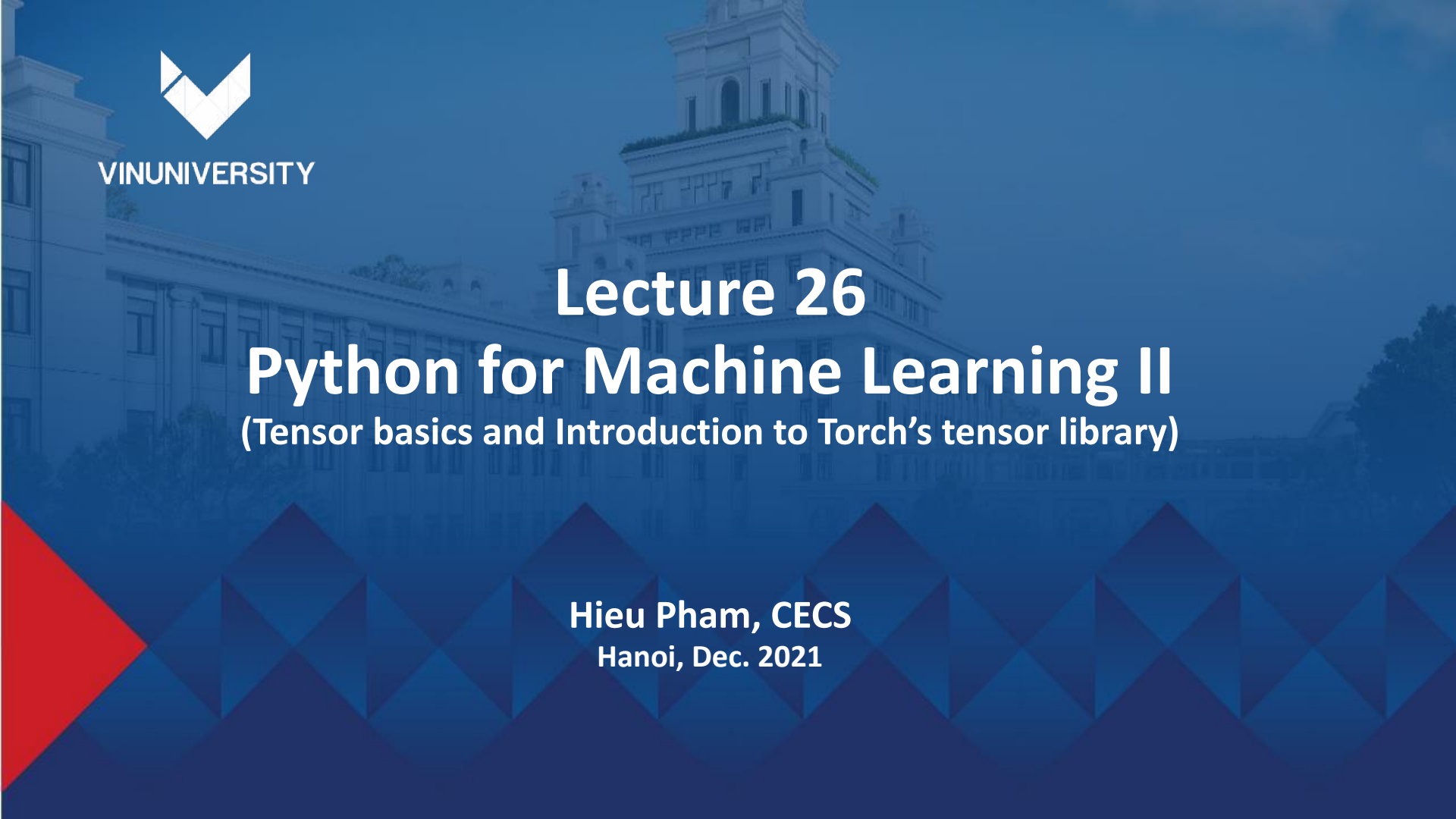


Lecture 26
Python for Machine Learning II
(Tensor basics and Introduction to Torch's tensor library)

Hieu Pham, CECS

Hanoi, Dec. 2021

Tensor basics

A tensor is a multi-way extension of a matrix. In particular, the following are all tensors:

- A 0D tensor is a scalar
- A 1D tensor is a vector (e.g. a sound sample)
- A 2D tensor is a matrix (e.g. a grayscale image)
- A 3D tensor can be seen as a vector of identically sized matrix (e.g. a multi-channel image)
- A 4D tensor can be seen as a matrix of identically sized matrices, or a sequence of 3D tensors (e.g. a sequence of multi-channel images), etc.

Tensor basics

Why Tensors?

Tensor basics

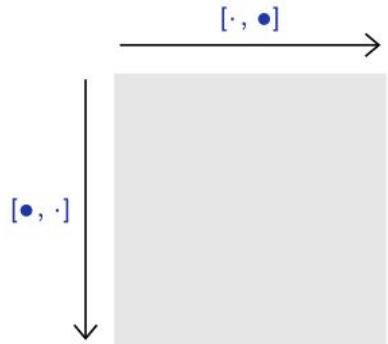
Why Tensors?

- Tensors can be used when matrices are not enough. We say a tensor is *N*-way array.

- A tensor can represent a series/set of matrices.

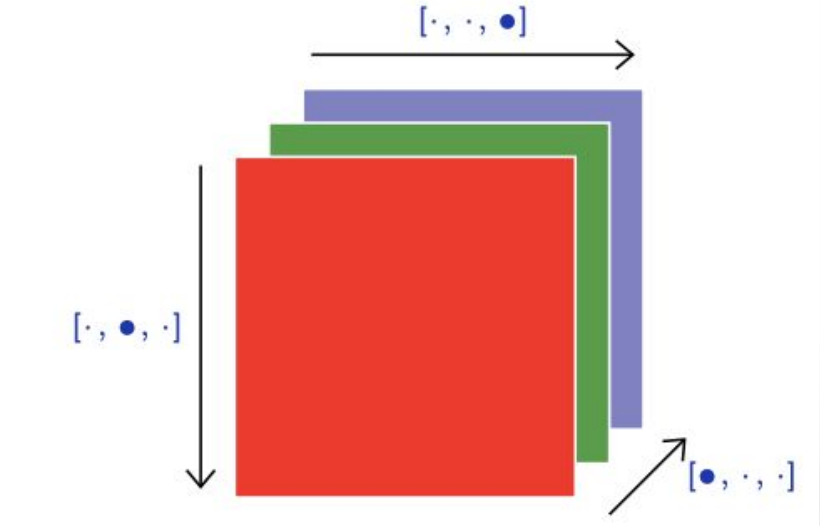
Tensor basics

2d tensor (e.g. grayscale image)

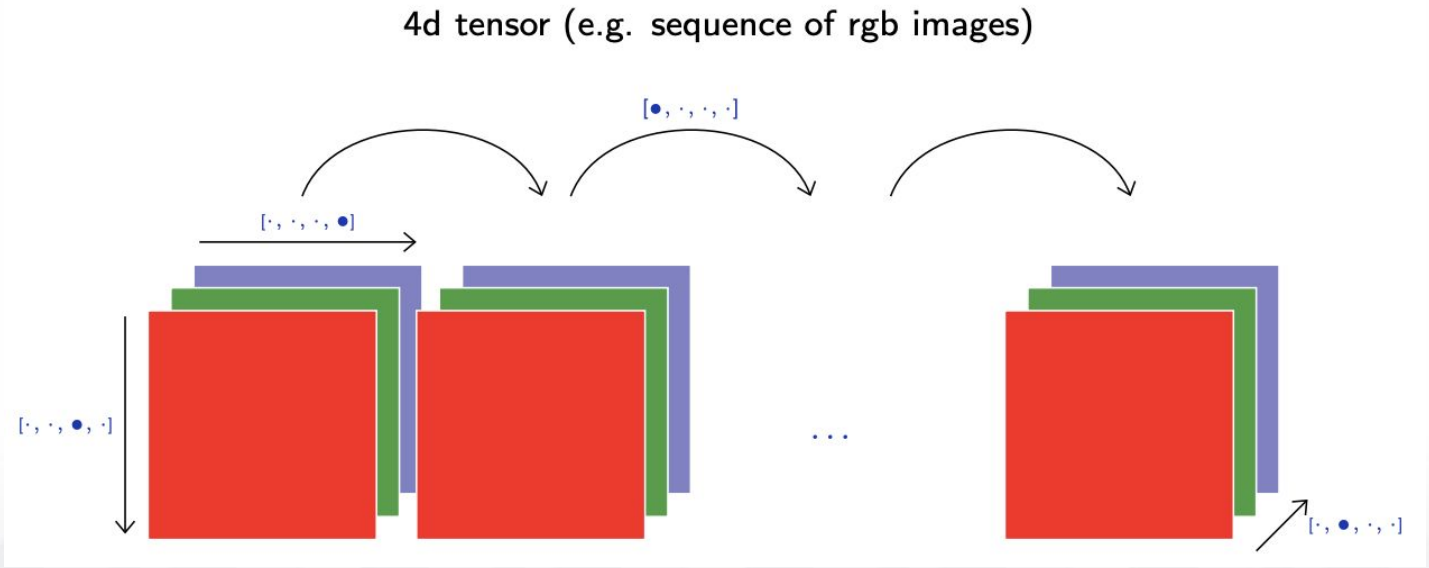


Tensor basics

3d tensor (e.g. rgb image)



Tensor basics



Tensor basics

Tensors are used to encode the signal to process, but also the internal states and parameters of models.

Introduction to Torch’s tensor library (Pytorch)

What is PyTorch?

- Open source machine learning library
- Developed by Facebook's AI Research lab
- It leverages the power of GPUs
- Automatic computation of gradients
- Makes it easier to test and develop new ideas.

Introduction to Torch's tensor library (Pytorch)

PyTorch's main features are:

- Efficient tensor operations on CPU/GPU
- Optimizers
- Data I/O.

Introduction to Torch's tensor library (Pytorch)

Why PyTorch?

- It is pythonic-concise, close to Python conventions
- Strong GPU support
- Many algorithms and components are already implemented
- Similar to NumPy

Introduction to Torch's tensor library (Pytorch)

Constructing our first tensors

Let's construct our first PyTorch tensor and see what it looks like.

```
# In[4]:
import torch
a = torch.ones(3)
a

# Out[4]:
tensor([1., 1., 1.])
```

Imports the torch module

Creates a one-dimensional tensor of size 3 filled with 1s

Introduction to Torch's tensor library (Pytorch)

Tensors can be created from Python lists with the **torch.tensor()** function.

```
# torch.tensor(data) creates a torch.Tensor object with the given data.
V_data = [1., 2., 3.]
V = torch.tensor(V_data)
print(V)

# Creates a matrix
M_data = [[1., 2., 3.], [4., 5., 6]]
M = torch.tensor(M_data)
print(M)

# Create a 3D tensor of size 2x2x2.
T_data = [[[1., 2.], [3., 4.]],
          [[5., 6.], [7., 8.]]]
T = torch.tensor(T_data)
print(T)
```

Introduction to Torch's tensor library (Pytorch)

Out:

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

        [[5., 6.],
         [7., 8.]]])
```

Introduction to Torch's tensor library (Pytorch)

You can create a tensor with random data and the supplied dimensionality with **torch.randn()**

```
x = torch.randn((3, 4, 5))
print(x)
```

Introduction to Torch's tensor library (Pytorch)

You can create a tensor with random data and the supplied dimensionality with **torch.randn()**

```
tensor([[[-1.5256, -0.7502, -0.6540, -1.6095, -0.1002],
         [-0.6092, -0.9798, -1.6091, -0.7121,  0.3037],
         [-0.7773, -0.2515, -0.2223,  1.6871,  0.2284],
         [ 0.4676, -0.6970, -1.1608,  0.6995,  0.1991]],

        [[ 0.8657,  0.2444, -0.6629,  0.8073,  1.1017],
         [-0.1759, -2.2456, -1.4465,  0.0612, -0.6177],
         [-0.7981, -0.1316,  1.8793, -0.0721,  0.1578],
         [-0.7735,  0.1991,  0.0457,  0.1530, -0.4757]],

        [[-0.1110,  0.2927, -0.1578, -0.0288,  0.4533],
         [ 1.1422,  0.2486, -1.7754, -0.0255, -1.0233],
         [-0.5962, -1.0055,  0.4285,  1.4761, -1.7869],
         [ 1.6103, -0.7040, -0.1853, -0.9962, -0.8313]]])
```

Introduction to Torch's tensor library (Pytorch)

PyTorch provides operators for component-wise and vector/matrix operations.

```
>>> x = torch.tensor([ 10., 20., 30.])
>>> y = torch.tensor([ 11., 21., 31.])
>>> x + y
tensor([ 21.,  41.,  61.])
>>> x * y
tensor([ 110.,  420.,  930.])
>>> x**2
tensor([ 100.,  400.,  900.])
```

Introduction to Torch's tensor library (Pytorch)

Indexing: As in NumPy, the : symbol defines a range of values for an index and allows to slice tensors.

```
>>> import torch
>>> x = torch.empty(2, 4).random_(10)
>>> x
tensor([[8., 1., 1., 3.],
        [7., 0., 7., 5.]])
>>> x[0]
tensor([8., 1., 1., 3.])
>>> x[0, :]
tensor([8., 1., 1., 3.])
>>> x[:, 0]
tensor([8., 7.])
>>> x[:, 1:3] = -1
>>> x
tensor([[ 8., -1., -1.,  3.],
        [ 7., -1., -1.,  5.]])
```

x[0,:] - Get the first row and all columns.

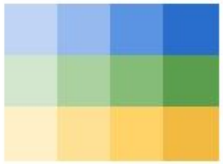
x[:, 0] - Get the first column and all rows.

Introduction to Torch's tensor library (Pytorch)

Reshaping Tensors

Original tensor `t`

`(3, 4)`



`t.view(2, 6)`

`t.view(1, 12)`

Introduction to Torch's tensor library (Pytorch)

Reshaping Tensors

Use the **.view()** method to reshape a tensor.

```
x = torch.randn(2, 3, 4)
print(x)
print(x.view(2, 12))  # Reshape to 2 rows, 12 columns
```

Introduction to Torch's tensor library (Pytorch)

torch.cat() is used for concatenating tensors

```
>>> x = torch.randn(2, 3)
>>> x
tensor([[ 0.6580, -1.0969, -0.4614],
        [-0.1034, -0.5790,  0.1497]])
>>> torch.cat((x, x, x), 0)
tensor([[ 0.6580, -1.0969, -0.4614],
        [-0.1034, -0.5790,  0.1497],
        [ 0.6580, -1.0969, -0.4614],
        [-0.1034, -0.5790,  0.1497],
        [ 0.6580, -1.0969, -0.4614],
        [-0.1034, -0.5790,  0.1497]])
>>> torch.cat((x, x, x), 1)
tensor([[ 0.6580, -1.0969, -0.4614,  0.6580, -1.0969, -0.4614,  0.6580,
         -1.0969, -0.4614],
        [-0.1034, -0.5790,  0.1497, -0.1034, -0.5790,  0.1497, -0.1034,
         -0.5790,  0.1497]])
```

the dimension over which the tensors are concatenated

References

https://pytorch.org/

https://fleuret.org/francois/teaching.html

Thank you for attending !