The Concept of Stride

Stride s for an N-dimensional array helps to map its multi-dimensional index to a single dimensional index in memory. For an N-dimensional array with shape $(d_1, d_2, \dots, d_N)$, stride is defined for each dimension.

Computation of Stride

The stride of an N-dim array is computed based on its shape. We calculate the stride in reverse order, from the last dimension to the first. For the last dimension (the innermost), the stride is always 1. As we move to outer dimensions, we multiply the current stride by the size of the current dimension.

Given an N-dim array shape $(d_1, d_2, \dots, d_N)$, the stride s for each dimension is calculated as:

$$\begin{gathered} s_N = 1 \\ s_{N-1} = s_N \times d_N \\ s_{N-2} = s_{N-1} \times d_{N-1} \\ \vdots \\ s_1 = s_2 \times d_2 \end{gathered}$$

Example

Consider a 2D array (matrix) with shape (3, 4):

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & h & l \end{bmatrix}$$

The stride for the second dimension (columns) is $s_2 = 1$ since you move one step in memory from a to b. For the first dimension (rows), $s_1 = s_2 \times d_2 = 1 \times 4 = 4$ as you jump 4 steps in memory to move from a to e.

Indexing using Stride

To find the memory location (1D index) of an element in the tensor given its multi-dimensional coordinates $(c_1, c_2, \dots, c_N)$, we use:

$$\text{index} = \sum_{i=1}^{N} c_i \times s_i$$

This equation gives the offset in the single-dimensional memory layout based on the multi-dimensional coordinates of the element.