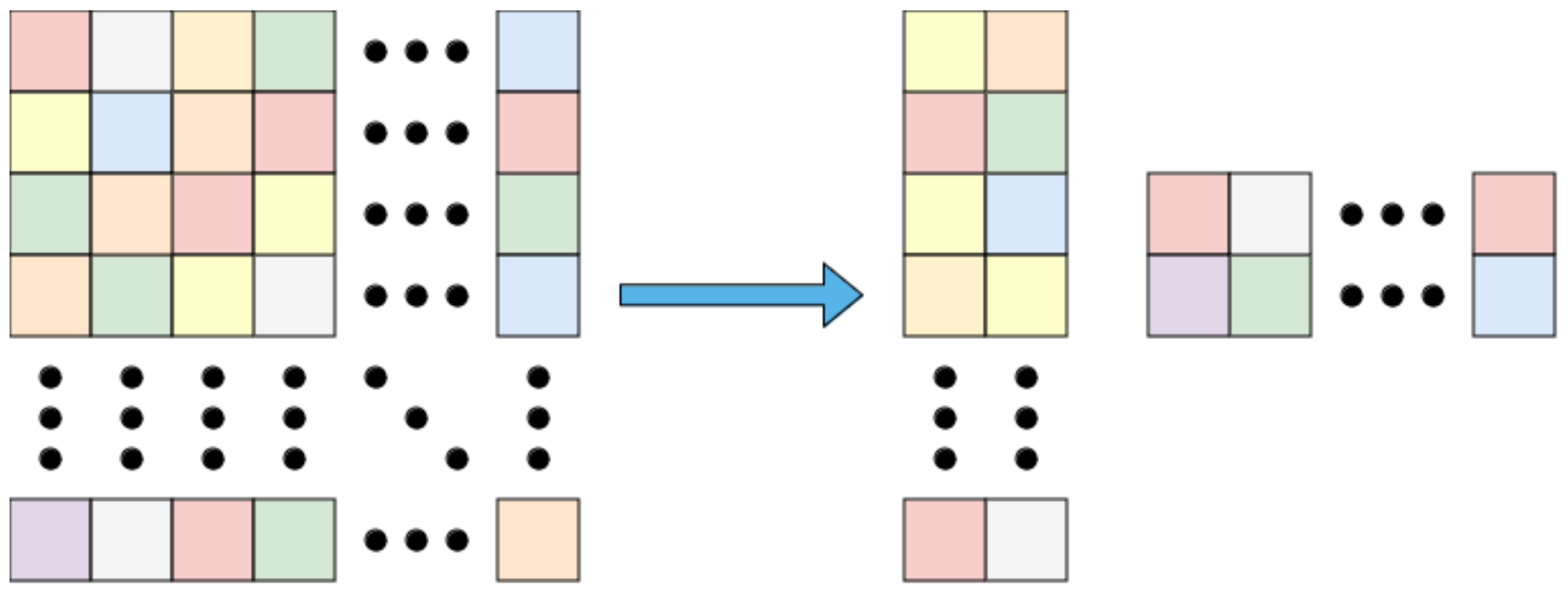
This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:



$$y_i = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + z_i$$

- **Vectors:** model $\underline{\mathbf{B}}$ as *sparse.*
- **Matrices:** model $\underline{\mathbf{B}}$ as *low rank.*



How do we impose structure on tensors?

What's in this talk

A preview of the rest of the talk

1. Tensor decompositions and where to find them
2. Regression with tensor-valued data and parameters
3. Dictionary learning with structured tensors
4. Some pointers to future directions

Tensor decompositions

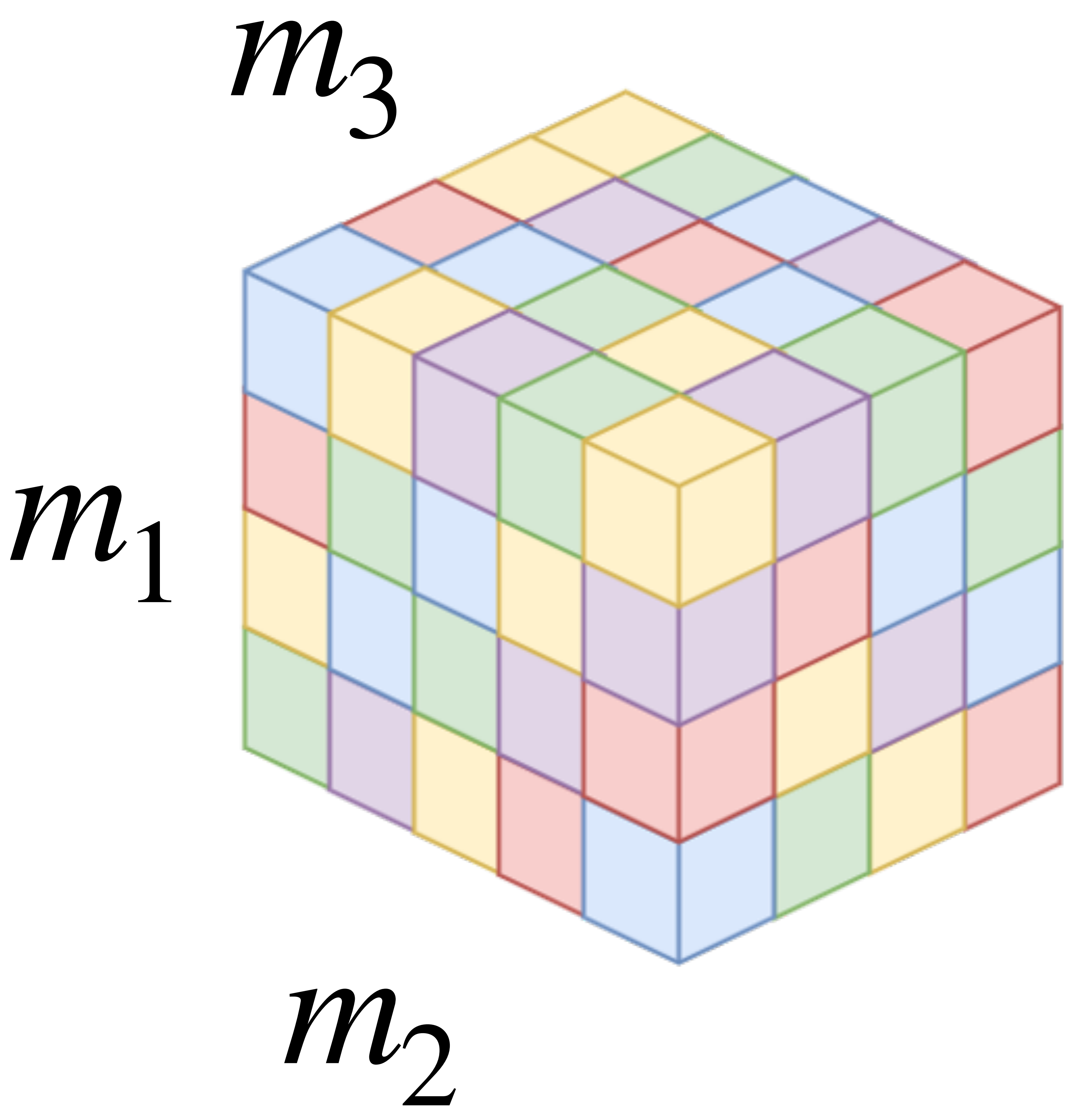
Some tensor terminology

A little jargon is unavoidable…

Kolda and Bader (2019)

Some tensor terminology

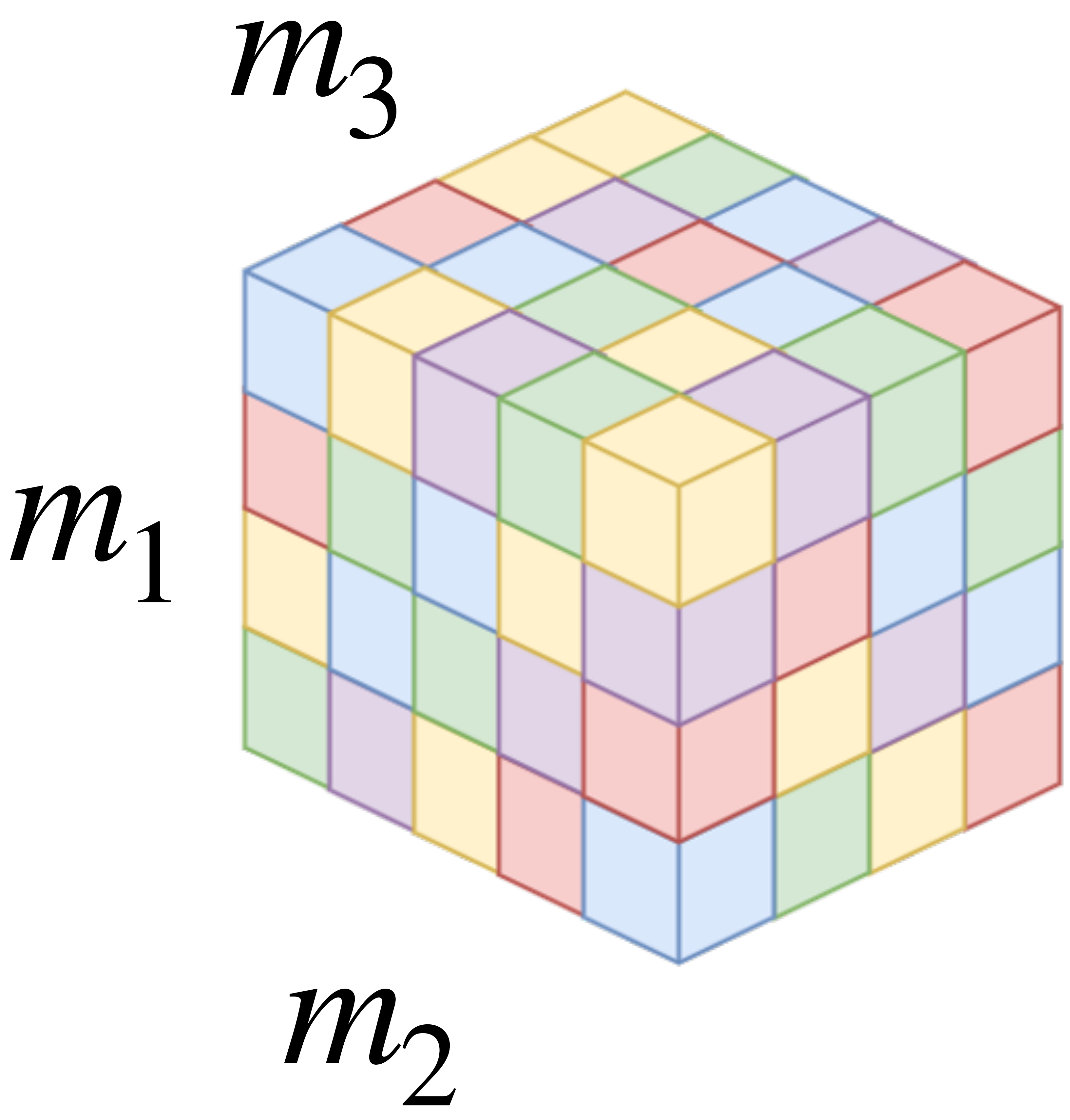
A little jargon is unavoidable…



Kolda and Bader (2019)

Some tensor terminology

A little jargon is unavoidable…

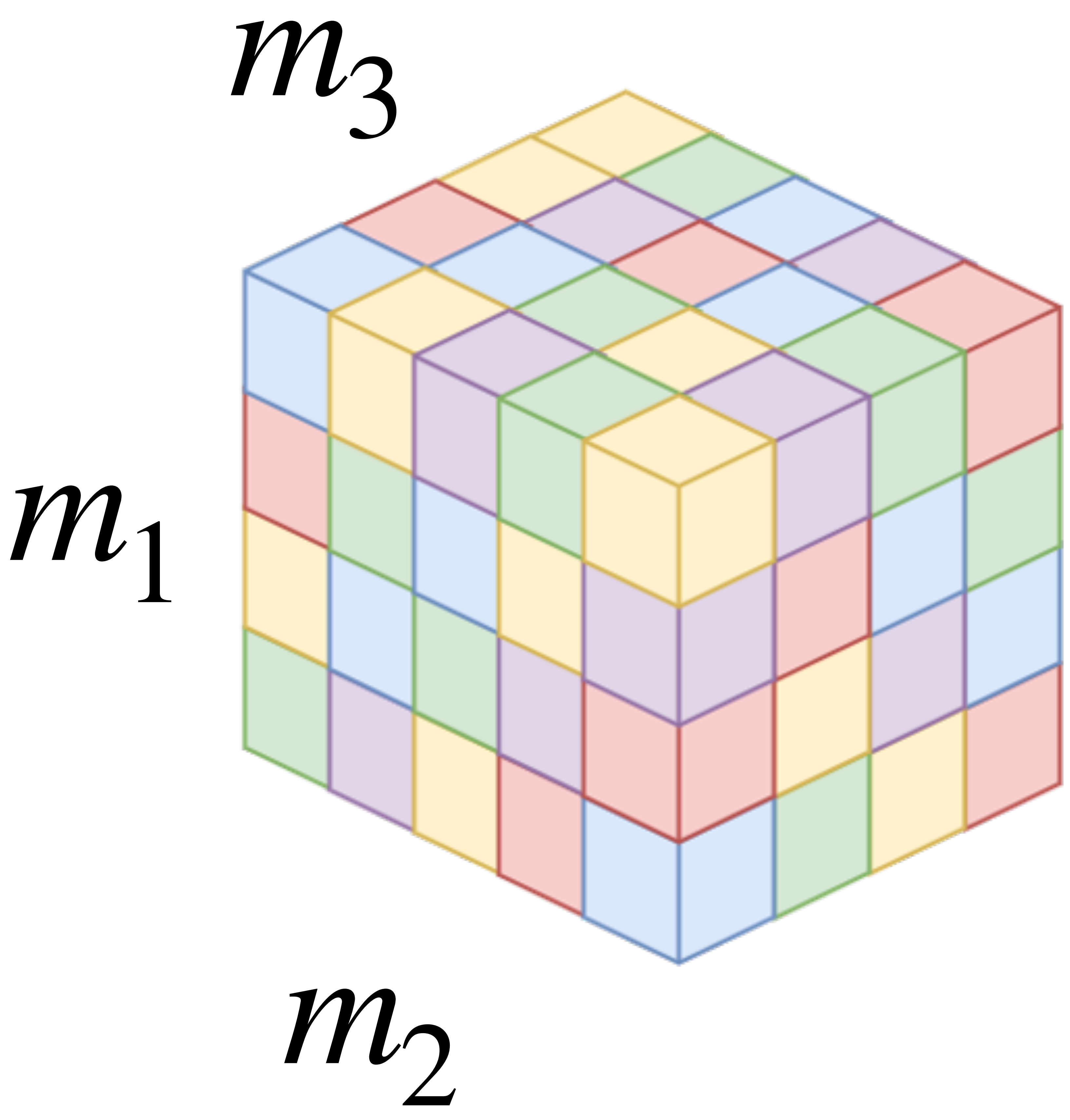


- **Mode:** each coordinate index
- **Order:** the number of modes of the tensor

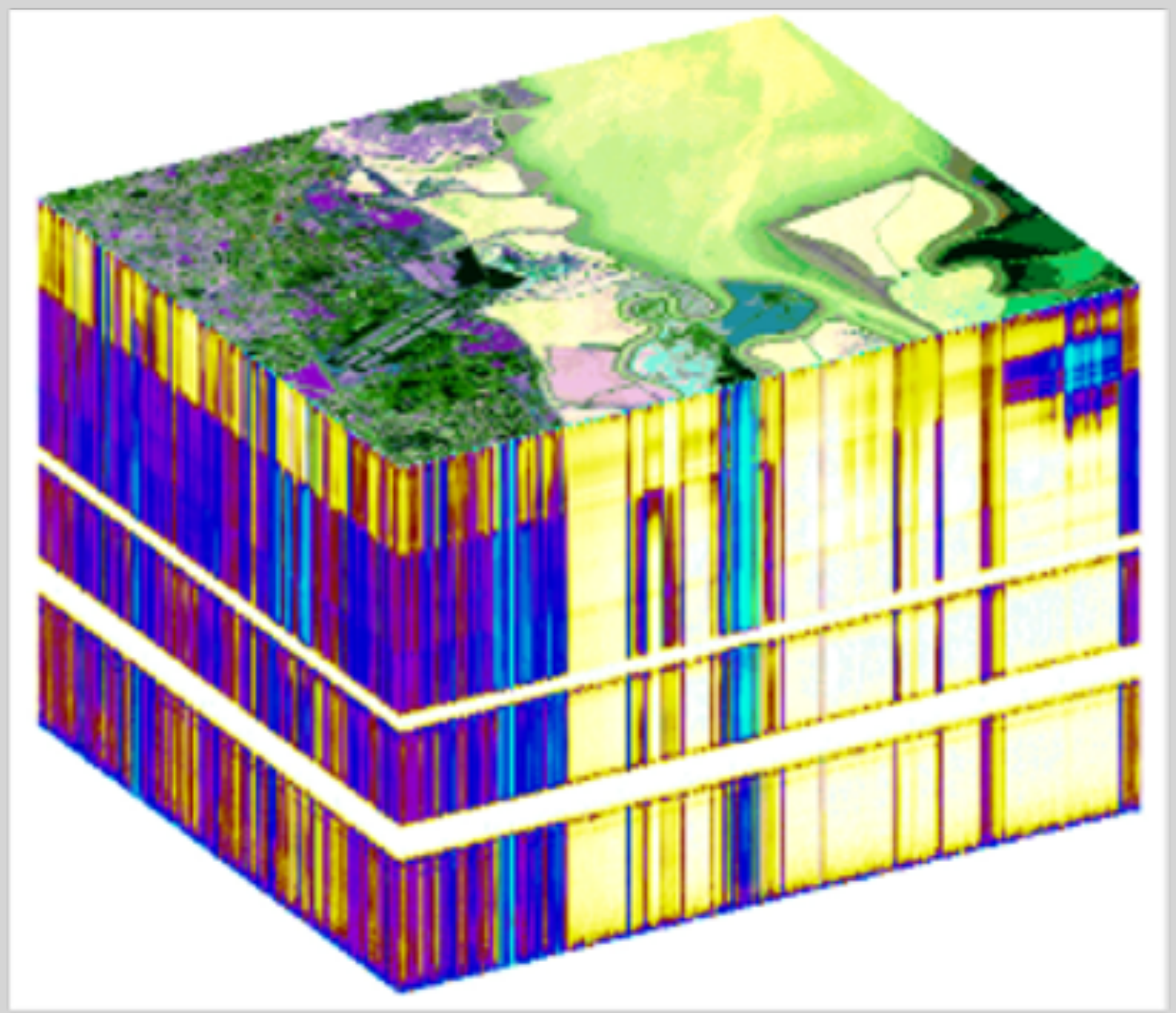
Kolda and Bader (2019)

Some tensor terminology

A little jargon is unavoidable…



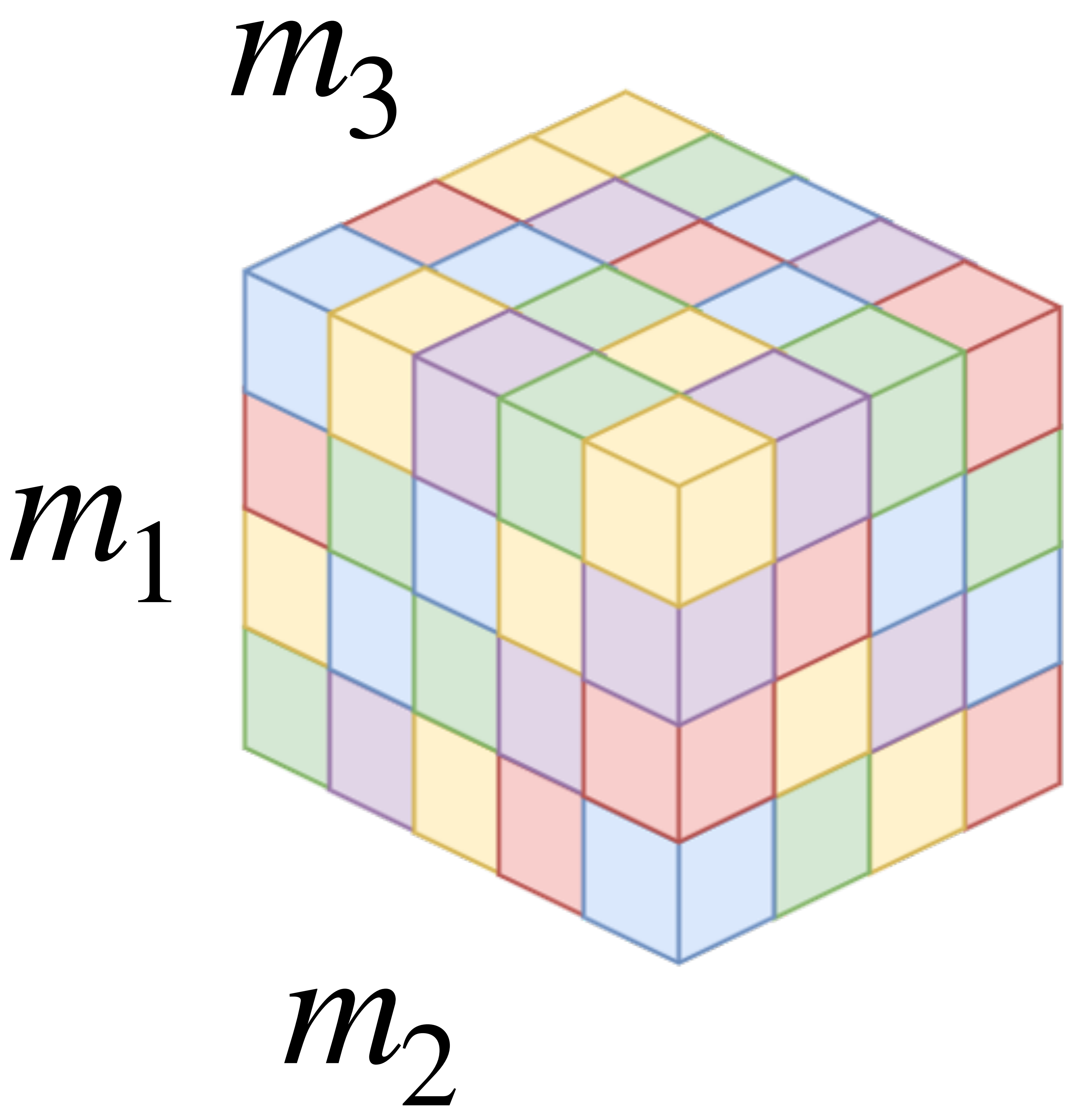
- **Mode:** each coordinate index
- **Order:** the number of modes of the tensor



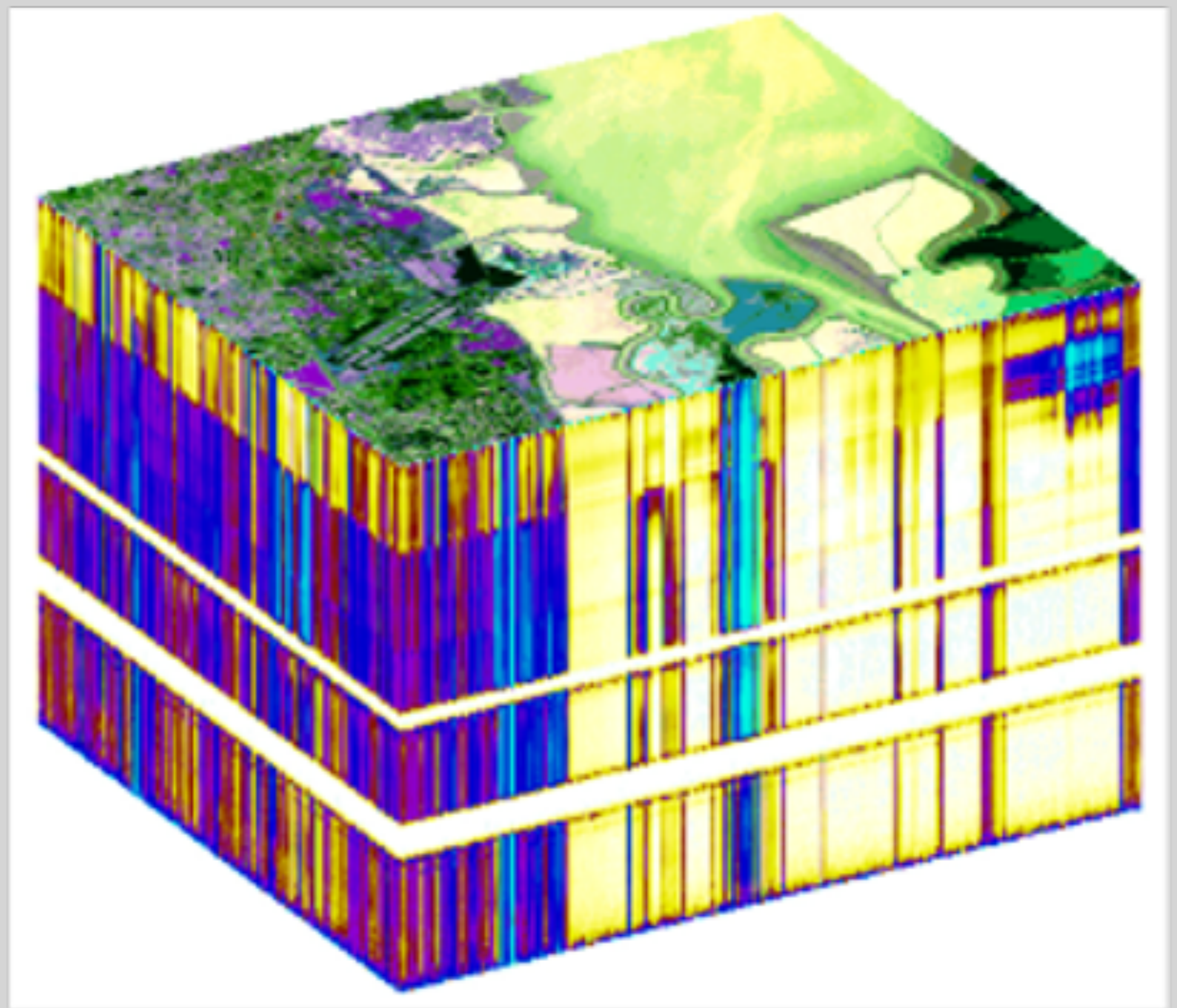
Kolda and Bader (2019)

Some tensor terminology

A little jargon is unavoidable…



- **Mode:** each coordinate index
- **Order:** the number of modes of the tensor



- Mode 1 = spectrum
- Mode 2 = longitude
- Mode 3 = latitude

Kolda and Bader (2019)

Matrix-tensor products

Mode-wise products

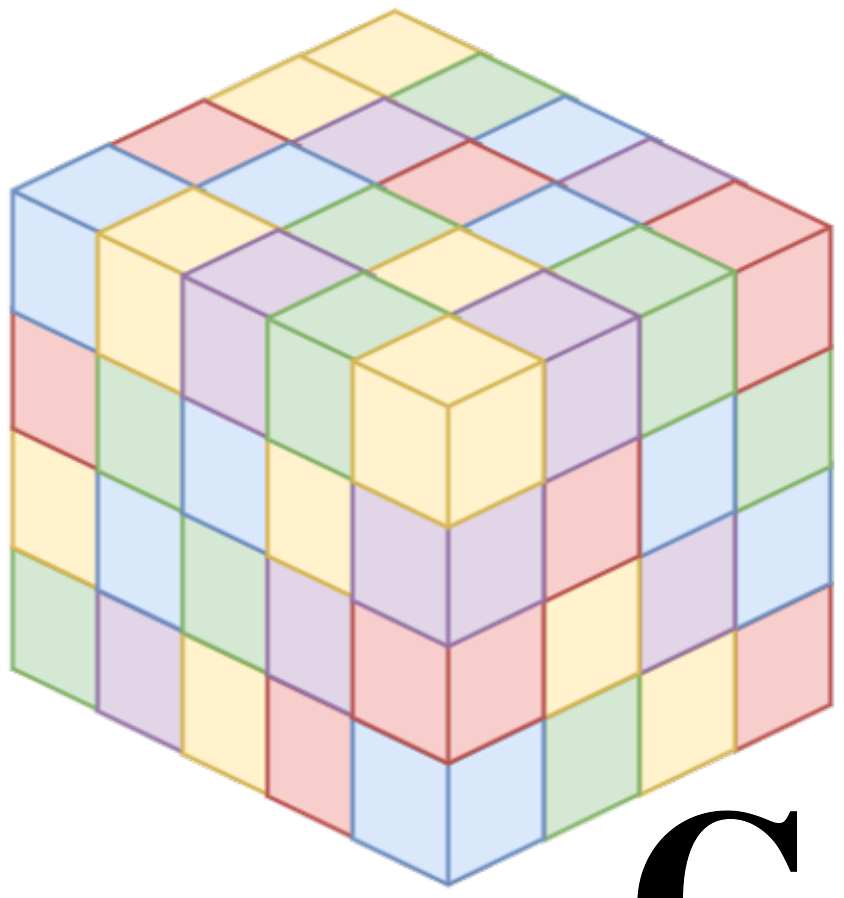
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



$\underline{\mathbf{G}}$

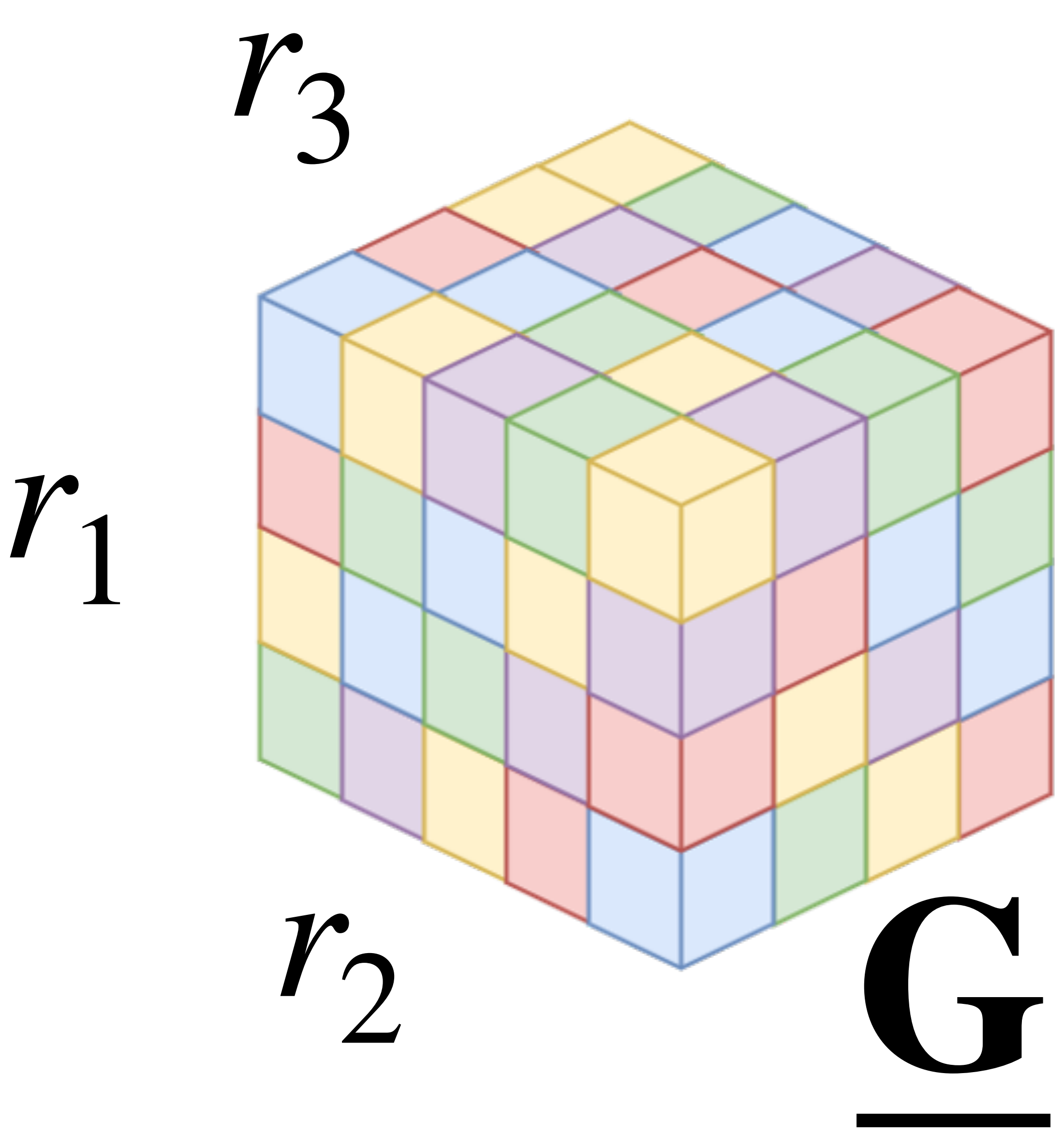
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



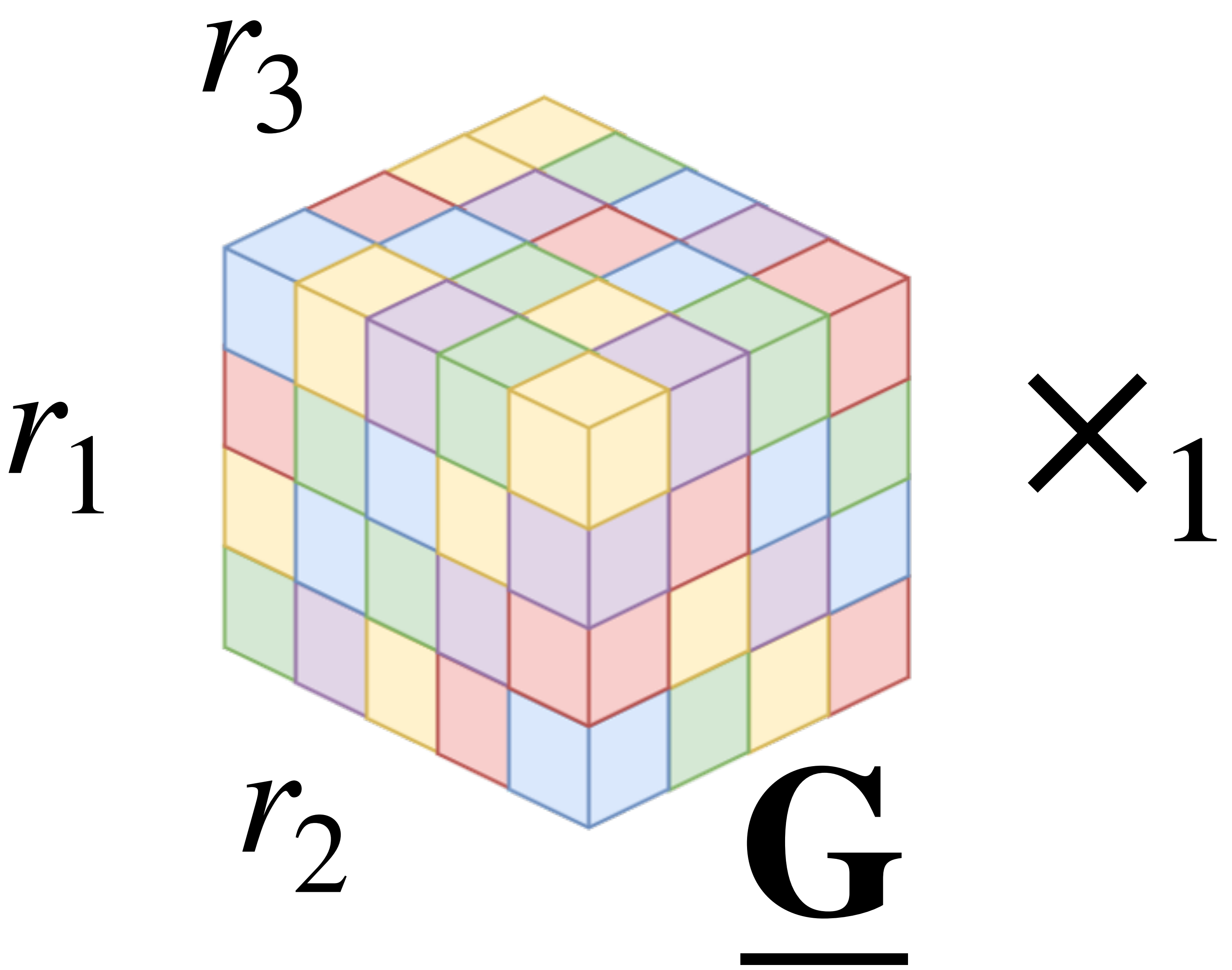
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



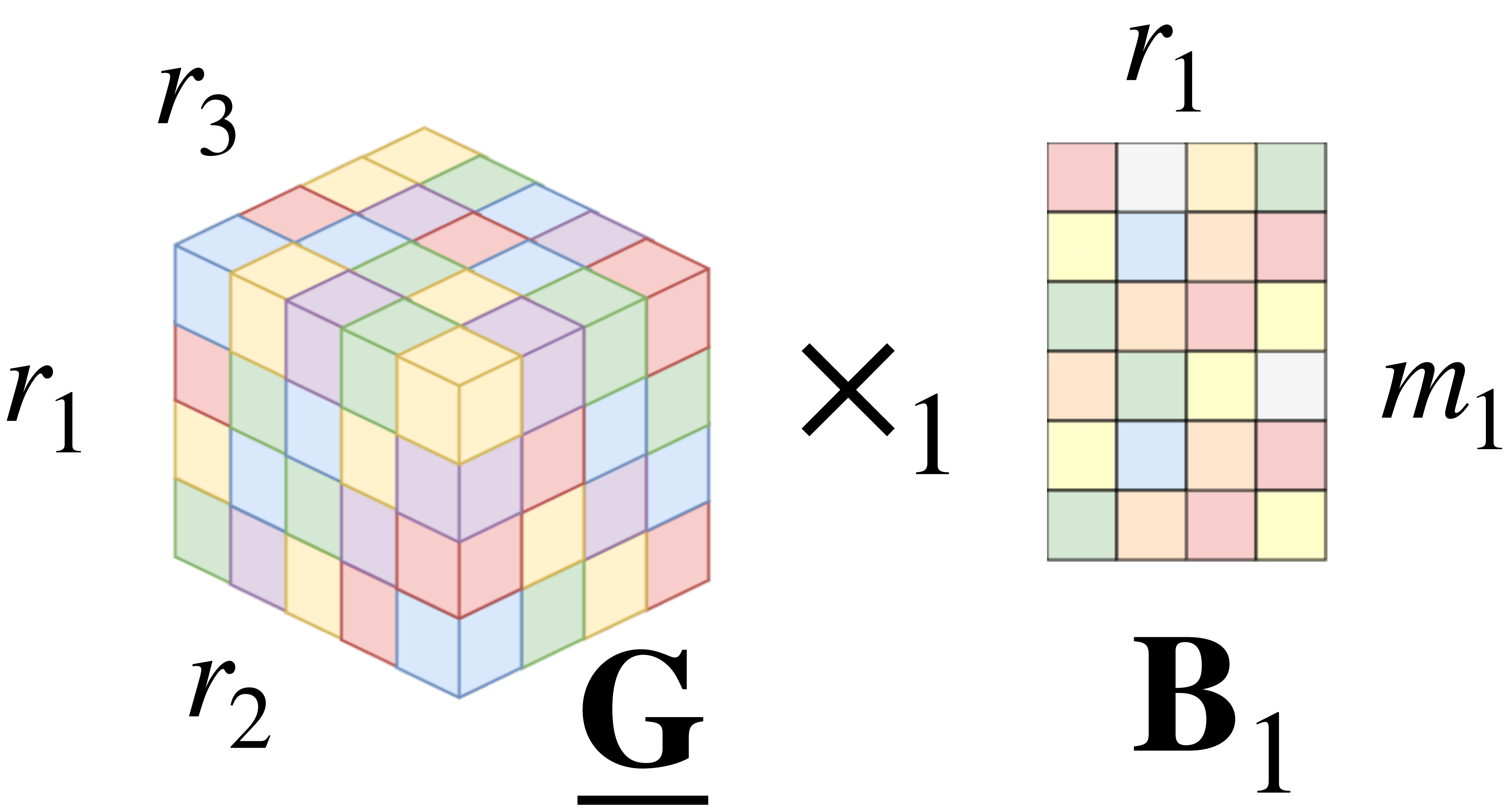
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



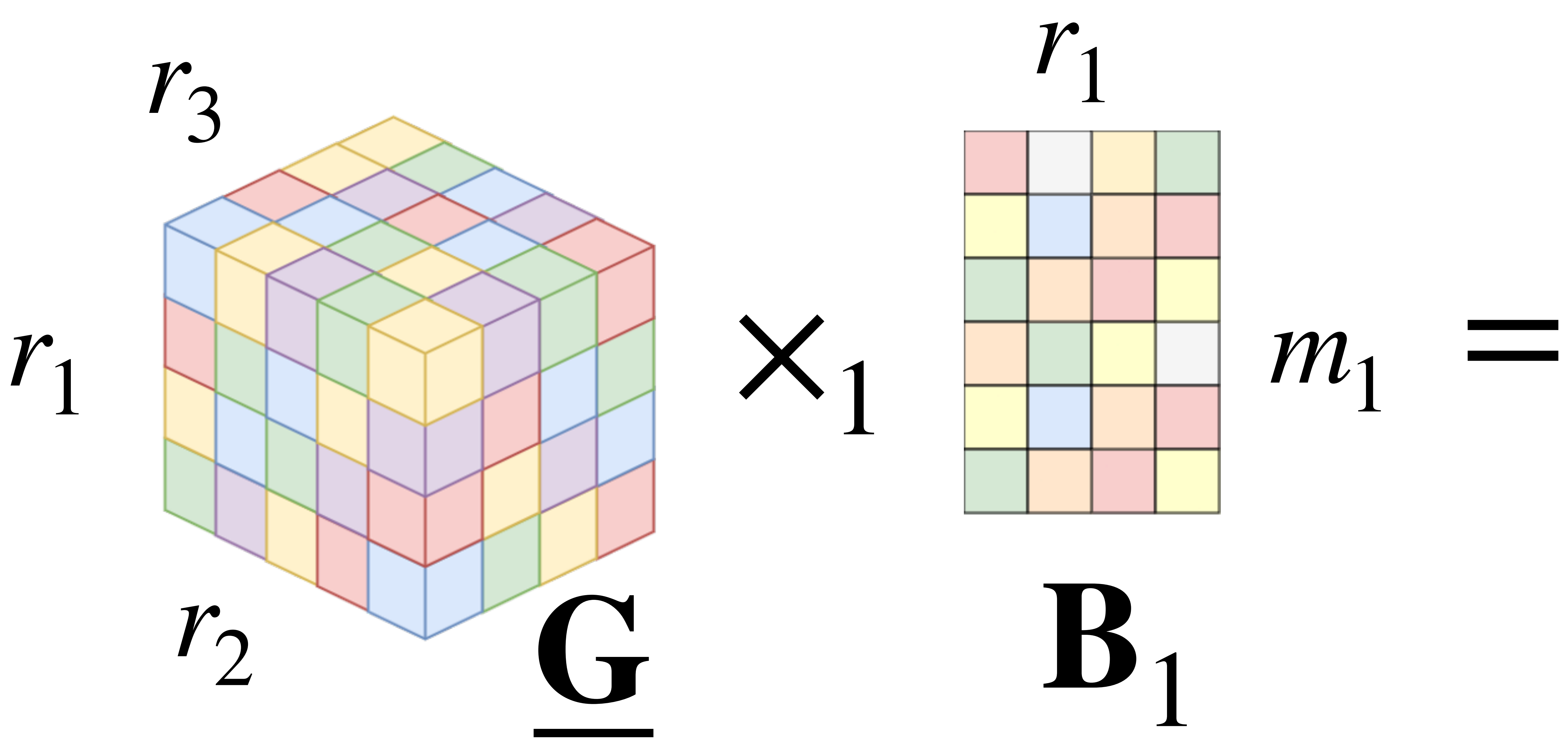
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



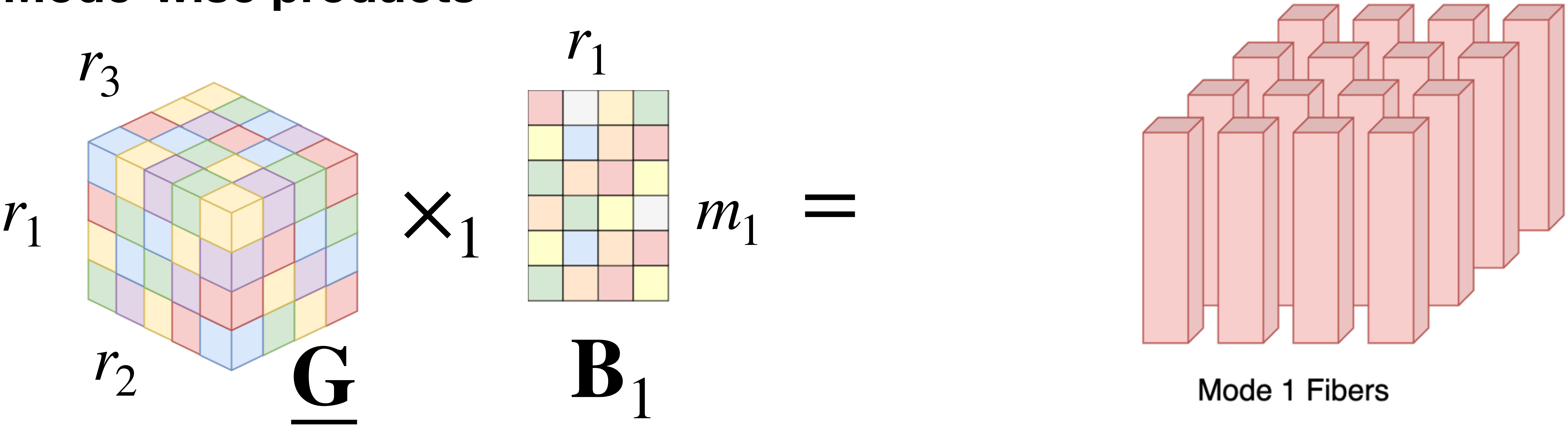
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



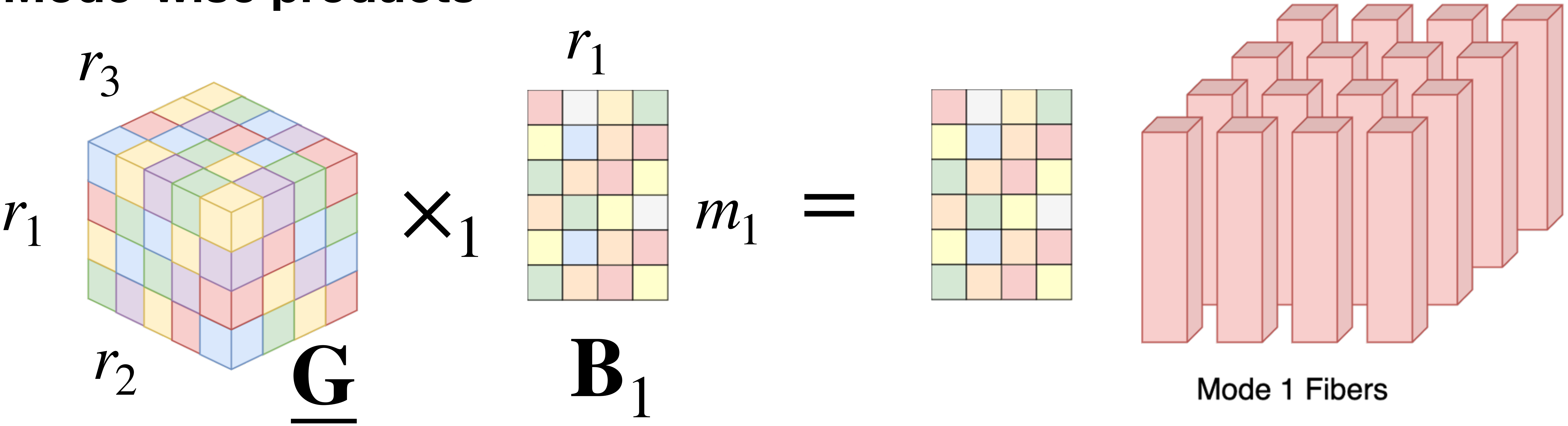
We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor products

Mode-wise products



We can multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

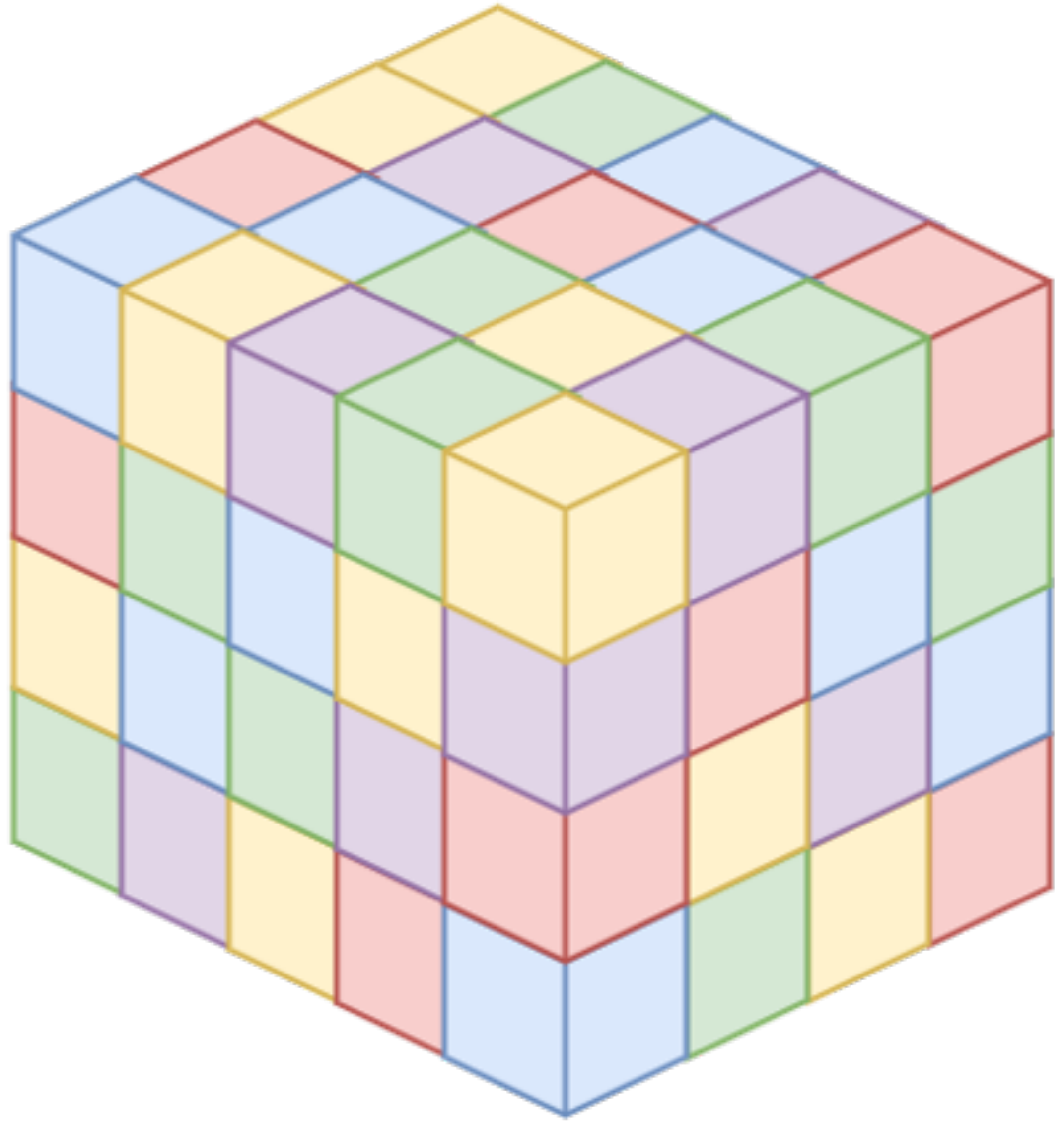
The result is a order-K tensor whose k-th mode is m_k dimensional.

Chaining matrix-tensor products

Processing multiple modes

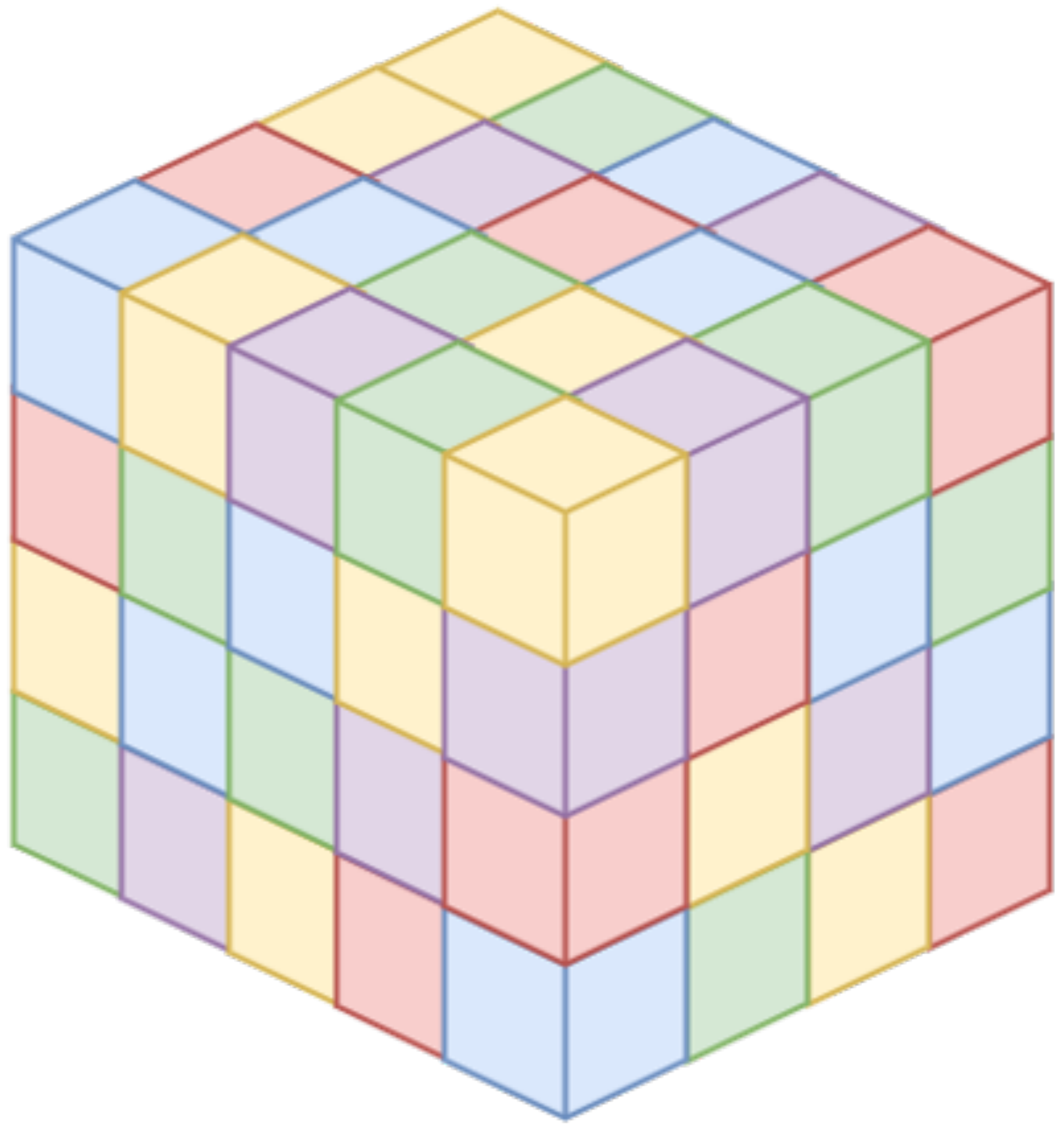
Chaining matrix-tensor products

Processing multiple modes

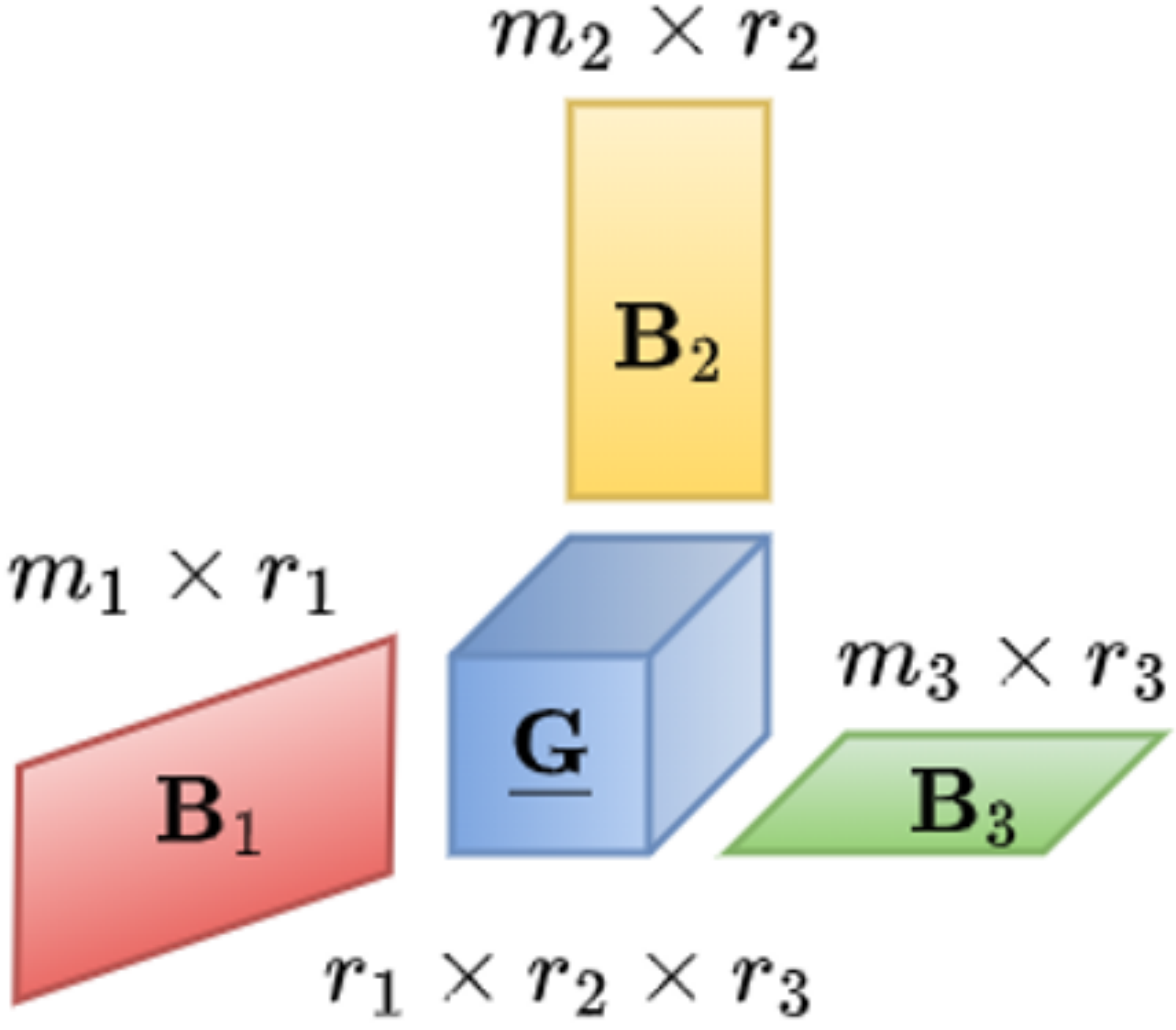


Chaining matrix-tensor products

Processing multiple modes

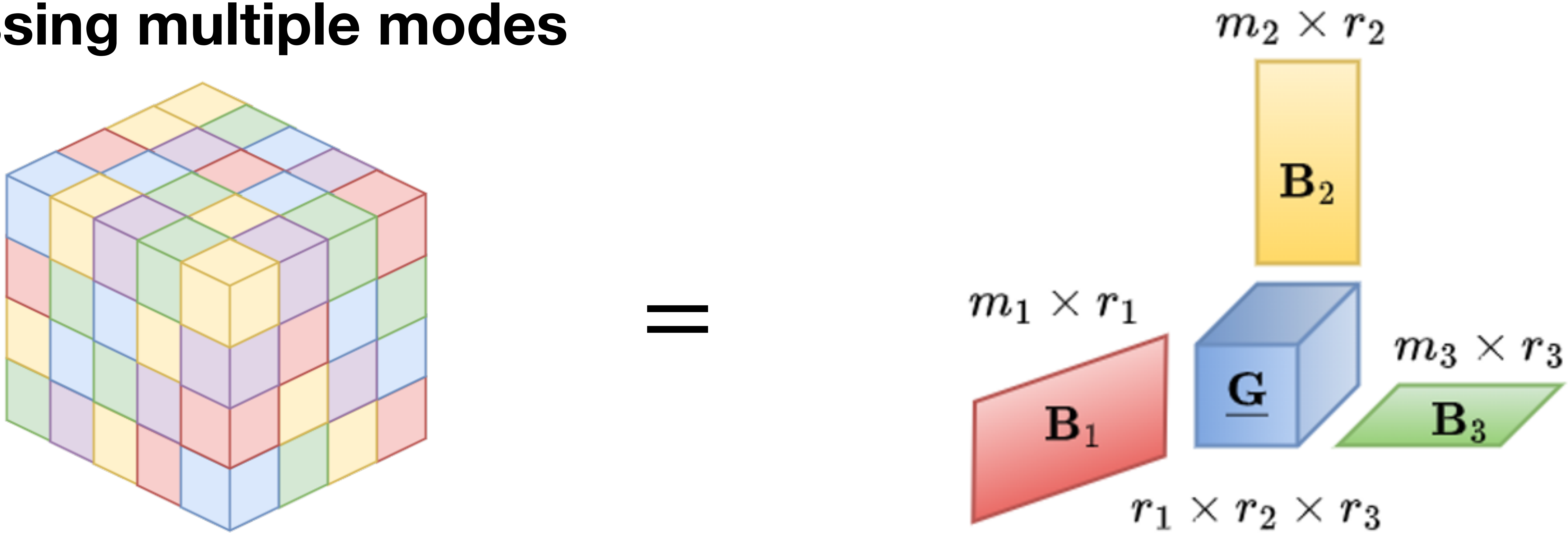


=



Chaining matrix-tensor products

Processing multiple modes

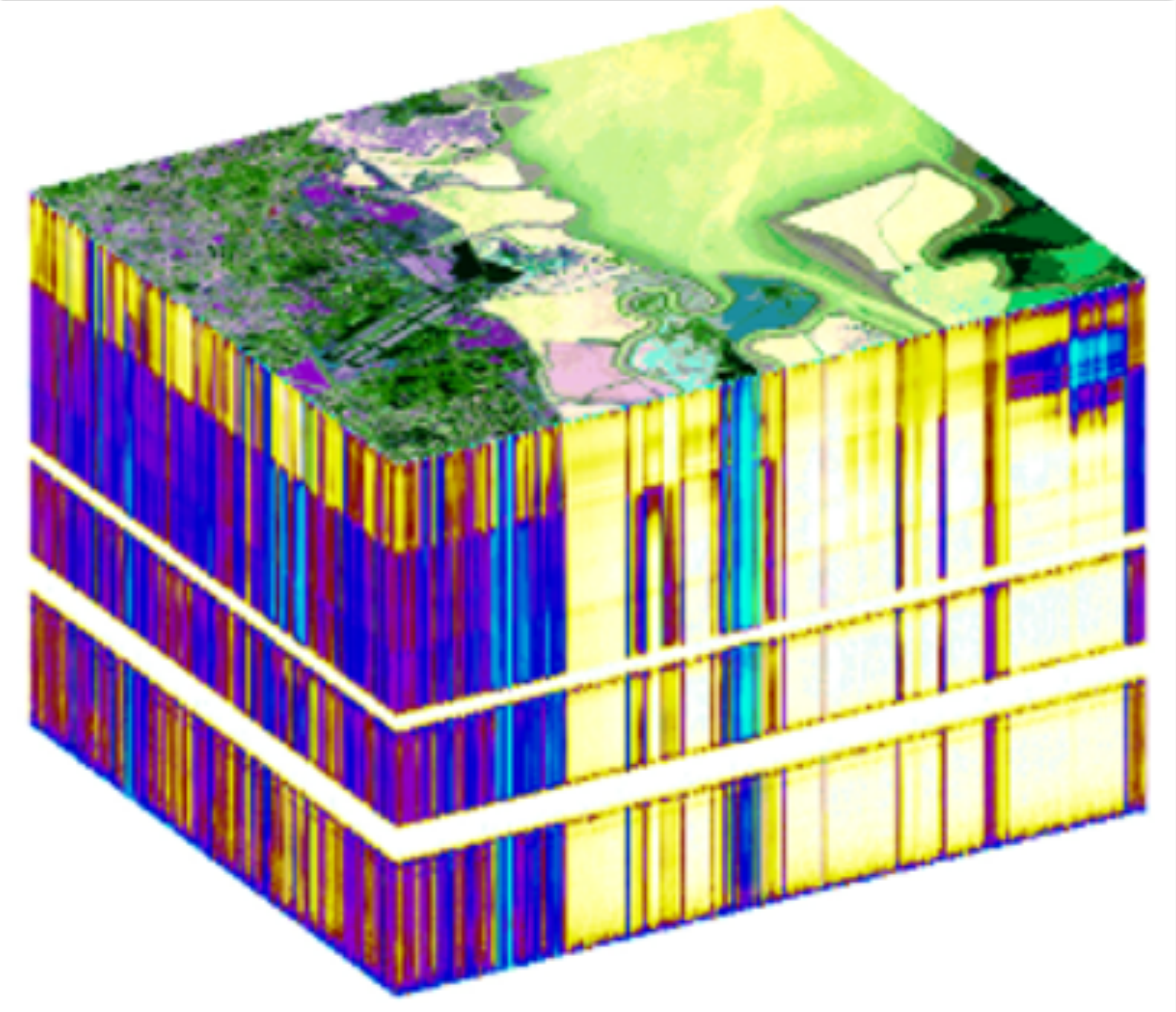


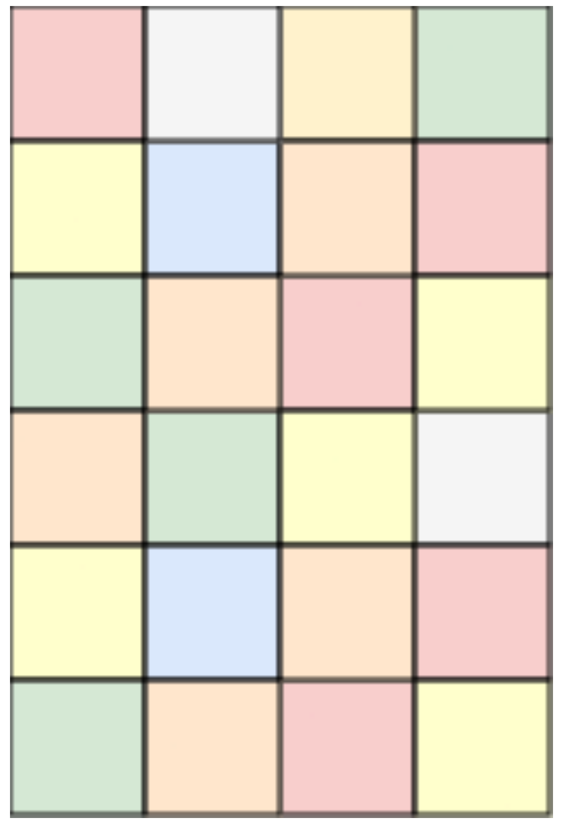
We can change the shape of a tensor with repeated matrix-tensor products

$$\underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K = \underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_K}$$

Matrix-tensor product example

Filtering hyperspectral images





$$\underline{\mathbf{X}} \quad \times_1 \quad \mathbf{L}$$

If $\underline{\mathbf{X}}$ is a hyperspectral image and $\mathbf{L}$ corresponds to the DFT of a lowpass filter, then

$$\underline{\mathbf{X}} \times_1 \mathbf{L}_1$$

Applies the lowpass filter to the spectrum at each location.

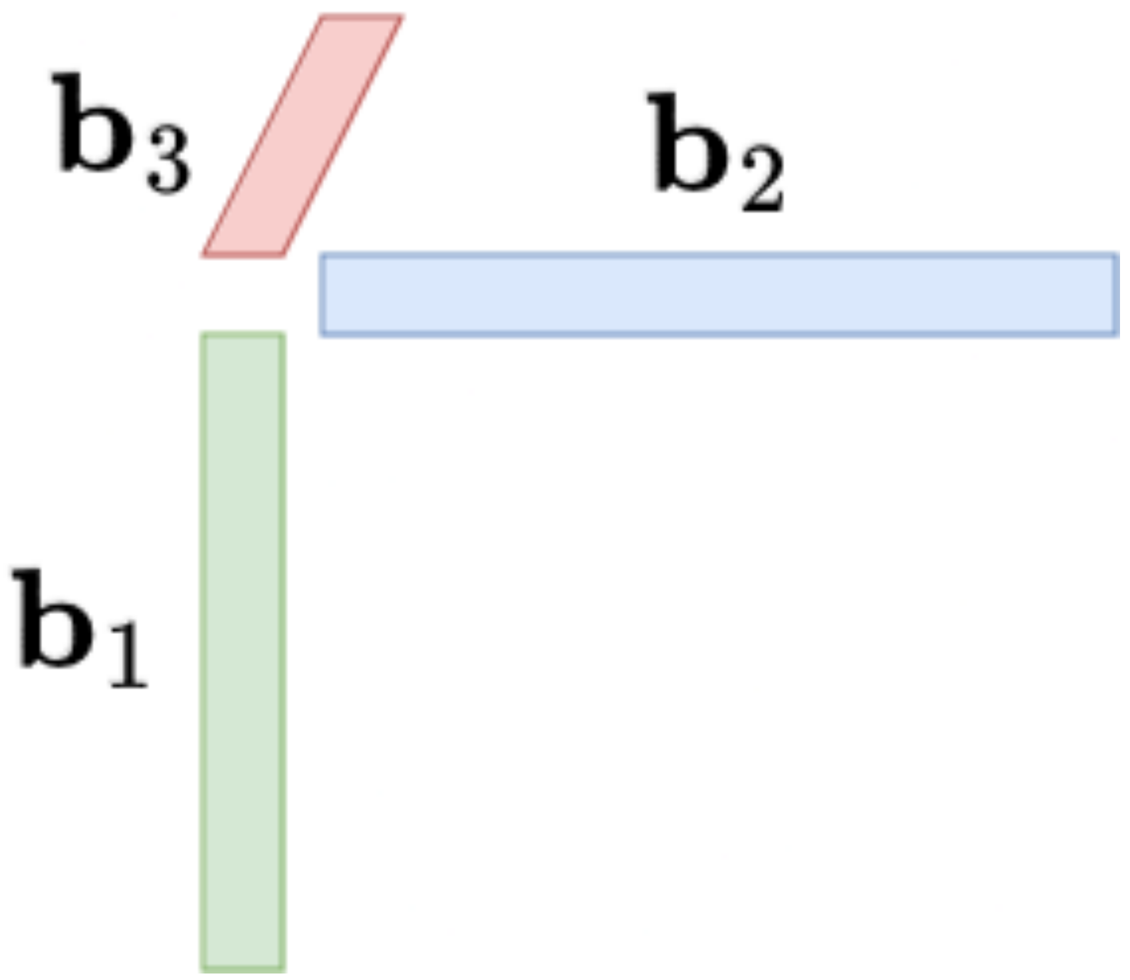
Rank-1 tensors are outer products

Trying to get a handle on rank

Rank-1 tensors are outer products

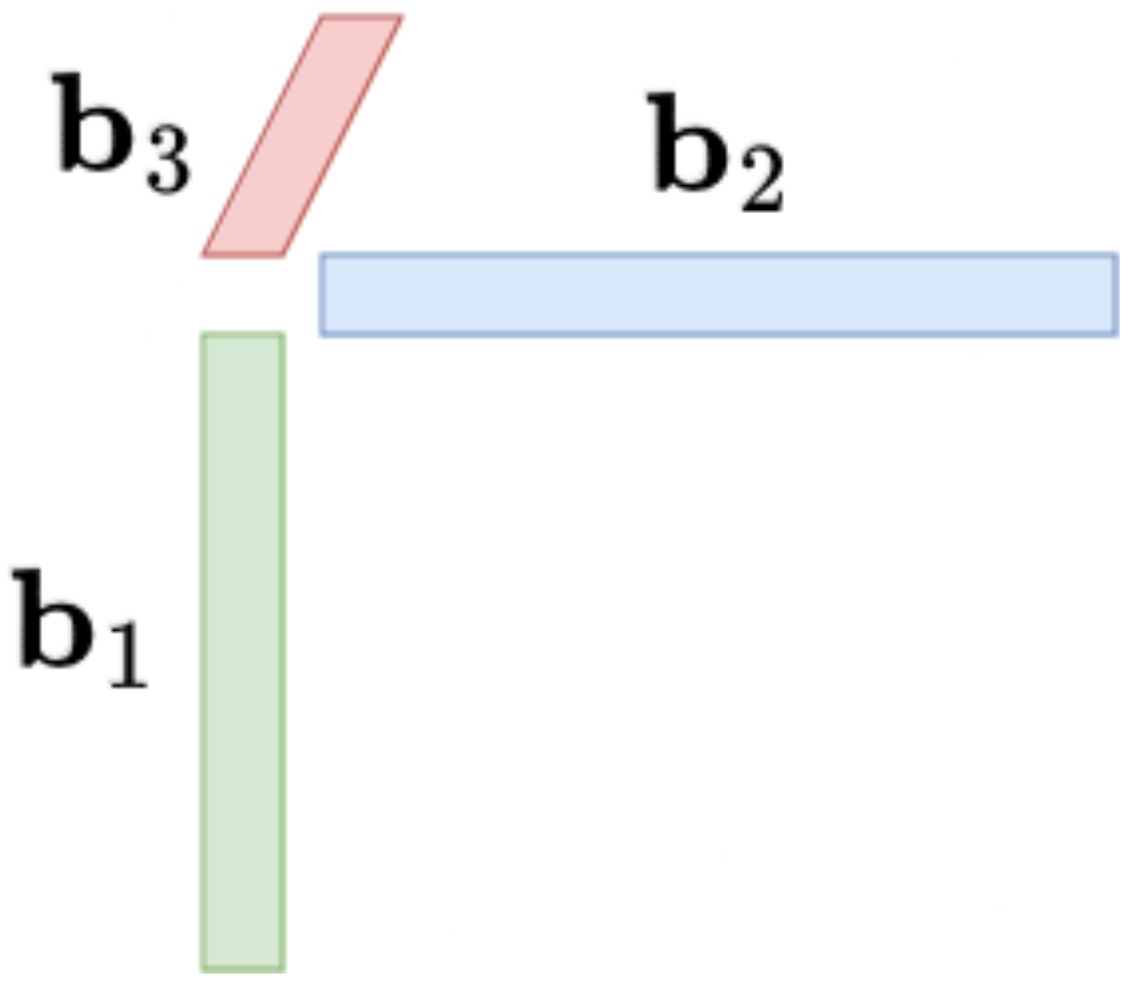
Trying to get a handle on rank

- In 2D this is a rank-1 matrix, and a rank-r matrix can be written as the sum of r rank-1 matrices.

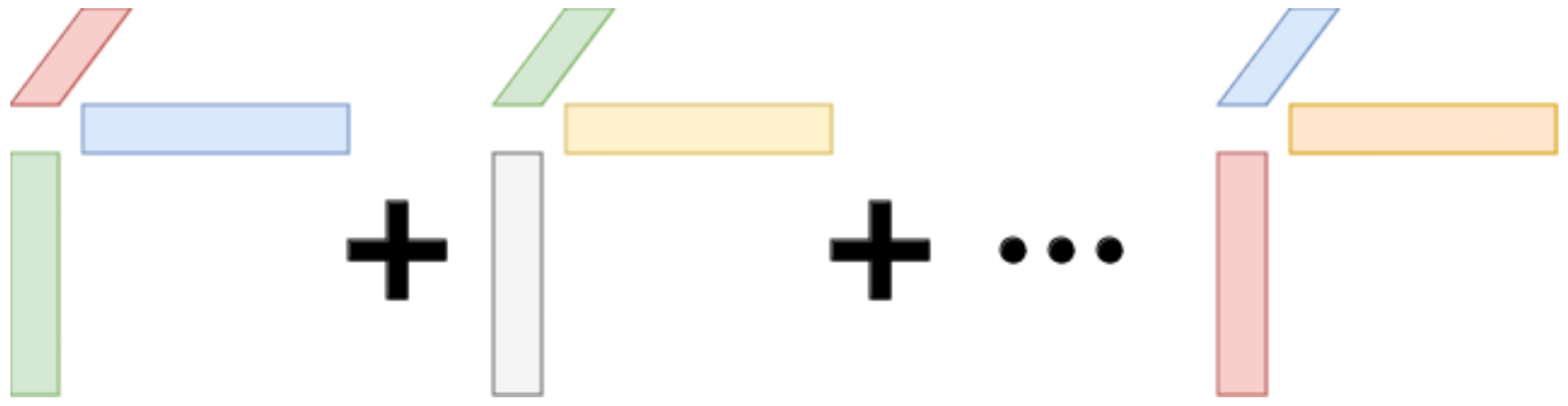


Rank-1 tensors are outer products

Trying to get a handle on rank



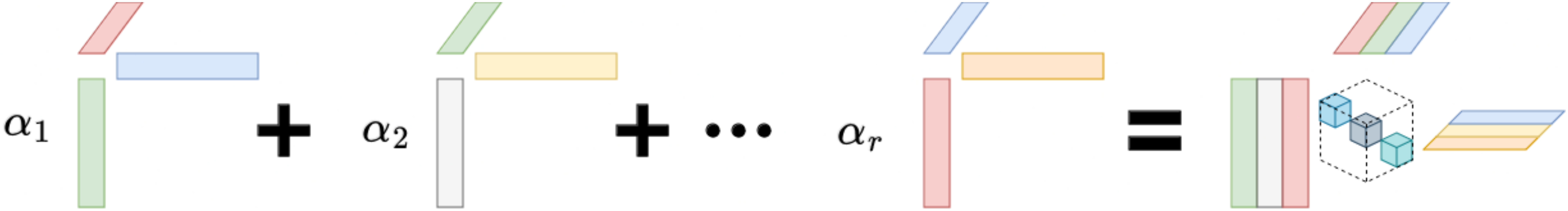
- In 2D this is a rank-1 matrix, and a rank-r matrix can be written as the sum of r rank-1 matrices.
- A matrix has a **CANDECOMP/ PARAFAC (CP)** representation of order r if we can write it as a sum of r rank-1 outer products.



CP Decomposition

CP factorization

Writing the decomposition with matrix-tensor products



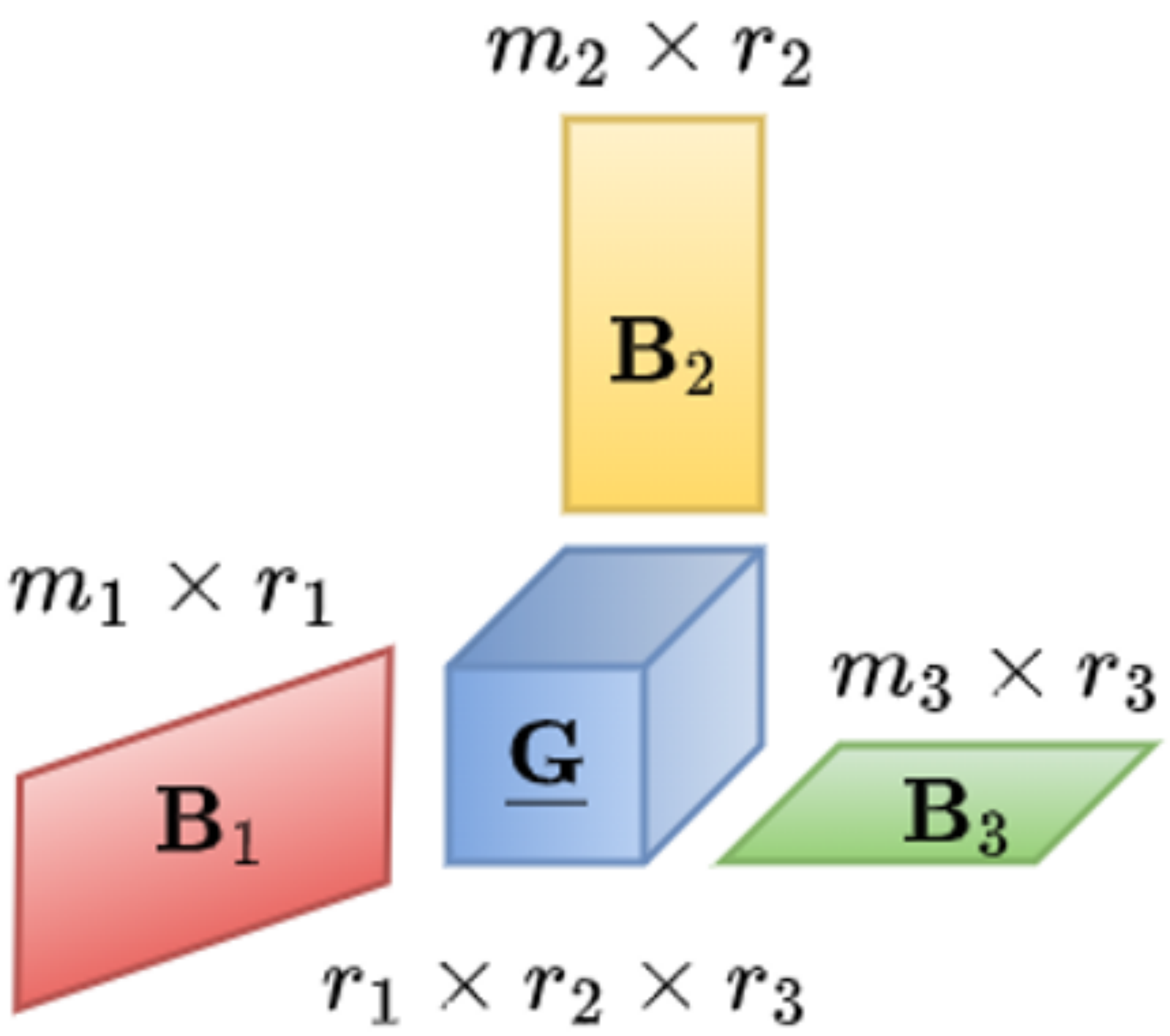
Gather the factors from each mode into matrices and define an $r \times r \times \cdots \times r$ **diagonal core tensor** $\underline{\mathbf{G}}$:

$$\underline{\mathbf{B}}_{\mathrm{CP}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K$$

The total number of parameters is $r\left(1 + \sum_{k=1}^{K} m_k\right)$ as opposed to $\prod_{k=1}^{K} m_k$.

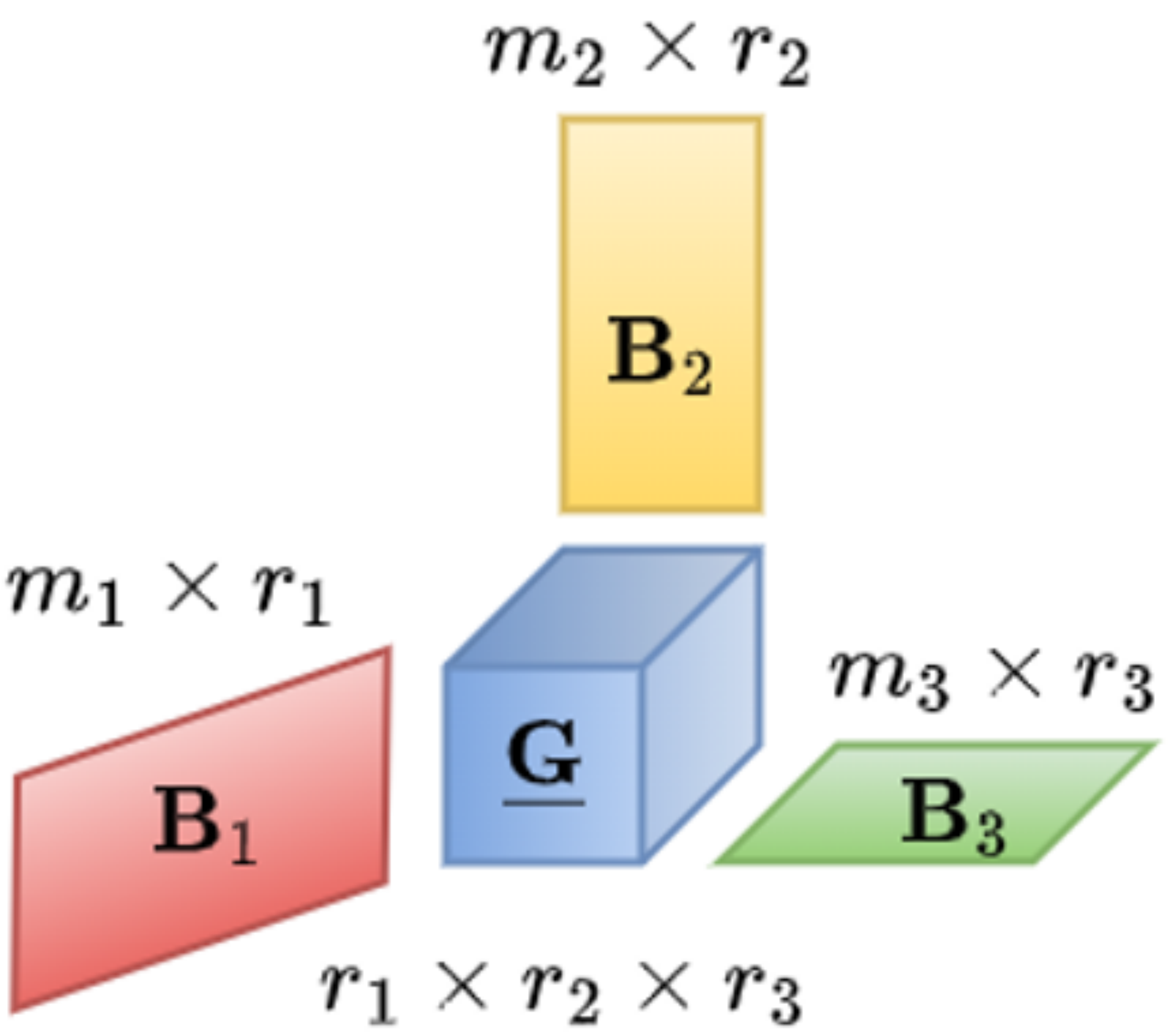
Tucker decomposition

Filling out the core tensor



Tucker decomposition

Filling out the core tensor



Suppose we have a **core tensor**

$$\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$$

and expand the dimensions using matrix-tensor products. This is the **Tucker decomposition**:

$$\underline{\mathbf{B}}_{\text{Tucker}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B} \times_3 \mathbf{B}_3$$

The total number of parameters is

$$\prod_{k=1}^{K} r_k + \sum_{k=1}^{K} m_k r_k$$

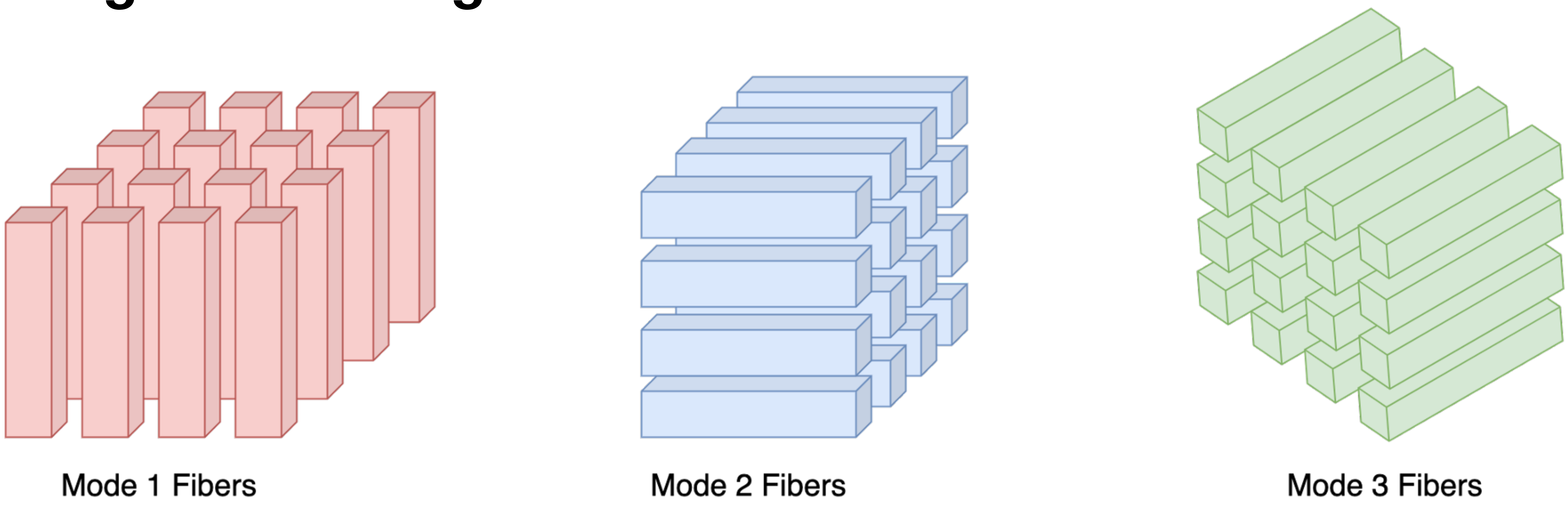
Issues with decompositions

There are many different definitions of “rank” for tensors

- **CP rank** of $\underline{\mathbf{B}}$ = smallest number of terms in a CP decomposition (Hitchcock 1927, Kruskal 1977).
    - The decomposition is (often) unique.
    - Computing the rank is NP-complete for finite fields and NP-hard for $\mathbb{Q}$ (Håstad 1990, resolving a conjecture of Gonzalez and Ja’Ja’ 1980).
- **Tucker rank** is a **vector**. Decomposition can be computed using the higher-order SVD [HOSVD] or other algorithms (De Lathauwer et al. 2000, also others).
    - Tucker rank is **not** unique.

Matricization

Unfolding or flattening a tensor

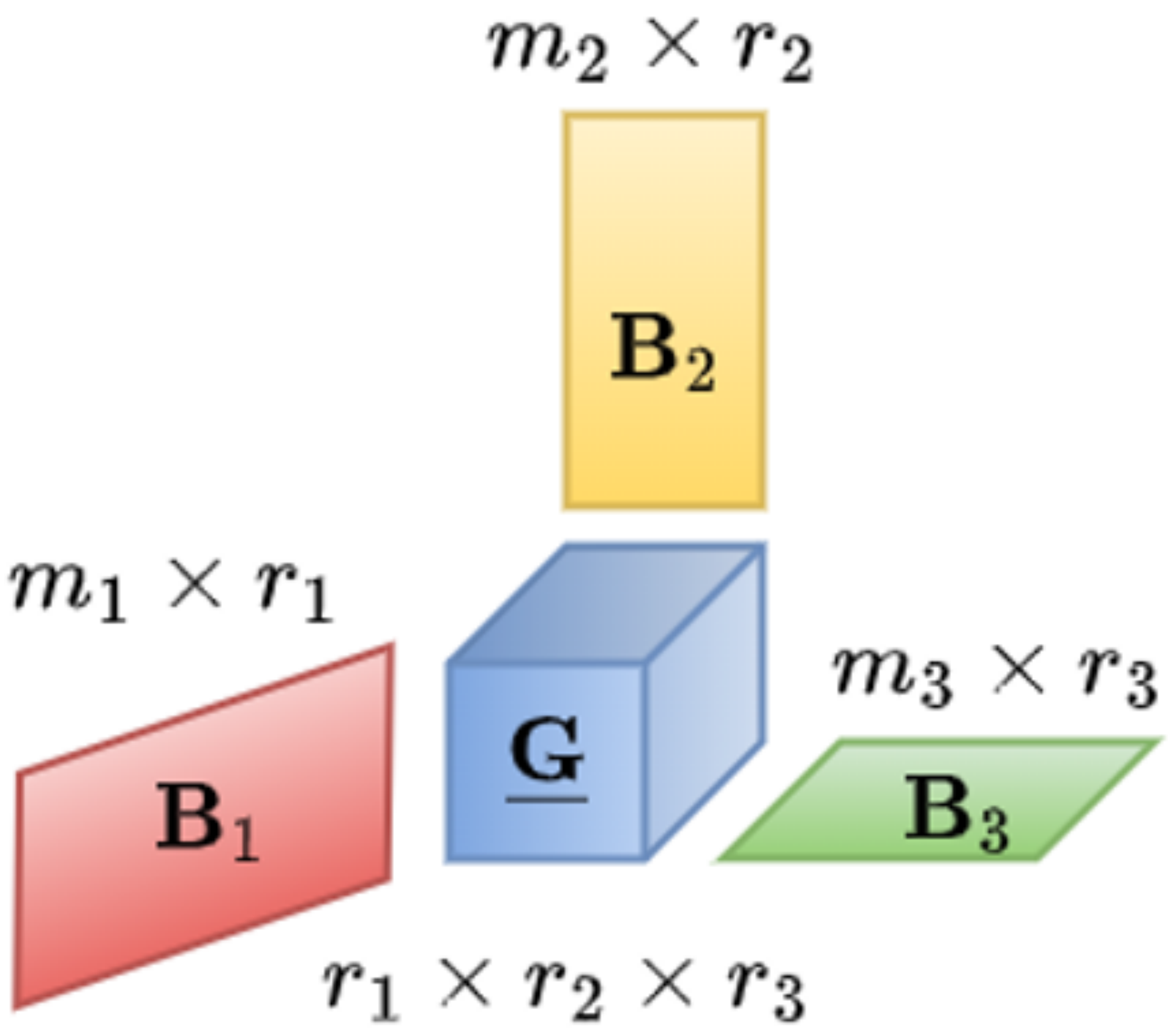


An order-K tensor can be rearranged into a matrix in K different ways by rearranging the 1-dimensional **fibers** in each dimension into a matrix.

We call these the **mode-k unfoldings** of the original tensor.

A different kind of matricization

Matrix-tensor products as a matrix vector product



Start with a Tucker factorization:

$$\underline{\mathbf{B}}_{\text{Tucker}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K$$

If we vectorize $\underline{\mathbf{B}}_{\text{Tucker}}$, we get get the following:

$$\text{vec}(\underline{\mathbf{B}}_{\text{Tucker}}) = \left(\mathbf{B}_K \otimes \cdots \otimes \mathbf{B}_1\right) \text{vec}(\underline{\mathbf{G}})$$

where $\otimes$ is the **Kronecker product**.

The Kronecker product

Matrix-tensor products as a matrix vector product

The Kronecker product makes “copies” of one matrix inside the other:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

Vectorizing shows that the Tucker decomposition

$$\text{vec}(\underline{\mathbf{B}}_{\text{Tucker}}) = \left(\mathbf{B}_K \otimes \cdots \otimes \mathbf{B}_2 \otimes \mathbf{B}_1\right) \text{vec}(\underline{\mathbf{G}})$$

Is somewhat restrictive.

Block tensor decompositions

Yet more generality

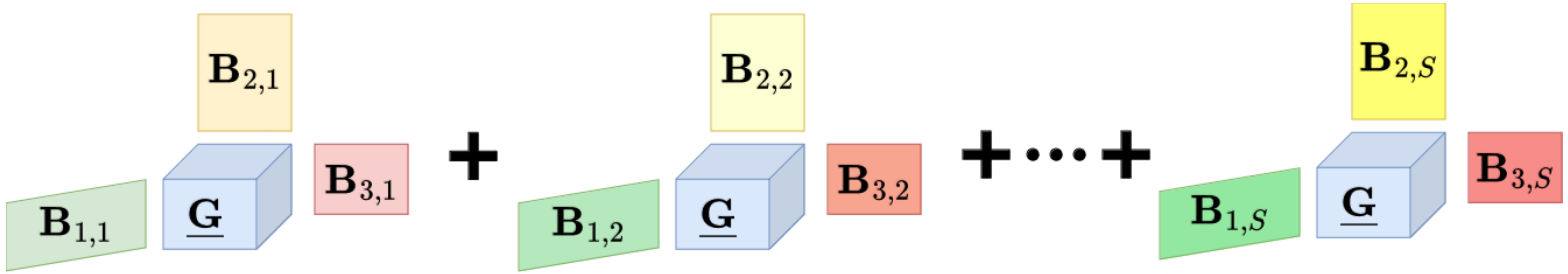
More recent work has studied **block tensor decompositions** (Section 5.7, Kolda and Bader 2009), which can written as a **mixture of Tucker models**:

$$\underline{\mathbf{B}}_{\mathrm{BTD}} = \sum_{s=1}^{S} \underline{\mathbf{G}}_s \times_1 \mathbf{B}_{1,s} \times_2 \mathbf{B}_{2,s} \cdots \times_K \mathbf{B}_{K,s},$$

This is definitely more flexible! But perhaps too flexible…

Proposal: low separation rank (LSR) tensors

BTD with a common core tensor



Special case of the BTD is a **low separation rank (LSR)** decomposition:

$$\underline{\mathbf{B}}_{\text{LSR}} = \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \underline{\mathbf{B}}_{1,s} \times_2 \underline{\mathbf{B}}_{2,s} \cdots \times_K \underline{\mathbf{B}}_{K,s}$$

We use the *same core tensor* $\underline{\mathbf{G}}$ for each term. We also assume (wlog) that the factor matrices $\{\underline{\mathbf{B}}_{k,s}\}$ have orthonormal columns.

What does separation rank mean?

Back to the matricization

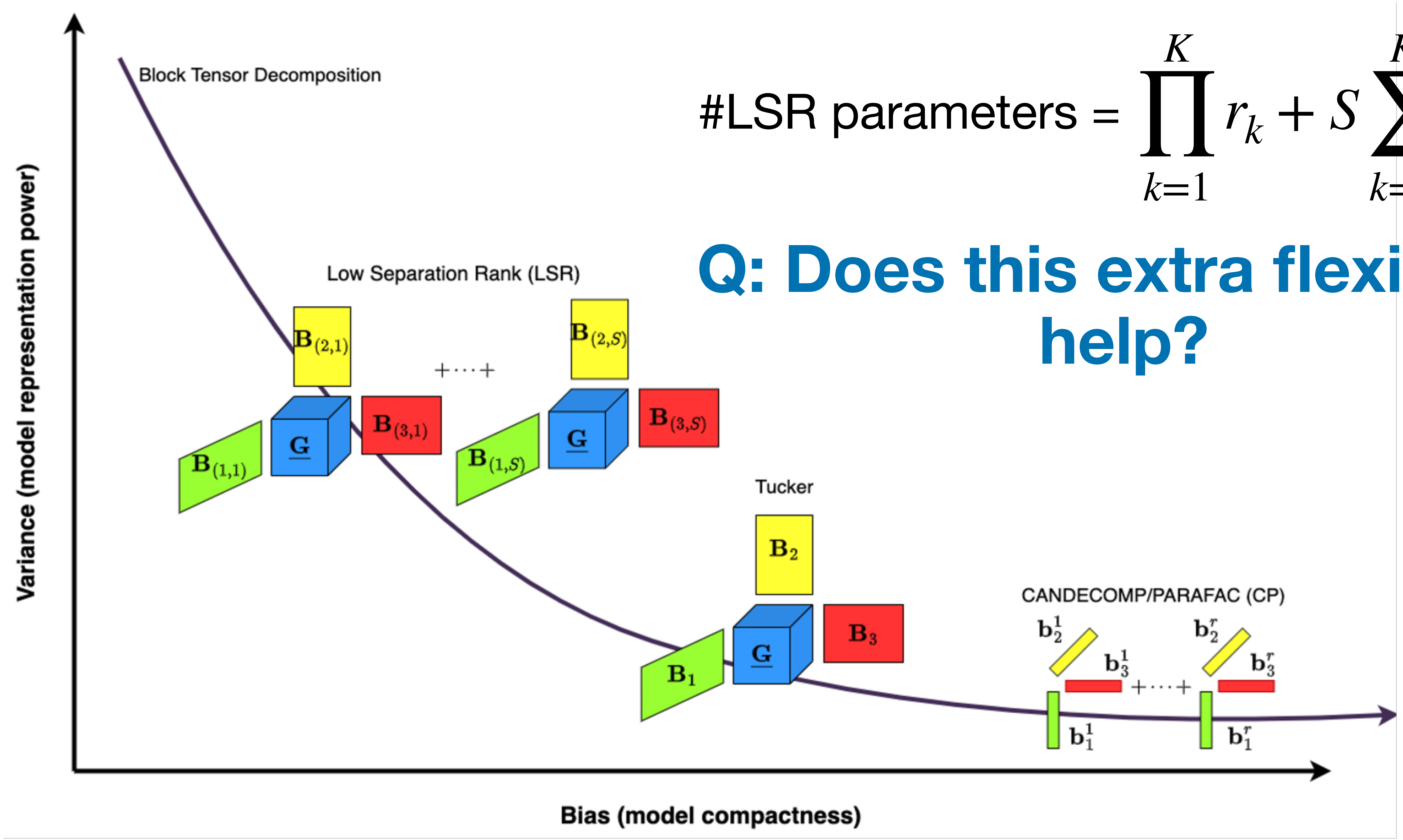
The **separation rank** (Tsiligkaridis and Hero, 2013) of a matrix is the minimum number S of terms needed so that

$$\mathbf{M} = \sum_{s=1}^{S} \mathbf{A}_{K,s} \otimes \cdots \otimes \mathbf{A}_{2,s} \otimes \mathbf{A}_{1,s}$$

Our LSR model corresponds assuming the matrix-vector product has a matrix with low separation rank

$$\sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \underline{\mathbf{B}}_{1,s} \times_2 \underline{\mathbf{B}}_{2,s} \cdots \times_K \underline{\mathbf{B}}_{K,s} = \underline{\mathbf{B}}_{\text{LSR}} \Longrightarrow \left(\sum_{s} \bigotimes_{k} \mathbf{B}_{\mathbf{k}} \right) \mathbf{g}$$

Comparing different decompositions



$$\#\text{LSR parameters} = \prod_{k=1}^{K} r_k + S \sum_{k=1}^{K} m_k r_k$$

Q: Does this extra flexibility help?

Regression and classification with structured tensors

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

We have a *training set* of n tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\}$ following a **generalized linear model (GLM)**. Our goal: estimate $\underline{\mathbf{B}}$ s.t. if $\eta = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle$ then

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

We have a *training set* of n tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\}$ following a **generalized linear model (GLM)**. Our goal: estimate $\underline{\mathbf{B}}$ s.t. if $\eta = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle$ then

$$p(y;\eta) = b(y)\exp\left(-\eta T(y) - a(\eta)\right).$$

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

We have a *training set* of n tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\}$ following a **generalized linear model (GLM)**. Our goal: estimate $\underline{\mathbf{B}}$ s.t. if $\eta = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle$ then

$$p(y;\eta) = b(y)\exp\left(-\eta T(y) - a(\eta)\right).$$

That is, y is from an *exponential family*. One example is *logistic regression*:

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

We have a *training set* of n tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\}$ following a **generalized linear model (GLM)**. Our goal: estimate $\underline{\mathbf{B}}$ s.t. if $\eta = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle$ then

$$p(y; \eta) = b(y) \exp\left(-\eta T(y) - a(\eta)\right).$$

That is, y is from an *exponential family*. One example is *logistic regression*:

$$y \sim \text{Bernoulli}\left(\frac{1}{1 + \exp(-\langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle)}\right)$$

Prior work using CP and Tucker tensors

Generalized linear models

Prior work using CP and Tucker tensors

Generalized linear models

We look **LSR** models for **GLMs**:

Prior work using CP and Tucker tensors

Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)

Prior work using CP and Tucker tensors

Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)
- **CP** + **GLMs** (Zhou et al. 2014)

Prior work using CP and Tucker tensors

Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)
- **CP** + **GLMs** (Zhou et al. 2014)
- **Tucker** + **linear regression** (Zhang et al. 2020, Ahmed et al. 2020)

Prior work using CP and Tucker tensors

Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)
- **CP** + **GLMs** (Zhou et al. 2014)
- **Tucker** + **linear regression** (Zhang et al. 2020, Ahmed et al. 2020)
- **Tucker** + **logistic regression** (Zhang et al. 2016)

Prior work using CP and Tucker tensors

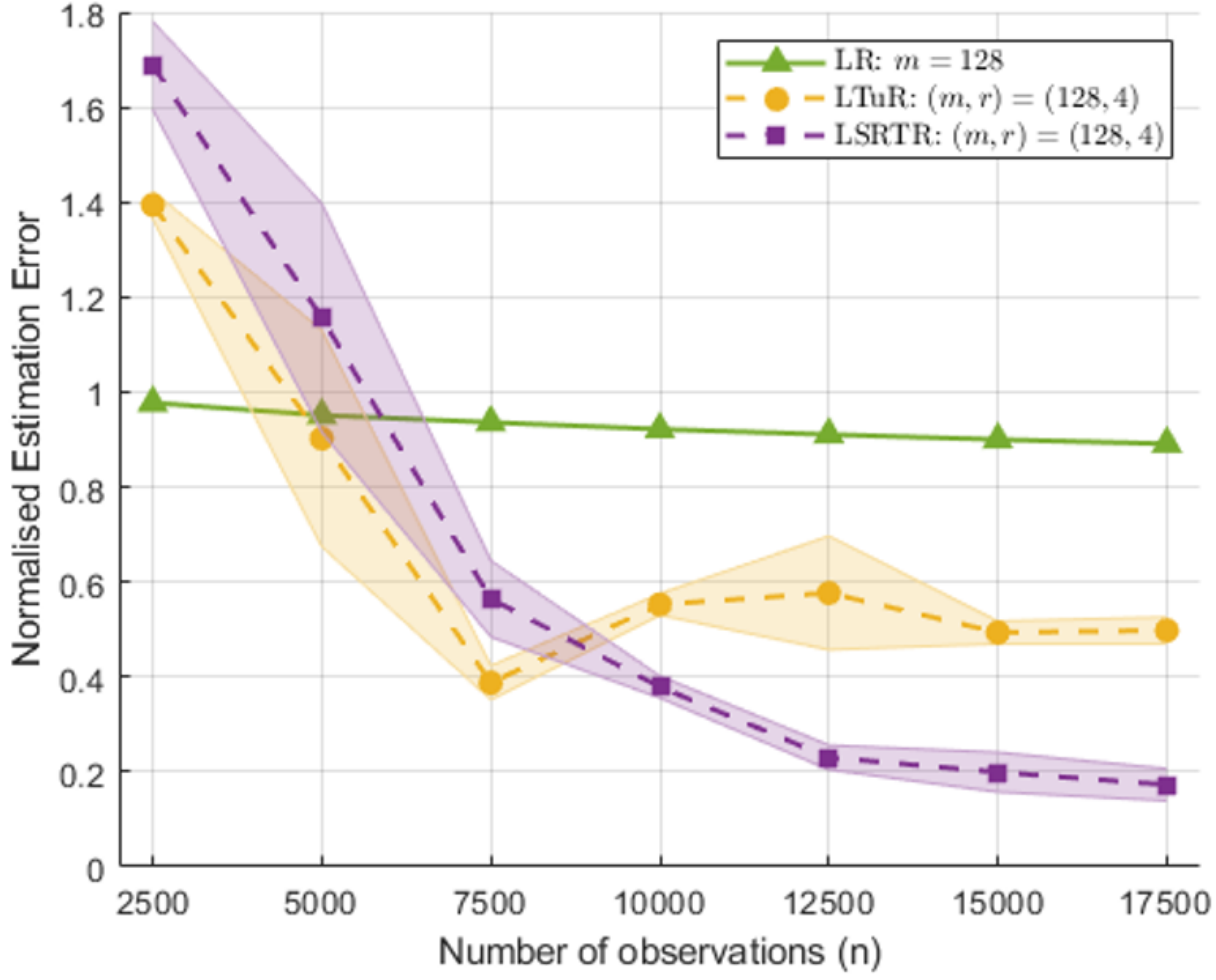
Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)
- **CP** + **GLMs** (Zhou et al. 2014)
- **Tucker** + **linear regression** (Zhang et al. 2020, Ahmed et al. 2020)
- **Tucker** + **logistic regression** (Zhang et al. 2016)
- **Tucker** + **GLMs** (Li et al., 2018; Zhou et al., 2013)

The benefits of more flexible modeling

Taking advantage of more data



LSR models let use scale the number of parameters to the data set size.

Synthetic data experiments show that with a modest number of samples, LSR models are better than vectorizing or using a Tucker model.

Mapping the tensor to a matrix

Using the LSR matrix in the vectorized problem

Mapping the tensor to a matrix

Using the LSR matrix in the vectorized problem

Under an LSR model, we have

$$\eta = \left\langle \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \mathbf{B}_{(1,s)} \times_2 \mathbf{B}_{(2,s)} \times_3 \cdots \times_K \mathbf{B}_{(K,s)}, \underline{\mathbf{X}} \right\rangle$$

Mapping the tensor to a matrix

Using the LSR matrix in the vectorized problem

Under an LSR model, we have

$$\eta = \left\langle \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \mathbf{B}_{(1,s)} \times_2 \mathbf{B}_{(2,s)} \times_3 \cdots \times_K \mathbf{B}_{(K,s)}, \underline{\mathbf{X}} \right\rangle$$

Vectorizing:

$$\eta = \left\langle \left(\sum_{s=1}^{S} \mathbf{B}_{(K,s)} \otimes \mathbf{B}_{(K-1,s)} \otimes \cdots \otimes \mathbf{B}_{(1,s)} \right) \mathbf{g}, \mathbf{x} \right\rangle$$

Space of LSR models

Using the LSR matrix in the vectorized problem

Suppose we are given $(r_1, r_2, \ldots, r_K, S)$. Then define

$$\mathscr{C}_{\mathsf{LSR},S} = \left\{ \underline{\mathbf{B}} : \underline{\mathbf{B}} = \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \mathbf{B}_{(1,s)} \times_2 \cdots \times_K \mathbf{B}_{(K,s)} \right\},$$

where for each (k, s), the columns of $\mathbf{B}_{(k,s)}$ are orthonormal.

This the the space we have to optimize over to select an LSR model for our regression parameter.

Maximum likelihood

Sorry, but it's really messy

The MLE can is computed by minimizing

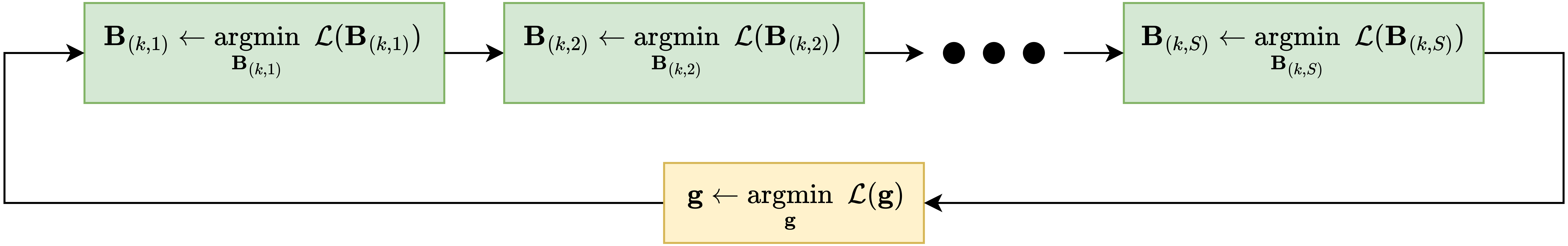
$$\sum_{i=1}^{n}\left[\left\langle\left(\sum_{s=1}^{S}\bigotimes_{k}\mathbf{B}_{(k,s)}\right)\mathbf{g},\mathbf{x}_i\right\rangle T(y_i) - a\left(\left\langle\left(\sum_{s=1}^{S}\bigotimes_{k}\mathbf{B}_{(k,s)}\right)\mathbf{g},\mathbf{x}_i\right\rangle\right)\right],$$

Over all $\mathbf{B}_{k,s} \in \mathbb{O}^{m_k \times r_k}$ and $\mathbf{g} \in \mathbb{R}^{r_1 r_2 \cdots r_K}$.

Note: if we fix all matrices but one and then optimize over that one, it is tractable…

Alternating minimization: LSR-TR

Seems to work well in practice



Use **alternating minimization** cycling through each $\mathbf{B}_{(k,s)}$ and then $\mathbf{g}$.

In particular, use projected gradient descent on each $\mathbf{B}_{(k,s)}$ and regular gradient descent on $\mathbf{g}$.

Convergence guarantees: work in progress.

Experiments on medical imaging data

Data sets and algorithms

Data sets: ABIDE Autism [fMRI] (Craddock et al., 2013 2020), Vessel MNIST 3D [MRA] (Yang et al., 2020).

Other algorithms:

- **TTR**: **Tucker** + **GLMs** using a ‘block relaxation’ algorithm (Li et al., 2018)
- **LTuR**: **Tucker** + **logistic regression** with Frobenius norm regularization (Zhang & Jiang, 2016)
- **LR**: **Unstructured** + **logistic regression** (Seber & Lee, 2003)
- **LCPR**: **CP** + **logistic regression** (Tan et al., 2013)

ABIDE Autism data set

A tiny data set: $K = 2$, $\mathbf{m} = (111{,}116)$, $n = 80$

	SVM	LR	LCPR	LTuR	LSRTR
Sensitivity	0.71	0.71	0.71	0.71	1
Specificity	0.14	0.71	0.85	0.85	0.85
F1 score	0.55	0.71	0.77	0.77	**0.93**
AUC	0.42	0.51	0.84	0.84	**0.9**
Average Accuracy	0.43	0.71	0.78	0.78	**0.92**

- Chose ranks $r_1 = 6$ and $r_2 = 6$ with $S = 2$.
- Unstructured models are quite bad in the undersampled regime.
- Adding one more Tucker component can give significant improvements.

VesselMNIST 3D

Comparing against a DNN too: $K = 3$, $\mathbf{r} = (28,28,28)$, $n = 1335$

	SVM	LR	LCPR	LTuR	LSRTR	ResNet 50 + 3D
Sensitivity	0.39	0.53	0.26	0.32	0.47	0.85
Specificity	0.95	0.55	0.946	0.94	0.96	0.86
F1 score	0.44	0.21	0.3	0.37	0.55	**0.57**
AUC	0.84	0.52	0.6	0.66	0.81	**0.9**
Average Accuracy	0.89	0.55	0.869	0.87	**0.91**	0.85

- Chose ranks $r_1 = 3$, $r_2 = 3$, $r_3 = 3$, and $S = 2$
- LSRTR has better accuracy but worse F1 and AUC (see paper).
- Issues such as overfitting, interpretability, etc. are still open.

What about the theory?

Lower bounds yes, upper bounds in progress…

Suppose our data was generated with an LSR tensor $\underline{\mathbf{B}}^*$ We (Taki, S. Bajwa, 2023) can prove a lower bound on the MSE of estimating $\underline{\mathbf{B}}^*$:

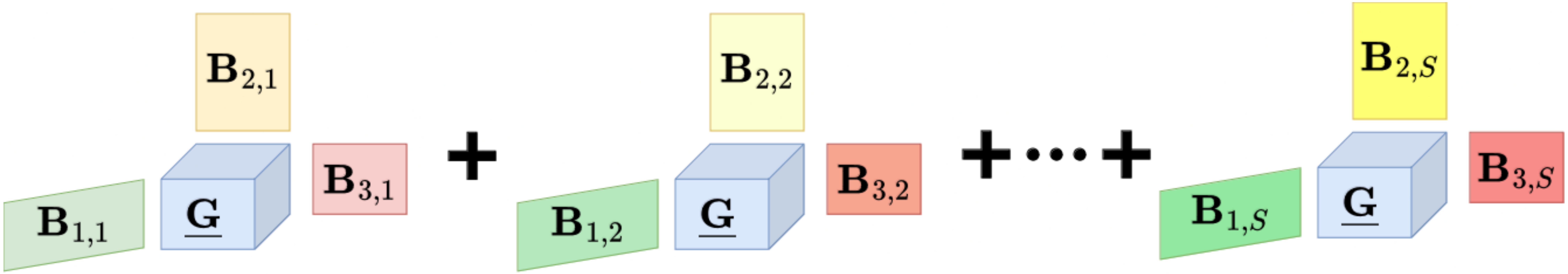
$$\mathbb{E}\left[\|\underline{\mathbf{B}}^* - \underline{\hat{\mathbf{B}}}\|_F^2\right] = \Omega\left(\frac{S\sum_k (m_k - 1)r_k + \prod_k (r_k - 1) - 1}{\|\boldsymbol{\Sigma}_x\|_2 n}\right)$$

We can specialize this result to the Tucker and CP cases as well.

	Structure of $\underline{\mathbf{B}}$			
Regression	**Unstructured**	**CP**	**Tucker**	**LSR**
Linear	$\frac{\sigma_y^2 \widetilde{m}}{n}$ (Raskutti et al., 2011)	–	$\frac{\sigma_y^2 \left(\sum_{k \in [K]} m_k r_k - r_k^2 + \widetilde{r} \right)}{n}$ (Zhang et al., 2020)	–
Logistic	$\frac{\widetilde{m}}{n}$ (Abramovich & Grinshtein, 2016)	–	–	–
GLM	$\frac{\sigma_y^2 \widetilde{m}}{Dn}$ (Lee & Courtade, 2020)	$\frac{\sum_{k \in [K]} m_k r + r}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Corollary 2	$\frac{\sum_{k \in [K]} m_k r_k + \widetilde{r}}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Corollary 1	$\frac{S \sum_{k \in [K]} m_k r_k + \widetilde{r}}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Theorem 6

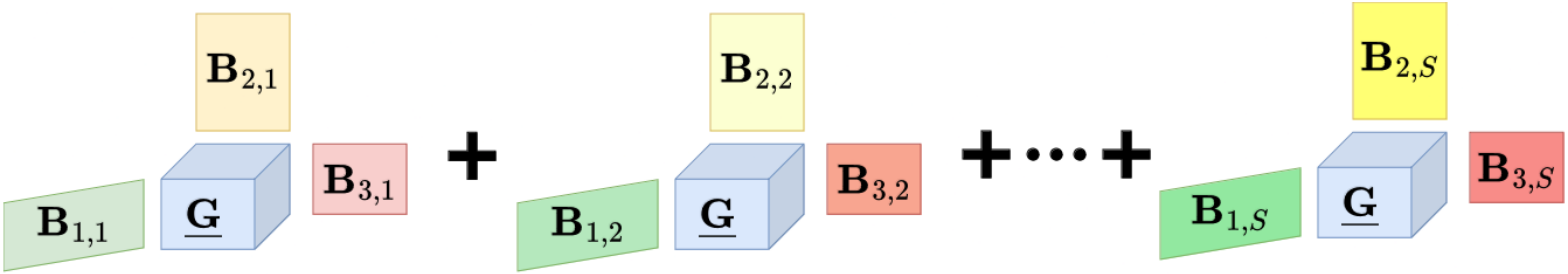
Ongoing/future work

Identifiability and beyond



Ongoing/future work

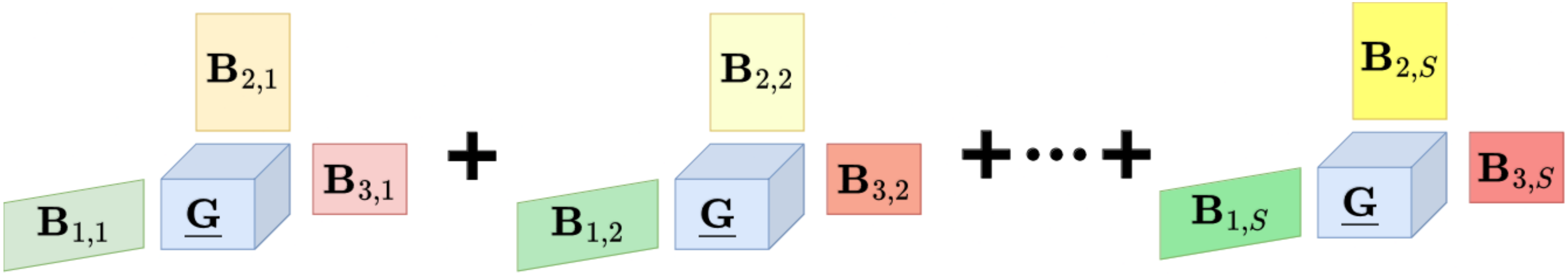
Identifiability and beyond



- Determine conditions so that LSR factors are (locally) identifiable.

Ongoing/future work

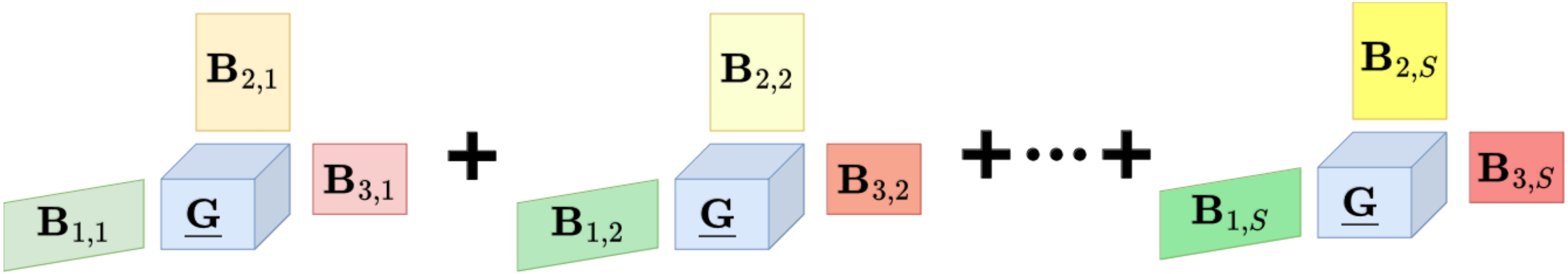
Identifiability and beyond



- Determine conditions so that LSR factors are (locally) identifiable.
- Understand the analytical properties of the LSR set.

Ongoing/future work

Identifiability and beyond



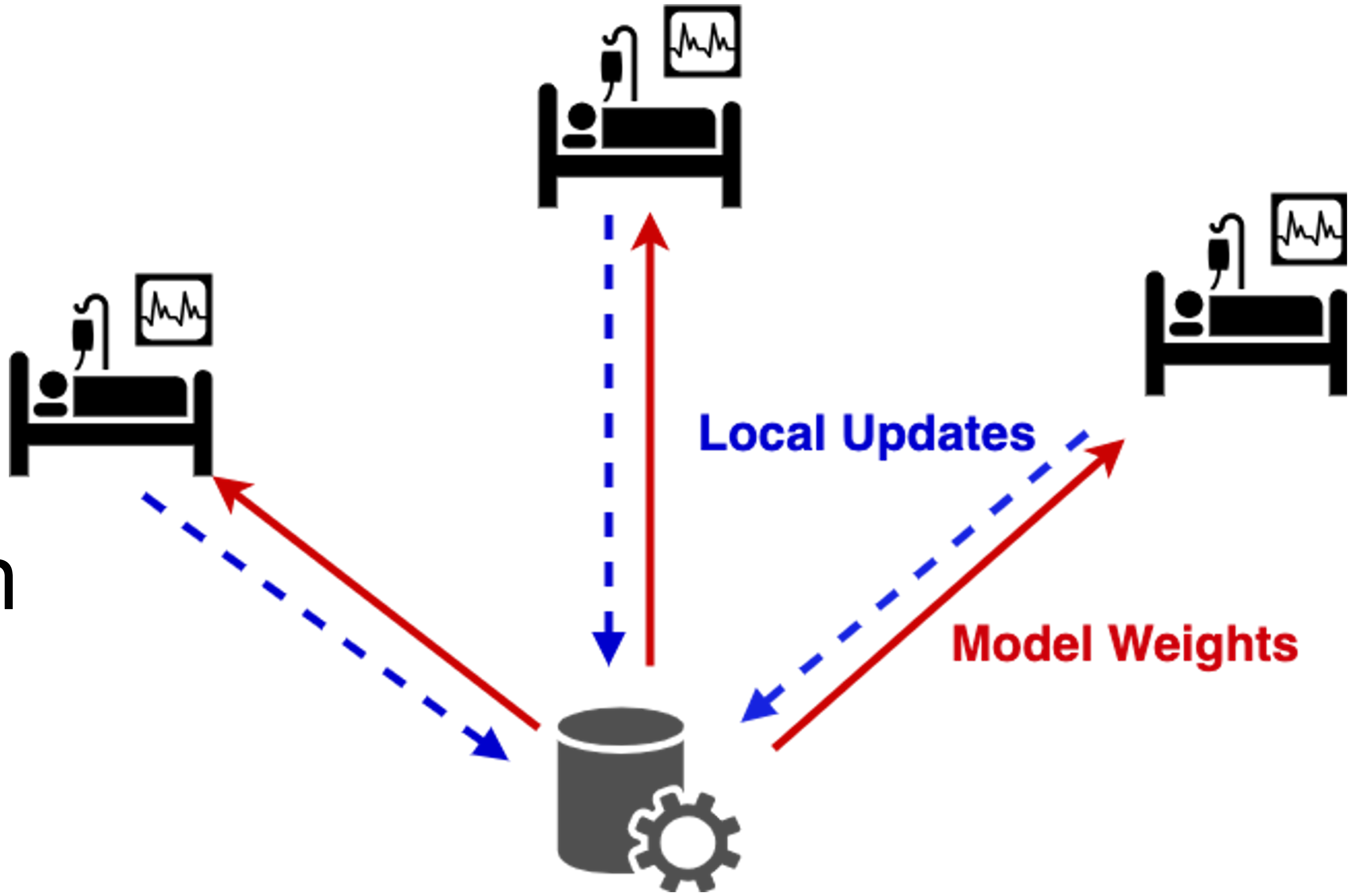
- Determine conditions so that LSR factors are (locally) identifiable.
- Understand the analytical properties of the LSR set.
- Find a convergence analysis for alternating minimization.

Federated learning from tensor valued data

Tensor data are often hard to acquire

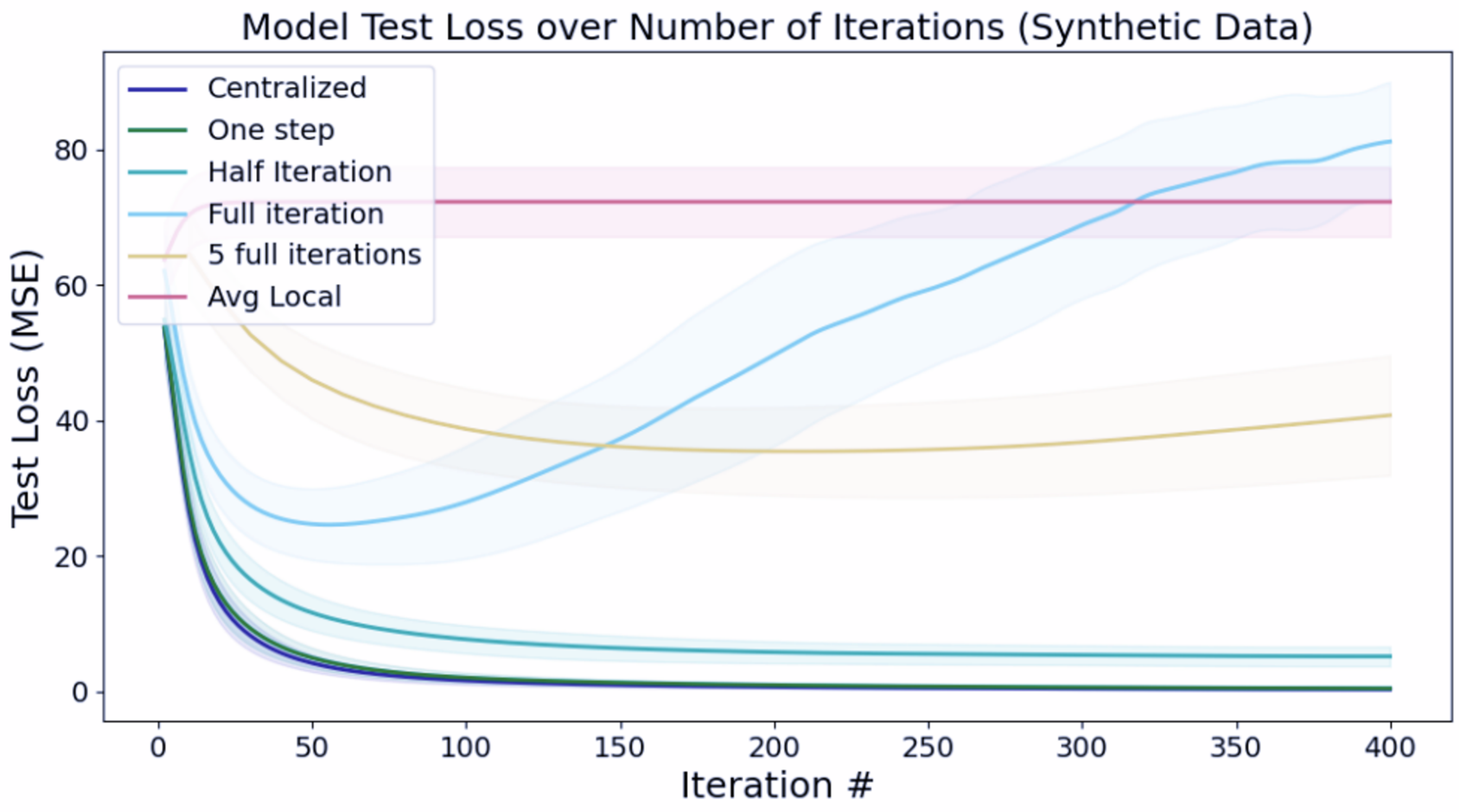
In "federated learning" we want to efficiently learn from data which are held at different sites.

If we have MRI data at different research groups, can we still train a regression model with limited communication?



Balancing local and global updates

Empirical results are promising but preliminary



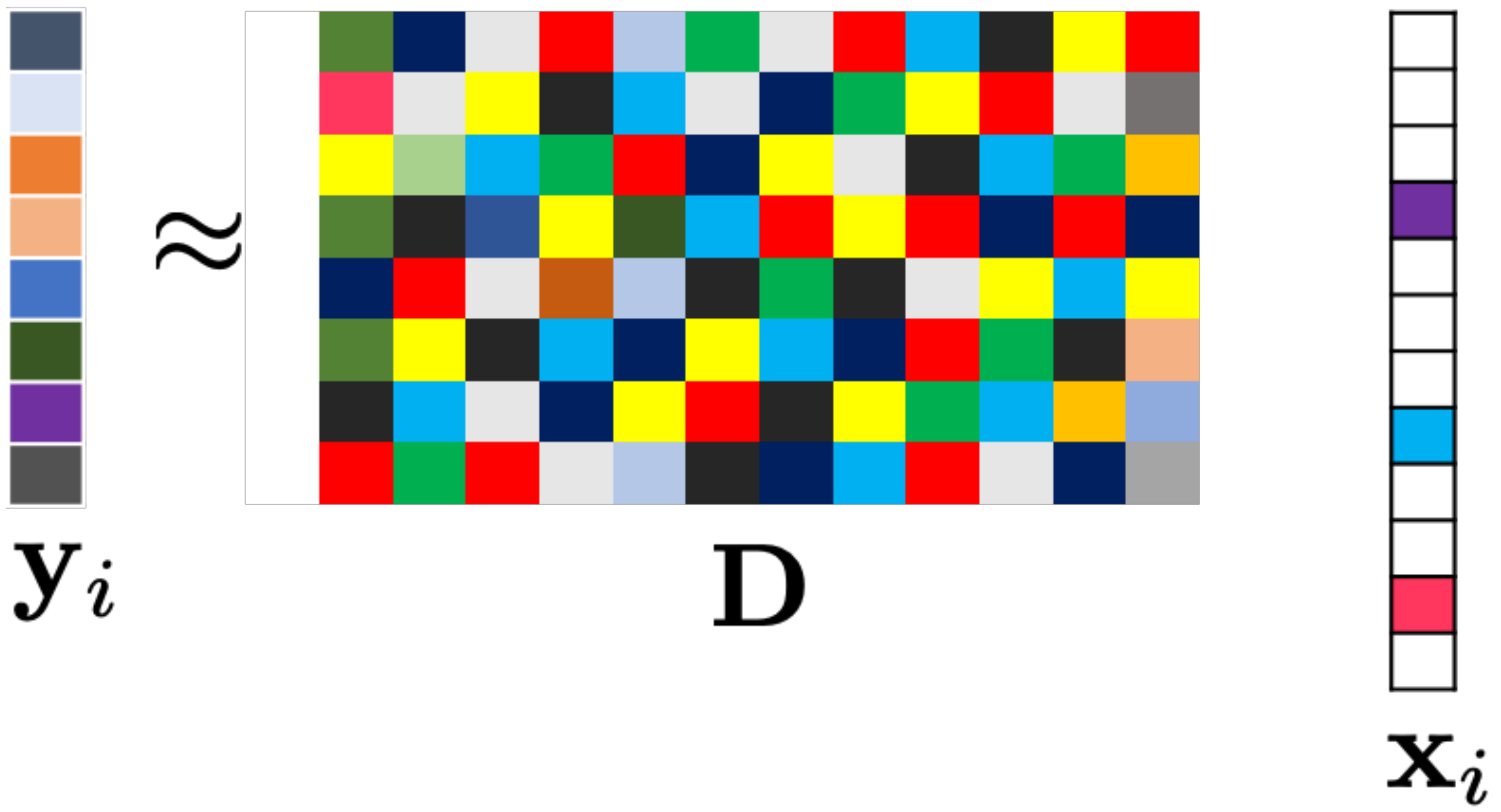
(Sanchez, Taki, Bajwa, S., 2024)

- Need tight coupling between local and centralized updates.
- Poses a challenge when communication reliability is a bottleneck.
- Lots of interesting work on the applications/engineering side!

Representation learning with structured tensors (optional)

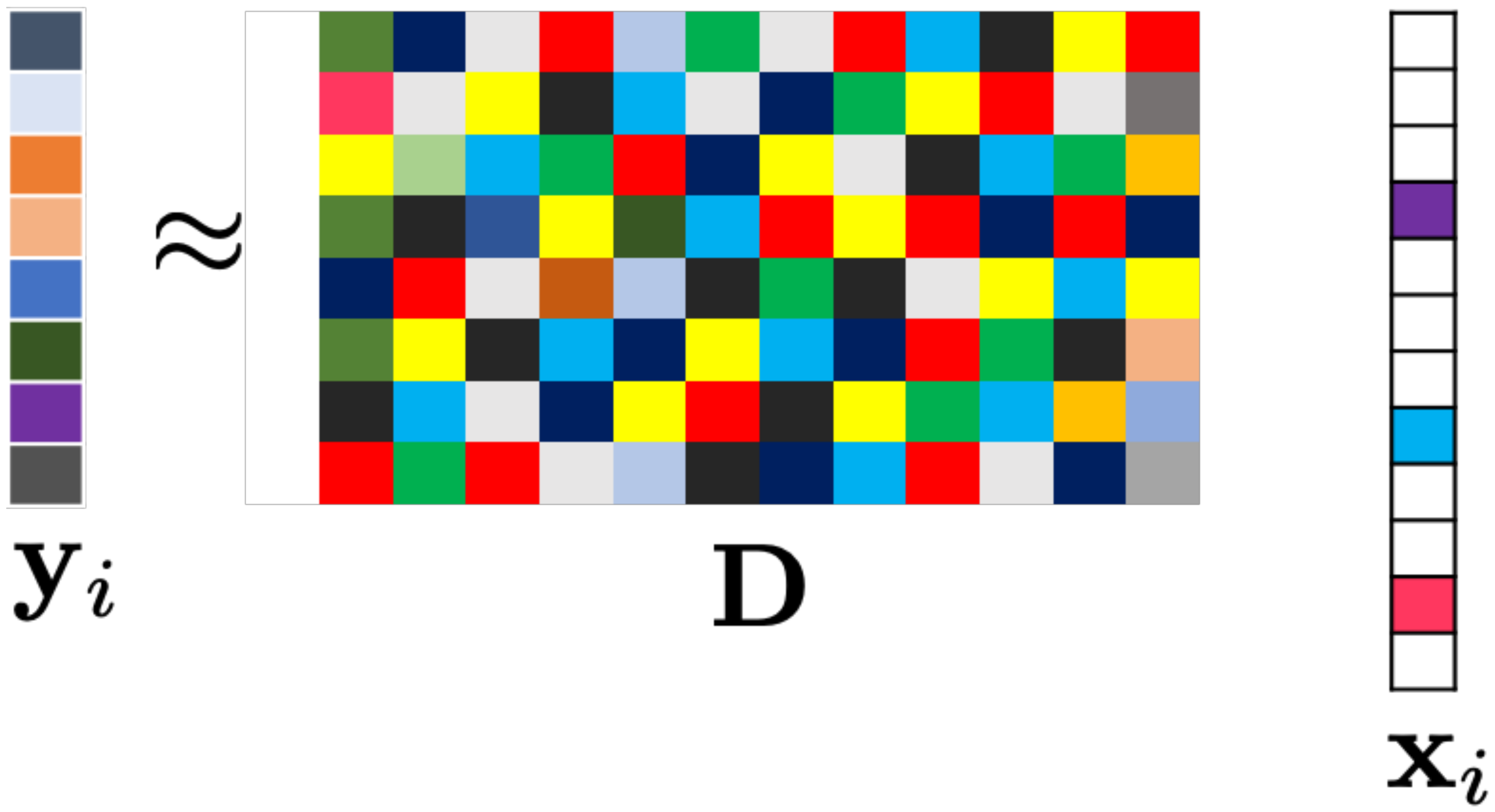
Dictionary learning

Sparse representation in one slide



Dictionary learning

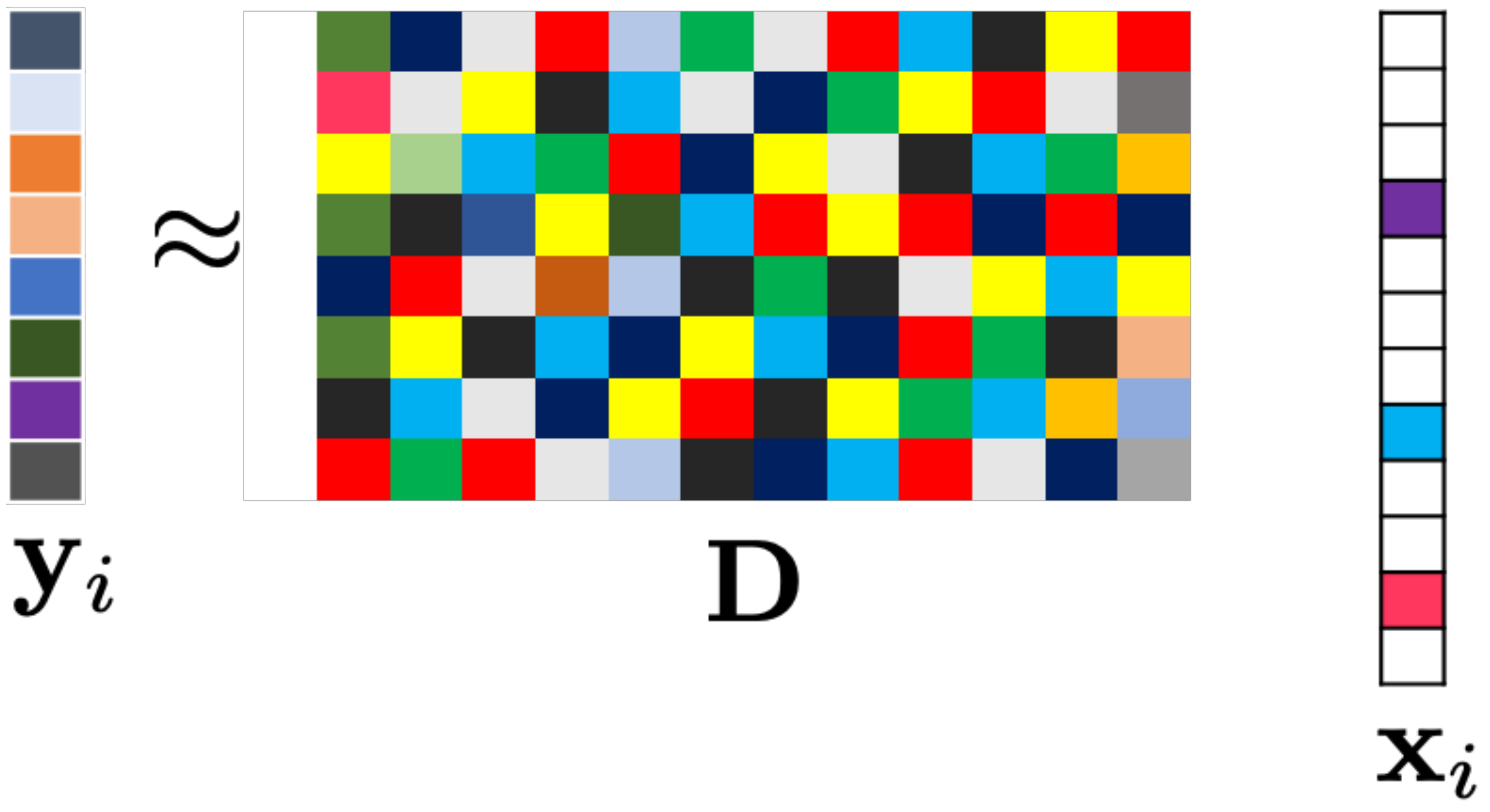
Sparse representation in one slide



Given data $\{\mathbf{y}_i\}$, learn a sparse representation:

Dictionary learning

Sparse representation in one slide

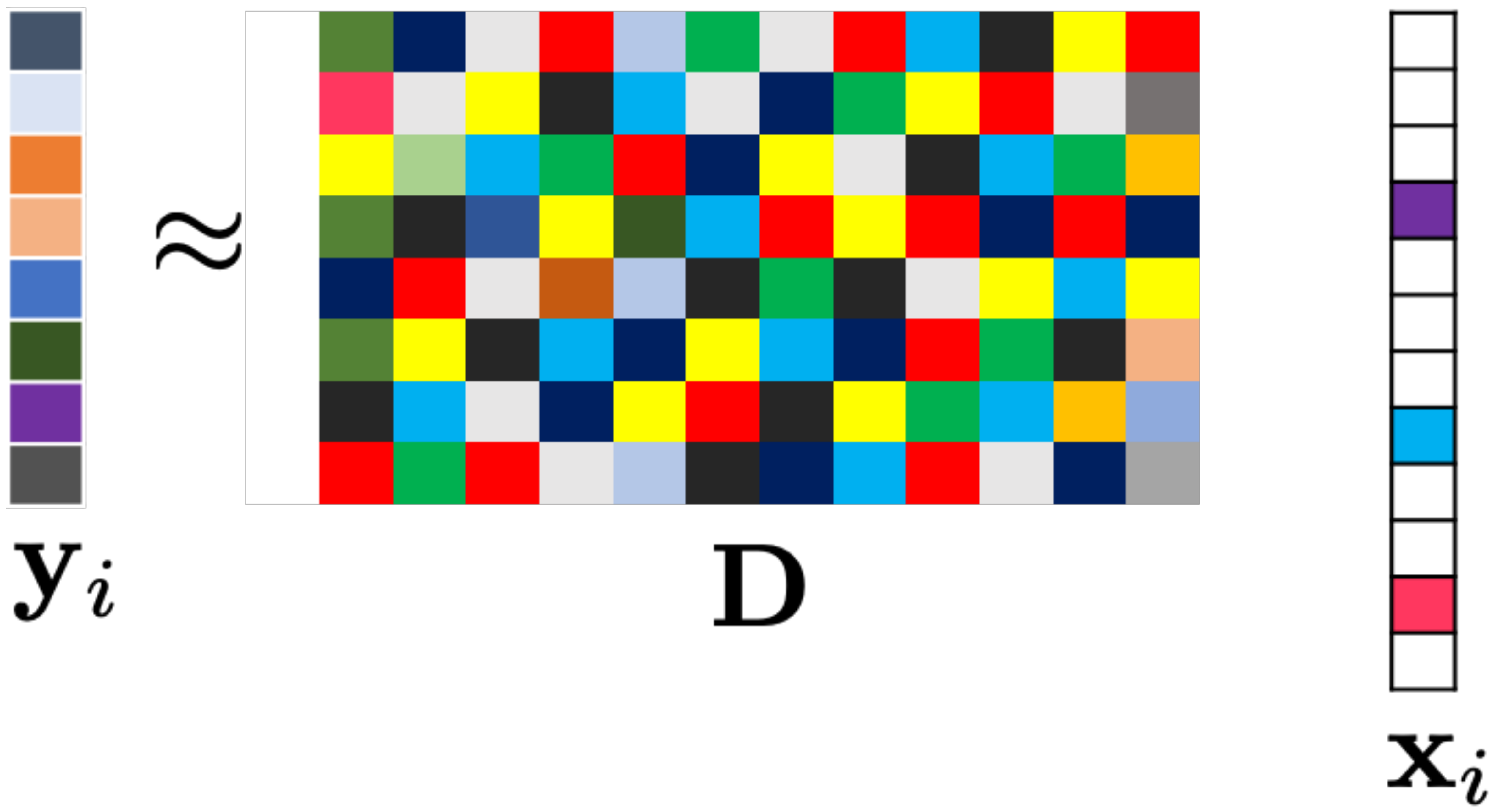


Given data $\{\mathbf{y}_i\}$, learn a sparse representation:

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i + \mathbf{w}_i.$$

Dictionary learning

Sparse representation in one slide



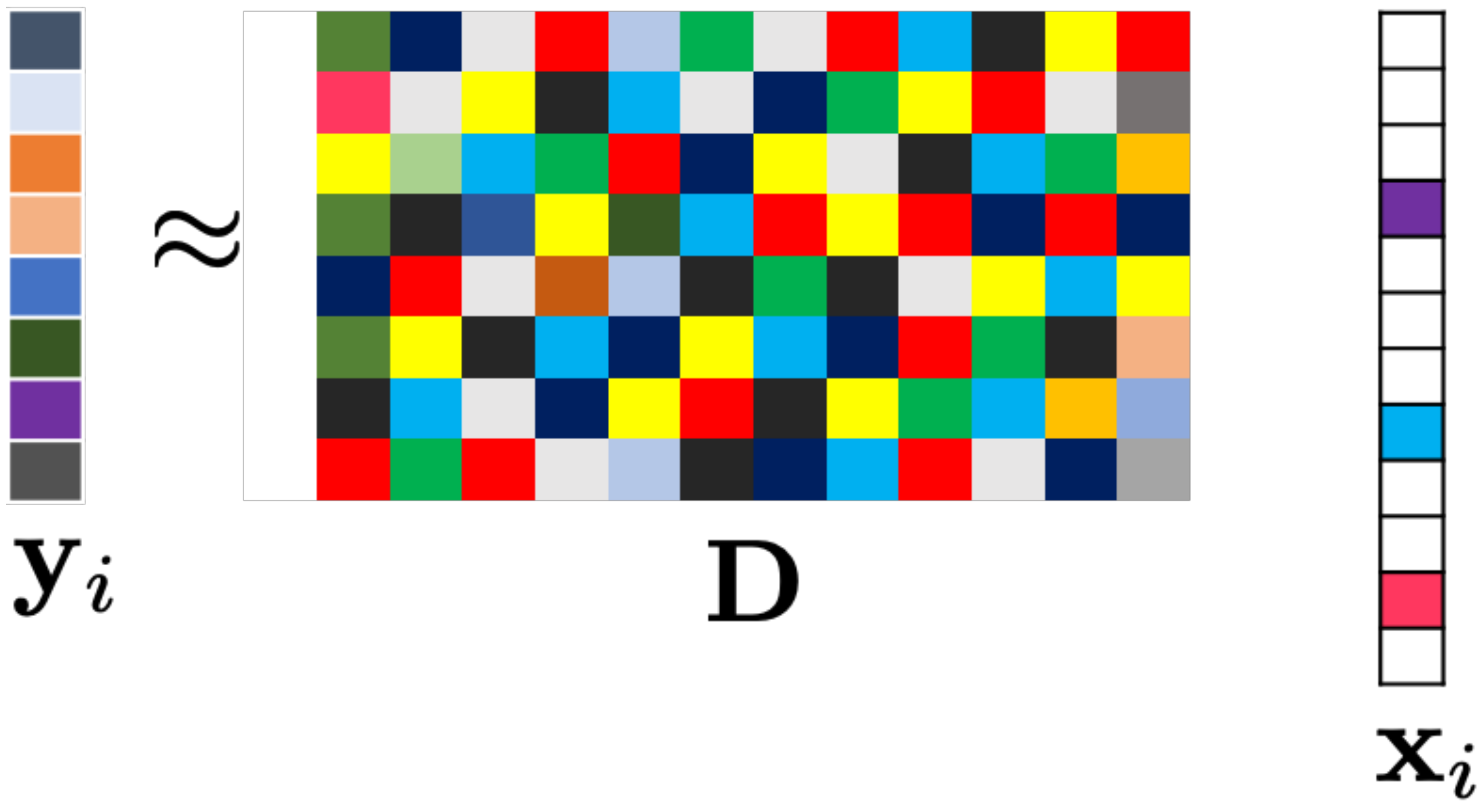
Given data $\{\mathbf{y}_i\}$, learn a sparse representation:

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i + \mathbf{w}_i.$$

$\mathbf{D}$ is a *dictionary* whose columns are *atoms*.

Dictionary learning

Sparse representation in one slide



Given data $\{\mathbf{y}_i\}$, learn a sparse representation:

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i + \mathbf{w}_i.$$

$\mathbf{D}$ is a *dictionary* whose columns are *atoms*.

Coefficient vector $\mathbf{x}_i$ selects s columns of $\mathbf{D}$.

Dictionary learning for tensor data

How can we do the same thing but for tensors?

Dictionary learning for tensor data

How can we do the same thing but for tensors?

We observe tensor data $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_L \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$. Can we learn a sparse representation for this data?

Dictionary learning for tensor data

How can we do the same thing but for tensors?

We observe tensor data $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_L \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$. Can we learn a sparse representation for this data?

Look at the vectorized model:

Dictionary learning for tensor data

How can we do the same thing but for tensors?

We observe tensor data $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_L \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$. Can we learn a sparse representation for this data?

Look at the vectorized model:

$$\mathrm{vec}(\underline{\mathbf{Y}}_i) = \mathbf{y}_i \approx \mathbf{D}\mathbf{x}_i$$

Dictionary learning for tensor data

How can we do the same thing but for tensors?

We observe tensor data $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_L \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$. Can we learn a sparse representation for this data?

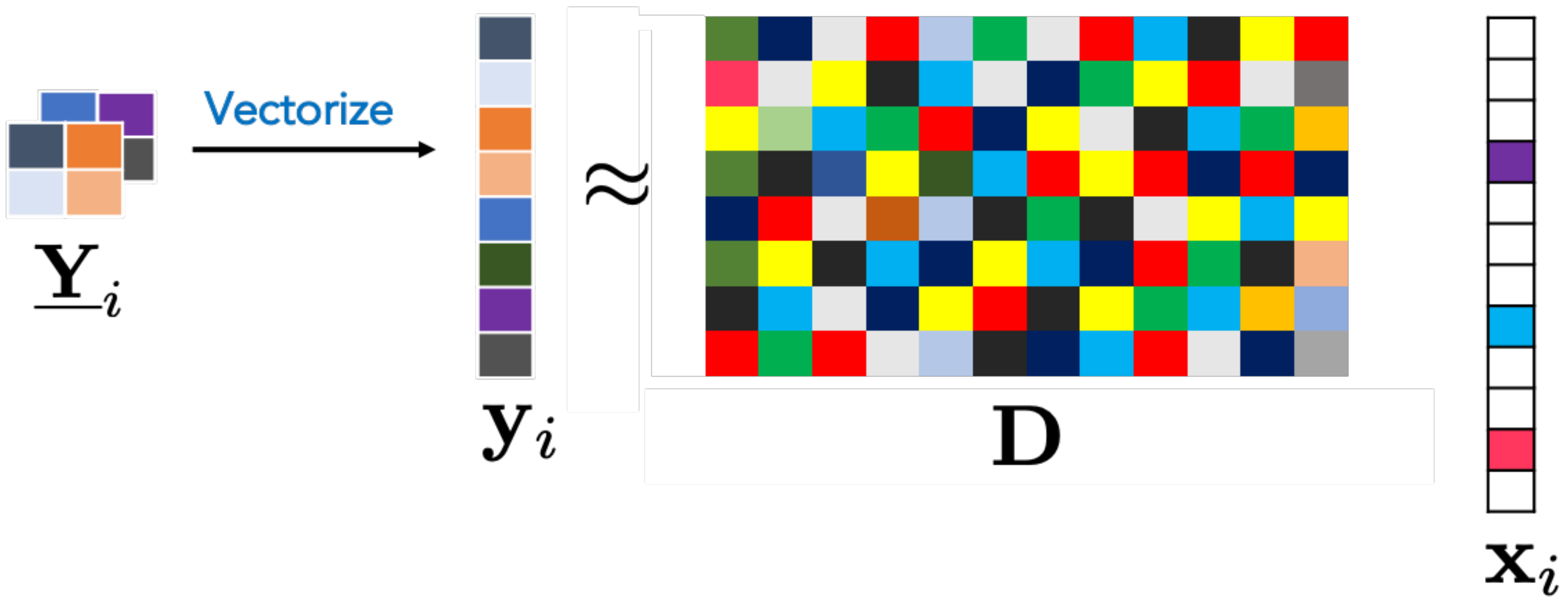
Look at the vectorized model:

$$\mathrm{vec}(\underline{\mathbf{Y}}_i) = \mathbf{y}_i \approx \mathbf{D}\mathbf{x}_i$$

We want to estimate a *dictionary* $\mathbf{D} \in \mathbb{R}^{m \times p}$ such that the coefficient vectors $\mathbf{x}_i$ are *sparse*. Here $m = \prod_k m_k$.

Default approach: vectorize

What if we ignore the tensor structure?



Tensor decompositions to the rescue

What if our dictionary has a Tucker structure?

Tensor decompositions to the rescue

What if our dictionary has a Tucker structure?

A Tucker-structured dictionary:

$$\underbrace{\underline{\mathbf{Y}}}_{\in \mathbb{R}^{m_1 \times \cdots \times m_N}} = \overbrace{\underbrace{\underline{\mathbf{X}}}_{\in \mathbb{R}^{p_1 \times \cdots \times p_N}}}^{\text{Sparse core tensor}} \times_1 \underbrace{\mathbf{D}_1}_{\in \mathbb{R}^{m_1 \times p_1}} \times_2 \cdots \times_N \underbrace{\mathbf{D}_K}_{\in \mathbb{R}^{m_K \times p_K}} + \underline{\mathbf{W}}$$

Tensor decompositions to the rescue

What if our dictionary has a Tucker structure?

A Tucker-structured dictionary:

$$\underbrace{\underline{\mathbf{Y}}}_{\in \mathbb{R}^{m_1 \times \cdots \times m_N}} = \overbrace{\underbrace{\underline{\mathbf{X}}}_{\in \mathbb{R}^{p_1 \times \cdots \times p_N}}}^{\text{Sparse core tensor}} \times_1 \underbrace{\mathbf{D}_1}_{\in \mathbb{R}^{m_1 \times p_1}} \times_2 \cdots \times_N \underbrace{\mathbf{D}_K}_{\in \mathbb{R}^{m_K \times p_K}} + \underline{\mathbf{W}}$$

What happens when we vectorize?

$$\underbrace{\mathbf{y}}_{\in \mathbb{R}^{m}} = \left(\mathbf{D}_K \otimes \mathbf{D}_{K-1} \otimes \cdots \otimes \mathbf{D}_1\right) \underbrace{\mathbf{x}}_{\in \mathbb{R}^{p}} + \mathbf{w}$$

Kronecker-structured (KS) dictionary learning

The difference that structure can make

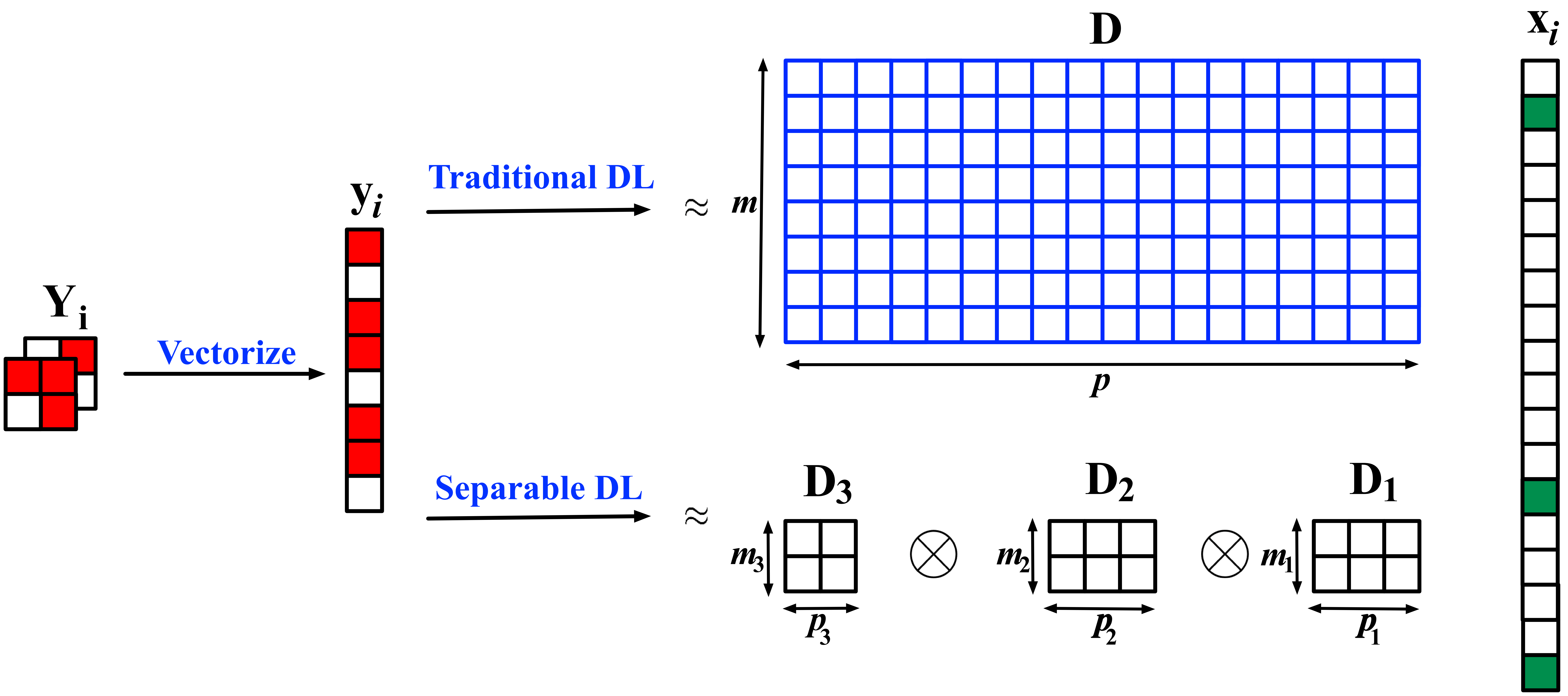
- **Traditional (unstructured) dictionary learning:** MOD (Engan, Rao, Kreutz-Delgado ’99), K-SVD (Aharon, Elad, Bruckstein ’06), Online DL (Mairal et al. ’09)

- **KS dictionary learning:** K-HOSVD (Roemer, Del Galdo, Haardt ’14), GradTensor (Zubair and Wang ’13), SeDiL (Hawe, Seibert, Kleinsteuber ’13), SuKro (Dantas, Da Costa, Lopes ’17)

- **Our work:** use LSR structure for the dictionary to allow more flexible parameterization.

Kronecker-structured (KS) dictionary learning

The difference that structure can make



Even a KS assumption can help

Reducing the number of parameters can make a huge difference

Original Image

Noisy Image

Unstructured DL:
147456 parameters

Separable DL:
265 parameters

Comparison to unstructured dictionaries

Using decompositions helps a lot!

	Vectorized DL	KS-DL
Minimax lower bound	$\frac{mp^2}{\varepsilon^2}$	$\frac{p \sum_k m_k p_k}{K\varepsilon^2}$
Achievability bound	$\frac{mp^3}{\varepsilon^2}$	$\max_k \frac{m_k p_k^3}{\varepsilon_k^2}$

Minimax bound for the vector case: Jung et al. (2015)

Achievability bound for the vector case: Gribonval et al. (2015)

Generative model for structured dictionaries

Defining the set of all LSR matrices

Generative model for structured dictionaries

Defining the set of all LSR matrices

Define the set of all LSR dictionaries:

Generative model for structured dictionaries

Defining the set of all LSR matrices

Define the set of all LSR dictionaries:

$$\mathscr{D}_{\mathsf{LSR},S} = \left\{ \mathbf{D} \in \mathbb{R}^{m \times p} : \mathbf{D} = \sum_{s=1}^{S} \bigotimes_{k} \mathbf{D}_{(k,s)} \right\}$$

Generative model for structured dictionaries

Defining the set of all LSR matrices

Define the set of all LSR dictionaries:

$$\mathscr{D}_{\mathsf{LSR},S} = \left\{ \mathbf{D} \in \mathbb{R}^{m \times p} : \mathbf{D} = \sum_{s=1}^{S} \bigotimes_{k} \mathbf{D}_{(k,s)} \right\}$$

where each $\mathbf{D}_{(k,s)} \in \mathbb{R}^{m_k \times p_k}$ has unit norm columns.

Generative model for structured dictionaries

Defining the set of all LSR matrices

Define the set of all LSR dictionaries:

$$\mathscr{D}_{\mathsf{LSR},S} = \left\{ \mathbf{D} \in \mathbb{R}^{m \times p} : \mathbf{D} = \sum_{s=1}^{S} \bigotimes_{k} \mathbf{D}_{(k,s)} \right\}$$

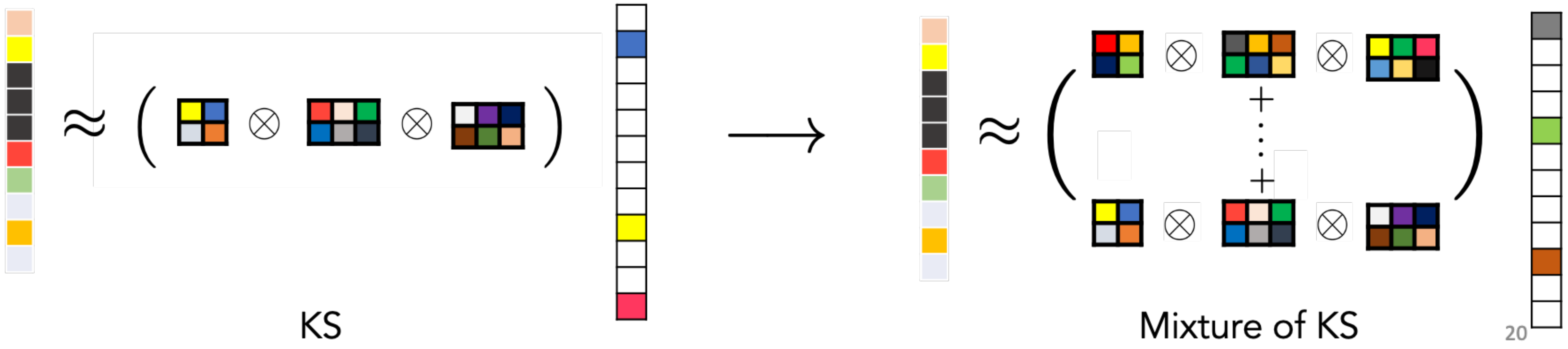
where each $\mathbf{D}_{(k,s)} \in \mathbb{R}^{m_k \times p_k}$ has unit norm columns.

Assume our data comes from a true model $\mathbf{D}^0 \in \mathscr{D}_{\mathsf{LSR},S}$:

$$\mathbf{y}_i = \mathbf{D}^0 \mathbf{x}_i + \mathbf{w}_i$$

But can we do better with higher S?

Extending to LSR dictionaries



Because the core tensor (coefficient vector) is sparse, we can apply the LSR decomposition to the dictionary:

$$\mathbf{D} = \sum_{s=1}^{S} \mathbf{D}_{(K,s)} \otimes \cdots \otimes \mathbf{D}_{(2,s)} \otimes \mathbf{D}_{(1,s)}$$

Identifiability for general S

Local recovery guarantees

For general S and LSR structured dictionaries, we can show (Ghassemi et al, 2020) the following upper bound on n:

$$n = O\left(\frac{p^2 \sum_k m_k p_k}{\varepsilon_k^2}\right)$$

Proof ingredients: need to understand topological properties of $\mathcal{D}_{\mathrm{LSR},S}$ and related spaces as well as covering numbers, etc.

Practical algorithms

Unfortunately, separation rank is also NP hard

We propose two estimators for learning LSR dictionaries (Ghassemi et al, 2020):

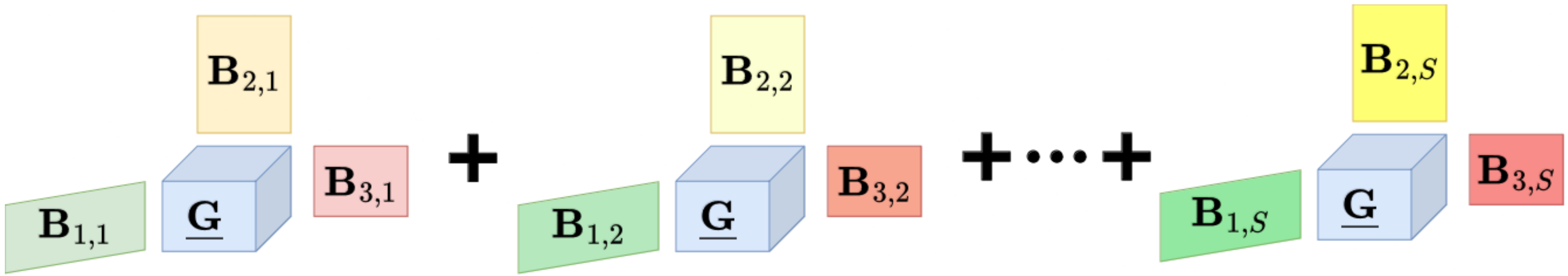
- **Regularization-based:** use a sum-trace-norm on unfolding together with ADMM.
- **Factorization-based:** explicitly optimize over the factors in the LSR decomposition.

Compares well to K-SVD (Aharon et al. 2006) and SediL (Hawe et al. 2013): see Ghassemi et al. (2020) for details.

Recap and looking forward

Recap of what we've seen

Tensor decompositions for everyone!



There is a whole continuum of tensor decompositions and **LSR structured tensors** can be very useful:

- Adapt parameterization to the data available.
- Efficiently (empirically) learnable/estimatable.

Many mathematical questions remain

So many fun (and fundamental) questions!

Many mathematical questions remain

So many fun (and fundamental) questions!

Approximation theory: how can we find a good approximation to a given tensor?

Many mathematical questions remain

So many fun (and fundamental) questions!

Approximation theory: how can we find a good approximation to a given tensor?

Topology: How can we practically manage pathologies in tensor-land?

Many mathematical questions remain

So many fun (and fundamental) questions!

Approximation theory: how can we find a good approximation to a given tensor?

Topology: How can we practically manage pathologies in tensor-land?

Optimization: What are the right ways to do a convex relaxation of an LSR constraint?

Many mathematical questions remain

So many fun (and fundamental) questions!

Approximation theory: how can we find a good approximation to a given tensor?

Topology: How can we practically manage pathologies in tensor-land?

Optimization: What are the right ways to do a convex relaxation of an LSR constraint?

RTT: What about random tensors or random tensors with low “rank” or “simpler” structure?

شُكْراً جَزيلاً

Parameters of interest

Along with some assumptions on the model

- **Sample size:** number of observations n
- **Tensor order:** K
- **Dictionary sizes:** $\{(m_k, p_k) : k = 1,2,\ldots, K\}$
- **Coefficient energy:** the $\mathbf{x}_i$ are i.i.d. with variance σ_x^2
- **SNR:** $\dfrac{q\sigma_x^2}{m\sigma^2}$, where q is the sparsity level

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Define the error of an dictionary estimator $\hat{\mathbf{D}}$:

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Define the error of an dictionary estimator $\hat{\mathbf{D}}$:

$$\varepsilon = \left\| \mathbf{D} - \hat{\mathbf{D}}(\mathbf{Y}) \right\|_F.$$

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Define the error of an dictionary estimator $\hat{\mathbf{D}}$:

$$\varepsilon = \left\| \mathbf{D} - \hat{\mathbf{D}}(\mathbf{Y}) \right\|_F.$$

For fixed SNR, and $S = 1$ we have the following lower bound on n (Zakeri, Bajwa, S. 2018):

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Define the error of an dictionary estimator $\hat{\mathbf{D}}$:

$$\varepsilon = \left\| \mathbf{D} - \hat{\mathbf{D}}(\mathbf{Y}) \right\|_F.$$

For fixed SNR, and $S = 1$ we have the following lower bound on n (Zakeri, Bajwa, S. 2018):

$$n = \Omega\left(\frac{p \sum_k m_k p_k}{K \varepsilon^2} \right)$$

Minimax lower bounds for $S = 1$

The special case of Kronecker-structured (KS) dictionaries

Define the error of an dictionary estimator $\hat{\mathbf{D}}$:

$$\varepsilon = \left\| \mathbf{D} - \hat{\mathbf{D}}(\mathbf{Y}) \right\|_F.$$

For fixed SNR, and $S = 1$ we have the following lower bound on n (Zakeri, Bajwa, S. 2018):

$$n = \Omega\left(\frac{p \sum_k m_k p_k}{K \varepsilon^2} \right)$$

Proof idea: construct a packing in $\mathscr{D}_{\mathsf{LSR},1}$ and use Fano's inqualilty.

Identifiability when $S = 1$

Local recovery guarantees

Identifiability when $S = 1$

Local recovery guarantees

Suppose we want to recover each factor dictionary $\mathbf{D}_k$:

Identifiability when $S = 1$

Local recovery guarantees

Suppose we want to recover each factor dictionary $\mathbf{D}_k$:

$$\varepsilon_k = \left\| \mathbf{D}_k - \hat{\mathbf{D}}_k(\mathbf{Y}) \right\|_F.$$

Identifiability when $S = 1$

Local recovery guarantees

Suppose we want to recover each factor dictionary $\mathbf{D}_k$:

$$\varepsilon_k = \left\| \mathbf{D}_k - \hat{\mathbf{D}}_k(\mathbf{Y}) \right\|_F.$$

Then we have (Zakeri, S., Bajwa, 2018) the following upper bound on n:

Identifiability when $S = 1$

Local recovery guarantees

Suppose we want to recover each factor dictionary $\mathbf{D}_k$:

$$\varepsilon_k = \left\| \mathbf{D}_k - \hat{\mathbf{D}}_k(\mathbf{Y}) \right\|_F.$$

Then we have (Zakeri, S., Bajwa, 2018) the following upper bound on n:

$$n = O\left(\max_k \frac{m_k p_k^3}{\varepsilon_k^2} \right)$$