Our goal is an optimisation of operations of type

$$g(f(\ldots), \ldots)$$

where f is a function that produces a big matrix and g is a reduce function that makes the big matrix small.

Example 1: $\boxed{A^T \times B \times v}$, where $A, B \in \mathbb{R}^{M \times N}$ and $v \in \mathbb{R}^N$

so our f here is

$$f: \mathbb{R}^{N \times M} \times \mathbb{R}^{M \times N} \to \mathbb{R}^{N \times N}$$
$$A^T, B \mapsto A^T \times B$$

and

$$g: \mathbb{R}^{N \times N} \times \mathbb{R}^N \to \mathbb{R}^N$$
$$(A^T \times B), v \mapsto (A^T \times B) \times v$$

the problematic matrix is a result of $f(A, B)$

Example 2: (Distance)

$$\boxed{f(a, b) = C \quad \text{and} \quad g(C, v) = C \times v}$$

where $C \in \mathbb{R}^{N \times N}$ and $C_{ij} = [a_i - b_j]_{ij}$,
$a \in \mathbb{R}^N$, $b \in \mathbb{R}^N$, $v \in \mathbb{R}^N$

Example 3: (multiplication)

The same as in Example 2, but
$C_{ij} = [a_i * b_j]_{ij}$

Example 4: (Squared distance)

$f(A, B) = C$, where $A \in \mathbb{R}^{N \times M}$, $B \in \mathbb{R}^{N \times M}$

$$C_{ij} = \left[\left(\sum_k A_{ik} - B_{jk} \right)^2 \right]_{ij}, \quad \text{s.t. } C \in \mathbb{R}^{N \times N}$$

$$g(C, v) = C \times v$$

In Example 1, for $A^T \times B \times v$ we could simply re-arrange computation, because of associativity property of the matrix product, so instead of computing $(A^T \times B)$ first that gives $N \times N$ matrix, we could compute $(B \times v)$ first that gives N vector, and then $A^T \times (B \times v)$.
It reduces the memory requirements and computational cost.

Re-arranging is a partial solution, the generic solution would be chunking/splitting/partitioning big matrix computations and never materialise them at once on the same device. This could be referred as cache-wise computation.

E.g. for matrix-vector multiplication (MVM):

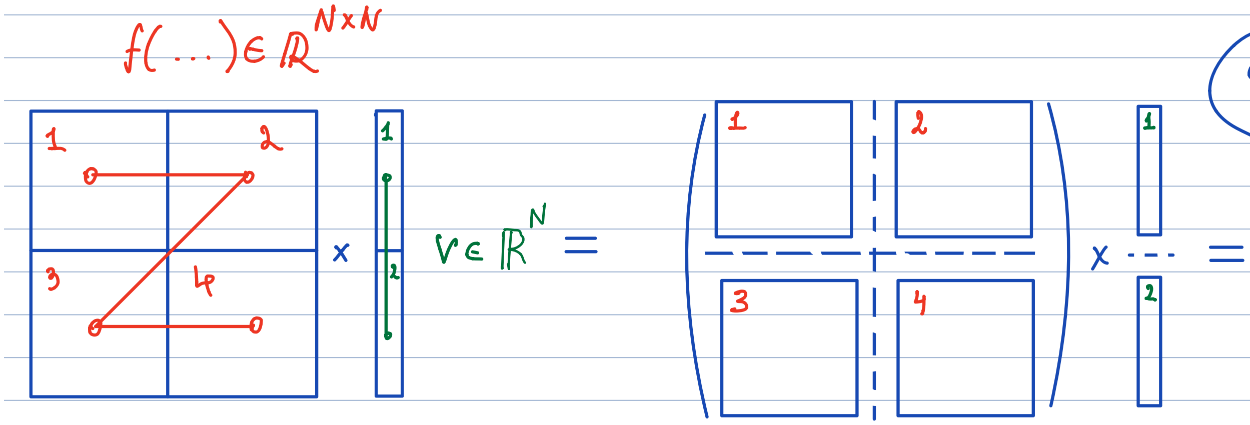
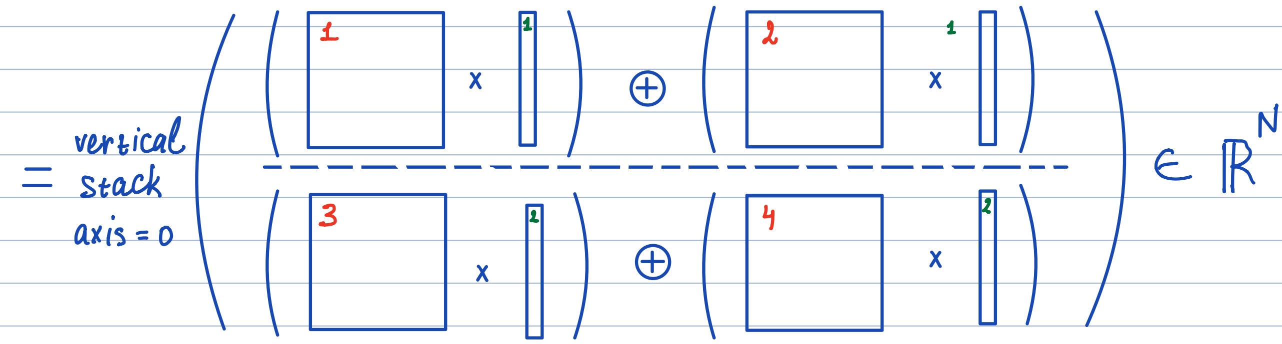


Diagram 1



$\oplus$ - is an element-wise addition operation

Now let's imagine that matrix $f(\dots)$ is a result of outer product $(x y^T)$ where $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^N$, and $xy^T \in \mathbb{R}^{N \times N}$

So for $f(x, y) = xy^T$:

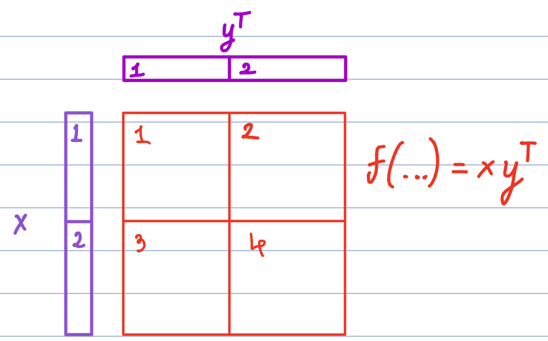
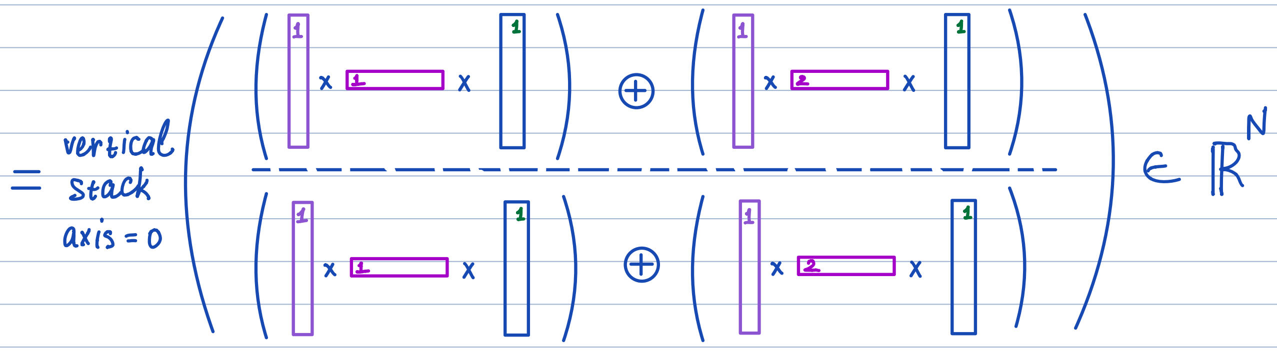


Diagram 1 becomes:



This is a super efficient computation as 2nd dot product gives a scalar and 1st dot product becomes simply an element-wise multiplication of a scalar and a vector.

The total cost of operation is $O(N)$ and memory $O(N)$