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°
***Universidad de Oviedo**
°TSK

Gijón
5 – 16 Abril 2021

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

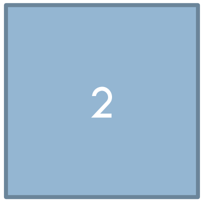
PyTorch

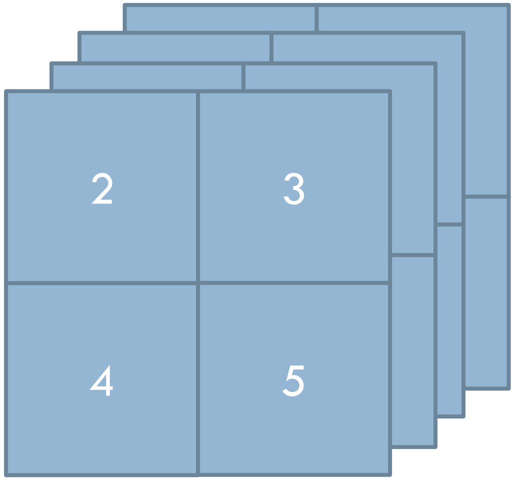
PyTorch

- PyTorch
  - Python-based scientific computing package
    - Open source
  - Main goal:
    - Facilitate building deep learning projects
    - Used for applications such as computer vision and natural language processing
  - Primarily developed by Facebook
    - Similar to Keras+TensorFlow (Google)
  - Based on two high-level features:
    - Hardware-accelerated tensor library
    - Automatic differentiation library

PyTorch

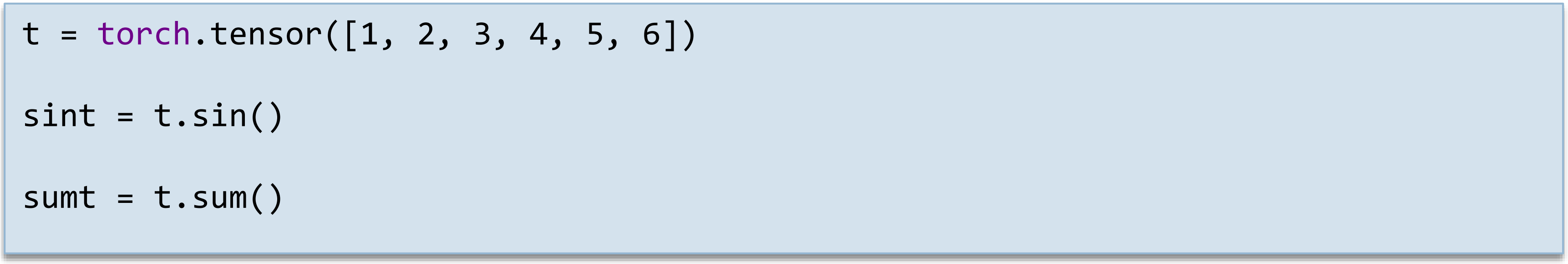
- A tensor
  - A number (scalar)
  - An array
  - A matrix
  - More dimensions





PyTorch

- □ A tensor



```
t = torch.tensor([1, 2, 3, 4, 5, 6])

sint = t.sin()

sumt = t.sum()
```

The same as NumPy?

PyTorch

- □ A tensor

```python
t = torch.tensor([1, 2, 3, 4, 5, 6]).to(device='cuda')

sint = t.sin()

sumt = t.sum()
```

The same as NumPy?

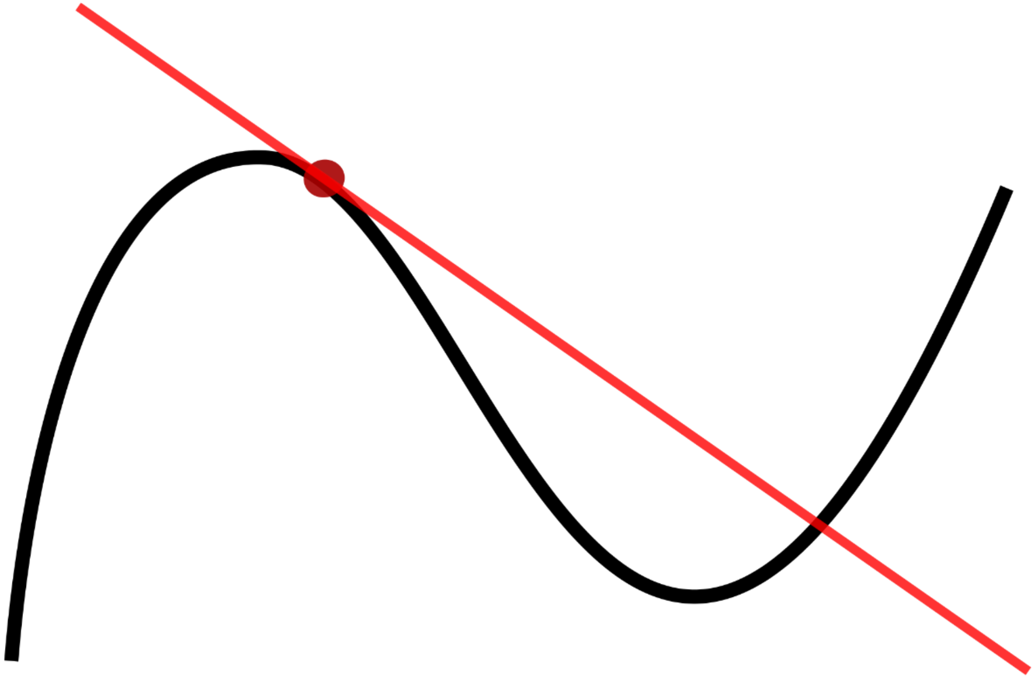
No, operations can run on the GPU

Also, NumPy is a much older lib, thus it has a vast ecosystem of supporting libraries

The rest is similar

PyTorch

- Derivatives
  - The slope of the tangent line to the graph of the function



https://en.wikipedia.org/wiki/Derivative

PyTorch

- Derivatives

$$y = f(x) = 3x^2 + 2x + 10$$

Derivate of y with respect to x at x = 5?

$$\frac{dy}{dx} = 6x + 2$$

32

PyTorch

- Derivatives

```
x = torch.tensor(5., requires_grad=True)

y = 3*x**2 + 2*x + 10

y.backward()

print(x.grad)

> tensor(32.)
```

PyTorch

□ Derivatives

```
x = torch.tensor(5., requires_grad=True)

y = torch.log(3*(x.sin())**2)/torch.exp(1/x)

y.backward()

print(x.grad)

> tensor(-0.4512)
```

PyTorch

- Derivatives

```
def function(x, y):
   return x**2 + 3*y**2

x = torch.tensor(5., requires_grad=True)
y = torch.tensor(5., requires_grad=True)

z = function(x, y)
z.backward()

print(x.grad)
print(y.grad)

>> tensor(10.)
>> tensor(30.)
```

PyTorch

□ Derivatives

```
def f1(x, y):
   return x**2 + 3*y**2

def f2(x):
   return x.sin()*x.cos()

x = torch.tensor(5., requires_grad=True)
y = torch.tensor(5., requires_grad=True)

z = f2(f1(x, y))
z.backward()

print(x.grad)
print(y.grad)

>> tensor(10.)
>> tensor(30.)
```