Mathemical Analysis of Backpropagation Networks

Linear layer backward pass: derivation

We have a batched linear layer (with bias broadcast across the batch):

- $X \in \mathbb{R}^{B \times I}$
- $W \in \mathbb{R}^{I \times O}$
- $b \in \mathbb{R}^{O}$
- $Y \in \mathbb{R}^{B \times O}$

Forward: $Y = XW + b$

Let the upstream gradient be: $G = \frac{\partial L}{\partial Y} \in \mathbb{R}^{B \times O}$

where I is input dimention, O is output dimenion and B is batch size

---

1) Derive $\frac{\partial L}{\partial W} = X^T G$

Elementwise forward equation: $Y_{b,o} = \sum_{i=1}^{I} X_{b,i} W_{i,o} + b_o$

Partial derivative of an output element w.r.t. a weight: $\frac{\partial Y_{b,o}}{\partial W_{i,o}} = X_{b,i}$

Chain rule: $\frac{\partial L}{\partial W_{i,o}} = \sum_{b=1}^{B} \frac{\partial L}{\partial Y_{b,o}} \frac{\partial Y_{b,o}}{\partial W_{i,o}} = \sum_{b=1}^{B} G_{b,o} X_{b,i}$

Now note that the (i, o) entry of $X^T G$ is: $(X^T G)_{i,o} = \sum_{b=1}^{B} X_{b,i} G_{b,o}$

So: $\boxed{\frac{\partial L}{\partial W} = X^T G}$

---

2) Derive $\frac{\partial L}{\partial X} = G W^T$

Partial derivative of an output element w.r.t. an input: $\frac{\partial Y_{b,o}}{\partial X_{b,i}} = W_{i,o}$

Chain rule: $\frac{\partial L}{\partial X_{b,i}} = \sum_{o=1}^{O} \frac{\partial L}{\partial Y_{b,o}} \frac{\partial Y_{b,o}}{\partial X_{b,i}} = \sum_{o=1}^{O} G_{b,o} W_{i,o}$

Now the (b, i) entry of GW^T is: $(GW^T)_{b,i} = \sum_{o=1}^{O} G_{b,o} (W^T)_{o,i} = \sum_{o=1}^{O} G_{b,o} W_{i,o}$

So: $\boxed{\frac{\partial L}{\partial X} = G W^T}$

In summary training uses $X^T G$ to update weights and GW^T to propagate gradients backward. As a corrolary to the above, we note that the derivatives are related via:

$$\left(\frac{\partial L}{\partial W}\right) W^T = \left(X^T G\right) W^T = X^T \left(G W^T\right) = X^T \left(\frac{\partial L}{\partial X}\right).$$

So: $\boxed{\left(\frac{\partial L}{\partial W}\right) W^T = X^T \left(\frac{\partial L}{\partial X}\right)}$

---

3) Derive $\frac{\partial L}{\partial b} = \sum_{b=1}^{B} G_{b,:}$

Because b_o is added to every row (broadcasting), we have: $\frac{\partial Y_{b,o}}{\partial b_o} = 1$

Chain rule: $\frac{\partial L}{\partial b_o} = \sum_{b=1}^{B} \frac{\partial L}{\partial Y_{b,o}} \frac{\partial Y_{b,o}}{\partial b_o} = \sum_{b=1}^{B} G_{b,o}$

Vector form: $\boxed{\frac{\partial L}{\partial b} = \sum_{b=1}^{B} G_{b,:}}$

At the end of a classifier, we usually apply softmax to turn the model's raw scores (logits) into a probability distribution, and then use cross-entropy to measure how well those predicted probabilities match the true label distribution. The true label is typically represented as a one-hot vector: 1 for the correct class and 0 for every other class. When we combine softmax and cross-entropy, the gradient w.r.t. the logits simplifies to

$dL/dZ = (Probabilities - OneHotLabels)/B$

in short form: $\boxed{dL/dZ = (P - Y)/B}$

This gradient is the standard starting signal we backpropagate through the rest of the network.

In nost MLP's between each linear layer there is a gating function such as ReLU which we will use for simplicity that acts as a 0/1 gate:

$A = \mathrm{ReLU}(Y) = \max(0, Y)$

with a backward formula:

$$\textbf{ReLU backward:} \quad \frac{\partial L}{\partial Y} = \frac{\partial L}{\partial A} \odot (Y > 0)$$

where $\odot (Y > 0)$ indicates only positive elements are passed through rest are set to 0.

Backprop for an MLP with more than 1 hidden layer

Notation (for L linear layers)

Let $A_0 = X$ (batch input).

For layers $\ell = 1, \ldots, L-1$ (hidden layers):

$$Z_\ell = A_{\ell-1} W_\ell + b_\ell, \qquad A_\ell = \max(0, Z_\ell)$$

Final (logits) layer $\ell = L$:

$$Z_L = A_{L-1} W_L + b_L, \qquad P = \text{softmax}(Z_L)$$

Backward pass (general form)

We use the shorthand notation
$dZ_\ell \triangleq \frac{\partial L}{\partial Z_\ell}, \quad dW_\ell \triangleq \frac{\partial L}{\partial W_\ell}, \quad dA_\ell \triangleq \frac{\partial L}{\partial A_\ell}, \quad dZ_L = \frac{\partial L}{\partial Z_L}.$

That is, the prefix d denotes the gradient of the scalar loss L with respect to the indicated variable.

Starting at the output:

$$dZ_L = \frac{P - Y}{B}$$

Then for $\ell = L, L-1, \ldots, 1$ do:

1. Parameter gradients

$$dW_\ell = A_{\ell-1}^T dZ_\ell, \qquad db_\ell = \sum(dZ_\ell,\ \text{axis} = 0)$$

2. Propagate to previous activation:

$$dA_{\ell-1} = dZ_\ell W_\ell^T$$

3. If $\ell > 1$, pass through ReLU gate:

$$dZ_{\ell-1} = dA_{\ell-1} * (Z_{\ell-1} > 0)$$

We just repeat the same linear-backward + ReLU-backward steps for each additional hidden layer.

Closed-form (non-recursive) expression for dW_ℓ

Define the ReLU masks for hidden layers:

$$M_k := \mathbf{1}[Z_k > 0], \qquad k = 1, \ldots, L-1.$$

Let $\mathcal{G}_k(\cdot)$ denote the ReLU "gate" operator (elementwise masking):

$$\mathcal{G}_k(U) := U \odot M_k.$$

Then for any layer $\ell \in \{1, \ldots, L\}$, we can write

$$dW_\ell = A_{\ell-1}^T \, dZ_\ell,$$

---

(2) Clear operator notation

ReLU gate operator (elementwise mask) $\mathcal{G}_k(U) := U \odot \mathbf{1}[Z_k > 0]$.

Define a single "backprop step" operator for layer j (multiply by W_j^T, then apply the ReLU gate for layer j-1) $T_j(U) := \mathcal{G}_{j-1}\left(UW_j^T\right)$.

Then the closed form for dZ_ℓ is: $\boxed{dZ_\ell = (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_L)(\Delta)}$ $(\ell \le L-1)$, and for the last layer, $dZ_L = \Delta$.

(3) Fully explicit nested-parentheses form

$\boxed{dZ_\ell = \mathcal{G}_\ell\Big(\mathcal{G}_{\ell+1}\big(\cdots \mathcal{G}_{L-2}\big(\mathcal{G}_{L-1}(\Delta W_L^T)\, W_{L-1}^T\big) \cdots W_{\ell+2}^T\big)\, W_{\ell+1}^T\Big)}$ $(\ell \le L-1)$, with $dZ_L = \Delta$.

We use the linear-layer gradient identity: $dW_\ell = A_{\ell-1}^T dZ_\ell$

Closed form for dZ_ℓ (for $\ell \le L-1$), and $dZ_L = \Delta$
$dZ_\ell = (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_L)(\Delta), \qquad dZ_L = \Delta.$

(2) Closed form for dW_ℓ $\boxed{dW_\ell = A_{\ell-1}^T \, (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_L)(\Delta)}$ $(\ell \le L-1)$, and for the last layer (empty composition), $\boxed{dW_L = A_{L-1}^T \Delta.}$

(3) Fully explicit nested-parentheses form

First write dZ_\ell explicitly:
$dZ_\ell = \mathcal{G}_\ell\Big(\mathcal{G}_{\ell+1}\big(\cdots \mathcal{G}_{L-2}\big(\mathcal{G}_{L-1}(\Delta W_L^T)\, W_{L-1}^T\big) \cdots W_{\ell+2}^T\big)\, W_{\ell+1}^T\Big), \qquad (\ell \le L-1)$, with $dZ_L = \Delta$.

Then plug into $dW_\ell = A_{\ell-1}^T dZ_\ell$:

$\boxed{dW_\ell = A_{\ell-1}^T \, \mathcal{G}_\ell\Big(\mathcal{G}_{\ell+1}\big(\cdots \mathcal{G}_{L-2}\big(\mathcal{G}_{L-1}(\Delta W_L^T)\, W_{L-1}^T\big) \cdots W_{\ell+2}^T\big)\, W_{\ell+1}^T\Big)}$ $(\ell \le L-1)$,

$$\boxed{dW_\ell = A_{\ell-1}^T \left[\left(\prod_{j=L}^{\ell+1} \left(\mathcal{G}_{j-1}(UW_j^T)\right)\right) \circ (\Delta)\right]}$$

and $\boxed{dW_L = A_{L-1}^T \Delta.}$

Bias gradient identity (chapter) $db_\ell = \sum(dZ_\ell,\ \text{axis} = 0).$

ReLU gate operator $\mathcal{G}_k(U) := U \odot \mathbf{1}[Z_k > 0].$

Backprop step operator: multiply by W_j^T then apply ReLU gate for layer (j-1)
$T_j(U) := \mathcal{G}_{j-1}\left(UW_j^T\right)$. uses $dX = dYW^T$ and ReLU gating

Closed form for dZ_ℓ $dZ_\ell = (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_L)(\Delta), \qquad dZ_L = \Delta.$

Closed form for db_ℓ $\boxed{db_\ell = \sum((T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_L)(\Delta),\ \text{axis} = 0)} \qquad (\ell \le L-1),$
and for the last layer, $\boxed{db_L = \sum(\Delta,\ \text{axis} = 0).}$

---

(3) Fully explicit nested form for db_ℓ

$$dZ_\ell = \mathcal{G}_\ell\Big(\mathcal{G}_{\ell+1}\big(\cdots \mathcal{G}_{L-2}\big(\mathcal{G}_{L-1}(\Delta W_L^T)\, W_{L-1}^T\big) \cdots W_{\ell+2}^T\big)\, W_{\ell+1}^T\Big), \qquad (\ell \le L-1),$$

with $dZ_L = \Delta.$

$$\boxed{db_\ell = \sum\Big(\mathcal{G}_\ell\Big(\mathcal{G}_{\ell+1}\big(\cdots \mathcal{G}_{L-2}\big(\mathcal{G}_{L-1}(\Delta W_L^T)\, W_{L-1}^T\big) \cdots W_{\ell+2}^T\big)\, W_{\ell+1}^T\Big),\ \text{axis} = 0\Big)} \qquad (\ell \le$$

and $\boxed{db_L = \sum(\Delta,\ \text{axis} = 0).}$

Residual Connections

Residual (skip) connection: $Y = X + F(X;\theta)$

Forward:

$y = X + F(X;\theta)$

Backward:

Let the upstream gradient be $G = \frac{\partial L}{\partial Y}.$

So the gradient passed back to each input of the add is:
$\frac{\partial L}{\partial X}\Big|_{\text{skip}} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X} = G \cdot 1 = G, \qquad \frac{\partial L}{\partial F} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial F} = G \cdot 1 = G$

The add node splits gradients: $\frac{\partial L}{\partial X}\Big|_{\text{skip}} = G, \qquad \frac{\partial L}{\partial F} = G.$

Backprop through the transform branch F: $\frac{\partial L}{\partial X}\Big|_{\text{through } F} = \frac{\partial L}{\partial F}\,\frac{\partial F(X;\theta)}{\partial X} = G\,\frac{\partial F(X;\theta)}{\partial X}.$

Combine contributions at X:
$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial X}\Big|_{\text{skip}} + \frac{\partial L}{\partial X}\Big|_{\text{through } F} = G + G\,\frac{\partial F(X;\theta)}{\partial X} = G\cdot\left(1 + \frac{\partial F(X;\theta)}{\partial X}\right).$

Parameter gradients for the transform branch: $\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial F}\,\frac{\partial F(X;\theta)}{\partial \theta} = G\,\frac{\partial F(X;\theta)}{\partial \theta}.$

Residual (identity skip) block: operator-composition notation

Residual block forward: $Y = X + F(X).$

Upstream gradient: $dY := \frac{\partial L}{\partial Y}.$

Split rule for the "+" node Since $Y = X + F$, the gradient splits and is copied to both inputs: $dX_{\text{skip}} = dY, \qquad dF = dY.$

Define operators for the residual branch F

$\mathcal{G}_k(U) := U \odot \mathbf{1}[Z_k > 0].$

Linear-backward operator for a weight matrix W: (this is the map $U \mapsto UW^T$, matching $dX = dYW^T$) $\mathcal{L}_W(U) := UW^T.$

One "Linear-backward + ReLU-gate" step operator: (multiply by W^T, then apply the ReLU gate for the previous layer) $T_j(U) := \mathcal{G}_{j-1}\big(\mathcal{L}_{W_j}(U)\big) = \mathcal{G}_{j-1}\left(UW_j^T\right).$

Closed form for the gradient through the residual branch If F is an (L_F)-layer MLP-like stack ending at layer L_F, and its output pre-activation gradient is $dZ_{L_F} = dF$, then for any internal layer ℓ in F: $dZ_\ell^{(F)} = (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_{L_F})(dF), \qquad dZ_{L_F}^{(F)} = dF.$

The gradient w.r.t. the input of F (i.e., X) is the linear-backward of the first layer of F applied to $dZ_1^{(F)}$: $dX_F = \mathcal{L}_{W_1}\left(dZ_1^{(F)}\right) = dZ_1^{(F)} W_1^T.$

Combine skip + residual paths $\boxed{dX = dX_{\text{skip}} + dX_F = dY + dX_F}$

If you want everything in one line (still operator form):
$\boxed{dX = dY + \mathcal{L}_{W_1}\big((T_2 \circ T_3 \circ \cdots \circ T_{L_F})(dY)\big)}$ (Here we used dF = dY for the identity skip.)

A (discrete) convolution layer is a $\emph{locallinearoperator}$ in the input: each output value is computed as a dot product between a small local input patch and a shared kernel, applied at every spatial position. Convolution is a local linear operator (a sliding dot product) and therefore admits the same backpropagation structure as a linear layer, although the corresponding linear map has a sparse, weight-sharing structure rather than a dense matrix.

For a 3-tier Residual Network stack

Linear-backward operator for a weight matrix W: (this is the map $U \mapsto UW^T$, matching $dX = dYW^T$) $\mathcal{L}_W(U) := UW^T$.

ReLU-gate operator for layer i (gates using the pre-activation Z_i): $\mathcal{G}_i(U) := U \odot \mathbf{1}[Z_i > 0]$.

One "Linear-backward + ReLU-gate" step operator: (multiply by W_j^T, then apply the ReLU gate for the previous layer) $T_j(U) := \mathcal{G}_{j-1}(\mathcal{L}_{W_j}(U)) = \mathcal{G}_{j-1}\left(UW_j^T\right)$.

Bottleneck residual MLP block (3 linears in the residual branch):

Forward: $Y = X + F(X)$.

Residual branch: $Z_1 = XW_1 + b_1, \quad A_1 = \mathrm{ReLU}(Z_1),$
$Z_2 = A_1W_2 + b_2, \quad A_2 = \mathrm{ReLU}(Z_2), Z_3 = A_2W_3 + b_3, \quad F(X) = Z_3.$

Upstream gradient: $dY := \frac{\partial L}{\partial Y}$.

Split at the residual add: $dF = dY, \qquad dX_{\text{skip}} = dY.$

Start backprop inside F at the output (no ReLU after last linear): $dZ_3 = dF = dY$.

Backprop through the residual branch using the operators: $dZ_2 = T_3(dZ_3) = \mathcal{G}_2\left(dY\, W_3^T\right)$.

$dZ_1 = T_2(dZ_2) = \mathcal{G}_1\left(dZ_2\, W_2^T\right) = \mathcal{G}_1\left(\mathcal{G}_2\left(dY\, W_3^T\right) W_2^T\right)$.

Gradient to X through the residual branch (first linear backward, no gate on X):
$dX_F = \mathcal{L}_{W_1}(dZ_1) = dZ_1W_1^T$.

Combine skip + residual: $\boxed{dX = dX_{\text{skip}} + dX_F = dY + \mathcal{L}_{W_1}\Big(T_2\big(T_3(dY)\big)\Big).}$

Same thing as one fully nested expression: $\boxed{dX = dY + \left[\mathcal{G}_1\left(\mathcal{G}_2\left(dY\, W_3^T\right) W_2^T\right)\right] W_1^T.}$

Bottleneck residual MLP block (3 linears in the residual branch)

Forward (residual add): $Y = X + F(X)$.

Residual branch (ReLU after first two linears, none after last):
$Z_1 = XW_1 + b_1, \quad A_1 = \mathrm{ReLU}(Z_1), Z_2 = A_1W_2 + b_2, \quad A_2 = \mathrm{ReLU}(Z_2),$
$Z_3 = A_2W_3 + b_3, \quad F(X) = Z_3.$

Upstream gradient: $dY := \frac{\partial L}{\partial Y}$.

Split at the residual add: $dZ_3 = dF = dY, \qquad dX_{\text{skip}} = dY.$

We use the linear-layer gradient identity: $dW_\ell = A_{\ell-1}^T dZ_\ell$.

Define operators:

Linear-backward operator for a weight matrix W: $\mathcal{L}_W(U) := UW^T$.

ReLU-gate operator for layer i: $\mathcal{G}_i(U) := U \odot \mathbf{1}[Z_i > 0]$.

One "Linear-backward + ReLU-gate" step operator:
$T_j(U) := \mathcal{G}_{j-1}\big(\mathcal{L}_{W_j}(U)\big) = \mathcal{G}_{j-1}\Big(UW_j^T\Big)$.

For this bottleneck block, the only gated steps are T_3 (gates Z_2) and T_2 (gates Z_1):
$T_3(U) = \mathcal{G}_2(UW_3^T), \qquad T_2(U) = \mathcal{G}_1(UW_2^T)$.

Let $\Delta := dZ_3 = dY$.

(1) Closed form for dZ_ℓ

$dZ_3 = \Delta$.

For $\ell \in \{1, 2\}$: $dZ_\ell = (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_3)(\Delta)$.

Concretely: $dZ_2 = T_3(\Delta), \qquad dZ_1 = (T_2 \circ T_3)(\Delta)$.

(2) Closed form for dW_ℓ

For $\ell \in \{1, 2\}$: $\boxed{dW_\ell = A_{\ell-1}^T \, (T_{\ell+1} \circ T_{\ell+2} \circ \cdots \circ T_3)(\Delta)}$.

For the last layer (empty composition): $\boxed{dW_3 = A_2^T \Delta}$.

(3) Fully explicit nested-parentheses form

First write dZ_ℓ explicitly:

$dZ_3 = \Delta$.

$dZ_2 = \mathcal{G}_2(\Delta W_3^T)$.

$dZ_1 = \mathcal{G}_1\Big(\mathcal{G}_2(\Delta W_3^T)\, W_2^T\Big)$.

Then plug into $dW_\ell = A_{\ell-1}^T dZ_\ell$:

$\boxed{dW_1 = X^T \, \mathcal{G}_1\Big(\mathcal{G}_2(\Delta W_3^T)\, W_2^T\Big)}$.

$\boxed{dW_2 = A_1^T \, \mathcal{G}_2(\Delta W_3^T)}$.

$\boxed{dW_3 = A_2^T \Delta}$.

(4) (Optional) Closed form for the block input gradient dX

Residual-path contribution: $dX_F = dZ_1 W_1^T = \mathcal{L}_{W_1}(dZ_1)$.

Combine skip + residual:

$$\boxed{dX = dX_{\text{skip}} + dX_F = dY + dZ_1 W_1^T = dY + \mathcal{L}_{W_1}\big((T_2 \circ T_3)(\Delta)\big).}$$

Multi-head / multi-branch linear projections from the same input

Let $X \in \mathbb{R}^{B \times d_{\text{in}}}$ be a shared input to H heads.
For each head $h \in \{1, \dots, H\}$, define a linear map
$Z_h = XW_h + b_h, \qquad W_h \in \mathbb{R}^{d_{\text{in}} \times d_h}, \quad b_h \in \mathbb{R}^{d_h}.$

Given upstream gradients $\frac{\partial L}{\partial Z_h}$ for each head, the parameter gradients are

$\frac{\partial L}{\partial W_h} = X^T \frac{\partial L}{\partial Z_h}, \qquad \frac{\partial L}{\partial b_h} = \sum_{i=1}^{B} \frac{\partial L}{\partial Z_h^{(i)}}.$

Each head contributes a gradient to the shared input:

$\left(\frac{\partial L}{\partial X}\right)_h = \frac{\partial L}{\partial Z_h} W_h^T.$

Since X feeds all heads, the total gradient at the branch point is the sum of contributions:

$\frac{\partial L}{\partial X} = \sum_{h=1}^{H} \frac{\partial L}{\partial Z_h} W_h^T.$

If your heads are then concatenated and passed to another linear layer, you'd additionally route the gradient back through the concat (i.e., split dL into per-head slices), and then apply the same formulas above per slice.

Max-pooling backprop (single pooling window)

Forward: $y = \max_{i \in \{1,\dots,m\}} x_i$

Backward: given upstream gradient $g = \frac{\partial L}{\partial y}$

$$\frac{\partial L}{\partial x_i} = \begin{cases} g, & i \in \arg\max\limits_j x_j \\ 0, & \text{otherwise.} \end{cases}$$

2D max-pooling (pool_dim = p), output index (r,c)

Forward: $y_{r,c} = \max\limits_{\substack{0 \le pr < p \\ 0 \le pc < p}} x_{rp+pr,\, cp+pc}$

Let (pr^*, pc^*) be the argmax offsets from the forward pass:
$(pr^*, pc^*) \in \arg\max_{\substack{0 \le pr < p \\ 0 \le pc < p}} x_{rp+pr,\ cp+pc}$

Backward: given upstream gradient $\delta_{r,c} = \frac{\partial L}{\partial y_{r,c}}$
$$\frac{\partial L}{\partial x_{rp+pr,\ cp+pc}} = \begin{cases} \delta_{r,c}, & (pr, pc) = (pr^*, pc^*) \\ 0, & \text{otherwise.} \end{cases}$$

Downsampling by slicing (a linear "pooling" alternative)

Unlike max pooling (which outputs the maximum value in each window and is nonlinear), a fixed slicing/downsampling operator is linear.

Let X be the input (vectorized or flattened tensor) and let S be a fixed selection (downsampling) operator that picks one predetermined element from each pooling window (e.g., the top-left element). Define $Y = SX.$

Since S is fixed, this operation is linear in X and has no learnable parameters. Given an upstream gradient $\frac{\partial L}{\partial Y}$, the backward pass is

$$\frac{\partial L}{\partial X} = S^T \frac{\partial L}{\partial Y}.$$

There is no parameter gradient term (no W or b) because S is not learned.

Single-head scaled dot-product self-attention: forward/backward

Shapes (one common convention)

- $X \in R^{B \times T \times d_model}$
- $W_Q, W_K, W_V \in R^{d_model \times d_k}$
- $Q, K, V \in R^{B \times T \times d_k}$
- $S, A \in R^{B \times T \times T}$
- $Y \in R^{B \times T \times d_k}$

Forward $Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V$

$S = \frac{1}{\sqrt{d_k}} QK^\top \qquad (\text{transpose over the } T \text{ dimension, per batch})$

$$\boxed{S = \frac{1}{\sqrt{d_k}}, X, (W_Q W_K \top), X \top}$$

$A = \mathrm{softmax}(S) \qquad (\text{row-wise over keys})$

$Y = AV$

Backward Upstream gradient: $G = \frac{\partial L}{\partial Y} \in \mathbb{R}^{B \times T \times d_k}$

(A) Backprop through Y = A V

$\frac{\partial L}{\partial V} = A^\top G$

$\frac{\partial L}{\partial A} = G V^\top$

(B) Backprop through $A = softmax(S)$, row-wise For each row i (each query position), let $a_i = A_{i,:}$ and $g_i = \left(\frac{\partial L}{\partial A}\right)_{i,:}$. $\left(\frac{\partial L}{\partial S}\right)_{i,:} = a_i \odot \left(g_i - (g_i^\top a_i)\mathbf{1}\right)$

(C) Backprop through $S = (1/sqrt(d_k))QK^T$ Let $H = dL/dS$. $H = \frac{\partial L}{\partial S}$

$\frac{\partial L}{\partial Q} = \frac{1}{\sqrt{d_k}} HK$

$\frac{\partial L}{\partial K} = \frac{1}{\sqrt{d_k}} H^\top Q$

(D) Backprop through linear projections $Q = XW_Q$, etc.
$\frac{\partial L}{\partial W_Q} = X^\top \frac{\partial L}{\partial Q}, \qquad \frac{\partial L}{\partial W_K} = X^\top \frac{\partial L}{\partial K}, \qquad \frac{\partial L}{\partial W_V} = X^\top \frac{\partial L}{\partial V}$

$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Q} W_Q^\top + \frac{\partial L}{\partial K} W_K^\top + \frac{\partial L}{\partial V} W_V^\top$

Residual connection around attention (common Transformer pattern) $Z = X + \mathrm{Attn}(X)$

Let $\frac{\partial L}{\partial Z} = G_Z$.

Then $\frac{\partial L}{\partial X}\Big|_{\text{skip}} = G_Z, \qquad \frac{\partial L}{\partial X}\Big|_{\text{through attn}} = \left(\frac{\partial L}{\partial X}\right)_{\text{Attn}}$

and $\frac{\partial L}{\partial X} = G_Z + \left(\frac{\partial L}{\partial X}\right)_{\text{Attn}}$.

Scaled dot-product attention scores $S = \frac{1}{\sqrt{d_k}} QK^\top$ Parameterization via an input matrix X and projection matrices $Q = XW_Q, \qquad K = XW_K$

Substitute Q and K into S $S = \frac{1}{\sqrt{d_k}} (XW_Q)(XW_K)^\top = \frac{1}{\sqrt{d_k}} XW_Q W_K^\top X^\top$

Define A and B to expose the bilinear / Gram-like structure
$A = XW_Q, \qquad B = XW_K \qquad \Longrightarrow \qquad S = \frac{1}{\sqrt{d_k}} AB^\top$

Special case: self-attention with tied projections (W_Q = W_K)

$W_Q = W_K \implies Q = K \qquad \Longrightarrow \qquad S = \frac{1}{\sqrt{d_k}} QQ^\top$

Spectral connection in the symmetric positive semidefinite case
$\text{If } Q = U\Sigma V^\top \text{ (SVD), then } QQ^\top = U\Sigma^2 U^\top$

Therefore, eigenpairs of S in the tied case Eigenvalues relate to singular values of Q

$$S = \frac{1}{\sqrt{d_k}} U\Sigma^2 U^\top \quad \Longrightarrow \quad \lambda_i(S) = \frac{\sigma_i(Q)^2}{\sqrt{d_k}}, \quad \text{eigvecs}(S) = U \qquad U^\top U = I,$$

$$QQ^\top = \sum_{i=1}^{r} \sigma_i(Q)^2 \, u_i u_i^\top, \quad r = \text{rank}(Q) \qquad \text{where } u_i \text{ is the } i\text{-th column of } U.$$

General case: $Q \neq K$ (S is not symmetric in general)
$S = \frac{1}{\sqrt{d_k}} QK^\top \quad (\text{not necessarily symmetric})$

Causal (decoder-only) mask: allow attending only to current/past tokens
$M \in \{0,1\}^{n \times n}, \qquad M_{ij} = \begin{cases} 1, & j \leq i, \\ 0, & j > i. \end{cases}$

Masked logits $\tilde{S}_{ij} = \begin{cases} S_{ij}, & M_{ij} = 1, \\ -\infty, & M_{ij} = 0. \end{cases}$

With a mask, you effectively do an elementwise modification:
$\tilde{S} = S + B, \quad \text{where } B_{ij} = \begin{cases} 0, & M_{ij} = 1 \\ -\infty, & M_{ij} = 0 \end{cases}$

Row-wise softmax to obtain attention weights $A_{i:} = \text{softmax}\left(\tilde{S}_{i:}\right)$.

Top-1 (Switch) Mixture-of-Experts (MoE): forward/backward (simplest form for magnitude analysis)

Setup / shapes

Let $X \in \mathbb{R}^{B \times d_{\text{in}}}$ be a batch of token vectors. There are E experts $f_e(\cdot;\theta_e)$, each mapping $f_e : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$.

Router weight matrix (learned parameters) $W_r \in \mathbb{R}^{d_{\text{in}} \times E}$

A router produces logits $R \in \mathbb{R}^{B \times E}$ and selects one expert per token: $e(b) \in \{1, \ldots, E\}$ for token b.

Forward $R = XW_r \in \mathbb{R}^{B \times E}$

$e(b) = \arg\max_{e \in \{1,\ldots,E\}} R_{b,e}$

$Y_b = f_{e(b)}(X_b; \theta_{e(b)}), \qquad Y \in \mathbb{R}^{B \times d_{\text{out}}}$

Backward Upstream gradient: $G = \frac{\partial L}{\partial Y} \in \mathbb{R}^{B \times d_{\text{out}}}$

(A) Gradients into expert outputs (only the selected expert receives gradient)
$\frac{\partial L}{\partial f_e(X_b)} = \begin{cases} G_b, & e = e(b) \\ 0, & e \neq e(b) \end{cases}$

(B) Expert parameter gradients $\frac{\partial L}{\partial \theta_e} = \sum_{b:\, e(b)=e} \left(\frac{\partial L}{\partial f_e(X_b)} \right) \frac{\partial f_e(X_b;\theta_e)}{\partial \theta_e}$

(C) Input gradients through experts $\frac{\partial L}{\partial X_b} = \left(\frac{\partial L}{\partial f_{e(b)}(X_b)} \right) \frac{\partial f_{e(b)}(X_b;\theta_{e(b)})}{\partial X_b} = G_b \frac{\partial f_{e(b)}(X_b;\theta_{e(b)})}{\partial X_b}$

Note on the router The hard top-1 selection $e(b) = argmax(\cdot)$ is non-differentiable. For magnitude analysis, it is common to treat $e(b)$ as fixed for the pass.

If you ignore router gradients, then dL/dW_r is not defined under pure argmax. If you use a differentiable relaxation (not shown), you would add a router term to dL/dX and compute dL/dW_r similarly to a linear layer.

(General chain-rule context: gradients propagate backward through the graph.)

Forward (batch size B):
$$
\begin{aligned}
Z_1 &= XW_1 + b_1 \\
A_1 &= \mathrm{ReLU}(Z_1) = \max(0, Z_1) \\
Z_2 &= A_1 W_2 + b_2 \\
P &= \mathrm{softmax}(Z_2)
\end{aligned}
$$

Softmax + cross-entropy gradient:
$$
\frac{\partial L}{\partial Z_2} = \frac{P - Y}{B}
$$

Linear layer rule ($Y = XW + b$):
$$
\begin{aligned}
\frac{\partial L}{\partial W} &= X^T \frac{\partial L}{\partial Y} \\
\frac{\partial L}{\partial b} &= \sum \left(\frac{\partial L}{\partial Y} \right) \ \text{(sum over batch axis)} \\
\frac{\partial L}{\partial X} &= \frac{\partial L}{\partial Y} W^T
\end{aligned}
$$

Apply linear rule to layer 2 ($Z_2 = A_1 W_2 + b_2$):
$$
\begin{aligned}
\frac{\partial L}{\partial W_2} &= A_1^T \frac{\partial L}{\partial Z_2} \\
\frac{\partial L}{\partial b_2} &= \sum \left(\frac{\partial L}{\partial Z_2} \right) \\
\frac{\partial L}{\partial A_1} &= \frac{\partial L}{\partial Z_2} W_2^T
\end{aligned}
$$

ReLU backward:
$$
\frac{\partial L}{\partial Z_1} = \frac{\partial L}{\partial A_1} \odot (Z_1 > 0)
$$

Apply linear rule to layer 1 ($Z_1 = XW_1 + b_1$):
$$
\begin{aligned}
\frac{\partial L}{\partial W_1} &= X^T \frac{\partial L}{\partial Z_1} \\
\frac{\partial L}{\partial b_1} &= \sum \left(\frac{\partial L}{\partial Z_1} \right) \\
\frac{\partial L}{\partial X} &= \frac{\partial L}{\partial Z_1} W_1^T
\end{aligned}
$$

Substitute into $\frac{\partial L}{\partial W_2} = A_1^T \frac{\partial L}{\partial Z_2}$: $\boxed{\frac{\partial L}{\partial W_2} = A_1^T \left(\frac{P - Y}{B} \right)}$

$$\boxed{\frac{\partial L}{\partial W_2} = A_1^T \frac{\partial L}{\partial Z_2} = A_1^T \left(\frac{P - Y}{B} \right)}$$

$$\boxed{\frac{\partial L}{\partial W_2} = A_1^T \left(\frac{P - Y}{B} \right) = (\mathrm{ReLU}(Z_1))^T \left(\frac{P - Y}{B} \right) = (\max(0, XW_1 + b_1))^T \left(\frac{P - Y}{B} \right)}$$

$$\boxed{\frac{\partial L}{\partial W_1} = X^T \left(\left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot \mathrm{ReLU}'(Z_1) \right) = X^T \left(\left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot (Z_1 > 0) \right)}$$

$$\boxed{\frac{\partial L}{\partial b_2} = \sum_{i=1}^{B} \left(\frac{\partial L}{\partial Z_2} \right)_{i,:} = \sum_{i=1}^{B} \left(\frac{P - Y}{B} \right)_{i,:}}$$

$$\boxed{\frac{\partial L}{\partial b_1} = \sum_{i=1}^{B} \left(\frac{\partial L}{\partial Z_1} \right)_{i,:} = \sum_{i=1}^{B} \left(\left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot (XW_1 + b_1 > 0) \right)_{i,:}}$$

Core identities (batch): $\nabla_{W_2} L = A_1^T G_2, \qquad G_2 := \frac{\partial L}{\partial Z_2} = \frac{P - Y}{B}$

$$G_1 := \frac{\partial L}{\partial Z_1} = (G_2 W_2^T) \odot \mathrm{ReLU}'(Z_1), \qquad \nabla_{W_1} L = X^T G_1$$

$$\boxed{\|\nabla_{W_2} L\|_F = \|A_1^T G_2\|_F \leq \|A_1\|_F \, \|G_2\|_F}$$

ReLU gate doesn't increase the Frobenius norm Since R e L U ' (Z 1) ∈ { 0 , 1 } elementwise,

$$\|G_1\|_F = \|(G_2 W_2^T) \odot \mathrm{ReLU}'(Z_1)\|_F \leq \|G_2 W_2^T\|_F$$

Propagating through W 2 Use ‖ A B ‖ F ≤ ‖ A ‖ F ‖ B ‖ 2 :

$$\|G_2 W_2^T\|_F \leq \|G_2\|_F \, \|W_2^T\|_2 = \|G_2\|_F \, \|W_2\|_2$$

Why $\|AB\|_F \leq \|A\|_F \|B\|_2$ (Hölder / Cauchy–Schwarz)
The mixed Frobenius–spectral bound above is the case $p = q = 2$ of Hölder's inequality. View the (i, j) entry of AB as an inner product: $(AB)_{i,j} = \langle \mathrm{row}_i(A), \mathrm{col}_j(B) \rangle$. By Cauchy–Schwarz, $|(AB)_{i,j}| \leq \|\mathrm{row}_i(A)\|_2 \|\mathrm{col}_j(B)\|_2$. Summing over i, j and using $\|B\|_2 = \max_{\|v\|_2 = 1} \|Bv\|_2$ gives $\|AB\|_F \leq \|A\|_F \|B\|_2$. So the bound we used is exactly Hölder (or Cauchy–Schwarz) in disguise.

ReLU gate: $\|(G_2 W_2^T) \odot (Z_1 > 0)\|_F \le \|G_2 W_2^T\|_F$ also follows from Hölder: for Hadamard products, $\|A \odot B\|_F \le \|A\|_F \|B\|_\infty$ (elementwise max), and the mask has $\|(Z_1 > 0)\|_\infty = 1$.

Bias gradient (optional): $\nabla_{b_2} L = \sum_{i=1}^{B} (G_2)_{i,:} = \mathbf{1}_B^T G_2$. By the same idea (linear map $\mathbf{1}_B^T$ applied to G_2), $\|\nabla_{b_2} L\|_2 \le \|\mathbf{1}_B\|_2 \|G_2\|_F = \sqrt{B}\,\|G_2\|_F$.

First layer Again ‖ X T G 1 ‖ F ≤ ‖ X ‖ F ‖ G 1 ‖ F , so chaining the above:

$$\boxed{\|\nabla_{W_1} L\|_F = \|X^T G_1\|_F \le \|X\|_F \, \|G_1\|_F \le \|X\|_F \, \|G_2\|_F \, \|W_2\|_2}$$

Convention: $\|\cdot\|$ denotes a matrix norm. We assume it is submultiplicative: $\|AB\| \le$

$$G_2 := \frac{\partial L}{\partial Z_2} = \frac{P - Y}{B}, \qquad \nabla_{W_2} L = A_1^T G_2, \qquad \nabla_{b_2} L = \sum_{i=1}^{B} (G_2)_{i,:}$$

$$G_1 := \frac{\partial L}{\partial Z_1} = (G_2 W_2^T) \odot (Z_1 > 0), \qquad \nabla_{W_1} L = X^T G_1, \qquad \nabla_{b_1} L = \sum_{i=1}^{B} (G_1)_{i,:}$$

$$\|\nabla_{W_2} L\| = \|A_1^T G_2\| \le \|A_1^T\| \, \|G_2\|$$

$$\|G_1\| = \|(G_2 W_2^T) \odot (Z_1 > 0)\| \le \|G_2 W_2^T\| \le \|G_2\| \, \|W_2^T\|$$

$$\|\nabla_{W_1} L\| = \|X^T G_1\| \le \|X^T\| \, \|G_1\| \le \|X^T\| \, \|G_2\| \, \|W_2^T\|$$

Forward: $Z_1 = XW_1 + b_1, \qquad A_1 = \mathrm{ReLU}(Z_1) = \max(0, Z_1), \qquad Z_2 = A_1 W_2 + b_2,$

$$\textbf{Substitute } G_2: \quad G_2 := \frac{\partial L}{\partial Z_2} = \frac{P - Y}{B}.$$

$$\textbf{Substitute } G_1: \quad G_1 := \frac{\partial L}{\partial Z_1} = (G_2 W_2^T) \odot (Z_1 > 0) = \left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot (XW_1 +$$

$$\nabla_{W_2} L = A_1^T G_2 = (\max(0, XW_1 + b_1))^T \left(\frac{P - Y}{B} \right).$$

$$\nabla_{W_1} L = X^T G_1 = X^T \left(\left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot (XW_1 + b_1 > 0) \right).$$

$$\textbf{Bias gradients (linear-layer rule):} \quad \nabla_{b_2} L = \sum_{i=1}^{B} (G_2)_{i,:} = \sum_{i=1}^{B} \left(\frac{P - Y}{B} \right)_{i,:}, \qquad \nabla_{b_1} L$$

Norm bounds (no subscripts): $\|\nabla_{W_2} L\| = \|A_1^T G_2\| \le \|A_1^T\| \, \|G_2\|, \qquad \|\nabla_{W_1} L\| = \|$

$$G_1 := \frac{\partial L}{\partial Z_1} = (G_2 W_2^T) \odot (Z_1 > 0)$$

Norm bound for W_2 : $\|\nabla_{W_2} L\| = \|A_1^T G_2\| \le \|A_1^T\| \, \|G_2\| = \|A_1^T\| \left\| \frac{P - Y}{B} \right\|$

Norm bound for W_1 : $\|\nabla_{W_1} L\| = \|X^T G_1\| \le \|X^T\| \, \|G_1\| = \|X^T\| \left\| \left(\left(\frac{P - Y}{B} \right) W_2^T \right.$

you can drop the ReLU mask using the fact that it's a 0/1 "gate" (it can only shrink gradients elementwise)

Since:

$$\left\| \left(\left(\frac{P - Y}{B} \right) W_2^T \right) \odot (Z_1 > 0) \right\| \le \left\| \left(\frac{P - Y}{B} \right) W_2^T \right\|$$

Becomes:

$$\|\nabla_{W_1} L\| \le \|X^T\| \left\| \left(\frac{P - Y}{B} \right) W_2^T \right\|$$

$$\boxed{\|\nabla_{W_1} L\| \le \|X^T\| \left\| \frac{P - Y}{B} \right\| \|W_2^T\|}$$

Assume $\|\cdot\|$ **is a submultiplicative matrix norm:** $\|AB\| \le \|A\| \, \|B\|$.

$$\textbf{Recall:} \quad G_2 = \frac{P - Y}{B}$$

Norm bound for W_2 : $\|\nabla_{W_2} L\| = \|A_1^T G_2\| \le \|A_1^T\| \, \|G_2\| = \|A_1^T\| \left\| \frac{P - Y}{B} \right\|$

Substitute $A_1 = \mathrm{ReLU}(Z_1) = \max(0, Z_1)$ with $Z_1 = XW_1 + b_1$: $\boxed{\|\nabla_{W_2} L\| \le \|\max($}$

since

$$\|\nabla_{W_2} L\| = \|A_1^T \Delta\| \le \|A_1^T\| \, \|\Delta\| \le \|Z_1^T\| \, \|\Delta\|$$

Let $Z_1 = XW_1 + b_1$, $A_1 = \max(0, Z_1)$, and $\Delta = \frac{P-Y}{B}$. (From backprop: $\nabla_{W_2} L = A_1^T \Delta$.)
【】

Assume $\|\cdot\|$ is a matrix norm that is (i) submultiplicative: $\|AB\| \le \|A\|\,\|B\|$, and (ii) satisfies the triangle inequality: $\|U+V\| \le \|U\| + \|V\|$.

$$\|\nabla_{W_2} L\| = \|A_1^T \Delta\| \le \|A_1^T\|\,\|\Delta\| \le \|Z_1^T\|\,\|\Delta\|$$

$$\|Z_1^T\| = \|(XW_1 + b_1)^T\| = \|(XW_1)^T + b_1^T\| \le \|(XW_1)^T\| + \|b_1^T\|$$

$$\|(XW_1)^T\| = \|W_1^T X^T\| \le \|W_1^T\|\,\|X^T\|$$

$$\boxed{\|\nabla_{W_2} L\| \le \left(\|W_1^T\|\,\|X^T\| + \|b_1^T\|\right) \left\|\frac{P-Y}{B}\right\|}$$

Norm bounds for a general L-layer MLP (ReLU between linears)

We use the same backprop identities as in the chapter:

- Linear layer gradients: $dW = X^T dY$, $dX = dY W^T$, $db = \sum(dY, \text{axis} = 0)$
- ReLU backward gate: $dZ = dA * (Z > 0)$
- Softmax + cross-entropy start gradient: $dZ_L = (P - Y)/B$

Assumptions on the norm

Let $\|\cdot\|$ be a matrix norm that is

- submultiplicative: $\|AB\| \le \|A\|\,\|B\|$
- and satisfies the triangle inequality: $\|U+V\| \le \|U\| + \|V\|$.

Forward definitions

Let $A_0 = X$.

For $\ell = 1, \ldots, L-1$:

$$Z_\ell = A_{\ell-1} W_\ell + b_\ell, \qquad A_\ell = \max(0, Z_\ell)$$

Final logits:

$$Z_L = A_{L-1} W_L + b_L, \qquad P = \text{softmax}(Z_L)$$

Backward definitions

Start:

$$dZ_L = \Delta = \frac{P - Y}{B}$$

For $\ell = L, \ldots, 1$:

$$dW_\ell = A_{\ell-1}^T dZ_\ell, \qquad dA_{\ell-1} = dZ_\ell W_\ell^T, \qquad db_\ell = \sum(dZ_\ell,\ \text{axis} = 0)$$

and for $\ell > 1$:

$$dZ_{\ell-1} = dA_{\ell-1} * (Z_{\ell-1} > 0)$$

---

1) Bound each weight-gradient norm $||dW_\ell||$

From $dW_\ell = A_{\ell-1}^T dZ_\ell$ (same pattern as `grad_W2 = A1.T @ grad_Z2`) :

$$||dW_\ell|| = ||A_{\ell-1}^T dZ_\ell|| \le ||A_{\ell-1}^T||\ ||dZ_\ell||$$

Also, since $A_{\ell-1} = \max(0, Z_{\ell-1})$ elementwise, you can use the monotone bound

$$||A_{\ell-1}|| \le ||Z_{\ell-1}|| \quad \Rightarrow \quad ||A_{\ell-1}^T|| \le ||Z_{\ell-1}^T||$$

So:

$$\boxed{||dW_\ell|| \le ||Z_{\ell-1}^T||\ ||dZ_\ell||}$$

If you want to expand the affine term with triangle inequality:

$$||Z_{\ell-1}^T|| = ||(A_{\ell-2}W_{\ell-1} + b_{\ell-1})^T|| \le ||(A_{\ell-2}W_{\ell-1})^T|| + ||b_{\ell-1}^T||$$

and then

$$||(A_{\ell-2}W_{\ell-1})^T|| = ||W_{\ell-1}^T A_{\ell-2}^T|| \le ||W_{\ell-1}^T||\ ||A_{\ell-2}^T||$$

$$||Z_{\ell-1}^T|| \le ||W_{\ell-1}^T A_{\ell-2}^T|| + ||b_{\ell-1}^T||$$

$$||Z_{\ell-1}^T|| \le ||W_{\ell-1}^T||||A_{\ell-2}^T|| + ||b_{\ell-1}^T||$$

---

2) Bound the backpropagated signal $||dZ_\ell||$ recursively

From $dA_{\ell-1} = dZ_\ell W_\ell^T$ (linear backward) :

$$||dA_{\ell-1}|| \le ||dZ_\ell||\ ||W_\ell^T||$$

Then ReLU gating $dZ_{\ell-1} = dA_{\ell-1} * (Z_{\ell-1} > 0)$ implies it can only shrink (elementwise mask of 0/1), so we use the safe bound:

$$||dZ_{\ell-1}|| \leq ||dA_{\ell-1}||$$

Combine:

$$\boxed{||dZ_{\ell-1}|| \leq ||dZ_{\ell}||\,||W_{\ell}^T||}$$

Unrolling from the output gradient $dZ_L = \Delta = (P - Y)/B$ gives, for any ℓ:

$$\boxed{||dZ_{\ell}|| \leq ||\Delta|| \prod_{j=\ell+1}^{L} ||W_j^T||}$$

---

3) Final combined bound for $||dW_\ell||$

Plug the dZ_ℓ bound into the dW_ℓ bound:

$$\boxed{||dW_{\ell}|| \leq ||A_{\ell-1}^T||\,||\Delta|| \prod_{j=\ell+1}^{L} ||W_j^T||}$$

where $\Delta = (P - Y)/B$.

if Frobeneius Norm

Assumptions (using a submultiplicative matrix norm, e.g. spectral 2-norm):

$$dW_l = A_{l-1}.T@dZ_l (linear - layerbackward : dL/dW = X.T@dL/dY)$$

$$||dZ_l|| <= ||Delta|| * \Pi_{j=l+1..L} ||W_j.T||$$

Then:

$$||dW_l|| <= ||A_{l-1}.T|| * ||Delta|| * \Pi_{j=l+1..L} ||W_j.T||$$

$$\boxed{||dW_{\ell}|| \leq (||W_{\ell-1}^T||||Z_{\ell-2}^T|| + \sqrt{B} \cdot ||b_{\ell-1}^T||)\,||\Delta|| \prod_{j=\ell+1}^{L} ||W_j^T||}$$

Bound (Frobenius for activations/gradients; spectral/operator-2 for weights & bias)

Notes:

- B = batch size (because bias is broadcast across the batch in Y = XW + b)
- ||M||_2 is the spectral norm; in NumPy: np.linalg.norm(M, 2)
- ||M||_F is the Frobenius norm; in NumPy: np.linalg.norm(M, 'fro')

Using equivalence to avoid mixed norms in products (but looser):

$$\textbf{For any matrix } M \in \mathbb{R}^{m\times n}: \qquad \|M\|_2 \le \|M\|_F \le \sqrt{\operatorname{rank}(M)}\,\|M\|_2 \le \sqrt{\min(m,n)}\,\|M$$

$$\|AB\|_F \le \|A\|_F\,\|B\|_2 \le \|A\|_F\,\|B\|_F.$$

We can use Frobenius for activations/gradients and spectral for weights (because spectral captures “worst-case amplification” through a linear map).