# PyTorch Tensors vs NumPy ndarrays — Cheat Sheet

## 1. Overview

| Feature | PyTorch Tensor | NumPy ndarray |
|---|---|---|
| Library | torch | numpy |
| Purpose | Deep learning, GPU computing | General scientific computing |
| GPU Support | Yes (.to('cuda')) | No |
| Autograd | Yes (requires_grad=True) | No |
| Backend | C++ / CUDA | C |
| Core Concept | Multi-dimensional array | Multi-dimensional array |

## 2. Creation

| Task | PyTorch | NumPy |
|---|---|---|
| From list | torch.tensor([[1,2],[3,4]]) | np.array([[1,2],[3,4]]) |
| Zeros | torch.zeros((2,3)) | np.zeros((2,3)) |
| Ones | torch.ones((2,3)) | np.ones((2,3)) |
| Random | torch.rand((2,3)) | np.random.rand(2,3) |
| Identity | torch.eye(3) | np.eye(3) |
| Range | torch.arange(0,10,2) | np.arange(0,10,2) |
| Linspace | torch.linspace(0,1,5) | np.linspace(0,1,5) |

## 3. Shape & Reshaping

| Task | PyTorch | NumPy |
|---|---|---|
| Shape | t.shape | a.shape |
| Reshape | t.reshape(2,3) or t.view(2,3) | a.reshape(2,3) |
| Transpose | t.T or t.transpose(0,1) | a.T |
| Expand dims | t.unsqueeze(0) | np.expand_dims(a,0) |
| Remove dims | t.squeeze() | np.squeeze(a) |

## 4. Indexing & Slicing

| Operation | PyTorch | NumPy |
|---|---|---|
| Single element | t[0,1] | a[0,1] |
| Slice rows | t[1:] | a[1:] |

| | | |
|---|---|---|
| Slice columns | t[:, 0] | a[:, 0] |
| Boolean mask | t[t > 0.5] | a[a > 0.5] |

## 5. Math Operations

| Operation | PyTorch | NumPy |
|---|---|---|
| Add | t1 + t2 | a1 + a2 |
| Subtract | t1 - t2 | a1 - a2 |
| Multiply | t1 * t2 | a1 * a2 |
| Dot product | torch.matmul(t1, t2) | np.dot(a1, a2) |
| Mean | t.mean() | a.mean() |
| Sum | t.sum() | a.sum() |
| Max/Min | t.max(), t.min() | a.max(), a.min() |

## 6. Conversion Between Tensor & ndarray

| Example Code | PyTorch | NumPy |
|---|---|---|
| NumPy → Tensor | torch.from_numpy(a) | np.array([1,2,3]) |
| Tensor → NumPy | t.numpy() | — |
| Note | Shared memory if on CPU | — |

## 7. GPU & Autograd (Tensor-only features)

| Task | PyTorch | NumPy |
|---|---|---|
| Move to GPU | t.to('cuda') | Not supported |
| Enable gradients | t.requires_grad_() | Not supported |
| Compute gradients | t.backward() | Not supported |

**8. Summary Formula**
Tensor = ndarray + GPU support + Autograd

**Quick Takeaways**
- Both support same shape, indexing, and math operations.
- Tensors are drop-in replacements for ndarrays in deep learning code.
- Use Tensors for PyTorch models, NumPy for data preprocessing.