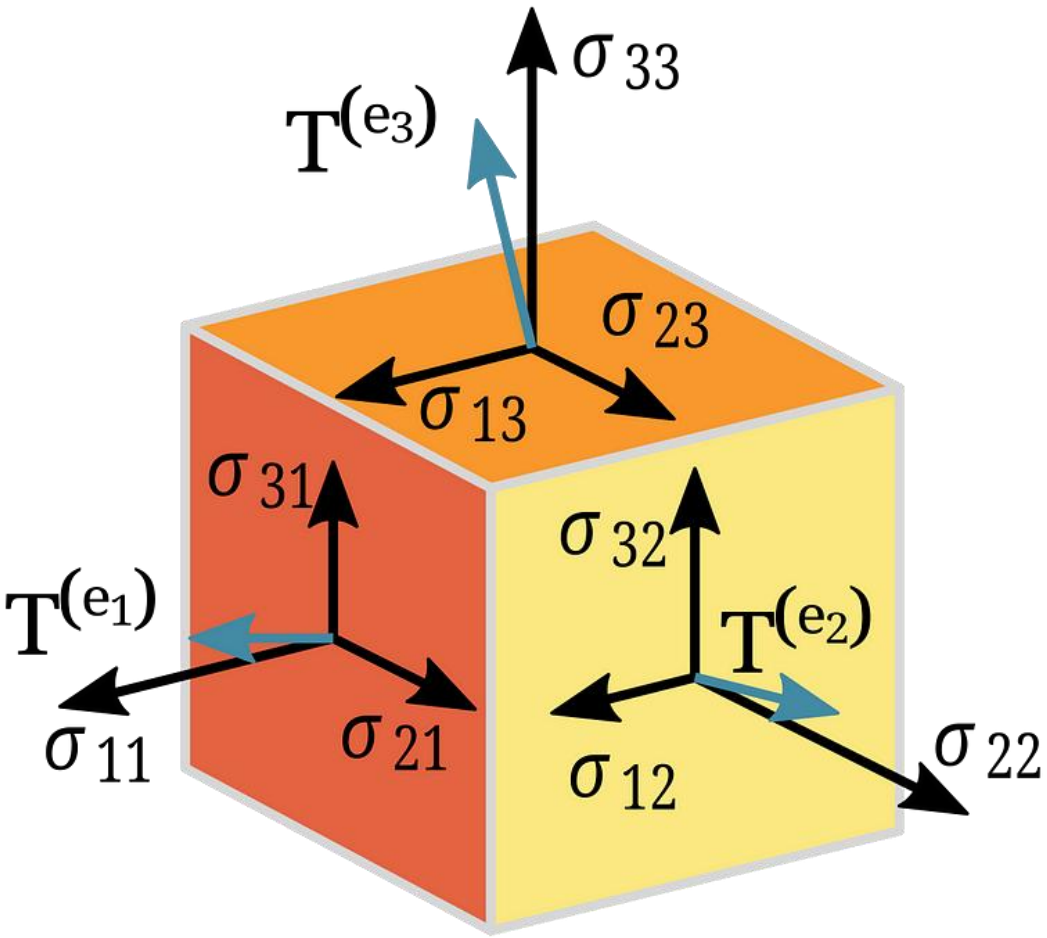
By M.Asmaad Saeed

Let's Start With TENSORS !?

Hmm… One would say that 'Tensor' is a generalization of matrices and vectors to *n* dimensions. But i would argue that it's not really true, because everyone sees *tensors* by their own perspectives and 'views'.



In this Article i'll be covering all the major aspects of tensors that i feel one should know and have a grip on.

> **Tensors** are fascinating mathematical objects that underpin many modern technologies—from the way our computers process images to how artificial intelligence models learn and make predictions. In this article, I will explore tensors from both an intuitive and a technical perspective, ensuring that even readers without a strong background in mathematics or computer science can grasp their

significance. By the end, you'll see how tensors form the backbone of deep learning, particularly in the architecture of neural networks.

The **Outline** of this Article would be :

1: A Non-Tecnical View on what a Tensor is (Might OverSimplify:)

2: A Technical View on Tensors.

3: Some main types of Tensors.

4: Tensors being used in Deep Learning(DL), NN, LLMs. Why ?

5: The ALGEBRA of Tensors (Most—Awaited).

6: Conclusion.

A Non-Tecnical View on *TENSORS*

Imagine you're organizing your belongings. You might use different kinds of containers depending on what you need to store. For example, you could use:

- **A single box** to keep one item, like your favorite book. In the world of tensors, this single box is like a **0-dimensional tensor**, also known as a **scalar**. It's just a single number. Think of it like the temperature you see on a weather app—just one number telling you how hot or cold it is.
- **A list of boxes** arranged in a row to store multiple books, maybe one for each day of the week. This row of boxes is similar to a **1-dimensional tensor**, or a **vector**. It's a list of numbers in a specific order. For example, if you note down the temperature each day for a week, you'd have a list of seven numbers, a 1D tensor.
- **A bookshelf** with rows and columns to organize many books. This bookshelf is like a **2-dimensional tensor**, called a **matrix**. It's a table of numbers arranged in rows and columns, much like a spreadsheet. Think of a black and white image; you can represent it as a grid of numbers where each number represents the shade of gray at a particular point.
- Now, imagine you have **multiple bookshelves stacked together**. This is getting into **3-dimensional tensors**. Think of a colored

image. It's not just height and width anymore. It also has color channels—red, green, and blue. So, for each point in the image (defined by height and width), you have three numbers telling you how much red, green, and blue color there is.

- You can even go beyond 3D! Imagine a **series of these stacks of bookshelves over time**, like pages in a flipbook creating a video. This starts to resemble **4-dimensional tensors**. A video is essentially a sequence of colored images (3D tensors) played over time, adding a fourth dimension—time or frames.

In essence, tensors are just ways to organize numbers in multiple dimensions. They are containers for data, arranged in a grid-like fashion. The number of dimensions tells you how many indices you need to locate a specific number within the tensor.

Why are tensors useful? They are incredibly useful because they provide a structured way for computers to understand and process complex data. Think about images, videos, sound, text—all these can be converted into numbers and organized as tensors. This organization allows computers to perform operations on this data in a meaningful way, especially in fields like image recognition, language understanding, and many more. For now, just remember tensors as versatile containers for numerical data in multiple dimensions.

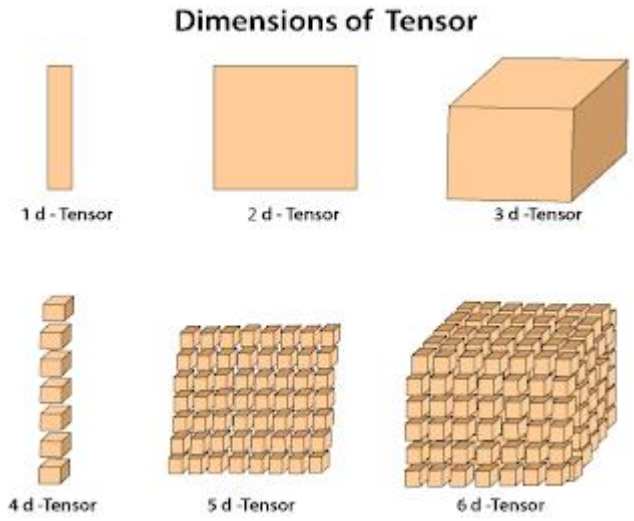
Technical Definition: Tensors in Mathematics

Mathematically, a tensor is a generalization of scalars, vectors, and matrices. To understand this better, let's break it down:

Scalars, Vectors, Matrices as Special Cases:

- A **scalar** is a 0-dimensional tensor. It's just a single number. Mathematically, it's an element from a field (like real numbers).
- A **vector** is a 1-dimensional tensor. It's an ordered list of numbers. In linear algebra, vectors are often thought of as arrows in space, having both magnitude and direction.
- A **matrix** is a 2-dimensional tensor. It's a rectangular array of numbers. Matrices are fundamental in linear transformations, representing mappings between vector spaces.
- A **tensor** extends this concept to any number of dimensions. It's a multi-dimensional array.

By M.Asmaad Saeed



Mathematical Representation:

Tensors are represented as multi-dimensional arrays. We describe a tensor by its **rank** (also called order or degree), which is the number of dimensions it has.

- A scalar has rank 0.
- A vector has rank 1.
- A matrix has rank 2.
- A 3D array has rank 3, and so on.

Each element in a tensor is identified by a set of indices. For a rank-n tensor, you need n indices to specify a component. For instance, in a matrix (rank-2 tensor) A, we use two indices, i and j, to denote an element A[i, j] at the i-th row and j-th column. For a 3D tensor T, we would use three indices T[i, j, k], and so forth.

Transformation Properties (Brief Overview):

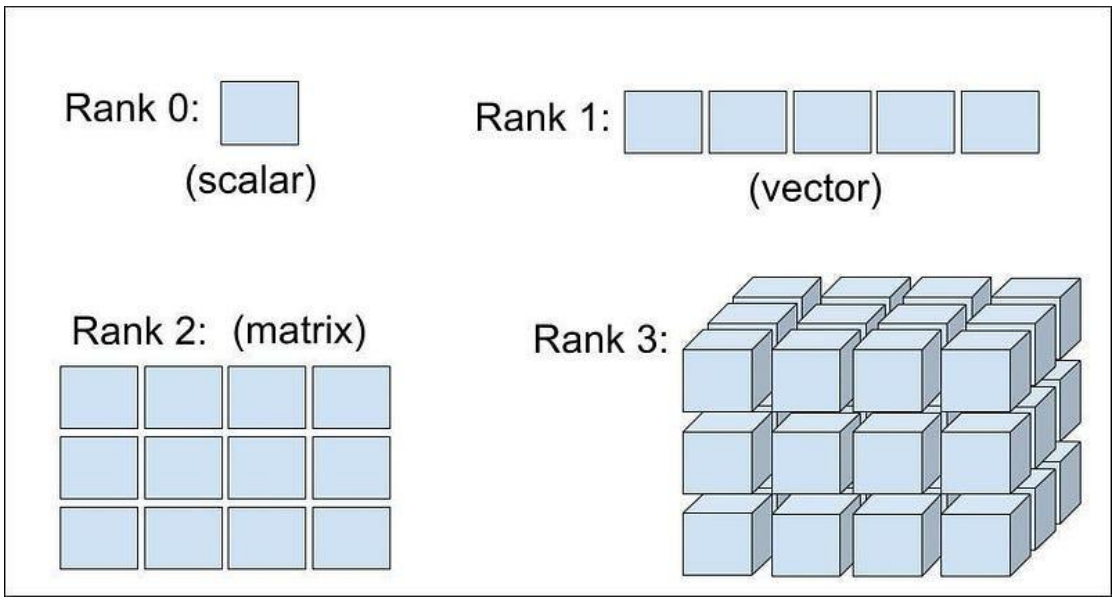
- In a more advanced mathematical context, tensors are defined by how they transform under changes of *coordinates*. This is a crucial aspect in fields like physics and differential geometry.
- Essentially, tensors are objects that transform in a specific, consistent way when you switch from one coordinate system to another. This ensures that tensors represent physical or geometric entities in a coordinate-independent manner.
- There are different types of tensors based on their transformation properties, such as *covariant, contravariant, and mixed tensors*. However, for a basic understanding, especially in the context of

deep learning, focusing on the rank and multi-dimensional array aspect is often sufficient.

A key aspect of tensors is that they are not just arbitrary collections of numbers, they transform in predictable ways under coordinate changes.

Types of Tensors

Tensors can be categorized in different ways, but the most common classification is based on their **rank** (number of dimensions). Let's explore some types with examples:



0-rank Tensor: Scalar

- **Definition:** A scalar is a single number. It has no dimensions.

Example :

- **Temperature:** The temperature at a specific point in time and location, like "25 degrees Celsius". This is just a single value.
- **Pressure:** The pressure at a point, like "1013 hPa". Again, a single numerical value.
- **Age:** Your age, like "30 years". A single number representing a quantity.

1-rank Tensor: Vector

By M.Asmaad Saeed

- **Definition:** A vector is an ordered list of numbers. It has one dimension. Vectors represent magnitude and direction.

Example:

- **Velocity:** The velocity of a car, like "60 km/h to the East". This has both a speed (magnitude) and a direction. You could represent this as a vector [60, East Direction]. If 'East Direction' is encoded numerically, it becomes a list of numbers.
- **Force:** A force acting on an object, like "10 Newtons downwards". Magnitude (10N) and direction (downwards).
- **Daily Stock Prices for a Week:** [Price on Monday, Price on Tuesday, …, Price on Friday]. This is a list of prices, ordered by day.

2-rank Tensor: Matrix

- **Definition:** A matrix is a rectangular array of numbers arranged in rows and columns. It has two dimensions. Matrices are used to represent linear transformations and relationships between vectors.

Example:

- **Transformation Matrix in Graphics:** In computer graphics, matrices are used to perform operations like scaling, rotation, and translation of images or objects. A 2x2 or 3x3 matrix can define how to rotate or scale a 2D or 3D object.
- **Covariance Matrix:** In statistics, a covariance matrix shows the covariance between different variables in a dataset. It's a square matrix where each element (i, j) is the covariance between the i-th and j-th variable.
- **Grayscale Image:** A grayscale image can be represented as a matrix where each element represents the intensity of gray at a pixel location.

3-rank and Higher Tensors

- **Definition:** Tensors with three or more dimensions. They are multi-dimensional arrays that extend the concepts of vectors and matrices to higher orders.

Example (3D Tensor):

- **Color Image:** As mentioned before, a color image can be seen as a 3D tensor (Height x Width x Color Channels). For each pixel (defined by height and width), there are three values representing the intensity of Red, Green, and Blue.
- **Volume Data (Medical Scans):** Medical imaging data like MRI or CT scans are often 3D tensors. They represent a 3D volume, where each point in 3D space has a value associated with it (e.g., tissue density).
- **Video Clip (short):** A short video clip can be thought of as a 4D tensor (Frames x Height x Width x Color Channels). It adds a time dimension to the 3D color image.

Higher Rank Tensors (beyond 3D): While harder to visualize directly, tensors of rank 4, 5, and higher are used in advanced fields like physics, engineering, and deep learning. For instance, in physics, the stress-energy tensor in general relativity is a rank-2 tensor in 4D spacetime. In deep learning, especially in complex models, you may encounter tensors with ranks higher than 3 or 4 when dealing with batches of videos or very high-dimensional feature representations.

Tensors in Deep Learning and Large Language Models (LLMs)

Tensors are absolutely fundamental to the field of Deep Learning and especially to Large Language Models (LLMs). They are not just a theoretical concept but the **basic building blocks** for how we represent and process data in neural networks.

Tensors: The Language of Deep Learning Frameworks:

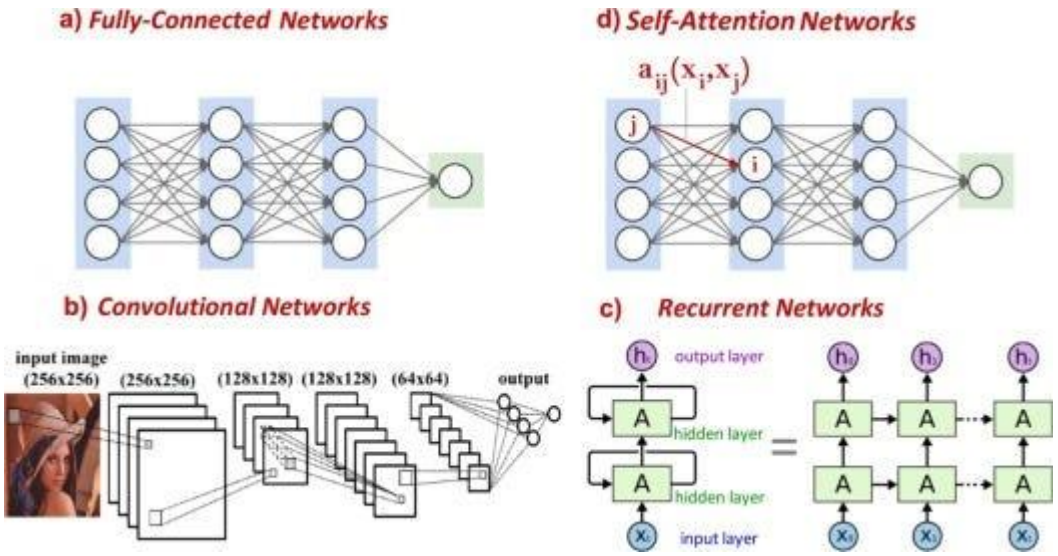
- Deep learning frameworks like **TensorFlow** and **PyTorch** are built around the concept of tensors. Everything, from the input data to the parameters of the models, and the outputs, is represented as tensors.
- When you work with these frameworks, you are essentially performing operations on tensors. These frameworks provide optimized functions and tools specifically designed for efficient tensor computations, especially on GPUs (Graphics Processing Units), which are crucial for deep learning.

Representing Data as Tensors in Deep Learning:

- **Images:** When you feed an image into a neural network for image recognition, the image is first converted into a tensor. For a color

image, it's typically a 3D tensor (Height x Width x Color Channels). A batch of images would be represented as a 4D tensor (Batch Size x Height x Width x Color Channels).

- **Text:** In natural language processing (NLP), text is transformed into numerical form to be processed by neural networks. This is done through techniques like word embeddings. Each word might be converted into a vector of numbers (a 1D tensor), and a sentence or document becomes a sequence of these vectors, forming a 2D tensor (Sequence Length x Embedding Dimension). For processing batches of sentences, you'd use 3D tensors (Batch Size x Sequence Length x Embedding Dimension).
- **Audio:** Audio data, like speech, is also converted into numerical representations, often as spectrograms or raw audio waveforms, which can be represented as tensors.
- **Structured Data:** Even tabular data, like datasets in spreadsheets, can be represented as 2D tensors (matrices), where rows are samples and columns are features.



Tensors in Neural Network Operations:

- **Weights and Biases are Tensors:** The learnable parameters of a neural network—weights and biases—are all stored as tensors. For example, in a fully connected layer, the weights connecting one layer to the next are typically represented as a matrix (a 2D tensor).
- **Layer Inputs and Outputs are Tensors:** As data flows through a neural network, the input and output of each layer are tensors. Each

layer performs some operation on the input tensor to produce an output tensor, which then becomes the input for the next layer.

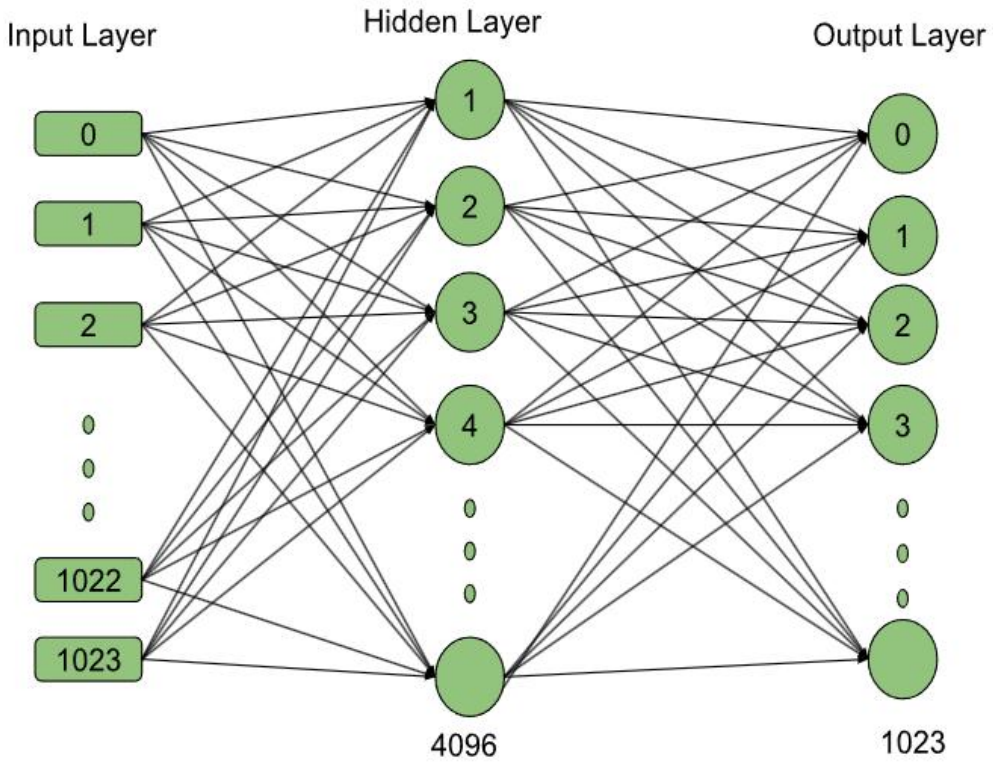
Tensor Operations are the Core Computations: The operations within neural networks are fundamentally tensor operations. These include:

- **Matrix Multiplication (and more generally, tensor contractions):** Crucial for linear transformations within layers.
- **Element-wise Operations:** Operations like addition, multiplication, activation functions (ReLU, sigmoid, etc.) are applied element-wise on tensors.
- **Convolutions:** In Convolutional Neural Networks (CNNs) for image processing, convolution operations are tensor operations that extract features from images.
- **Pooling:** Operations like max-pooling or average-pooling reduce the dimensionality of tensors, also a tensor operation.

Tensors in Large Language Models (LLMs):

- **Word Embeddings as Tensors:** In LLMs, words are first converted into word embeddings, which are vectors (1D tensors) representing the semantic meaning of words in a high-dimensional space. For example, a word embedding might be a vector of 300 numbers.
- **Transformer Networks and Attention Mechanisms:** The architecture of modern LLMs, like those based on Transformers, heavily relies on tensor operations, especially in the **attention mechanism**. The attention mechanism calculates relationships between words in a sentence (or tokens in a sequence) using tensor operations like matrix multiplications and dot products. These operations are crucial for understanding context and dependencies in language.
- **LLMs Process and Generate Text with Tensors:** When an LLM processes text, it's manipulating tensors at every step. It takes input text, converts it to tensors (word embeddings), performs a series of tensor operations through its layers (including attention mechanisms), and finally generates output text by converting the output tensors back into words.
- **Data Scientists' and ML Engineers' Perspective:** For us, tensors are the fundamental data structure we work with daily. We think about data, model parameters, and computations in terms of tensors. When we design and train models, we are essentially designing and optimizing sequences of tensor operations to achieve a specific task,

like language translation, text generation, or image classification. We perceive tensors as the way machines "understand" and process complex data, enabling them to learn patterns and make intelligent decisions.



The ALGEBRA of Tensors

Tensors. The very word can evoke a mix of awe and confusion, especially when encountered for the first time. Often described as "*multidimensional arrays*" by physicists and "*multilinear maps*" by mathematicians, tensors seem to live in a realm where intuition struggles to keep pace. This article aims to cut through the mystery and explain, in a clear and accessible way, what tensors truly are, drawing upon insightful perspectives and the underlying algebra.

The Initial Confusion: Two Sides of the Same Coin

One of the primary sources of confusion stems from the seemingly different languages used by physicists and mathematicians when

describing tensors. Physicists often define tensors through their *transformation properties* under coordinate changes. They talk about sets of numbers that transform in a specific way when you switch from one coordinate system to another. This definition can feel abstract and detached from a concrete object.

Mathematicians, on the other hand, offer a more direct approach, defining tensors as *multilinear maps*. A covariant 2-tensor, for example, is simply defined as a bilinear map that takes two vectors and returns a scalar. This definition, while elegant, might not immediately reveal the "multidimensional array" aspect that physicists emphasize.

> The truth, as we will explore, is that both perspectives are describing the *same fundamental object*. They are just emphasizing different facets of tensors: their behavior under transformations (physicists) and their inherent mathematical nature as mappings (mathematicians).

1: Building Intuition with Bases and Linear Maps

Before we tackle tensors, recall the familiar setting of linear maps. If you have a linear map:

$$\mathbf{L:V\to U}\ ,$$

its action is completely determined by what it does to a basis of V. Given a basis $\{e_i\}$ for V and any vector

$$\mathbf{x=\sum_i x_i\ e_i\ ,}$$

linearity tells us that

$$\mathbf{L(x)=L(\sum_i x_i e_i\ )=\sum_i x_i L(e_i\ )\ .}$$

In coordinates, the coefficients of $L(e_i)$ (with respect to some basis of U) form the columns of the matrix representing L.

> This idea is the first step toward understanding tensors. Instead of considering linear maps on single vectors, we now turn to **multilinear maps**—functions that are linear in each of several arguments.

By M.Asmaad Saeed

2. From Bilinear Maps to the Tensor Product

Consider a bilinear map

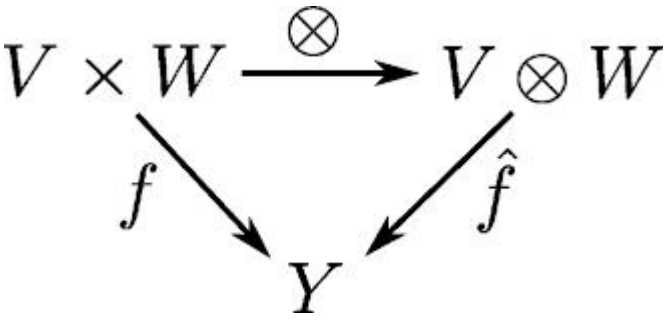
B:V×W→U.

If $\{e_i\}$ is a basis for V and $\{f_j\}$ for W, then for any vectors

$\mathbf{x=\sum_i x_i e_i}$ **and** $\mathbf{y=\sum_j y_j f_j}$ **,**

bilinearity implies:

$\mathbf{B(x,y)=B(\sum_i x_i e_i\ ,\ \sum_j y_j f_j)=\ \sum_{i,j} x_i y_j B(e_i\ ,\ f_j)}$ **.**



In the special case when U=*R* , the N = *mn* real numbers {B(ei,fj)} completely determine the bilinear map B. Notice that to compute B(x,y), you need not know the full vectors x and y individually, but rather the products xi yj for all i and j.

This observation leads to a two-step process:

1. **Formation of an Intermediate Object:** Construct an N-dimensional vector space T (all such spaces are isomorphic) with a basis $\{g_k\}$. Map the pair (x,y) into T by forming a new vector whose coordinates are given by the products
   $(x_1 y_1, x_1 y_2, \ldots, x_i y_j, \ldots)$.
2. **Reduction via a Linear Map:** Define a linear map

B~:T→R

whose action is determined by the entries $B(e_i, f_j)$. Then, by construction,

B(x,y)=B~(x ⊗ y),

where $x \otimes y$ denotes the image of (x,y) in T.

By M.Asmaad Saeed

We define the space T as the **tensor product** of V and W, written

$T=V \otimes W$.

An element of **$V \otimes W$** is called a **tensor**. The simple tensor **$x \otimes y$** is called an **elementary tensor** or **decomposable tensor**. Importantly, while every elementary tensor comes from a single pair (x,y), a general tensor is a linear combination of elementary tensors and typically does not decompose into a single product.

3. The Universal Mapping Property

The construction above is more than just a convenient bookkeeping device—it satisfies a profound and elegant property known as the **universal mapping property**. In abstract terms, the tensor product **$V \otimes W$** is defined together with a bilinear map

$\otimes : V \times W \to V \otimes W$

such that for any vector space X and any bilinear map

$f: V \times W \to X$,

there exists a **unique linear map**

$\tilde{f} : V \otimes W \to X$

that makes the diagram commute:

$f = \tilde{f} \circ \otimes$.

In simpler terms, every bilinear map from V×W factors uniquely through $V \otimes W$. *This property is the heart of why the tensor product is so powerful*—it converts the nonlinear problem of dealing with bilinear maps into the linear realm, where we have more tools at our disposal.

4. Coordinate Independence and Basis Transformations

A practical construction of the tensor product might initially use bases to define the mapping $\otimes$. However, one of the most important properties of the tensor product is its **basis independence**. Although the coordinates of a tensor may change under a change of basis, the abstract tensor itself remains unchanged.

By M.Asmaad Saeed

For example, if a bilinear map **B:V×W→R** is given and we compute its coordinates B(ei ,fj)=Bij with respect to bases {ei } and {fj }, then under a change of basis these coordinates transform in a prescribed way:

$$t_{ij}' = \sum_{a,b} R_{ia} R_{j\_b} t_{a\_b},$$

where R_{ia} represents the **change-of-basis matrix**. This coordinate transformation property is exactly what physicists refer to when they say that tensors "transform in a particular way under change of coordinates."

5. Two Perspectives: Mathematicians vs. Physicists

Mathematicians typically define a tensor in a coordinate-free manner. For instance, a **covariant 2-tensor** is defined as a bilinear map:

$$t: V \times V \to R.$$

Once a basis {ei } is chosen, the bilinear map can be expressed in coordinates:

$$t(v,w) = \sum_{i,j} v_i w_j t(e_i, e_j),$$

where t(ei ,ej) are the components of the tensor. In more abstract language, this tensor is identified with an element of the tensor product space $V^* \otimes V^*$.

The Physicist's View

Physicists, on the other hand, often start with the components of a tensor in a given coordinate system. For example, a covariant 2-tensor is given as a set of numbers {tij } that satisfy a specific transformation law under a change of coordinates:

$$t_{ij}' = \sum_{a,b} R_{ia} R_{jb} t_{ab}.$$

For physicists, these numbers are "**ephemeral**" in the sense that they depend on the choice of basis, whereas the abstract tensor—viewed as a bilinear map—is the "real" object.

> Both perspectives describe the same underlying structure,
> but the mathematician's approach emphasizes the

coordinate-free, universal nature of tensors, while the physicist's approach focuses on how tensors behave under changes of coordinates.

Conclusion

Tensors, though sometimes introduced as mysterious arrays of numbers, are best understood as the fundamental objects that arise when one replaces bilinear (or, more generally, multilinear) maps with linear maps via the tensor product. This construction, characterized by the universal mapping property, allows us to study complex interactions in a coordinate-free way. Whether you are a mathematician who cherishes abstract elegance or a physicist who needs to track how quantities transform under change of coordinates, the **tensor framework** provides a powerful, unifying language.

In summary, a tensor is not merely a multidimensional array but an abstract element of a tensor product space—an object defined by how it interacts with linear maps and coordinate changes. Once you grasp this algebraic foundation, many of the applications in geometry, physics, and even machine learning become more transparent.

In my opinion, By understanding the algebra behind tensors, we see that the seemingly different definitions from various disciplines are simply different faces of the same deep mathematical concept.

*Don't Forget follow '*www.linkedin.com/in/asmaad-saeed-981524302*' for more insightful Articles on Data Science, Machine Learning, Neural Nets, Mathematics, RAGs, GANs and NLPs.*

References

1: Images : Wikipedia
2: Algebra Intuitons : https://math.stackexchange.com/questions/10282/an-introduction-to-tensors

By M.Asmaad Saeed