The Ultimate Guide to Tensors

1. Historical and Conceptual Origins

- Word "Tensor" comes from Latin *tendere* = "to stretch."
- First formalized in 19th century by Gregorio Ricci-Curbastro & Tullio Levi-Civita → developed Tensor Calculus (the language of Einstein's General Relativity).
- In physics: Tensors describe physical laws independent of coordinate systems.
- In machine learning: Tensors became the de facto way of representing multi-dimensional data + model parameters.

2. What Exactly is a Tensor?

Two complementary views:

Applied/ML View

- A multi-dimensional array of numbers.
- Example: NumPy arrays, PyTorch tensors.

Mathematical View

- A multilinear map:
$$T : V_1 \times V_2 \times \cdots \times V_n \to \mathbb{R}$$
where each V_i is a vector space.
- This captures tensors as coordinate-independent objects.

Both views are valid; in ML, we mostly use the array view but the math underpins transformations & derivatives.

3. Hierarchy of Tensors

Tensor Order	Object	Example
0	Scalar	$7, \pi, -3.14$
1	Vector	$[2, -5, 3]$

2	Matrix	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
3	3D Tensor	RGB image (224×224×3)
N	N-D Tensor	Sequence of 3D videos (batch, frames, height, width, channels)

4. Attributes of a Tensor

Every tensor has:

1. **Rank (order):** Number of axes. Vector = rank 1, Matrix = rank 2.
2. **Shape:** Length of each axis. Example: (32, 224, 224, 3).
3. **Axes:** Independent dimensions. Axis 0 = rows, axis 1 = columns, etc.
4. **Dimension names (sometimes):** In high-level ML frameworks like `xarray`, axes can have names (time, features).
5. **Data type (dtype):** float32, float64, int64, bool.
6. **Device:** CPU, GPU, TPU.

5. Tensor Operations

Operations fall into categories:

5.1 Elementwise

Addition: $C_{ij} = A_{ij} + B_{ij}$
Other: subtraction, multiplication, division.

5.2 Contraction

Generalization of dot/matrix multiplication:

- Vector dot product: $a_i b_i$.
- Matrix multiplication: $C_{ik} = \sum_j A_{ij} B_{jk}$.
- Higher contractions: sum over shared axes.

5.3 Reshape & Transpose

- **Reshape:** Rearrange shape without changing data.
- **Transpose/Permute:** Swap axes.

5.4 Broadcasting

Rules for aligning smaller/larger tensors. Example:

```
[3, 1] + [1, 4] -> [3, 4]
```

5.5 Norms & Reductions

- L2 norm: $\|v\|_2 = \sqrt{\sum_i v_i^2}$
- Frobenius norm (matrix).
- Reductions: sum, mean, max, min along axes.

6. Tensor Calculus

Where tensors shine: derivatives.

- Gradient (∇f): Derivative of scalar w.r.t. tensor.
- Jacobian (J): Derivative of vector w.r.t. vector (matrix).
- Hessian (H): 2nd derivative (higher-order tensor).
- Chain Rule: Applied across tensors during backpropagation.

Example in ML: If

$$y = Wx + b$$

where W (matrix), x (vector), b (vector):

- Gradient wrt W: outer product of input/output derivatives.
- Frameworks like PyTorch automate this.

7. Geometric Meaning

- Scalars: measure magnitude (length, weight).
- Vectors: represent direction & magnitude in space.
- Matrices: linear transformations (rotation, scaling, projection).
- Higher tensors: encode multi-directional relations (stress, curvature).

In ML:

- Tensor encodes input, weights, features, attention patterns.
- They transform spaces $\rightarrow$ map input domain to hidden spaces.

8. Tensors in ML Practice

8.1 Input Data

- Images: (Batch, Channels, Height, Width).
- Text: (Batch, Sequence length, Embedding dim).
- Audio: (Batch, Samples, Features).

8.2 Model Parameters

- Fully connected layer: weight tensor (out_features, in_features).
- Convolution layer: (num_filters, in_channels, kernel_h, kernel_w).

8.3 Computations

- Intermediate activations = tensors.
- Loss function returns scalar tensor.
- Gradients = tensors flowing backward.

9. Implementation in Frameworks

9.1 NumPy

- ndarray → general N-D array.
- No gradient support.

9.2 PyTorch

- torch.Tensor.
- Gradient tracking (requires_grad=True).
- GPU support.

9.3 TensorFlow

- tf.Tensor, tf.Variable.
- Static/dynamic computation graphs.

9.4 JAX

- Functional style, JIT compilation.
- Autograd with tensors (DeviceArray).

10. Advanced Topics

10.1 Einstein Summation

Compact notation for contractions.

```
import numpy as np
a = np.random.randn(2,3)
b = np.random.randn(3,4)
c = np.einsum('ik,kj->ij', a, b)  # matrix multiplication
```

10.2 Sparse Tensors

- Store only non-zeros.
- Used in graphs, NLP.

10.3 Tensor Decomposition

- CP, Tucker decomposition.
- Reduce tensor complexity.
- Applications: recommender systems, compression.

10.4 Tensor Networks

- Physics → represent quantum states.
- ML → efficient factorization of high-dimensional data.

10.5 Tensors on Manifolds

- Riemannian geometry → curved spaces.
- ML uses: Geometric Deep Learning, Graph Neural Networks.

11. Examples Across ML

CNN (Computer Vision)

- Input: (batch, channels, H, W).
- Conv kernel: (filters, channels, kh, kw).
- Output: (batch, filters, H_out, W_out).

RNN/Transformers (NLP)

- Input: (batch, seq_len, embed_dim).
- Attention tensor: (batch, heads, seq_len, seq_len).

Reinforcement Learning

- State tensor: (batch, features).
- Policy logits: (batch, actions).

12. Numerical & Computational Aspects

- Memory layout: Row-major (C/NumPy) vs column-major (Fortran/Matlab).
- Contiguity: Important for GPU speed.
- Precision: float16, bfloat16 for efficiency.
- Parallelism: Tensor cores (NVIDIA GPUs) accelerate tensor ops.

13. Why Tensors are Essential in ML

- Universal representation: everything is a tensor.
- Efficient computation: GPUs optimized for tensor ops.
- Auto-differentiation: backprop relies on tensor calculus.
- Scalability: can handle billions of parameters.

14. Complete Concept Map (Mindset)

- Tensors unify data + models.
- Everything flows as tensors through the network.
- Training = forward tensor flow + backward gradient flow.
- Advanced ML = manipulating tensors at scale.

Final Summary

Tensors are not just "multidimensional arrays" but a deep mathematical, geometric, and computational structure. They extend scalars, vectors, and matrices; they encode data, parameters, and transformations; they make deep learning possible by enabling efficient computation, differentiation, and representation across domains from vision to NLP to reinforcement learning. Advanced topics like sparse tensors, decomposition, and tensor networks connect ML with physics and geometry.