

# Artificial Models for Music Creativity

## Lesson 4 - Introduction to high-dimensional data structure

Artificial Models for Multimodal Creativity  
Alessandro Anatrini - 4.1.2025

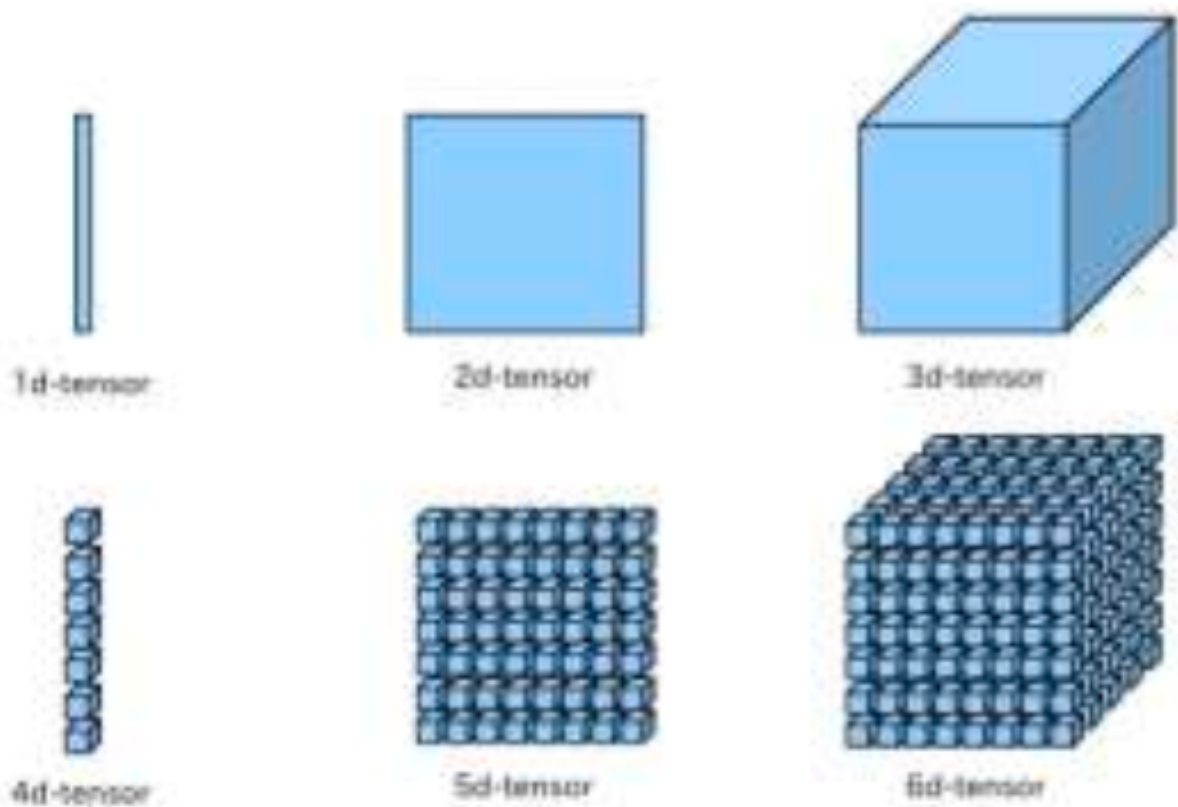
# FOUR MYTHS ON ARTIFICIAL INTELLIGENCE

1. Tensor: Dimension
2. Tensor: Shape
3. Real world data: Which tensor?

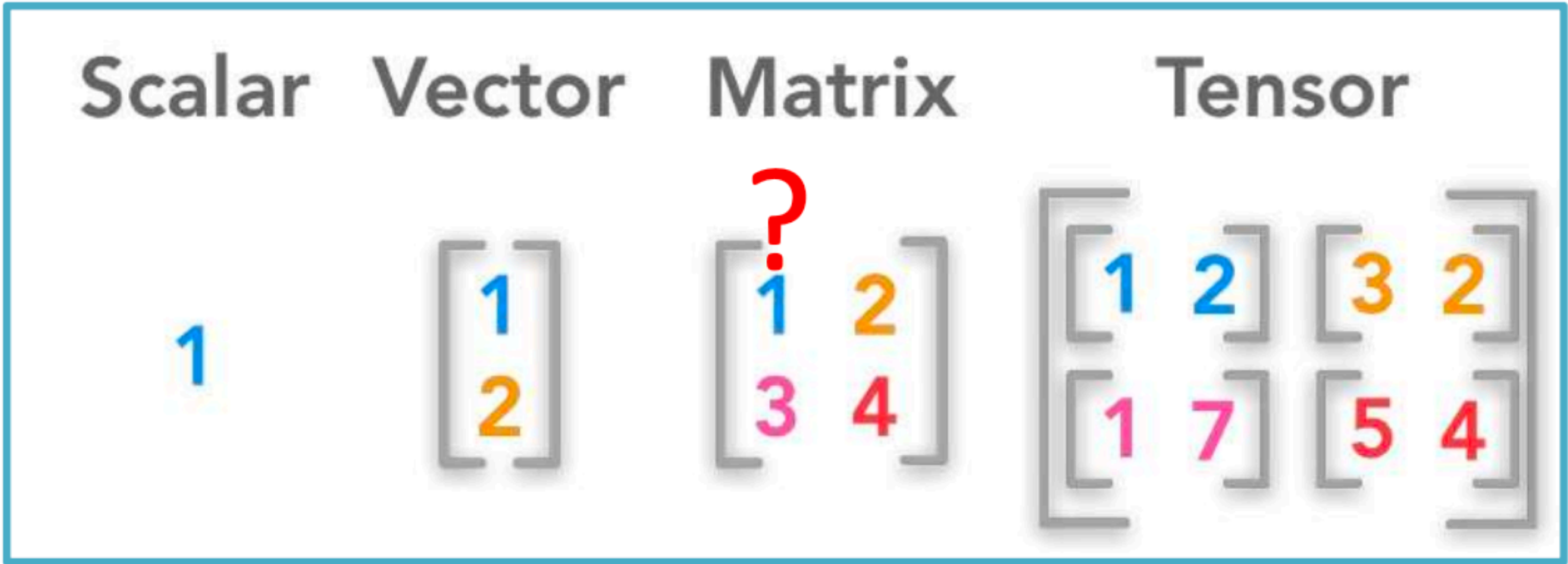
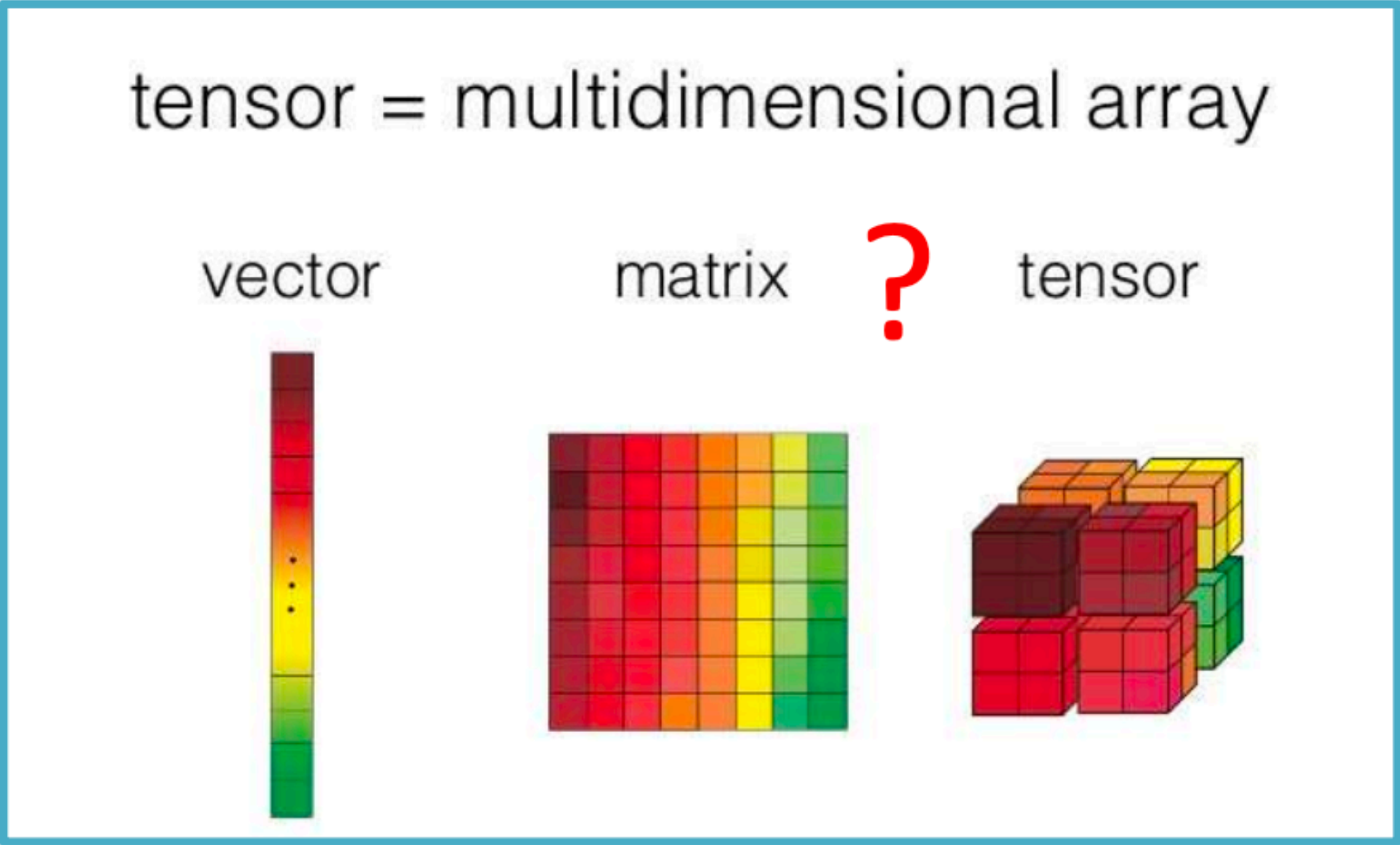
# WTF IS A TENSOR?

A tensor is a container for data

Data are almost numerical data

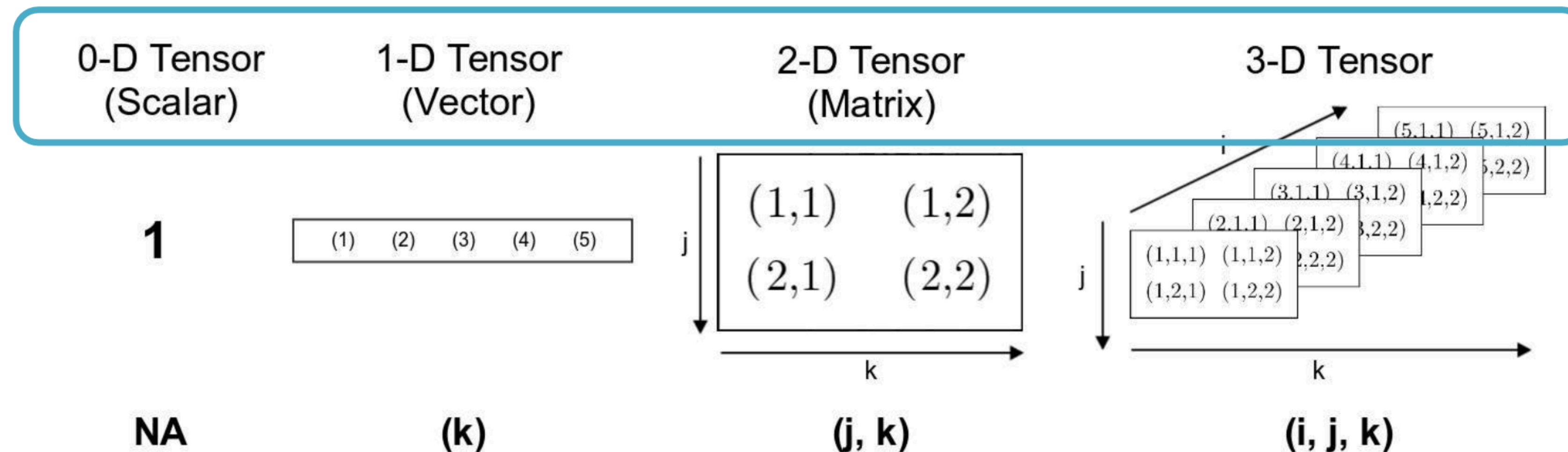


A tensor is a general name for a multi-way array data. 1d-tensor is a vector, 2d is a matrix, 3d is a cube. We can imagine a 4d-tensor as a vector of cubes, 5d as a matrix of cubes and 6d as a cube of cubes.



## KEYWORDS

1. Generalisation of matrices to an arbitrary number of dimensions
2. In tensor dimension is often called axis
3. No. of dimensions (=axis) is called ranks



# SCALAR - 0D TENSOR

A tensor that contains only one number is called scalar

```
In [1]: import numpy as np
```

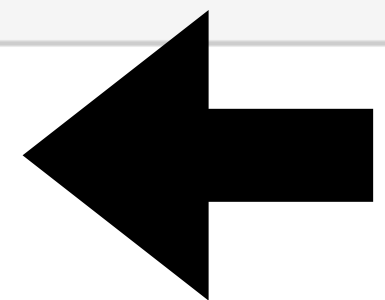
```
In [2]: x = np.array(12)
```

```
In [3]: x
```

```
Out[3]: array(12)
```

```
In [4]: x.ndim
```

```
Out[4]: 0
```



dimension can be shown using the `ndim` method

## VECTOR - 1D TENSOR

An array of numbers is called vector or 1d tensor

```
In [1]: import numpy as np
```

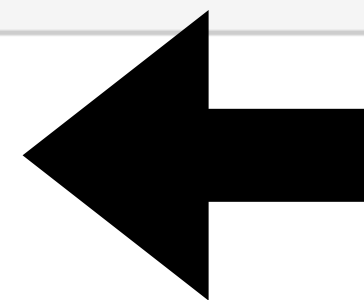
```
In [5]: x = np.array([23, 45, 66, 2])
```

```
In [6]: x
```

```
Out[6]: array([23, 45, 66,  2])
```

```
In [7]: x.ndim
```

```
Out[7]: 1
```





# MATRIX - 2D TENSOR

An array of vectors is called matrix or 2d tensor

```
In [1]: import numpy as np
```

```
In [26]: x = np.array([[23, 45, 66, 2],  
                      [12, 44, 31, 89],  
                      [72, 49, 20, 3]])
```

```
In [27]: x
```

```
Out[27]: array([[23, 45, 66,  2],  
               [12, 44, 31, 89],  
               [72, 49, 20,  3]])
```

```
In [28]: x.ndim
```

```
Out[28]: 2
```



# 3D TENSOR

From 3d-tensor on it is just  
nd-tensor

```
In [1]: import numpy as np
```

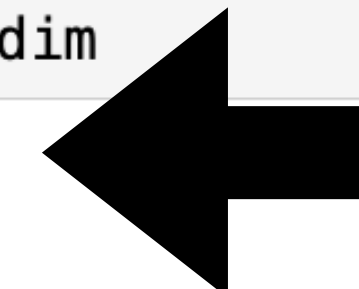
```
In [12]: x = np.array([[[1, 3, 5, 7],  
                        [2, 4, 6, 8],  
                        [3, 6, 9, 12]],  
                      [[1, 3, 5, 7],  
                       [2, 4, 6, 8],  
                       [3, 6, 9, 12]],  
                      [[1, 3, 5, 7],  
                       [2, 4, 6, 8],  
                       [3, 6, 9, 12]]])
```

```
In [13]: x
```

```
Out[13]: array([[[ 1,  3,  5,  7],  
                [ 2,  4,  6,  8],  
                [ 3,  6,  9, 12]],  
               [[ 1,  3,  5,  7],  
                [ 2,  4,  6,  8],  
                [ 3,  6,  9, 12]],  
               [[ 1,  3,  5,  7],  
                [ 2,  4,  6,  8],  
                [ 3,  6,  9, 12]]])
```

```
In [14]: x.ndim
```

```
Out[14]: 3
```





1. Number of dimensions (=axes)
2. Shape: how many dimensions the tensor has along each axis
3. Data type: dtype in python (float32, float64, unit8...)
4. Shape is a key element in DL programming

# SCALAR - 0D TENSOR

## Scalar has empty shape

```
In [1]: import numpy as np
```

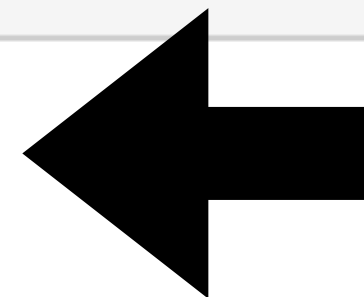
```
In [17]: x = np.array(3)
```

```
In [18]: x.ndim
```

```
Out[18]: 0
```

```
In [19]: x.shape
```

```
Out[19]: ()
```



# VECTOR - 1D TENSOR

1d tensor has a shape with a single element

```
In [1]: import numpy as np
```

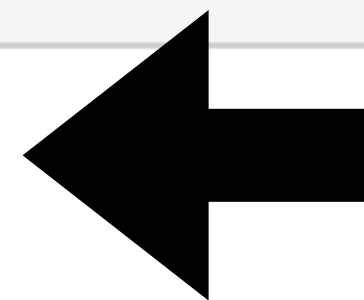
```
In [20]: x = np.array([23, 45, 66, 2])
```

```
In [21]: x.ndim
```

```
Out[21]: 1
```

```
In [22]: x.shape
```

```
Out[22]: (4,)
```



# VECTOR - 1D TENSOR

## 2d tensor is a matrix

```
In [1]: import numpy as np
```

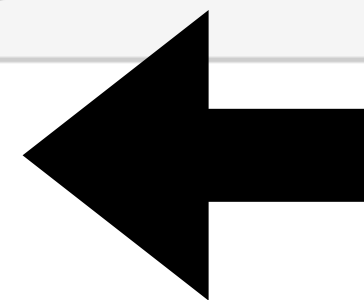
```
In [23]: x = np.array([[23, 45, 66, 2],  
                      [12, 44, 31, 89],  
                      [72, 49, 20, 3]])
```

```
In [24]: x.ndim
```

```
Out[24]: 2
```

```
In [25]: x.shape
```

```
Out[25]: (3, 4)
```



# 3D TENSOR

a 3d tensor's shape  
is represented with  
3 numbers

```
In [1]: import numpy as np
```

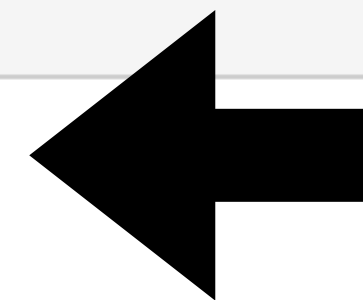
```
In [12]: x = np.array([[[1, 3, 5, 7],  
                        [2, 4, 6, 8],  
                        [3, 6, 9, 12]],  
                      [[1, 3, 5, 7],  
                       [2, 4, 6, 8],  
                       [3, 6, 9, 12]],  
                      [[1, 3, 5, 7],  
                       [2, 4, 6, 8],  
                       [3, 6, 9, 12]]])
```

```
In [15]: x.ndim
```

```
Out[15]: 3
```

```
In [16]: x.shape
```

```
Out[16]: (3, 3, 4)
```



dimension	0	1	2	3	4
name	scalar	vector	matrix	3d tensor	4d tensor
aka	0d tensor	1d tensor	2d tensor	3d tensor	4d tensor
example	12	[23, 45, 66, 2]	[[23, 45, 66, 2], [12, 44, 31, 89], [72, 49, 20, 3]]	[[[1, 3, 5, 7], [2, 4, 6, 8], [3, 6, 9, 12]], [[1, 3, 5, 7], [2, 4, 6, 8], [3, 6, 9, 12]], [[1, 3, 5, 7], [2, 4, 6, 8], [3, 6, 9, 12]]]	...
shape	()	(4)	(3,4)	(3, 3, 4)	(5, 3, 3, 4)





# REAL WORLD EXAMPLES

Name	Tensor	Shape
Vector data*	2D tensor	(samples, feature)
Timeseries data or sequence data	3D tensor	(samples, timesteps, features)
Images	4D tensor	(samples, height, width, channels)
Video	5D tensor	(samples, frames, height, width, channels)



# REAL WORLD EXAMPLES

2d tensor: actual personal data

3

Samples	Age	ZIP code	Income
1	12	123-324	10k
2	34	234-567	13k
3	12	349-874	20k
...			
9,999	45	874-988	30k
10,000	56	888-234	12k

10,000

- Numpy array

[[12, 123-324,10k],  
[34,234-567,13k],  
...  
[56 ,888-234, 12k]]

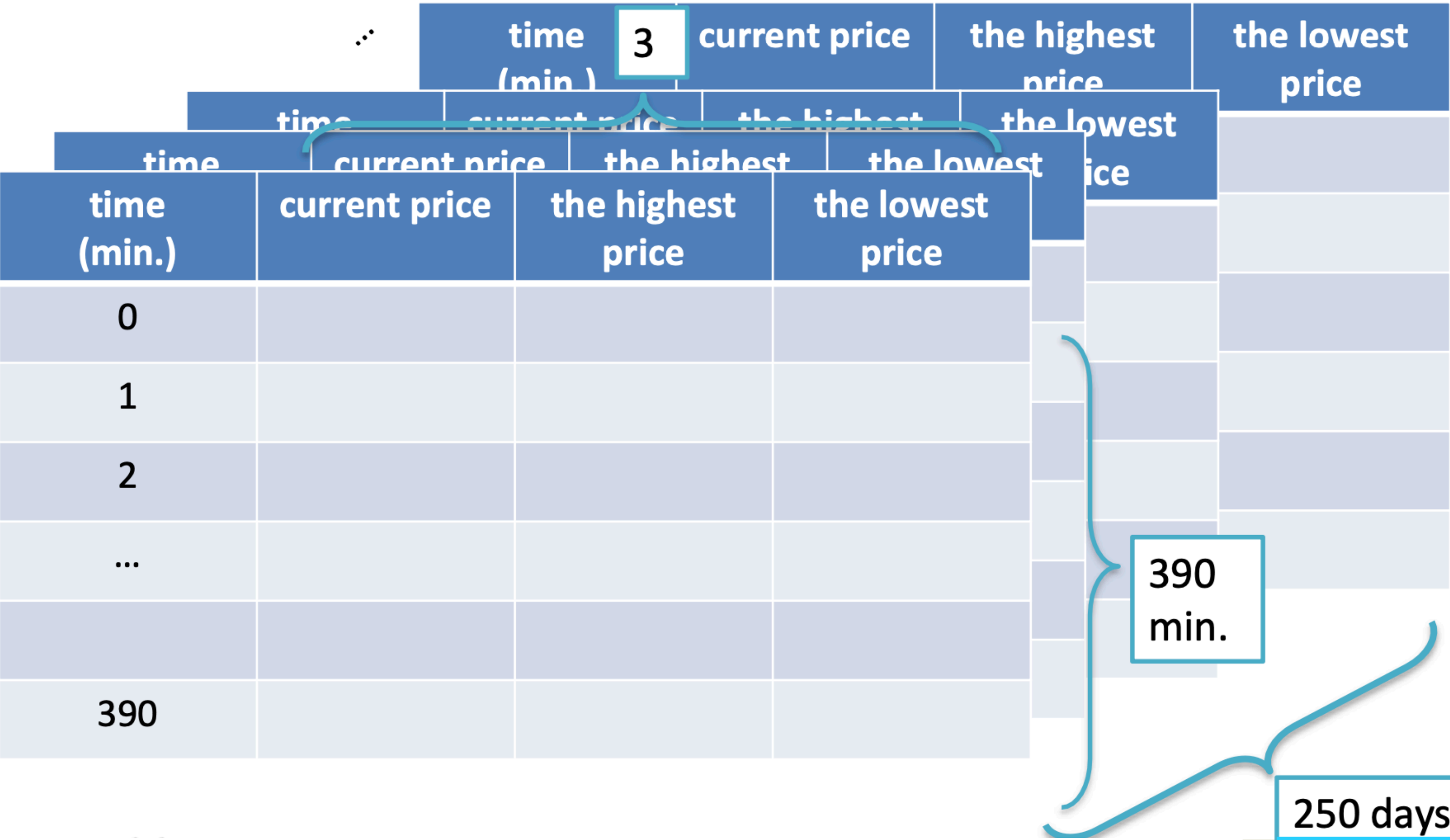
Shape (samples, features) = (10000, 3)



3d tensor: stock price dataset

REAL WORLD EXAMPLES

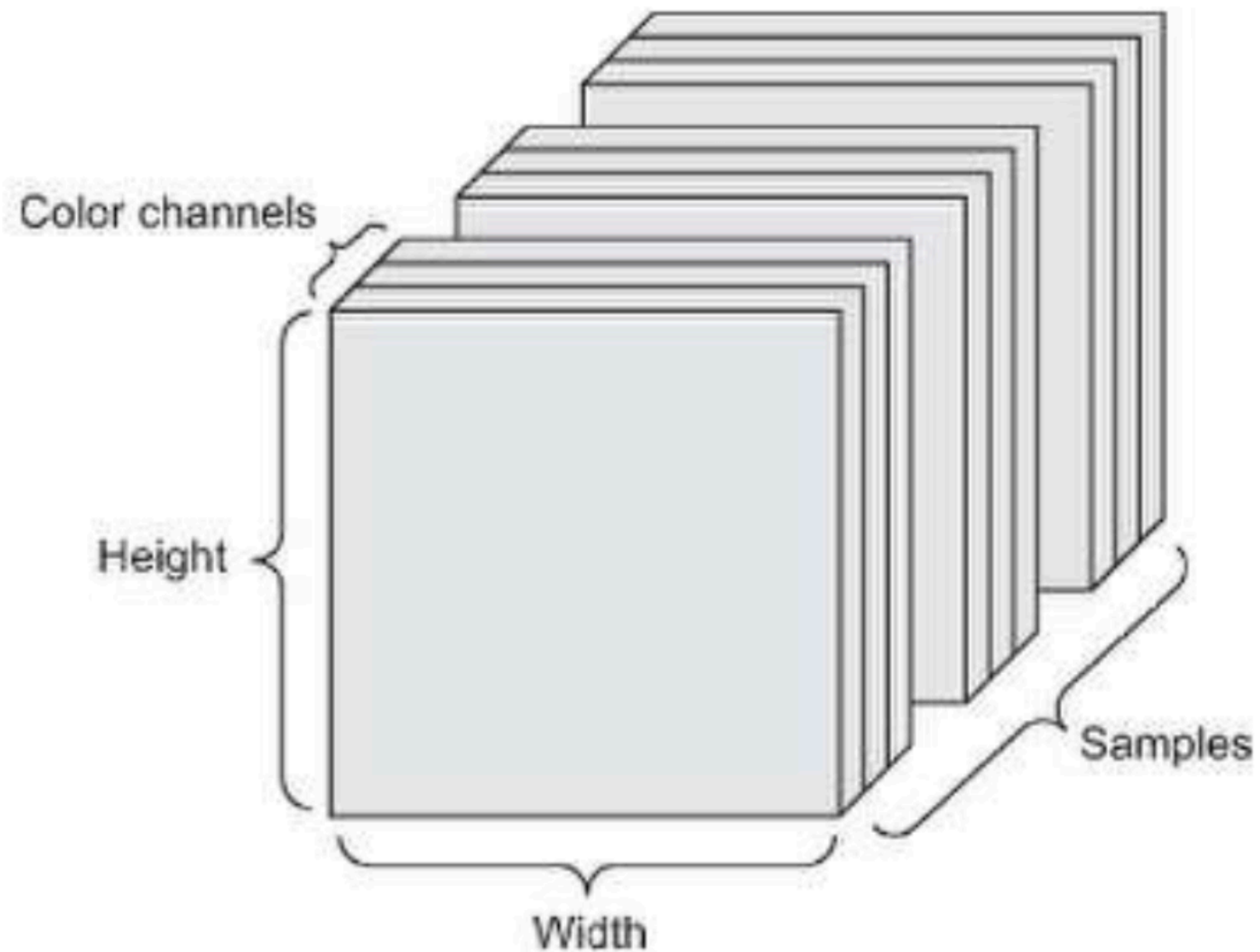
Shape (samples, time steps, features) = (250, 390, 3)



# REAL WORLD EXAMPLES

4d tensor: a batch of 128 colour  
images of size 256\*256

Shape (samples, height, width,  
channels) = (128, 256, 256, 3)



# REAL WORLD EXAMPLES

5d tensor: 60 sec, 144\*156 Yt video clip at 4 fps would be 240 frames. A batch of 4 such video clips

Shape (samples, frames, height, width, channels) = (4, 240, 144, 156, 3)

Total =  $4 * 240 * 144 * 156 * 3 = 106.168.320$

if type of tensor is float32, total memory will be about 405 MB