PyTorch Basics

Presenter: Akshay Daydar
Interdisciplinary Research Scholar (ME & CSE)

Pytorch Basics

What is Tensor?

A tensor is a mathematical object that generalizes the concepts of scalars, vectors, and matrices to higher dimensions. It can be thought of as a multi-dimensional array of numbers that can represent data or physical quantities.

Order	Example	Shape	Interpretation
0 (Scalar)	$s = 5$	No shape	Single number
1 (Vector)	$[3, 4, 5]$	3×1	List of numbers
2 (Matrix)	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	2×2	Grid of numbers
3 (Tensor)	RGB image	$32 \times 32 \times 3$	Cube of numbers
4+ (Tensor)	Batch of images	$100 \times 32 \times 32 \times 3$	Multi-dimensional array

Pytorch Basics

1. Tensor Intitalization

2. Tensor Maths

3. Tensor Indexing

4. Tensor Reshaping

Why Pytorch ?

Dynamic Computation Graphs – Build networks on the fly.

Pythonic – Easy to learn and implement.

Extensive Community Support – Strong community contributions and resources.

Research to Production – PyTorch serves both research purposes and production-level deployment.

Seamless GPU Support – Simplifies GPU-based computation.

Pytorch Basics

1. Tensor Intialization

```
device = "cuda" if torch.cuda.is_available() else "cpu"
```

```
torch_tensor = torch.tensor([[1,2,3],[4,5,6]], dtype =
torch.float32, device = device, requires_grad = True)
```

Other methods for torch tensor declaration

```
X = torch.empty((2,2)),
X = torch.rand((2,2))= torch.empty((2,2)).uniform_(0,1),
X = torch.ones((2,2)),
X = torch.zeros((2,2)),
X = torch.eye((2,2)) = torch.diag(torch.ones((2,2))),
X = torch.arange(start=0,end=5, step=1),
X = torch.linspace(start=0.1, end=1, steps=10)
```

Pytorch Basics

1. Tensor Intialization

Datatype conversion in torch

```
X = torch.empty((5,5)), # dtype = int-32
X = X.bool()            # Convert into binary form
X = X.short()           # Convert into dtype int-16
X = X.long()            # Convert into dtype int-64
X = X.half()            # Convert into dtype float-16
X = X.float()           # Convert into dtype float-32
X = X.double()          # Convert into dtype float-64
```

Torch to array conversion and vise-versa

```
X = np.zeros((6,6))       # Declare numpy array
X = torch.from_numpy(X)   # Convert to torch tensor
X = X.numpy()             # Convert to numpy array
```

Pytorch Basics

2. Tensor Maths

```
X1 = torch.tensor([1,2,3]), X2 = torch.tensor([4,5,6])

Addition: X1 + X2  = torch.add(X1, X2)

Inplace operation --> X2 += X1 or X2._add(X1)

Subraction: X1 - X2

Division: torch.true_divide( X1, X2)

Exponential: z = X1.pow(2) = X1 ** 2

Comparision: Z = X1>0, X2>X1, X2<X1
```

Pytorch Basics

2. Tensor Maths

```
X1 = torch.tensor([1,2,3]), X2 = torch.tensor([4,5,6])

Metrix Multiplication: torch.mm(X1, X2) = X1.mm(X2)

Element-wise Multiplication: X1 * X2

Dot product: torch.dot(X1, X2)

Batch Metrix Multiplication:

batch = 2, a = 3, b = 5, c = 4

X1 = torch.rand((batch, a, b)), X2 = torch.rand((batch, b,
c))
```

Pytorch Basics

2. Tensor Maths

```
X1 = torch.tensor([1,2,3]), X2 = torch.tensor([4,5,6])

indices, values = torch.max(X1, dim=0), indices =
torch.argmax(X1, dim=0)

Absolute tensor: torch.abs(X1)

Tensor Clamping: torch.clamp(X1, min = 0)
```

Pytorch Basics

3. Tensor Indexing

```
batch_size = 2, features = 5

x = torch.rand((batch_size), features)

z = x[0].shape      # Get the shape of first tensor

z = x[:,0].shape    # Get the shape of all tensors at the
first dimension

Conditional Formatting: Z = torch.arange(10)

Z[(Z>2) | (Z<8)]      # Get the elements greater than 2 and
less than 8
```

Pytorch Basics

4. Tensor Reshaping

```
X = torch.tensor([1,2,3,4])
Z = X.view(2,2)      # When X is contiguous
Z = X.reshape(2,2) # Independent of contiguity of X
Concatenation: X1 = torch.rand([2,4]), X2 =
torch.rand([2,3])
Z1 = torch.cat((X1, X2), dim=0), Z2 = torch.cat(X1, X2,
dim=1)
T = torch.tensor([1,2,3])
```

Akshay Daydar

Research Scholar, IITG

Research Areas: Medical Imaging, Deep Learning, Biomechanics

https://adaydar.github.io/