Convolution backward: CNN backprop

Ten problems on the backward pass of a convolutional layer: the kernel gradient as a correlation of the input with dy, the input gradient as a full convolution with the flipped kernel, the im2col and Toeplitz matrix views, 2-D and multi-channel layers on a batch, stride, same padding and pooling, with worked solutions and the mistakes that forget the flip or the sum over positions.

BEFORE YOU START

A convolutional layer is a dense layer with most of its weights forced to zero and the rest tied together: the same small kernel is applied at every position. Both constraints show up in the backward pass. Because the kernel is shared, its gradient is a sum over every position it was used at, which turns out to be another correlation. Because each output sees only a window of the input, the input gradient sends each output's gradient back to its own window, which turns out to be a convolution with the kernel flipped. These ten problems derive both in one dimension, check them by hand, and then carry them to two dimensions, several channels, a batch, stride, padding and pooling. The five mistakes at the end are the ones that produce an array of the right size or a plausible number: an unflipped kernel, a weight gradient averaged over positions, a valid correlation where a full one belongs, a stride ignored on the way back, and a max-pool gradient spread over the window.

- Indices start at 0 on this page, as in array code. The 1-D layer has input $x \in \mathbb{R}^n$, kernel $w \in \mathbb{R}^k$ with $k \le n$, a scalar bias b, and output $y \in \mathbb{R}^m$ with $y_i = \sum_{a=0}^{k-1} w_a \, x_{i+a} + b$ for $i = 0, \dots, m-1$.
- Deep-learning libraries call this a convolution. Mathematically it is a cross-correlation: the kernel is read in the same direction as the input. A true convolution flips the kernel, $(u * v)_j = \sum_i u_i \, v_{j-i}$, and one appears in Problem 3.
- $\operatorname{corr}(u, v)$ is the valid cross-correlation of a vector u of length p with a shorter vector v of length q: $\operatorname{corr}(u, v)_i = \sum_a v_a \, u_{i+a}$ for $i = 0, \dots, p-q$, so it has length $p - q + 1$. The layer is $y = \operatorname{corr}(x, w) + b\mathbf{1}$. For matrices, $\operatorname{corr}(U, V)_{ij} = \sum_{a,c} V_{ac} \, U_{i+a, j+c}$, with both offsets running over V's shape.
- $\operatorname{pad}_p(u)$ is u with p zeros added at each end; for a matrix, $\operatorname{pad}_{p_1, p_2}(U)$ adds p_1 rows of zeros above and below and p_2 columns of zeros left and right. $\operatorname{flip}(w)_a = w_{k-1-a}$ reverses a vector, and $\operatorname{rot}(K)_{ac} = K_{k_1-1-a,\, k_2-1-c}$ turns a $k_1 \times k_2$ matrix through $180°$.
- As on the minibatch page, dA is the code-style name for $\nabla_A L$, the array of $\partial L / \partial A_{\dots}$ with the shape of A, for a scalar loss L computed from the layer's output by the layers above. The backward pass receives dy and returns dw, db and dx.
- $\mathbf{1}$ is the all-ones vector and $\lfloor t \rfloor$ the largest integer at most t.

Builds on: Jacobians and the chain rule, Batched backprop: dense layers on a minibatch

Problems

1. Give the output length m in terms of n and k. Then compute y for $x = (1, 2, 0, -1, 3)$, $w = (2, -1, 1)$ and $b = 1$. difficulty ·

2. Compute dw and db. Write dw as a correlation. difficulty ··

3. Compute dx. Show that it is the full correlation of dy with the flipped kernel, $\mathrm{corr}(\mathrm{pad}_{k-1}(dy), \mathrm{flip}(w))$. difficulty ··

4. Write the layer as $y = Pw + b\mathbf{1}$ with $P \in \mathbb{R}^{m \times k}$, $P_{ia} = x_{i+a}$, and as $y = Tx + b\mathbf{1}$ with $T \in \mathbb{R}^{m \times n}$, $T_{ij} = w_{j-i}$ when $0 \le j - i \le k - 1$ and $T_{ij} = 0$ otherwise. Use the two forms to compute dw and dx, and compare with Problems 2 and 3. difficulty ··

5. By hand: $x = (2, -1, 3, 0, 1)$, $w = (1, 2, -1)$, and the layers above return $dy = (1, 0, -2)$. Compute dw, db and dx. difficulty ··

6. Two dimensions, one channel: $X \in \mathbb{R}^{n_1 \times n_2}$, $K \in \mathbb{R}^{k_1 \times k_2}$, a scalar bias b, and difficulty ···

$$Y_{ij} = \sum_{a=0}^{k_1-1} \sum_{c=0}^{k_2-1} K_{ac}\, X_{i+a,\, j+c} + b.$$

Give the shape of Y and compute dK, db and dX.

7. Channels and a batch: N inputs $X^{(n)} \in \mathbb{R}^{C \times n_1 \times n_2}$ with C channels, F filters, and biases $b \in \mathbb{R}^F$. Write $X_c^{(n)}$ for channel c of input n and $K_{f,c} \in \mathbb{R}^{k_1 \times k_2}$ for the slice of filter f that reads channel c. Output channel f of example n is $Y_f^{(n)} = \sum_{c=1}^{C} \mathrm{corr}(X_c^{(n)}, K_{f,c}) + b_f$. Compute $dK_{f,c}$, db_f and $dX_c^{(n)}$. difficulty ···

8. Stride s: $y_i = \sum_a w_a\, x_{si+a} + b$ for $i = 0, \dots, m-1$, with $m = \lfloor (n - k)/s \rfloor + 1$. Compute dw and dx. For $n = 8$, $k = 3$, $s = 2$, which entry of dx is zero whatever dy is? difficulty ···

9. Same padding: k is odd, $p = (k-1)/2$, and $y = \mathrm{corr}(\mathrm{pad}_p(x), w) + b\mathbf{1}$, so y has length n. Show that $dx = \mathrm{corr}(\mathrm{pad}_p(dy), \mathrm{flip}(w))$, and give dw. difficulty ..

10. Pooling with 2×2 windows and stride 2. For difficulty ..

$$X = \begin{bmatrix} 1 & 3 & 2 & 0 \\ 4 & 2 & 1 & 5 \\ 0 & -1 & 3 & 2 \\ 2 & 6 & 1 & 1 \end{bmatrix}$$

and

$$dY = \begin{bmatrix} 1 & -2 \\ 3 & \frac{1}{2} \end{bmatrix},$$

compute dX for max pooling, $Y_{ij} = \max_{a,c \in \{0,1\}} X_{2i+a,\,2j+c}$, and for average pooling, $Y_{ij} = \frac{1}{4}\sum_{a,c \in \{0,1\}} X_{2i+a,\,2j+c}$.

Answers

1. $m = n - k + 1$; here $m = 3$ and $y = (1, 4, 5)$
2. $dw = \operatorname{corr}(x, dy)$, that is $dw_a = \sum_i dy_i \, x_{i+a}$; $db = \mathbf{1}^\top dy$
3. $dx = \operatorname{corr}(\operatorname{pad}_{k-1}(dy), \operatorname{flip}(w))$, that is $dx_j = \sum_i dy_i \, w_{j-i}$
4. $dw = P^\top dy$ and $dx = T^\top dy$, the same vectors as Problems 2 and 3
5. $dw = (-4, -1, 1)$, $db = -1$, $dx = (1, 2, -3, -4, 2)$
6. $dK = \operatorname{corr}(X, dY)$; $db = \sum_{i,j} dY_{ij}$; $dX = \operatorname{corr}(\operatorname{pad}_{k_1-1,k_2-1}(dY), \operatorname{rot}(K))$
7. $dK_{f,c} = \sum_n \operatorname{corr}(X_c^{(n)}, dY_f^{(n)})$; $db_f = \sum_n \sum_{i,j} dY_{f,ij}^{(n)}$; $dX_c^{(n)} = \sum_f \operatorname{corr}(\operatorname{pad}_{k_1-1,k_2-1}(dY_f^{(n)}), \operatorname{rot}(K_{f,c}))$
8. $dw_a = \sum_i dy_i \, x_{si+a}$; $dx_j = \sum_i dy_i \, w_{j-si}$ (over $0 \le j - si \le k-1$), which is Problem 3 applied to dy with $s - 1$ zeros inserted between its entries; for $n = 8$, $k = 3$, $s = 2$, $dx_7 = 0$
9. $dx = \operatorname{corr}(\operatorname{pad}_p(dy), \operatorname{flip}(w))$ and $dw = \operatorname{corr}(\operatorname{pad}_p(x), dy)$
10. Max pooling: $dX = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix}$; average pooling: every entry of window (i, j) gets $dY_{ij}/4$, that is $\frac{1}{4}$, $-\frac{1}{2}$, $\frac{3}{4}$ and $\frac{1}{8}$ in the four windows

Worked solutions

Problem 1

> Give the output length m in terms of n and k. Then compute y for $x = (1, 2, 0, -1, 3)$, $w = (2, -1, 1)$ and $b = 1$.

Output i reads $x_i, x_{i+1}, \dots, x_{i+k-1}$, so it needs $i + k - 1 \le n - 1$, that is $i \le n - k$.	The window has k entries starting at i, and a valid correlation uses no padding, so the window must lie inside x.
i runs over $0, 1, \dots, n - k$, which is $n - k + 1$ values.	Counting from 0 adds one to the largest index.
$y_0 = 2 \cdot 1 + (-1) \cdot 2 + 1 \cdot 0 + 1 = 1.$	Window $(x_0, x_1, x_2) = (1, 2, 0)$, each entry multiplied by the kernel entry in the same position, plus b.
$y_1 = 2 \cdot 2 + (-1) \cdot 0 + 1 \cdot (-1) + 1 = 4.$	The window moves one step: $(x_1, x_2, x_3) = (2, 0, -1)$.
$y_2 = 2 \cdot 0 + (-1) \cdot (-1) + 1 \cdot 3 + 1 = 5.$	Window $(x_2, x_3, x_4) = (0, -1, 3)$, the last one that fits since $n - k = 2$.
$m = n - k + 1$; **here** $m = 3$ **and** $y = (1, 4, 5)$	Each w_a was used three times, once per window: this is the weight sharing whose consequence for the gradient is Problem 2.

Problem 2

> Compute *dw* and *db*. Write *dw* as a correlation.

$\partial y_i / \partial w_a = x_{i+a}$ for every $i = 0, \dots, m - 1$.	Only the term $w_a x_{i+a}$ of the sum contains w_a, and every output has such a term because the same kernel is applied at every position.

$\dfrac{\partial L}{\partial w_a} = \sum_{i=0}^{m-1} \dfrac{\partial L}{\partial y_i}\, x_{i+a} = \sum_i dy_i\, x_{i+a}.$	L depends on w only through y, and by step 1 every one of the m outputs contains w_a, so the chain rule has m terms.
$\sum_i dy_i\, x_{i+a} = \operatorname{corr}(x, dy)_a.$	The definition of $\operatorname{corr}(u, v)_a = \sum_i v_i\, u_{a+i}$ with $u = x$ and $v = dy$: dy plays the kernel and slides over x, giving $n - m + 1 = k$ outputs, one per kernel entry.
$\partial y_i / \partial b = 1$ for every i, so $\partial L / \partial b = \sum_i dy_i$.	b is added to every output, so the chain rule sums dy over all of them.
$dw = \operatorname{corr}(x, dy)$, **that is** $dw_a = \sum_i dy_i\, x_{i+a}$; $db = \mathbf{1}^\top dy$	The kernel gradient is a sum over positions for the same reason the minibatch page's bias gradient is a sum over examples: one parameter, many uses, and the chain rule adds the contributions.

Problem 3

> Compute dx. Show that it is the full correlation of dy with the flipped kernel, $\operatorname{corr}(\operatorname{pad}_{k-1}(dy), \operatorname{flip}(w))$.

x_j appears in y_i exactly when $j = i + a$ for some $0 \le a \le k - 1$, and then its coefficient is w_{j-i}.	Put $a = j - i$ in the term $w_a x_{i+a}$: kernel entry $j - i$ is the one that lands on x_j when the window starts at i.
$dx_j = \sum_i dy_i\, w_{j-i}$, summing over the i with $0 \le j - i \le k - 1$.	The chain rule sums over the outputs whose windows contain x_j. Inputs near the ends lie in fewer windows: x_0 lies only in window 0.
Let $\tilde{d} = \operatorname{pad}_{k-1}(dy)$, so $\tilde{d}_t = dy_{t-k+1}$ for $k - 1 \le t \le m + k - 2$ and $\tilde{d}_t = 0$ otherwise.	With $k - 1$ zeros at each end, every term w_{j-i} in step 2 can be written without a range condition: indices that fall outside dy hit a zero.

$\operatorname{corr}(\tilde{d}, \operatorname{flip}(w))_j = \sum_c w_{k-1-c}\, \tilde{d}_{j+c} = \sum_c w_{k-1-c}\, dy_{j+c-k+1}$.	The definition of corr with kernel $\operatorname{flip}(w)$, then step 3; the result has $(m + 2k - 2) - k + 1 = m + k - 1 = n$ entries, one per input.
Put $i = j + c - k + 1$; then $k - 1 - c = j - i$ and the sum becomes $\sum_i dy_i\, w_{j-i}$.	A change of summation index; as c runs over $0, \dots, k-1$, $j - i$ runs over the same range, so the terms match step 2 one for one.
$dx = \operatorname{corr}(\operatorname{pad}_{k-1}(dy), \operatorname{flip}(w))$, **that is** $dx_j = \sum_i dy_i\, w_{j-i}$	The second form is the true convolution $dy * w$, which is why the flip appears: the forward pass reads the kernel forwards as the window moves right, so seen from a fixed input x_j the kernel index $j - i$ decreases as the window index i increases. The forward pass is a correlation and the input gradient is a convolution.

Problem 4

> Write the layer as $y = Pw + b\mathbf{1}$ with $P \in \mathbb{R}^{m \times k}$, $P_{ia} = x_{i+a}$, and as $y = Tx + b\mathbf{1}$ with $T \in \mathbb{R}^{m \times n}$, $T_{ij} = w_{j-i}$ when $0 \le j - i \le k - 1$ and $T_{ij} = 0$ otherwise. Use the two forms to compute dw and dx, and compare with Problems 2 and 3.

$(Pw)_i = \sum_a x_{i+a}\, w_a = y_i - b$.	Row i of P is window i of x, so its product with w is the sum in the definition.
$(Tx)_i = \sum_j T_{ij}\, x_j = \sum_{a=0}^{k-1} w_a\, x_{i+a} = y_i - b$.	Only the j with $0 \le j - i \le k - 1$ contribute; put $a = j - i$.
$\partial y / \partial w = P$ and $\partial y / \partial x = T$.	y is affine in w for fixed x, and affine in x for fixed w; the Jacobian of $Mv + c$ with respect to v is M.
$dw = P^\top dy$ and $dx = T^\top dy$.	For a scalar L, the gradient with respect to an input is the transposed Jacobian times the gradient at the output.

$(P^\top dy)_a = \sum_i x_{i+a}\, dy_i$ and $(T^\top dy)_j = \sum_i w_{j-i}\, dy_i$.	$(M^\top v)_j = \sum_i M_{ij} v_i$; these are Problem 2, step 2 and Problem 3, step 2.
$dw = P^\top dy$ and $dx = T^\top dy$, the same vectors as Problems 2 and 3	P is what libraries call the im2col matrix, one window per row, and building it turns the layer and its kernel gradient into ordinary matrix products. T is the layer as a dense $m \times n$ matrix: banded, with the kernel copied along each row and zeros elsewhere. Its transpose sends each dy_i back to window i, which is Problem 3's flipped kernel; T itself is never formed.

Problem 5

> By hand: $x = (2, -1, 3, 0, 1)$, $w = (1, 2, -1)$, and the layers above return $dy = (1, 0, -2)$. Compute dw, db and dx.

$dw_0 = 1 \cdot 2 + 0 \cdot (-1) + (-2) \cdot 3 = -4.$	Problem 2: dw_a is dy dotted with the window of x starting at a; for $a = 0$ that is $(x_0, x_1, x_2) = (2, -1, 3)$.
$dw_1 = 1 \cdot (-1) + 0 \cdot 3 + (-2) \cdot 0 = -1$ and $dw_2 = 1 \cdot 3 + 0 \cdot 0 + (-2) \cdot 1 = 1.$	The windows $(x_1, x_2, x_3) = (-1, 3, 0)$ and $(x_2, x_3, x_4) = (3, 0, 1)$.
$db = 1 + 0 - 2 = -1.$	Problem 2: the sum of dy.
$\mathrm{pad}_2(dy) = (0, 0, 1, 0, -2, 0, 0)$ and $\mathrm{flip}(w) = (-1, 2, 1).$	Problem 3 with $k - 1 = 2$ zeros at each end.
$dx_0 = 0 \cdot (-1) + 0 \cdot 2 + 1 \cdot 1 = 1$, $dx_1 = 0 \cdot (-1) + 1 \cdot 2 + 0 \cdot 1 = 2$, $dx_2 = 1 \cdot (-1) + 0 \cdot 2 + (-2) \cdot 1 = -3.$	Slide $\mathrm{flip}(w)$ along the padded dy, starting at positions 0, 1, 2.

$dx_3 = 0 \cdot (-1) + (-2) \cdot 2 + 0 \cdot 1 = -4$ and $dx_4 = (-2) \cdot (-1) + 0 \cdot 2 + 0 \cdot 1 = 2$.	Positions 3 and 4, the last that fit in a length-7 vector. Spot check with $dx_j = \sum_i dy_i \, w_{j-i}$: x_0 is only in window 0, under $w_0 = 1$, so $dx_0 = dy_0 = 1$.
$dw = (-4, -1, 1)$, $db = -1$, $dx = (1, 2, -3, -4, 2)$	Five entries in dx and three in dw, the shapes of x and w.

Problem 6

Two dimensions, one channel: $X \in \mathbb{R}^{n_1 \times n_2}$, $K \in \mathbb{R}^{k_1 \times k_2}$, a scalar bias b, and

$$Y_{ij} = \sum_{a=0}^{k_1-1} \sum_{c=0}^{k_2-1} K_{ac} \, X_{i+a,\, j+c} + b.$$

Give the shape of Y and compute dK, db and dX.

Y is $(n_1 - k_1 + 1) \times (n_2 - k_2 + 1)$.	Problem 1 in each direction separately: the window must fit vertically and horizontally.
$\partial L / \partial K_{ac} = \sum_{i,j} dY_{ij} \, X_{i+a,\, j+c} = \operatorname{corr}(X, dY)_{ac}$.	Problem 2 in two dimensions: K_{ac} is used at every output position (i, j) with coefficient $X_{i+a,j+c}$, and the chain rule sums over all of them.
$\partial L / \partial b = \sum_{i,j} dY_{ij}$.	b is added to every entry of Y.
X_{pq} appears in Y_{ij} with coefficient $K_{p-i,\, q-j}$ whenever $0 \le p - i \le k_1 - 1$ and $0 \le q - j \le k_2 - 1$, so $dX_{pq} = \sum_{i,j} dY_{ij} \, K_{p-i,\, q-j}$.	Problem 3, steps 1 and 2, with one offset per direction.
$\sum_{i,j} dY_{ij} \, K_{p-i,\, q-j} = \operatorname{corr}(\operatorname{pad}_{k_1-1, k_2-1}(dY), \operatorname{rot}(K))_{pq}$.	Problem 3, steps 3 to 5, in each direction: flipping both axes of K is turning it through 180°, and the padding gives $n_1 \times n_2$ outputs.

$dK = \operatorname{corr}(X, dY)$; $db = \sum_{i,j} dY_{ij}$; $dX = \operatorname{corr}(\operatorname{pad}_{k_1-1,k_2-1}(dY), \operatorname{rot}(K))$	The shapes are $k_1 \times k_2$, a scalar and $n_1 \times n_2$, those of K, b and X. Nothing new happens in two dimensions: each axis behaves like the 1-D layer.

Problem 7

> Channels and a batch: N inputs $X^{(n)} \in \mathbb{R}^{C \times n_1 \times n_2}$ with C channels, F filters, and biases $b \in \mathbb{R}^F$. Write $X_c^{(n)}$ for channel c of input n and $K_{f,c} \in \mathbb{R}^{k_1 \times k_2}$ for the slice of filter f that reads channel c. Output channel f of example n is $Y_f^{(n)} = \sum_{c=1}^{C} \operatorname{corr}(X_c^{(n)}, K_{f,c}) + b_f$. Compute $dK_{f,c}$, db_f and $dX_c^{(n)}$.

$K_{f,c}$ appears only in output channel f, through the term $\operatorname{corr}(X_c^{(n)}, K_{f,c})$, and it appears there for every example n.	Each filter makes one output channel, its slice c reads only input channel c, and the filters are shared across the batch.
$dK_{f,c} = \sum_n \operatorname{corr}(X_c^{(n)}, dY_f^{(n)})$.	Problem 6, step 2, for each example, summed over examples because the parameter is shared, as on the minibatch page.
$db_f = \sum_n \sum_{i,j} dY_{f,ij}^{(n)}$.	b_f is added to every position of output channel f in every example.
$X_c^{(n)}$ appears in every output channel f of example n, through $\operatorname{corr}(X_c^{(n)}, K_{f,c})$, and in no other example.	Every filter reads every input channel; examples do not interact.
$dX_c^{(n)} = \sum_f \operatorname{corr}(\operatorname{pad}_{k_1-1,k_2-1}(dY_f^{(n)}), \operatorname{rot}(K_{f,c}))$.	Problem 6, step 5, for each output channel the input reaches, added because the chain rule sums over every path.

$dK_{f,c} = \sum_n \operatorname{corr}(X_c^{(n)}, dY_f^{(n)})$; $db_f = \sum_n \sum_{i,j} dY_{f,ij}^{(n)}$; $dX_c^{(n)} = \sum_f \operatorname{corr}(\operatorname{pad}_{k_1-1,k_2-1}(dY_f^{(n)}), \operatorname{rot}(K_{f,c}))$	The three sums are over the three kinds of sharing: the kernel gradient sums over positions (inside corr) and examples, the bias gradient over positions and examples, and the input gradient over the filters that read the channel. No gradient sums over input channels, because each slice $K_{f,c}$ reads one channel only.

Problem 8

> Stride s: $y_i = \sum_a w_a\, x_{si+a} + b$ for $i = 0, \ldots, m-1$, with $m = \lfloor (n-k)/s \rfloor + 1$. Compute dw and dx. For $n = 8$, $k = 3$, $s = 2$, which entry of dx is zero whatever dy is?

$dw_a = \sum_i dy_i\, x_{si+a}$.	Problem 2 with window i starting at si instead of i: w_a is still used once per output, now with coefficient x_{si+a}.
x_j appears in y_i with coefficient w_{j-si} when $0 \le j - si \le k-1$, so $dx_j = \sum_i dy_i\, w_{j-si}$ over those i.	Problem 3, steps 1 and 2, with window i starting at si.
Let u be dy with $s-1$ zeros inserted between neighboring entries: $u_{si} = dy_i$, other entries 0, length $s(m-1)+1$. Then $dx_j = \sum_t u_t\, w_{j-t}$.	The nonzero entries of u sit at $t = si$, so the sum over t is the sum over i in step 2; the zeros contribute nothing.
That is Problem 3's formula with u in place of dy, which gives the first $s(m-1)+k$ entries of dx; inputs x_j with $j \ge s(m-1)+k$ lie in no window, and their gradient is 0.	The last window starts at $s(m-1)$ and ends at $s(m-1)+k-1$; when s does not divide $n-k$, the inputs after it are never read.

For $n = 8$, $k = 3$, $s = 2$: $m = \lfloor 5/2 \rfloor + 1 = 3$, the last window covers x_4, x_5, x_6, and x_7 is never read.	Step 4 with $s(m-1) + k = 7$.
$dw_a = \sum_i dy_i \, x_{si+a}$; $dx_j = \sum_i dy_i \, w_{j-si}$ **(over** $0 \le j - si \le k-1$**), which is Problem 3 applied to** dy **with** $s - 1$ **zeros inserted between its entries; for** $n = 8$, $k = 3$, $s = 2$, $dx_7 = 0$	This is why the backward pass of a strided convolution is called a transposed convolution: it is $T^\top dy$ for the strided T of Problem 4, and the inserted zeros are how a stride-1 routine computes it.

Problem 9

> Same padding: k is odd, $p = (k-1)/2$, and $y = \mathrm{corr}(\mathrm{pad}_p(x), w) + b\mathbf{1}$, so y has length n. Show that $dx = \mathrm{corr}(\mathrm{pad}_p(dy), \mathrm{flip}(w))$, and give dw.

Let $\tilde{x} = \mathrm{pad}_p(x)$, of length $n + 2p = n + k - 1$; then y has $(n + k - 1) - k + 1 = n$ entries.	Problem 1 applied to $\tilde{x}$.
$d\tilde{x} = \mathrm{corr}(\mathrm{pad}_{k-1}(dy), \mathrm{flip}(w))$ and $dw = \mathrm{corr}(\tilde{x}, dy)$.	Problems 3 and 2 hold for any input vector, padded or not.
$x_j = \tilde{x}_{j+p}$, and the padding entries are constants, so $dx_j = d\tilde{x}_{j+p}$.	x reaches L only through $\tilde{x}$, entry for entry; the gradients that land on the zeros are discarded because nothing upstream produced them.
$d\tilde{x}_{j+p} = \sum_c w_{k-1-c} \, dy_{j+p+c-(k-1)} = \sum_c \mathrm{flip}(w)_c \, dy_{j+c-p}$.	Problem 3, step 4, at index $j + p$, with $k - 1 = 2p$.
$\sum_c \mathrm{flip}(w)_c \, dy_{j+c-p} = \mathrm{corr}(\mathrm{pad}_p(dy), \mathrm{flip}(w))_j$.	Entry $j + c$ of $\mathrm{pad}_p(dy)$ is dy_{j+c-p}, or 0 when that index falls outside dy.

$dx = \operatorname{corr}(\operatorname{pad}_p(dy), \operatorname{flip}(w))$ **and** $dw = \operatorname{corr}(\operatorname{pad}_p(x), dy)$	The backward pass of a same convolution is a same convolution with the flipped kernel, which is why frameworks can reuse the forward routine for it. In two dimensions the flip becomes $\operatorname{rot}(K)$.

Problem 10

Pooling with 2×2 windows and stride 2. For

$$X = \begin{bmatrix} 1 & 3 & 2 & 0 \\ 4 & 2 & 1 & 5 \\ 0 & -1 & 3 & 2 \\ 2 & 6 & 1 & 1 \end{bmatrix}$$

and

$$dY = \begin{bmatrix} 1 & -2 \\ 3 & \frac{1}{2} \end{bmatrix},$$

compute dX for max pooling, $Y_{ij} = \max_{a,c \in \{0,1\}} X_{2i+a,\,2j+c}$, and for average pooling, $Y_{ij} = \frac{1}{4} \sum_{a,c \in \{0,1\}} X_{2i+a,\,2j+c}$.

Where a window's largest entry is unique, Y_{ij} equals that entry for every small enough change of X, so $\partial Y_{ij} / \partial X_{pq}$ is 1 at the largest entry and 0 at the other three.	A small perturbation cannot change which entry is largest when the largest is strictly larger than the rest. With a tie the max is not differentiable, and libraries pick one of the tied entries.
The largest entries are 4 at $(1,0)$, 5 at $(1,3)$, 6 at $(3,1)$ and 3 at $(2,2)$.	The windows are rows 0–1 and 2–3 crossed with columns 0–1 and 2–3: $\{1,3,4,2\}$, $\{2,0,1,5\}$, $\{0,-1,2,6\}$, $\{3,2,1,1\}$. The two 1s in the last window are not its largest entry, so they cause no tie.
Max pooling: $dX_{1,0} = 1$, $dX_{1,3} = -2$, $dX_{3,1} = 3$, $dX_{2,2} = \frac{1}{2}$, every other entry 0.	The windows do not overlap, so each X_{pq} lies in one window only and receives that window's dY_{ij} if it is the largest entry, 0 otherwise.

Average pooling: $\partial Y_{ij}/\partial X_{pq} = \frac{1}{4}$ for all four entries of window (i, j).	The mean is linear with equal weights.
Max pooling: $dX = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix}$; **average pooling: every entry of window** (i, j) **gets** $dY_{ij}/4$**, that is** $\frac{1}{4}$**,** $-\frac{1}{2}$**,** $\frac{3}{4}$ **and** $\frac{1}{8}$ **in the four windows**	Max pooling routes the whole gradient to one entry per window, so the forward pass must save where the maximum was; average pooling spreads it evenly and needs to save nothing.

Where this goes wrong

1. Input gradient with the kernel unflipped

The forward pass is a correlation with w, and it is natural to expect the backward pass to be one too.

$dx_j = \sum_i dy_i \, w_{j-i}$, and for Problem 5 $\mathrm{pad}_2(dy) = (0, 0, 1, 0, -2, 0, 0)$	Right so far: Problem 3, step 2, and Problem 5, step 4.
"Backward is the forward operation run on the gradient: correlate the padded dy with w."	The analogy that causes the mistake: the forward pass reads the kernel forwards, so the backward pass is assumed to read it forwards too.
$dx = \mathrm{corr}(\mathrm{pad}_2(dy), w) = (-1, 2, 3, -4, -2)$ ✗	The correct answer is $(1, 2, -3, -4, 2)$ (Problem 5). From x_j's point of view the kernel index $j - i$ runs backwards as the window index i runs forwards, so the kernel must be flipped. A symmetric kernel hides the error, and so does a shape check: the length is right.

2. Averaging the kernel gradient over positions

Weight sharing makes one kernel stand in for m copies, and averaging the copies' gradients sounds like the fair way to combine them.

$\partial L/\partial w_a = \sum_i \frac{\partial L}{\partial y_i}\,\frac{\partial y_i}{\partial w_a}$, with one term per output position	Right so far: Problem 2, step 2.
"The kernel is shared by m positions, so its gradient is the average of the m per-position gradients."	The shortcut that causes the mistake: confusing the $1/N$ of a mean loss, which is part of the loss, with the way the chain rule combines the uses of a shared parameter.
$dw_a = \frac{1}{m}\sum_i dy_i\, x_{i+a}$ ✗	The chain rule adds the contributions of every use; it never divides by their number. If the loss is a mean, its $1/N$ is already inside dy. With the extra $\frac{1}{m}$ the kernel learns m times more slowly than the bias, and the factor changes with the image size.

3. Input gradient from a valid correlation

Problem 2 found the kernel gradient as a valid correlation, and the input gradient looks as if it should be built the same way.

$dx_j = \sum_i dy_i\, w_{j-i}$ for $j = 0, \ldots, n-1$	Right so far: Problem 3, step 2. There is one entry per input.
"Correlate dy with the flipped kernel; the kernel gradient needed no padding, so neither does this."	The analogy that causes the mistake: in Problem 2 the kernel dy is shorter than x and the result has length k; here the kernel is w and the input it slides over is dy itself, which is shorter than the answer.

$dx = \text{corr}(dy, \text{flip}(w))$ ✗	That has length $m - k + 1 = n - 2k + 2$, not n: for Problem 5 it is the single number -3, which is dx_2 alone. The inputs near the ends lie in fewer windows but still get gradients, and the $k - 1$ zeros at each end of dy are what supply the missing windows.

4. Strided input gradient without the inserted zeros

The stride-1 input gradient is a correlation with the flipped kernel, and with a stride it is tempting to reuse it unchanged.

$dx_j = \sum_i dy_i \, w_{j - si}$	Right so far: Problem 8, step 2.
"The stride only changes where the windows start, which the forward pass handled; the backward pass is the same correlation as before."	The shortcut that causes the mistake: treating the stride as a detail of the forward loop, when it is in the coefficient w_{j-si}.
$dx = \text{corr}(\text{pad}_{k-1}(dy), \text{flip}(w))$ ✗	That is $dx_j = \sum_i dy_i \, w_{j-i}$, which sends dy_i back to window i instead of window si, and it has length $m + k - 1$: for $n = 8$, $k = 3$, $s = 2$ that is 5 entries for 8 inputs. Insert $s - 1$ zeros between the entries of dy first (Problem 8).

5. Max-pool gradient spread over the window

Average pooling and max pooling both reduce a window to one number, and their gradients are easy to treat as the same.

Y_{ij} is the largest entry of window (i, j), and for Problem 10 the top-left window is $\{1, 3, 4, 2\}$ with $dY_{00} = 1$	Right so far: Problem 10, step 2.

“Every entry of the window took part in the pooling, so each gets an equal share of the gradient.”	The analogy that causes the mistake: average pooling's rule, Problem 10, step 4, applied to a maximum.
$dX_{00} = dX_{01} = dX_{10} = dX_{11} = \frac{1}{4}$ ✗	A small change to $X_{00} = 1$, $X_{01} = 3$ or $X_{11} = 2$ does not change the maximum 4, so their derivatives are 0; only X_{10} gets the gradient, all of it: $dX_{10} = 1$ (Problem 10, step 3).

More at eulerpad.app. This is one of 82 free practice sets with worked solutions at eulerpad.app/practice, alongside guides to the topics behind them.

EulerPad, the app, reads your handwritten math on iPad, iPhone and Android and checks it line by line. It's free to start.

This set is also Chapter 14 of *Deep Learning Math by Hand*, a 250-problem workbook with worked solutions: eulerpad.app/workbooks

eulerpad.app/practice/convolution-backward