Four Pictures of One Matmul:
Regrouping $C = WV$, with an Attention Reading

Japjot Singh

derived by pen, October 26, 2026; typeset from the handwritten original; extended from the code session of October 27

Abstract

A matrix product is a single three-index sum, $C[i,j] = \sum_k W[i,k]\,V[k,j]$, and every standard "picture" of it (element, row, column, outer product) is nothing but a choice of which index to group first. I derive the first three pictures by hand, through the one move that was genuinely non-obvious to me: pulling a summation outside a vector bracket (sum-then-stack equals stack-then-sum). I then read the result back into attention, where W is a softmax weight matrix and V the value matrix: the row picture says each context vector is a convex combination of visible value rows, so attention is a soft gather that interpolates and never extrapolates.

Color code, used consistently below: the summed index k wears one color per value, k=0 teal, k=1 orange, k=2 violet. Column k of W and row k of V wear the *same* color because they are consumed together; watching the colors regroup *is* the derivation.

1 Setup and notation

Throughout, $W \in \mathbb{R}^{T\times T}$ is an attention weights matrix and $V \in \mathbb{R}^{T\times d}$ is the value matrix (row k is token k's payload); $C = \text{ctx} = WV \in \mathbb{R}^{T\times d}$. Both axes of W index the same T tokens, in two roles: row i is token i *acting as the query*, and entry $W[i,j]$ is how much query-token i attends to key-token j. So "position i's softmax recipe" means: row i is token i's distribution over the tokens it may read from. A *token* is one position in the sequence (in real models a subword piece, not a word). In every (T,d) tensor, rows index tokens and columns index feature dimensions. Everything in §2–7 holds for any matmul; attention enters in §8.

Scale and asymmetry (re-read questions, 10/27). W is $T \times T$ over the **current sequence's positions**, never the vocabulary: its size is context length squared, independent of vocab size (the vocabulary appears only at the embedding and unembedding ends). Token i attends to every visible j *by construction*: the score $q_i \cdot k_j$ is computed for all pairs and only the causal mask removes $j > i$; there is nothing to ensure. The price is exactly that T^2 (the FLOPs remark in §4); making it scale IS the long-context problem (T10). Asymmetry: beyond masking, $W[i,j] \neq W[j,i]$ because relevance is directional ("it" must look at "cat"; "cat" has no need of "it"), and mechanically because scores form $S = XW_qW_k^{\mathsf{T}}X^{\mathsf{T}}$ with $W_q \neq W_k$: an *asymmetric* bilinear form. Setting $W_q = W_k$ would force a symmetric (Gram) score matrix. The backprop formulation gets its own note (queued).

Vectors are column vectors by convention, so row i of W is properly written w_i^T and a matrix is a stack of transposed rows. Because w_i in the wild sometimes means column i, I use slice notation, which is unambiguous and matches code:

$$W[i,:] = \text{row } i \text{ of } W, \qquad W[:,j] = \text{column } j \text{ of } W. \tag{1}$$

Working discipline that resolved repeated confusion: first decide *row or column*, only then worry about transposes.

Shape is readable straight off the brackets, no arrows needed: two fixed indices ($W[i,k]$) is a **scalar**; one colon ($V[k,:]$, $W[:,k]$) is a **vector**; a bare capital letter is a matrix. Standalone lowercase vectors, which carry no bracket cue, wear an arrow ($\vec{v}_2$), matching the handwritten original.

Nomenclature, fixed once (imprecision here cost real time):

- a **term** is one product $W[i,k]V[k,j]$: the atom of the whole computation;
- an **entry** is one position of an output vector or matrix (e.g. $C[i,j]$); terms get summed *into* entries;
- a **cell** is one position of a term grid (§3); one cell holds one term;
- a **sheet** is all terms sharing one value of k (§4); each sheet is a matrix of rank *at most* one — rank 0 exactly when either factor vanishes, as the example's sheet 0 does (10/27 harness catch). (Reminder: rank one = an outer product $\vec{u}\,\vec{w}^\mathsf{T}$: every row is a multiple of the same $\vec{w}^\mathsf{T}$, every column a multiple of the same $\vec{u}$. Example: $\left(\begin{smallmatrix}2&4\\3&6\end{smallmatrix}\right) = \left(\begin{smallmatrix}2\\3\end{smallmatrix}\right)(1 \;\; 2)$.)

The *element picture* keeps its conventional name; an element is just an entry.

Geometric remark (for the Axler reader). A matmul is a **composition of linear maps**. Acting on column vectors, $V : \mathbb{R}^d \to \mathbb{R}^T$ and $W : \mathbb{R}^T \to \mathbb{R}^T$, so $C = WV$ is $W \circ V$, and the shared axis is the **middle space** of the composition: the codomain of V glued to the domain of W. The contracted index k runs over the middle space's basis, and summing over k is the handoff: resolve the intermediate vector into coordinates along that basis, feed the coordinates onward. Two things to store: (i) everything must squeeze through the middle space, so $\operatorname{rank}(WV) \le \dim(\text{middle}) = J$: the sheets of §4 are precisely one rank-one contribution per middle-basis direction; (ii) a deliberately *thin* middle space is what low-rank structure means (LoRA's BA: rank r because the handoff happens in $\mathbb{R}^r$). For attention, the middle space is **token space**: row i's recipe is position i expressed in coordinates-over-tokens, and V's payload rows re-embed those token coordinates into feature space. There is no geometric picture of "the k axis" inside the output plane, and that's the point: k lives in a different space, the one the computation passes through.

The running example ("the triangle numbers"), written in full. Column k of W and row k of V share a color: the shared axis, made visible:

$$W = \begin{pmatrix} \frac{1}{10} & \frac{7}{10} & \frac{2}{10} \\ 1 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}, \qquad V = \begin{pmatrix} 0 & 0 \\ 4 & 0 \\ 0 & 4 \end{pmatrix}, \qquad C = WV = \begin{pmatrix} 2.8 & 0.8 \\ 0 & 0 \\ \frac{4}{3} & \frac{4}{3} \end{pmatrix}. \tag{2}$$

2 Element picture (the straight product)

Computing $C[0,0]$, $C[0,1]$, $C[1,0]$ by hand on (2) and closing the form:

$$\boxed{C[i,j] \;=\; \sum_{k=0}^{J-1} W[i,k]\, V[k,j]} \qquad J := \#\text{cols of } W = \#\text{rows of } V. \tag{3}$$

In words: take all k across row i of W and all k down column j of V and match them up. Each entry of C is one completed dot product (row of W against column of V). The summed index k must be shared, which is exactly the shape rule; here k runs over tokens.

> ***Stuck point.*** *I wrote row i of C as a bracket of entries,*
>
> $$C[i,:] \;=\; \Big[\; W[i,:]\cdot V[:,0] \quad W[i,:]\cdot V[:,1] \;\Big], \tag{4}$$
>
> *and tried to read a linear combination directly off it. Impossible: each entry is a completed* scalar*; the dot products have already collapsed the grouping. A linear combination needs $\sum(\text{scalar}\cdot\text{vector})$ terms, and (4) has no vectors left in sight.*

3 The regrouping lemma

Lemma 1 (sum-then-stack = stack-then-sum)**.** *Vector addition is entrywise, so a vector whose entries are each a sum of J terms equals a sum of J vectors, one per term position:*

$$\begin{pmatrix} a+b+c \\ d+e+f \end{pmatrix} \;=\; \begin{pmatrix} a \\ d \end{pmatrix} + \begin{pmatrix} b \\ e \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix}. \tag{5}$$

The bracket is purely *organizational*: it stores numbers in entry positions and does no arithmetic of its own. All the arithmetic (the +) happens entry-by-entry, which is why the bracket and the summation pass through each other freely. Geometrically: each column vector is a displacement arrow in the plane. The left side walks *one* arrow whose coordinates were pre-totaled; the right side walks three arrows tip-to-tail. Same endpoint, because adding displacements is path-joining and the resultant does not care how the walk is grouped: that is commutativity and associativity, drawn.

Why it is legal. First fix i; everything below is the anatomy of **one output row**, $C[i,:]$ (there is one such grid per row of C; i names which grid you are holding). Every cell is one term $W[i,k]\,V[k,j]$, and the recognition that unlocks the whole grid: **the k is shared**: the same k sits in W's column index and V's row index, so the color below sits on the shared index itself. Margins accumulate each reading:

row i's grid	$j=0$	$j=1$	$\xrightarrow{\text{stack along } j}$ (nothing summed)
$k=0$	$W[i,0]\,V[0,0]$	$W[i,0]\,V[0,1]$	$W[i,0]\;V[0,:]$ (scalar · vector)
$k=1$	$W[i,1]\,V[1,0]$	$W[i,1]\,V[1,1]$	$W[i,1]\;V[1,:]$
$k=2$	$W[i,2]\,V[2,0]$	$W[i,2]\,V[2,1]$	$W[i,2]\;V[2,:]$
$\downarrow$ sum over k	$C[i,0]$	$C[i,1]$	$C[i,:]$ (both readings land here)

Read **down** a j-column: sum its three cells; that sums k away and yields the single entry $C[i,j]$: sum-then-stack, the element picture. Read **across** a k-row: the two cells are *not* summed; they are stacked side by side into the vector $W[i,k]\,V[k,:]$ (j is never summed by any picture: it labels entries); the three row-vectors are then summed over k: stack-then-sum, the row picture. **The only index ever summed is k.** Finite sums may be evaluated in any order (commutativity and associativity; finite Fubini), so the two readings agree on the corner total. The column picture of §6 lives on a sibling grid: fix j instead of i, and the grid's axes become (k,i). □

4 The term block: one object, four readings

All the grids above are slices of one object. The three-index sum ranges over a $T \times J \times d$ **block of terms** with axes (i,k,j); in code, C = `einsum("ik,kj->ij", W, V)`, where i,j are the *free* indices (they survive into C) and k is the *contracted* index (it is summed away). "The grid is fixed per i" now has a precise meaning: §3's grid is the i-th slice of this block, one grid per value of the frozen free index.

Fixing k instead slices the block the third way, into **sheets**, and each sheet has rank at most one. On the example:

$$C = \underbrace{\begin{pmatrix}0&0\\0&0\\0&0\end{pmatrix}}_{W[:,0]\otimes V[0,:]} + \underbrace{\begin{pmatrix}2.8&0\\0&0\\\frac{4}{3}&0\end{pmatrix}}_{W[:,1]\otimes V[1,:]} + \underbrace{\begin{pmatrix}0&0.8\\0&0\\0&\frac{4}{3}\end{pmatrix}}_{W[:,2]\otimes V[2,:]} \tag{6}$$

Sheet 0 vanishes because $\vec{v}_1 = (0,0)$: attending to a token with a zero payload fetches nothing, at any attention weight. Attention reading of a sheet (10/27): sheet k is token k's payload $V[k,:]$ broadcast to every output row and scaled, row by row, by $W[:,k]$ — how much each token attends to token k. One sheet = **token k's total contribution to every context vector**, and truncating the sheet sum to the largest contributors is what a low-rank approximation of C means. Every picture is now one sentence: **slice the block, group what remains, sum.** Figure 1 draws the block and the three cuts.

picture	slice	group & sum	result
element	fix (i,j)	the J cells on that skewer	one entry of C
row	fix i	the (k,j) grid, by k	$C[i,:] = \sum_k$ (scalar) · (row of V)
column	fix j	the (k,i) grid, by k	$C[:,j] = \sum_k$ (scalar) · (column of W)
outer	nothing	the whole block, by k	$C = \sum_k \text{sheet}_k$, each rank one

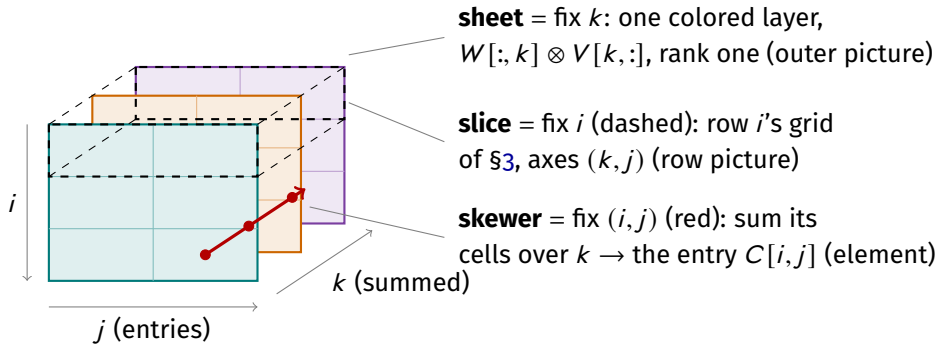


Figure 1: The term block for T=3, J=3, d=2: one cell per term $W[i,k]V[k,j]$, axes (i,k,j). The three cuts: fix $k \to$ a **sheet** (one colored layer, rank one); fix $i \to$ a **slice** (dashed), which is exactly the marginal-totals grid of §3; fix $(i,j) \to$ a **skewer** (red), whose J cells sum to one entry. k is the only summed axis; i and j survive into C. The column picture is the vertical sibling slice (fix j), not drawn.

Fluency remark (pictures are loop orders; found in code, 10/27). In code the three-index sum is a triple loop, and each picture is a **nesting order**, with the loops not written by hand absorbed into slices and broadcasting: element (i,j,k), row (i,k,j), column (j,k,i), outer (k,i,j) — writing the outer picture in primitive operations is the same three loops with k outermost, nothing more. Reordering is legal by exactly Lemma 1 (finite sums commute and associate). Open (queued in Still open): $3! = 6$ orderings but only four pictures — which orderings coincide, and what does that say a "picture" is?

Fluency remark (FLOPs are terms). Counting a matmul's work is counting the block's cells: $T \cdot J \cdot d$ multiply-adds, $\approx 2TJd$ FLOPs. In attention, QK^T and weights $\cdot V$ are blocks of size $T \times d_k \times T$ and $T \times T \times d$; the quadratic-in-T cost of attention is literally the size of these blocks.

Fluency remark (free-index sums are marginals; 10/27). Only k is ever contracted; summing a *free* index is legal arithmetic answering a different question, and it factors cleanly because a free index lives in just one factor: $\sum_i W[i,k]\,V[k,j] = \left(\sum_i W[i,k]\right) V[k,j]$, so B summed over i has row k equal to $V[k,:]$ scaled by the **column sum of** W — token k's payload times the total attention it *receives* (§7). Likewise B summed over j has entry (i,k) equal to $W[i,k]\sum_j V[k,j]$: the attention weight times token k's total feature mass — dubious units, since it adds features together. The contracted index is special precisely because it is the only index appearing in *both* factors; summing it is the only sum that couples W to V. These are the margins of §3's grid, grown to the full block. (Part 2 of `drill_matmul_pictures.py` — block, sheets, slices, skewers — all implemented and PASS, 10/27.)

5 Row picture

Fix i: the output row comes from row i of W, and its entries walk the columns of V. Expand each entry of (4) into its $\sum_k$ form, apply Lemma 1 (watch the colors regroup), and factor the shared

scalar out of each colored bracket. Compact subscripts as in §6: $W_{ik} := W[i,k]$.

$$\begin{aligned} C[i,:] &= \begin{bmatrix} W_{i0}V_{00} + W_{i1}V_{10} + W_{i2}V_{20}\,, & W_{i0}V_{01} + W_{i1}V_{11} + W_{i2}V_{21} \end{bmatrix} \\ &= W_{i0}\begin{bmatrix} V_{00} & V_{01} \end{bmatrix} + W_{i1}\begin{bmatrix} V_{10} & V_{11} \end{bmatrix} + W_{i2}\begin{bmatrix} V_{20} & V_{21} \end{bmatrix} \\ &= \boxed{\sum_k \underbrace{W[i,k]}_{\text{scalar}}\ \underbrace{V[k,:]}_{\text{vector}}} \end{aligned} \tag{7}$$

Scalars come from row i of W; the vectors are the *rows* of V. Numeric check on the example's row 0 (recipe $[\frac{1}{10}, \frac{7}{10}, \frac{2}{10}]$):

$$\tfrac{1}{10}\,(0,0) + \tfrac{7}{10}\,(4,0) + \tfrac{2}{10}\,(0,4) = (2.8,\ 0.8), \quad \text{matching } C \text{ in (2).} \tag{8}$$

> ***Stuck point.*** *Pulling $\sum_k$ outside the bracket. Resolution: write all six products out with real numbers, then pattern-match Lemma 1 by pairing terms that share a coefficient (= share a color). Also: keep the dummy index k reserved; never reuse it as a free row name.*

6 Column picture

Freeze j: the output column comes from column j of V, and the stack walks the rows of W. Same lemma, applied vertically ("sum then stack = stack then sum"), then factor out the shared scalar $V[k,j]$. Compact subscripts in the big display: $W_{ik} := W[i,k]$.

$$\begin{aligned} C[:,j] &= \begin{pmatrix} W_{00}V_{0j} + W_{01}V_{1j} + W_{02}V_{2j} \\ W_{10}V_{0j} + W_{11}V_{1j} + W_{12}V_{2j} \\ W_{20}V_{0j} + W_{21}V_{1j} + W_{22}V_{2j} \end{pmatrix} = \begin{pmatrix} W_{00} \\ W_{10} \\ W_{20} \end{pmatrix} V_{0j} + \begin{pmatrix} W_{01} \\ W_{11} \\ W_{21} \end{pmatrix} V_{1j} + \begin{pmatrix} W_{02} \\ W_{12} \\ W_{22} \end{pmatrix} V_{2j} \\ &= \boxed{\sum_k \underbrace{V[k,j]}_{\text{scalar}}\ \underbrace{W[:,k]}_{\text{vector}}} \end{aligned} \tag{9}$$

Scalars come from column j of V; the vectors are the *columns* of W. Column j of C is a linear combination of the columns of W, mixed by the elements in column j of V.

7 The symmetry, and where the output lives

In both pictures the summed index k is the same shared axis (columns of W = rows of V = tokens; one color per k throughout). What flips is which matrix donates scalars and which donates vectors:

picture	scalar donor	vector donor
row (7)	W (row i: the recipe)	V (its rows)
column (9)	V (column j)	W (its columns)

Influence (added on re-read, October 27). Fix a single coefficient $W[i,k]$: it scales exactly one donated vector, $V[k,:]$, and through it touches **every** entry of row $C[i,:]$. One coefficient, one vector, full span of the output slice. Symmetrically, $V[k,j]$ touches every entry of column $C[:,j]$, only through $W[:,k]$. The correction worth engraving: within one donated term, k does not move. *k enumerates* which vector is donated; the axis a donated vector *extends along* is j (row picture) or i (column picture). This influence map is also the seed of backprop: $\partial C[i,j]/\partial W[i,k] = V[k,j]$, so gradients flow back along exactly these one-coefficient-one-vector paths. (The queued backprop note starts here.)

Terminology, on purpose: *recipe* is reserved for a softmax row of W (entries ≥ 0, summing to 1: convex weights). Column j of V is an arbitrary coefficient vector with neither property, so the column picture says "mixed by," never "recipe." The asymmetry is semantic (it comes from attention), not algebraic: as plain matmul the two pictures are perfectly symmetric. And the same caution applies to W itself (10/27): softmax normalizes **rows** only, so the columns of W are not distributions — nothing constrains them — and a column sum $\sum_i W[i,k]$ has its own meaning, the total attention token k *receives*.

Fact 1. *Rows of WV live in the row space of V (the right factor). Columns of WV live in the column space of W (the left factor). Both statements hold of the same matrix simultaneously.*

Information flow, and the batch reading (10/27). Stronger than the donor table: entry $C[i,j]$ reads **only** $W[i,:]$ and $V[:,j]$ — the element formula (3) touches nothing else — so no grouping can ever mix two columns of V or two rows of W. Hence WV is a **batch of independent subproblems**, two ways. Column reading: W is the operator and V's columns are the batch, $C[:,j] = W\,V[:,j]$. Row reading: V is the operator and W's rows are the batch, $C[i,:] = W[i,:]\,V$. The two pictures disagree only about who is the machine and who is the data. In attention this independence is load-bearing: each row of the weight matrix is an independent query into the shared V, which is exactly why single-token decoding against a KV cache is legal — the new token's row is a new batch item and cannot disturb any earlier output.

8 The attention reading

The row picture carries attention's meaning: $\mathrm{ctx}[i,:]$ is row i's *recipe* over token payloads, and attention is a content-addressed soft *gather*: a one-hot recipe copies one row of V exactly; a uniform recipe averages them; softmax temperature is the dial between the two.

Fluency remark (attention is an expectation). Row i's recipe is a probability distribution p_i over the visible tokens, so

$$\mathrm{ctx}[i,:] \;=\; \mathbb{E}_{k\sim p_i}\, V[k,:]. \tag{10}$$

The hull statement below is then the standard probability fact that an expectation lies in the convex hull of its support, and temperature tunes the entropy of p_i: zero entropy = hard gather, maximum entropy = centroid.

The center-of-mass reading (code-session aha, 10/27, kept in the words it was re-earned in). "That row of W did a **weighted average** of the rows of V — a center of mass — so the output is stuck in the convex hull of V's rows; the 0.7 drags row 0's output toward $\vec{v}_2 = (4,0)$." Formally:

Because the recipe is a softmax row (entries ≥ 0, summing to 1), Fact 1 tightens: $\mathrm{ctx}[i,:]$ lies in the *convex hull* of the value rows visible to position i. The hull constraint is a byproduct of exactly softmax's two properties: nonnegativity and sum-to-one are the definition of convex weights. How much tighter than the span? In the example, maximally: $\vec{v}_2$ and $\vec{v}_3$ are linearly independent, so the span of the value rows is *all of* $\mathbb{R}^2$ and Fact 1 says nothing; the softmax alone shrinks the reachable set to the triangle. That is why $(4,4)$ is worth marking in Figure 2: it is in the span but not the hull ($1 \cdot \vec{v}_2 + 1 \cdot \vec{v}_3$ has coefficients summing to 2), so **no pattern of attention scores can make position i output it**: a hard expressivity limit. Attention interpolates, never extrapolates; escaping the hull requires different values (W_v), the MLP, or the residual stream. (Causal row 0 sees only itself, so its recipe is forced to $[1,0,\ldots]$.)

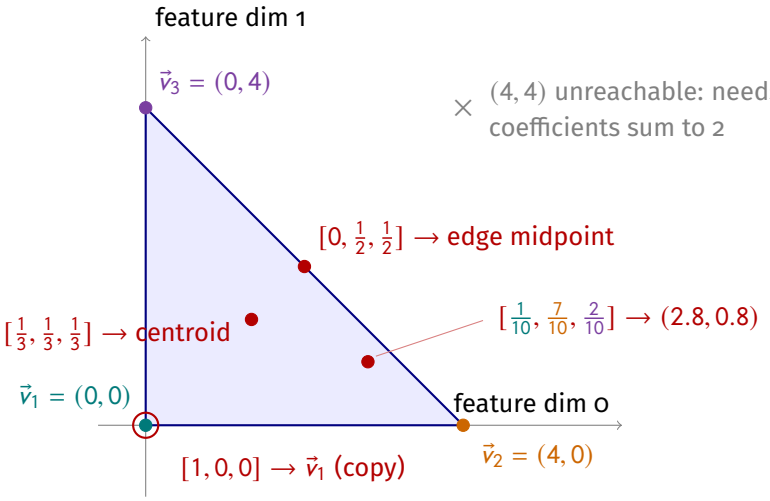


Figure 2: The convex-hull geometry of one attention row, on the running example (2). The value rows $\vec{v}_1, \vec{v}_2, \vec{v}_3$ (colored by their k, matching every equation above) span the shaded triangle; every realizable context vector (red) is a recipe-weighted center of mass and must lie inside it. One-hot recipes sit on vertices (hard gather / `index_select`), the uniform recipe at the centroid, recipes with a zero entry on the opposite edge. No softmax recipe reaches $(4,4)$.

The four pictures at a glance

One sum, four groupings of $C[i,j] = \sum_k W[i,k]V[k,j]$:

element	fix (i,j)	$C[i,j] = \sum_k W[i,k]\, V[k,j]$	(a dot product; all scalars)
row	fix i, group by k	$C[i,:] = \sum_k W[i,k]\, V[k,:]$	rows of V, recipe from W
column	fix j, group by k	$C[:,j] = \sum_k V[k,j]\, W[:,k]$	columns of W, mixed by V
outer	group all by k	$C = \sum_k W[:,k] \otimes V[k,:]$	sum of rank-one matrices

All four are readings of one $T \times J \times d$ **term block** (axes i,k,j; k contracted): slice, group, sum. Attention = the row picture with a softmax recipe: a soft gather, $\mathrm{ctx}[i,:] = \mathbb{E}_{k\sim p_i} V[k,:]$, confined to the convex hull of visible value rows.

9 Still open

1. **Outer-product picture**: implemented 10/27 in `drill_matmul_pictures.py` (primitive triple loop with k outermost; PASS), with the sheet reading in §4. Remaining: the pen derivation from Lemma 1 in the style of §5–6, and the loop-order question — sort all $3! = 6$ orderings into the four pictures and state, in one sentence, what a picture *is*. Truncating the sheet sum is what low-rank structure means (LoRA; $W_q W_k^\top$ as a rank-d_{head} factorized bilinear form).

2. **Rank question**: given Fact 1, what bounds $\text{rank}(WV)$ relative to W and V? (From the sheets view of (6) this is nearly immediate.)