Attention Backward Pass

v0.3 · June 2026

Matthew Willetts, with assistance from Codex and Claude

Self-attention forward and backward, every step in index notation, with explicit Kronecker-delta bookkeeping.

Setup

Let $i, j \in [T]$ be sequence positions (query and key), $k \in [d]$ the head dimension, $p \in [d_v]$ the value dimension. Inputs $Q, K \in \mathbb{R}^{T\times d}$, $V \in \mathbb{R}^{T\times d_v}$. Given: $G_{ip} := \partial L/\partial O_{ip}$.

Forward.

$$S_{ij} = \frac{1}{\sqrt{d}} \sum_k Q_{ik} K_{jk}, \quad A_{ij} = \frac{\exp S_{ij}}{\sum_l \exp S_{il}}, \quad O_{ip} = \sum_j A_{ij} V_{jp}.$$

Index convention

When applying the chain rule, write the full sum over upstream indices using fresh **dummy** labels, distinct from the indices on the LHS. The derivative $\partial M_{ab}/\partial M_{jp}$ of a matrix entry with respect to another entry of the same matrix is $\delta_{aj}\delta_{bp}$ — one delta per index slot. Each delta collapses one summation. Everything below is tracking where the deltas appear and which sums they collapse.

Step 1 — $\partial L/\partial V$ (brute force)

L depends on V_{jp} only through O. Write the full chain rule with dummy indices a (output position) and b (output dimension):

$$\frac{\partial L}{\partial V_{jp}} = \sum_{a,b} \frac{\partial L}{\partial O_{ab}} \frac{\partial O_{ab}}{\partial V_{jp}} = \sum_{a,b} G_{ab} \frac{\partial O_{ab}}{\partial V_{jp}}.$$

Now the forward equation, with its own dummy summation index c:

$$O_{ab} = \sum_c A_{ac} V_{cb}.$$

To compute $\partial O_{ab}/\partial V_{jp}$, two sub-steps. First, pull the derivative inside the sum. Since A depends only on Q and K (not on V), the factors A_{ac} are constants w.r.t. V:

$$\frac{\partial O_{ab}}{\partial V_{jp}} = \frac{\partial}{\partial V_{jp}} \left[\sum_c A_{ac}\, V_{cb} \right] = \sum_c A_{ac}\, \frac{\partial V_{cb}}{\partial V_{jp}}.$$

Second, the derivative of one entry of V w.r.t. another. Entries of V are algebraically independent, so

$$\frac{\partial V_{cb}}{\partial V_{jp}} = \delta_{cj}\,\delta_{bp}$$

— one delta per index slot: the position indices c and j must match, and the dimension indices b and p must match. Combining:

$$\frac{\partial O_{ab}}{\partial V_{jp}} = \sum_c A_{ac}\,\delta_{cj}\,\delta_{bp}.$$

The δ_{cj} collapses the c-sum, surviving only $c = j$:

$$\frac{\partial O_{ab}}{\partial V_{jp}} = A_{aj}\,\delta_{bp}.$$

Substitute back:

$$\frac{\partial L}{\partial V_{jp}} = \sum_{a,b} G_{ab}\,A_{aj}\,\delta_{bp}.$$

Now δ_{bp} collapses the b-sum, surviving only $b = p$:

$$\frac{\partial L}{\partial V_{jp}} = \sum_a G_{ap}\,A_{aj}.$$

Relabel the surviving dummy $a \to i$ (purely cosmetic, makes it match the rest of the notes):

$$\boxed{\frac{\partial L}{\partial V_{jp}} = \sum_i G_{ip}\,A_{ij}.}$$

Reading the result. The sum over i is "what survived" from the dummy-output-position sum: every query i contributes to $\partial L/\partial V_{jp}$ through how much it attended to position j (the factor A_{ij}). The p index "just passes through" because δ_{bp} killed the dummy-output-dimension sum — value dimensions don't mix during attention.

Step 2 — $\partial L/\partial A$

Same chain rule, same forward equation $O_{ab} = \sum_c A_{ac}V_{cb}$. Differentiate O_{ab} w.r.t. A_{ij} in two sub-steps. First, pull the derivative inside the sum — V doesn't depend on A:

$$\frac{\partial O_{ab}}{\partial A_{ij}} = \frac{\partial}{\partial A_{ij}}\left[\sum_c A_{ac}V_{cb}\right] = \sum_c V_{cb}\,\frac{\partial A_{ac}}{\partial A_{ij}}.$$

Second, the entry-vs-entry derivative gives two deltas:

$$\frac{\partial A_{ac}}{\partial A_{ij}} = \delta_{ai}\,\delta_{cj}.$$

Combining:

$$\frac{\partial O_{ab}}{\partial A_{ij}} = \sum_c V_{cb}\,\delta_{ai}\,\delta_{cj} = V_{jb}\,\delta_{ai}.$$

The δ_{cj} collapsed the c-sum; δ_{ai} remains, ready to collapse the a-sum upstream. Substitute into the chain rule for L:

$$\frac{\partial L}{\partial A_{ij}} = \sum_{a,b} G_{ab}\,V_{jb}\,\delta_{ai} = \sum_b G_{ib}\,V_{jb}.$$

Relabel $b \to p$ to match notation:

$$D_{ij} := \frac{\partial L}{\partial A_{ij}} = \sum_p G_{ip}\,V_{jp}.$$

Here the surviving sum is over p (the output dimension), because the position index was killed by δ_{ai}. Mirror image of Step 1: there it was the position sum that survived and the dimension delta that killed.

Step 3 — $\partial L/\partial S$ through the row-wise softmax

Chain rule from A to S, with dummies a, c:

$$\frac{\partial L}{\partial S_{ik}} = \sum_{a,c} D_{ac}\,\frac{\partial A_{ac}}{\partial S_{ik}}.$$

Now the softmax Jacobian. Write $A_{ac} = e^{S_{ac}}/Z_a$ with $Z_a := \sum_l e^{S_{al}}$. Apply the quotient rule, taking each derivative explicitly.

The numerator: $\partial e^{S_{ac}}/\partial S_{ik} = e^{S_{ac}}\,(\delta_{ai}\,\delta_{ck})$ — the chain rule through exp gives the function back, times the derivative of its argument S_{ac} w.r.t. S_{ik}, which is two deltas (one per index slot).

The denominator (with its own dummy summation l):

$$\frac{\partial Z_a}{\partial S_{ik}} = \frac{\partial}{\partial S_{ik}} \sum_l e^{S_{al}} = \sum_l e^{S_{al}}\,\delta_{ai}\,\delta_{lk} = e^{S_{ak}}\,\delta_{ai}$$

— the δ_{lk} collapses the l-sum.

Quotient rule:

$$\frac{\partial A_{ac}}{\partial S_{ik}} = \frac{e^{S_{ac}}\,\delta_{ai}\,\delta_{ck}\,Z_a - e^{S_{ac}}\,e^{S_{ak}}\,\delta_{ai}}{Z_a^2} = \delta_{ai}\left[\frac{e^{S_{ac}}}{Z_a}\delta_{ck} - \frac{e^{S_{ac}}}{Z_a}\frac{e^{S_{ak}}}{Z_a}\right] = \delta_{ai}\,A_{ac}(\delta_{ck} - A_{ak}).$$

The factored-out δ_{ai} encodes "rows are independent" — Z_a depends only on $S_{a\cdot}$, so S_{ik} affects A_{ac} only if $i = a$.

Substitute:

$$\frac{\partial L}{\partial S_{ik}} = \sum_{a,c} D_{ac}\,\delta_{ai}\,A_{ac}(\delta_{ck} - A_{ak}) = \sum_c D_{ic}\,A_{ic}(\delta_{ck} - A_{ik}).$$

The δ_{ai} killed the a-sum (surviving $a = i$). Now split the parenthesis:

$$\frac{\partial L}{\partial S_{ik}} = \sum_c D_{ic}\, A_{ic}\, \delta_{ck} \;-\; A_{ik} \sum_c D_{ic}\, A_{ic}.$$

The δ_{ck} in the first term collapses the c-sum (surviving $c = k$): $D_{ik}\, A_{ik}$.

The second term has no delta; the c-sum stays. Define

$$r_i := \sum_c D_{ic}\, A_{ic}.$$

Then

$$\boxed{\frac{\partial L}{\partial S_{ik}} = A_{ik}\big(D_{ik} - r_i\big).}$$

Simplifying r_i further. Plug in $D_{ic} = \sum_p G_{ip} V_{cp}$:

$$r_i = \sum_c A_{ic} \sum_p G_{ip} V_{cp} = \sum_p G_{ip} \sum_c A_{ic} V_{cp} = \sum_p G_{ip}\, O_{ip}.$$

So r_i is the inner product of the row's incoming gradient with the row's output. Doesn't need D materialised.

Centering identity (sanity check)

Sum the boxed equation over k:

$$\sum_k \frac{\partial L}{\partial S_{ik}} = \sum_k A_{ik} D_{ik} - r_i \sum_k A_{ik} = r_i - r_i \cdot 1 = 0.$$

The gradient in S is **zero-mean per row**. Physically, shifting all logits in a row by the same constant doesn't change the softmax output, so the gradient must have no component in the all-ones direction within each row. Same flavour as the LayerNorm centering identity.

Step 4 — $\partial L/\partial Q$ and $\partial L/\partial K$ through the score

Chain rule from S to Q, with dummies a, b:

$$\frac{\partial L}{\partial Q_{ik}} = \sum_{a,b} \frac{\partial L}{\partial S_{ab}} \frac{\partial S_{ab}}{\partial Q_{ik}}.$$

Forward equation with dummy c:

$$S_{ab} = \frac{1}{\sqrt{d}} \sum_c Q_{ac} K_{bc}.$$

Differentiate w.r.t. Q_{ik} in two sub-steps. First, pull the derivative inside the sum — K doesn't depend on Q, so K_{bc} is constant:

$$\frac{\partial S_{ab}}{\partial Q_{ik}} = \frac{1}{\sqrt{d}} \sum_c K_{bc} \frac{\partial Q_{ac}}{\partial Q_{ik}}.$$

Second, the entry-vs-entry derivative:

$$\frac{\partial Q_{ac}}{\partial Q_{ik}} = \delta_{ai}\,\delta_{ck}.$$

Combine:

$$\frac{\partial S_{ab}}{\partial Q_{ik}} = \frac{1}{\sqrt{d}} \sum_c K_{bc}\,\delta_{ai}\,\delta_{ck} = \frac{1}{\sqrt{d}}\,\delta_{ai}\,K_{bk}.$$

The δ_{ck} killed the c-sum; δ_{ai} will kill the a-sum upstream. Substitute:

$$\frac{\partial L}{\partial Q_{ik}} = \frac{1}{\sqrt{d}} \sum_{a,b} \frac{\partial L}{\partial S_{ab}}\,\delta_{ai}\,K_{bk} = \frac{1}{\sqrt{d}} \sum_b \frac{\partial L}{\partial S_{ib}}\,K_{bk}.$$

Relabel $b \to j$:

$$\frac{\partial L}{\partial Q_{ik}} = \frac{1}{\sqrt{d}} \sum_j \frac{\partial L}{\partial S_{ij}}\,K_{jk} = \frac{1}{\sqrt{d}} \sum_j A_{ij}(D_{ij} - r_i)\,K_{jk}.$$

For K: differentiate the same forward equation w.r.t. K_{jk}. Pull the derivative through the sum (Q is constant w.r.t. K), then apply $\partial K_{bc}/\partial K_{jk} = \delta_{bj}\,\delta_{ck}$:

$$\frac{\partial S_{ab}}{\partial K_{jk}} = \frac{1}{\sqrt{d}} \sum_c Q_{ac}\,\delta_{bj}\,\delta_{ck} = \frac{1}{\sqrt{d}}\,\delta_{bj}\,Q_{ak}.$$

Symmetric to Q's case: δ_{ck} killed the c-sum, and δ_{bj} will kill the b-sum upstream. Substitute:

$$\frac{\partial L}{\partial K_{jk}} = \frac{1}{\sqrt{d}} \sum_{a,b} \frac{\partial L}{\partial S_{ab}}\,\delta_{bj}\,Q_{ak} = \frac{1}{\sqrt{d}} \sum_a \frac{\partial L}{\partial S_{aj}}\,Q_{ak}.$$

Relabel $a \to i$:

$$\frac{\partial L}{\partial K_{jk}} = \frac{1}{\sqrt{d}} \sum_i \frac{\partial L}{\partial S_{ij}}\,Q_{ik} = \frac{1}{\sqrt{d}} \sum_i A_{ij}(D_{ij} - r_i)\,Q_{ik}.$$

Summary

Let $E_{ij} := A_{ij}(D_{ij} - r_i)$ with $D_{ij} = \sum_p G_{ip}V_{jp}$ and $r_i = \sum_p G_{ip}O_{ip}$. Then:

$$\frac{\partial L}{\partial V_{jp}} = \sum_i A_{ij}\,G_{ip}, \quad \frac{\partial L}{\partial Q_{ik}} = \frac{1}{\sqrt{d}} \sum_j E_{ij}\,K_{jk}, \quad \frac{\partial L}{\partial K_{jk}} = \frac{1}{\sqrt{d}} \sum_i E_{ij}\,Q_{ik}.$$

Matrix-form translation

For the whiteboard:

$$\frac{\partial L}{\partial V} = A^\top G, \quad \frac{\partial L}{\partial A} = G V^\top, \quad E = A \odot (D - r\mathbf{1}^\top),$$

$$\frac{\partial L}{\partial Q} = \frac{EK}{\sqrt{d}}, \quad \frac{\partial L}{\partial K} = \frac{E^\top Q}{\sqrt{d}}.$$

Where the moving parts come from

- Two outer-product-shape gradients ($\partial L/\partial V$, $\partial L/\partial A$) from $O = AV$ — linear, immediate.
- One row-local softmax Jacobian step, which reduces to multiply-by-A then subtract the per-row dot product r_i — that's the "softmax minus mean" pattern.
- Two more outer-product-shape gradients ($\partial L/\partial Q$, $\partial L/\partial K$) from the bilinear score $S = QK^\top/\sqrt{d}$.

Five contractions total, each $O(T^2 d)$ — the standard quadratic-in-sequence-length cost. (This is exactly what FlashAttention restructures to avoid materialising A and D: it precomputes $r_i = \sum_p G_{ip} O_{ip}$ in a cheap $O(T d_v)$ pass, then recomputes blocks of A on the fly.)

Where the implicit deltas show up

Per step, the rule is the same: differentiate a matrix entry w.r.t. another matrix entry to produce a product of one Kronecker delta per index slot, then watch which sums get collapsed. In each of the five contraction steps above:

Step	Deltas	Each kills
$\partial L/\partial V$	$\delta_{cj}\,\delta_{bp}$	c-sum (position match), b-sum (dim match)
$\partial L/\partial A$	$\delta_{ai}\,\delta_{cj}$	a-sum (query-row match), c-sum (key-pos match)
$\partial A/\partial S$	δ_{ai} + softmax-shape $(\delta_{ck} - A_{ak})$	a-sum (row-independence)
$\partial L/\partial Q$	$\delta_{ai}\,\delta_{ck}$	a-sum (query-row match), c-sum (head-dim match)
$\partial L/\partial K$	$\delta_{bj}\,\delta_{ck}$	b-sum (key-row match), c-sum (head-dim match)

The pattern: shared indices on input and output are killed by the position/row deltas; the surviving sum is over the "other dimension" — i.e. the index that gets contracted to form the output.