Computer vision in the new era of Artificial Intelligence and Deep Learning

Visión por computador en la nueva era de la Inteligencia Artificial y el Deep Learning

Rubén Usamentiaga*, Alberto Fernández°
*** University of Oviedo**
° TSK

Gijón (Spain)
5 – 16 April 2021

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

Numpy

NumPy

numpy_introduction.ipynb

Open in Colab numpy_introduction.ipynb

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

Numpy introduction

- Numpy is the core library for scientific computing in Python. It provides:
  - A high-performance multidimensional array object
  - Tools for working with these arrays
- The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure
  - These arrays are homogeneously typed (all elements of a single array must be of the same type)
- Use the following import convention

```
import numpy as np
```

Numpy introduction

- Most of the popular Machine Learning, Deep Learning, and Data Science libraries use NumPy under the hood:
  - Scikit-learn, Matplotlib, Pandas, OpenCV
- NumPy arrays are stored at one continuous place in memory unlike lists
  - Processes can access and manipulate them very efficiently
  - This behavior is called "locality of reference" in computer science.

numpy.ndarray

- The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure
  - It is also known by the alias array
  - numpy.array() is just a method which returns an array object of the type ndarray
  - The more important attributes of an ndarray object are:
    - a.shape: tuple of integers indicating the size of the array in each dimensions
    - a.ndim: the number of axes (dimensions) of the array (also called as rank)
    - a.size: the total number of elements of the array
    - a.dtype: an object describing the type of the elements in the array
    - a.data: the buffer containing the actual elements of the array

Array creation and arithmetic with arrays

`np.array([1, 2, 3])`: This creates an array from a list of values

`np.zeros((3,4))`: This creates an array of zeros with the specified shape

`np.ones((3,4))`: This creates an array of ones with the specified shape

`np.eye(2)`: This creates a diagonal 2x2 array

`np.random.random((2,2))`: This creates a random array with the specified shape

`np.linspace(0,10,21)`: This creates an array with 21 elements from 0 to 10

`np.arange(0,10.5,0.5)`: This creates an array from 0 to 10 (the upper interval 10.5 is not included) with a step of 0.5

```
array_a = np.arange(1, 10, 1)
array_b = np.arange(1, 10, 1)
```

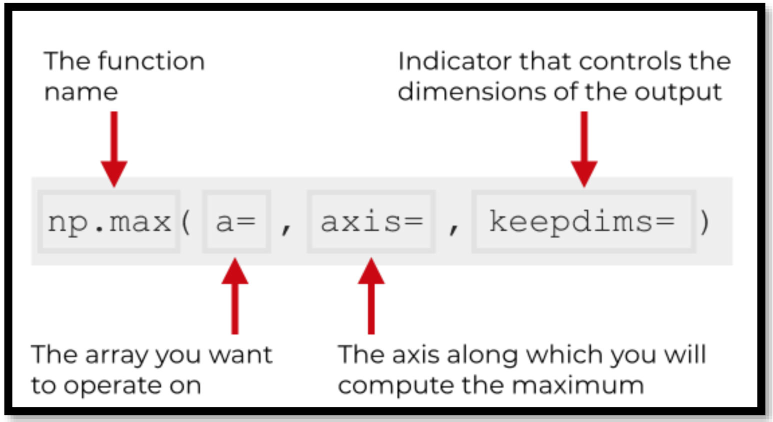
```
array_a + array_b
array_b - array_a

# Example of broadcasting:
array_a + 5

# Element-wise multiplications:
array_a * array_b
np.multiply(array_a, array_b)

# Dot product:
array_a.dot(array_b)
```

```
my_matrix_1 = np.ones((3, 2))
my_matrix_2 = np.ones((2, 4))
```

```
# Matrix multiplication:
np.matmul(my_matrix_1, my_matrix_2)
```

Axes operations

Example with np.max()



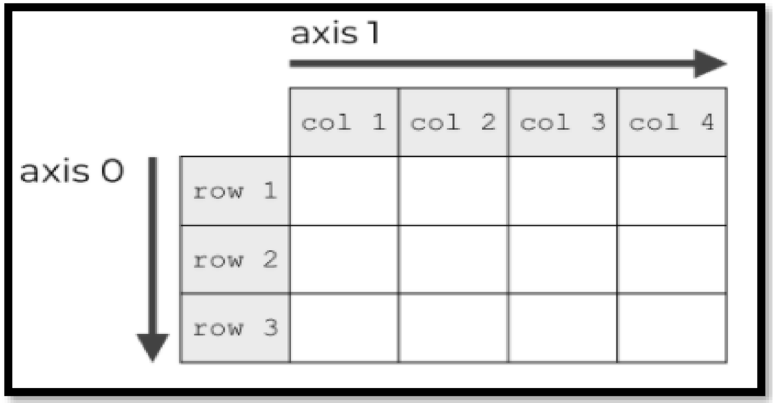
The function name

Indicator that controls the dimensions of the output

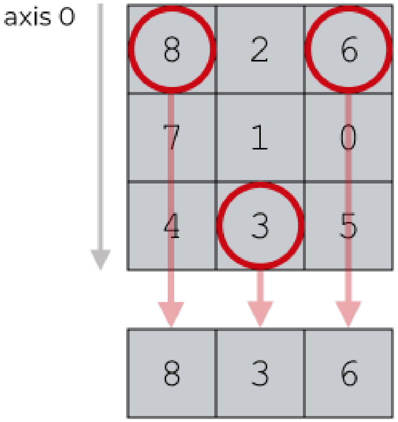
```
np.max( a= , axis= , keepdims= )
```

The array you want to operate on

The axis along which you will compute the maximum

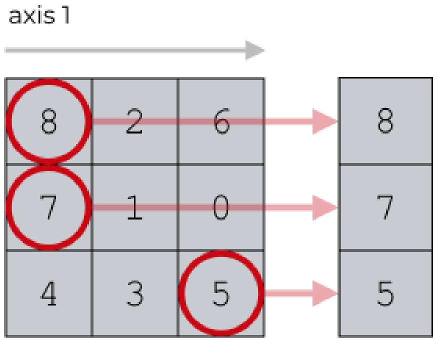


np.max(axis = 0)



np.max calculates the maxima downward when we set axis = 0

np.max(axis = 1)



np.max calculates the maxima horizontally when we set axis = 1

Indexing and slicing

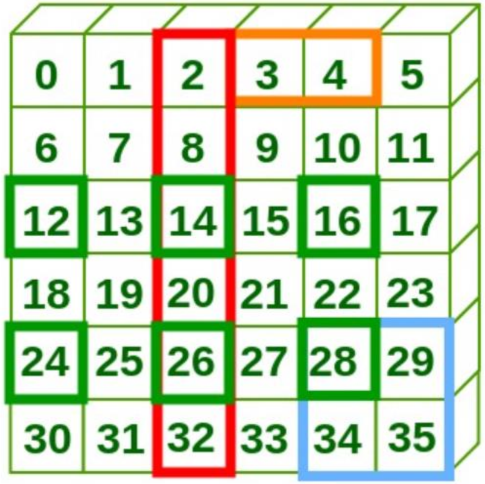
Indexing: Accessing elements of an array by using their indices.

- For one-dimensional Numpy arrays, you only need to specify one index value, which is the position of the element in the Numpy array (e.g. `arrayname[index]`).
- For two-dimensional Numpy arrays, you need to specify both a row index and a column index for the element (e.g. `arrayname[indexrow,indexcol]`).

Slicing: Slicing means taking elements from one given index to another given index:

- We pass slice instead of index like this: `[start:end]`
- We can also define the step, like this: `[start:end:step]`
- If we don't pass start its considered 0. If we don't pass end its considered length of array in that dimension. If we don't pass step its considered 1

```
>>> a[0,3:5]
array( [3,4] )

>>> a[4:, 4:]
array( [ 28, 29],
       [ 34, 35] ] )

>>> a[ :, 2]
array( [2, 8, 14, 20, 26, 32] )

>>> a[2 : : 2, : : 2]
array( [ 12, 14, 16],
       [ 24, 26, 28] ] )
```



0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Numpy

NumPy

Recommended lectures

- NumPy: the absolute basics for beginners: https://numpy.org/doc/stable/user/absolute_beginners.html
- NumPy quickstart: https://numpy.org/doc/stable/user/quickstart.html
- NumPy basics: https://numpy.org/doc/stable/user/basics.html

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course