

Backpropagation by Hand

A tiny CNN, start to finish

Name: _____ Date: _____

You will push one 5×5 image through a tiny convolutional network, measure how wrong the prediction is, send the error backward through every layer, and take one gradient-descent step. At the end you run the network again to check that the loss went down. Every layer type from the lecture appears once: convolution, ReLU, max-pool, dense layer, and softmax with cross-entropy.



Given

X (input, 5×5)

x	0	1	2	3	4
0	1	0	2	1	0
1	0	1	1	0	2
2	3	1	0	1	1
3	0	2	1	1	1
4	1	0	1	2	0

K (filter, 2×2)

K	0	1
0	1	-1
1	0.5	1

W (dense weights, 2×4)

w	p0	p1	p2	p3
A	0.2	-0.1	0.1	0.3
B	-0.2	0.1	0.2	-0.1

p0 to p3: pooled values, row by row

True class: B, so $y = (0, 1)$ **Learning rate:** 0.1 **Biases:** $b = 0$ (conv), $c = (0, 0)$ (dense)
 Indices are (row, column) and start at 0. Round to **four decimals**. If your last digit is off by 1 (0.0001), that is just rounding.

Part 1 · Forward pass

Step 1 · Convolution

Slide the 2×2 filter over X one cell at a time, with no padding. Each output cell is the sum of four products from a 2×2 patch of X , plus b . A 5×5 input and a 2×2 filter give a 4×4 output.

```
conv[i][j] = X[i][j] · K[0][0] + X[i][j+1] · K[0][1] + X[i+1][j] · K[1][0] +
X[i+1][j+1] · K[1][1] + b
```

Example: $\text{conv}[0][0] = 1 \cdot 1 + 0 \cdot (-1) + 0 \cdot 0.5 + 1 \cdot 1 = 2$ (already filled in)

conv (4×4)

conv	0	1	2	3
0	2			
1				
2				
3				

Step 2 · ReLU and max-pool

ReLU replaces every negative number with 0. Then cut the 4×4 map into four non-overlapping 2×2 windows and keep the largest value in each. Also write down where each winner sat in the 4×4 map.

Backpropagation needs it.

ReLU(conv) (4×4)

ReLU	0	1	2	3
0				
1				
2				
3				

Pooled values (2×2)

pool	0	1
0		
1		

Winner position in the 4×4 map

win	0	1
0	(,)	(,)
1	(,)	(,)

Step 3 · Dense layer, softmax, loss

Flatten the pooled map row by row into four numbers, multiply by W , apply softmax, and take the negative log of the probability that the network gave to the true class.

$$z[n] = \sum W[n][i] \cdot \text{flat}[i] + c[n] \quad p[n] = e^{z[n]} / (e^{z[A]} + e^{z[B]}) \quad L = -\ln p[B]$$

flat

--	--	--	--

pooled values, row by row

z = (A, B)

--	--

$z = W \cdot \text{flat} + c$

p = (A, B)

--	--

$p = \text{softmax}(z)$; the two numbers add up to 1

L

--

$L = -\ln p[B]$, because the true class is B

Part 2 · Backward pass

Now run the network in reverse. Use the old weights from the Given section. Nothing changes until Step 6.

Step 4 · Output layer and dense layer

$dz = p - y$ with $y = (0, 1)$. This is the softmax plus cross-entropy gradient. $dc = dz$
 $dW[n][i] = dz[n] \cdot flat[i]$ $dflat[i] = \sum \text{over } n \text{ of } dz[n] \cdot W[n][i]$

$dz = (A, B)$

--	--

p minus y

dW (2×4)

dW	$p0$	$p1$	$p2$	$p3$
A				
B				

$dflat$

--	--	--	--

how the loss changes with each pooled value

Step 5 · Max-pool and convolution

Pool routing: each $dflat$ value belongs to one window ($dflat[0]$ to the top-left window, $dflat[1]$ to top-right, $dflat[2]$ bottom-left, $dflat[3]$ bottom-right). Put it in the cell that won that window. Every other cell gets 0.
 ReLU mask: if that cell was ≤ 0 before ReLU, its gradient becomes 0.

$dconv$ (4×4)

$dconv$	0	1	2	3
0				
1				
2				
3				

dK (2×2)

dK	0	1
0		
1		

db

--

$dK[a][d] = \sum \text{over all } (i, j) \text{ of } dconv[i][j] \cdot X[i+a][j+d]$ $db = \sum dconv[i][j]$

The same filter weight was used at every position, so you add up its contribution from all of them. Only four cells of $dconv$ are non-zero, so each sum has just four terms.

Work space

Part 3 · Update and re-check

Step 6 · Gradient descent step

$w \leftarrow w - 0.1 \cdot dw$ for every weight and bias: K, b, W and c.

K new (2×2)

K	0	1
0		
1		

b new

W new (2×4)

W	p0	p1	p2	p3
A				
B				

c new (A, B)

Step 7 · Run the forward pass again

Use K new, b new, W new and c new. Repeat Steps 1 to 3 and write down the results. Did the loss drop from 1.1711?

Work space

pooled max of each window, row by row

z = (A, B)

p = (A, B)

L new before: 1.1711

Think about it

1. Twelve of the sixteen entries of $dconv$ are 0. Name two different reasons a cell can end up with a zero gradient.

2. Row B of dW is exactly the negative of row A. Why does that always happen with two classes?

3. Why does the forward pass have to remember which cell won each max-pool window?

4. The loss fell from 1.1711 to about 0.018 in a single step. Would you expect drops like that when training on thousands of images? Why or why not?
