Convolutional Neural Networks

One learned local question, asked everywhere

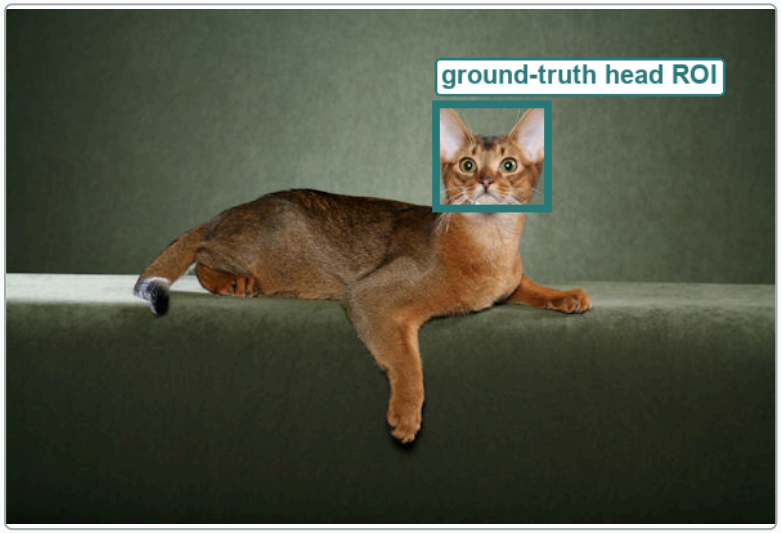
Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

Architecture can regularize before training begins

L7 changed the training recipe; L8 changes the hypothesis class

V VISUAL



REAL PHOTO · actual ground-truth head ROI · not a model prediction

L7 · FIX THE PROTOCOL

Keep loss, optimizer, split roles, and sealed test fixed.

L8 · CHANGE THE MAP

Encode locality and reuse one learned question across positions because nearby pixels interact and useful patterns recur.

Commit: can ten parameters produce 1,024 responses?

Q QUESTION

A 32×32 grayscale image must produce one 32×32 response map.

Which claim can be true? Keep your choice until the closing revisit.

A. Ten parameters can produce only ten output values

B. Ten shared parameters can produce 1,024 values by reusing one local rule

C. Any ten-parameter map can reproduce an arbitrary dense layer

D. Parameter sharing makes the computation independent of image size

Commit: can ten parameters produce 1,024 responses?

Q QUESTION

A 32×32 grayscale image must produce one 32×32 response map.

Which claim can be true? Keep your choice until the closing revisit.

A. Ten parameters can produce only ten output values

B. Ten shared parameters can produce 1,024 values by reusing one local rule

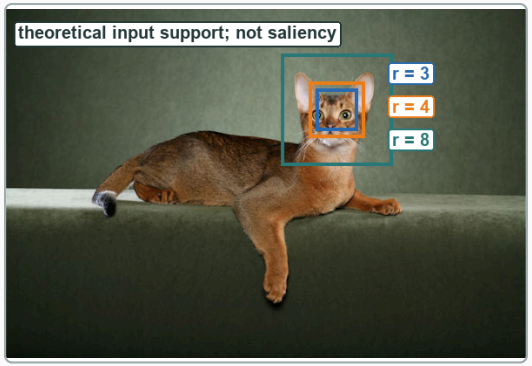
C. Any ten-parameter map can reproduce an arbitrary dense layer

D. Parameter sharing makes the computation independent of image size

> Commit to A, B, C, or D, then write the assumption your choice requires.

Outline

V VISUAL



REAL PHOTO · the local-to-global story stays attached to pixels

01 · LOCAL OPERATOR

Ask one learned question in a small neighbourhood.

02 · SHAPE GEOMETRY

Track padding, stride, dilation, and channel axes.

03 · RESPONSE BEHAVIOUR

Test shifts, pooling, and the limits of equivariance.

04 · RECEPTIVE FIELD

Trace which input pixels can influence a feature.

05 · CNN LEDGER

Count shapes, parameters, MACs, and classifier inputs.

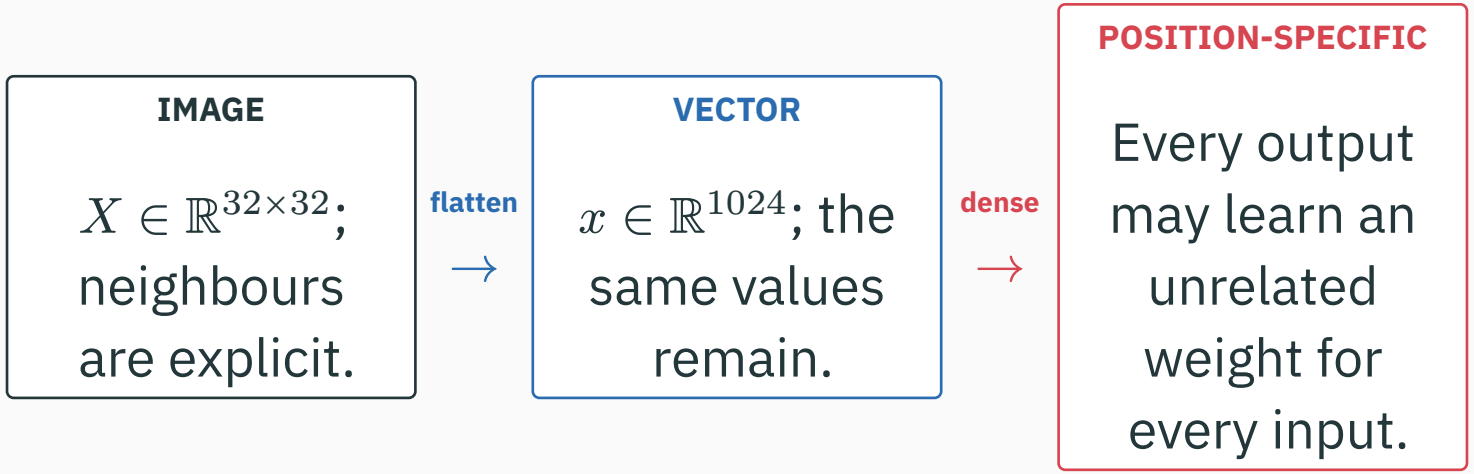
06 · LEARNING + DEBUG

Accumulate a shared gradient; verify layout and mode.

input · shared parameter · geometry · response · verified · failure

Flattening preserves values but removes the useful constraint

V VISUAL



Flattening preserves values but removes the useful constraint

V VISUAL

IMAGE

$X \in \mathbb{R}^{32\times32}$; neighbours are explicit.

flatten →

VECTOR

$x \in \mathbb{R}^{1024}$; the same values remain.

dense →

POSITION-SPECIFIC

Every output may learn an unrelated weight for every input.

LOCALITY

look at a small neighbourhood

SHARING

reuse the same detector everywhere

HIERARCHY

compose local evidence through depth

Compare like with like: include one bias per output

D DERIVATION

Both maps produce $32 \times 32 = 1024$ scalar outputs. The shared 3×3 map therefore uses $p = 1$, $s = 1$.

Compare like with like: include one bias per output

D DERIVATION

Both maps produce $32 \times 32 = 1024$ scalar outputs. The shared 3×3 map therefore uses $p = 1$, $s = 1$.

DENSE MAP

weights: 1024×1024

biases: 1024

total: $1{,}049{,}600$

SHARED 3×3 MAP

weights: $3 \times 3 = 9$

one shared bias: 1

total: 10

Compare like with like: include one bias per output

D DERIVATION

Both maps produce $32 \times 32 = 1024$ scalar outputs. The shared 3×3 map therefore uses $p = 1$, $s = 1$.

DENSE MAP

weights: 1024×1024
biases: 1024
total: $1{,}049{,}600$

SHARED 3×3 MAP

weights: $3 \times 3 = 9$
one shared bias: 1
total: 10

$$\frac{1{,}049{,}600}{10} = 104{,}960 \times \textbf{fewer parameters}$$

Compare like with like: include one bias per output

D DERIVATION

Both maps produce $32 \times 32 = 1024$ scalar outputs. The shared 3×3 map therefore uses $p = 1$, $s = 1$.

DENSE MAP

weights: 1024×1024
biases: 1024
total: $1{,}049{,}600$

SHARED 3×3 MAP

weights: $3 \times 3 = 9$
one shared bias: 1
total: 10

$$\frac{1{,}049{,}600}{10} = 104{,}960 \times \textbf{fewer parameters}$$

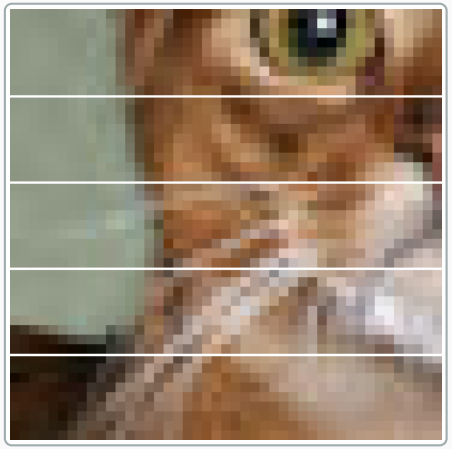
> The saving comes from two restrictions together: local connectivity and the same weights at every output location.

One local question, reused across the image

A declared real crop becomes the fixed 5×5 teaching image

V VISUAL

RUNNING CASE · REAL → PEDAGOGICAL Zero-based crop $x \in [328, 373), y \in [120, 165)$ is observed evidence; five band means are quantized deliberately for hand calculation.



REAL 45×45 RGB crop

gray + band mean →

band	mean $(0–255)$
1	113.83
2	112.37
3	145.50
4	145.78
5	109.32

≥ 128 →

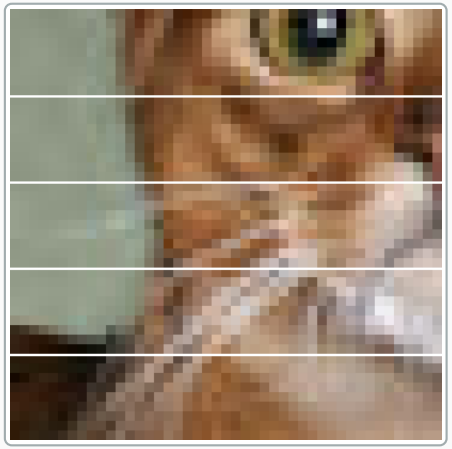
0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

X: teaching image

A declared real crop becomes the fixed 5×5 teaching image

V VISUAL

RUNNING CASE · REAL → PEDAGOGICAL Zero-based crop $x \in [328, 373), y \in [120, 165)$ is observed evidence; five band means are quantized deliberately for hand calculation.



REAL 45×45 RGB crop

gray + band mean →

band	mean $(0–255)$
1	113.83
2	112.37
3	145.50
4	145.78
5	109.32

≥ 128 →

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

X: teaching image

Quantization yields rows $(0, 0, 1, 1, 0)$; X now stays fixed for exact arithmetic.

Meet the shared 3×3 question

V VISUAL

RUNNING CASE · INPUT + PARAMETER X is the declared teaching construction; K is the shared parameter reused through convolution, geometry, pooling, and RF.

X: constructed input

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

K: shared kernel

1	1	1
0	0	0
−1	−1	−1

INK = input values · TEAL = learned kernel

Meet the shared 3×3 question

V VISUAL

RUNNING CASE · INPUT + PARAMETER X is the declared teaching construction; K is the shared parameter reused through convolution, geometry, pooling, and RF.

X: constructed input

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

K: shared kernel

1	1	1
0	0	0
−1	−1	−1

INK = input values · TEAL = learned kernel

The signed kernel compares the row above with the row below.

Library “convolution” is cross-correlation

D DERIVATION

For one channel, no padding, and stride 1,

Library “convolution” is cross-correlation

D DERIVATION

For one channel, no padding, and stride 1,

$$Y[i,j] = \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} K[u,v]X[i+u, j+v].$$

Library “convolution” is cross-correlation

D DERIVATION

For one channel, no padding, and stride 1,

$$Y[i,j] = \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} K[u,v]X[i+u, j+v].$$

CROSS-CORRELATION

uses $K[u,v]$ in the displayed orientation

MATHEMATICAL CONVOLUTION

flips the kernel before sliding

Library “convolution” is cross-correlation

D DERIVATION

For one channel, no padding, and stride 1,

$$Y[i,j] = \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} K[u,v]X[i+u, j+v].$$

CROSS-CORRELATION

uses $K[u,v]$ in the displayed orientation

MATHEMATICAL CONVOLUTION

flips the kernel before sliding

PyTorch and other DL libraries implement cross-correlation but call the layer convolution. Because K is learned, either orientation can represent the required detector.

Predict the first response before multiplying

Q QUESTION

RUNNING CASE · FIRST PATCH $X[0:3, 0:3]$ meets the same K; predict sign, then exact value.

first patch

0	0	0
0	0	0
1	1	1

kernel K

1	1	1
0	0	0
−1	−1	−1

Predict the first response before multiplying

Q QUESTION

RUNNING CASE · FIRST PATCH $X[0:3, 0:3]$ meets the same K; predict sign, then exact value.

first patch

0	0	0
0	0	0
1	1	1

kernel K

1	1	1
0	0	0
−1	−1	−1

Will $Y[0, 0]$ be negative, zero, or positive? What is its exact value?

Answer: the bottom row contributes -3

A ANSWER

RUNNING CASE · FIRST RESPONSE Every displayed number comes from the fixed image and kernel.

Answer: the bottom row contributes -3

A ANSWER

RUNNING CASE · FIRST RESPONSE Every displayed number comes from the fixed image and kernel.

$$Y[0,0] = (0\cdot 1 + 0\cdot 1 + 0\cdot 1) + (0\cdot 0 + 0\cdot 0 + 0\cdot 0) + (1\cdot(-1) + 1\cdot(-1) + 1\cdot(-1)).$$

Answer: the bottom row contributes -3

A ANSWER

RUNNING CASE · FIRST RESPONSE Every displayed number comes from the fixed image and kernel.

$$Y[0,0] = (0\cdot 1 + 0\cdot 1 + 0\cdot 1) + (0\cdot 0 + 0\cdot 0 + 0\cdot 0) + (1\cdot(-1) + 1\cdot(-1) + 1\cdot(-1)).$$

$$Y[0,0] = -3$$

Negative means bright pixels lie below dark pixels for this kernel orientation.

Predict the full valid feature map

Q QUESTION

RUNNING CASE · SLIDE $H = W = 5, k = 3, p = 0, s = 1$; the kernel is reused at every legal start.

x

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

Predict the full valid feature map

Q QUESTION

RUNNING CASE · SLIDE $H = W = 5, k = 3, p = 0, s = 1$; the kernel is reused at every legal start.

x

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

1. What is the output shape?
2. Why are values constant across each output row?
3. Which row is positive?

Answer: nine local questions form one feature map

A ANSWER

RUNNING CASE · FEATURE MAP Y stores one signed response for every legal 3×3 patch.

$X: 5 \times 5$

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

Answer: nine local questions form one feature map

A ANSWER

RUNNING CASE · FEATURE MAP Y stores one signed response for every legal 3×3 patch.

$X: 5 \times 5$

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0

$\rightarrow$

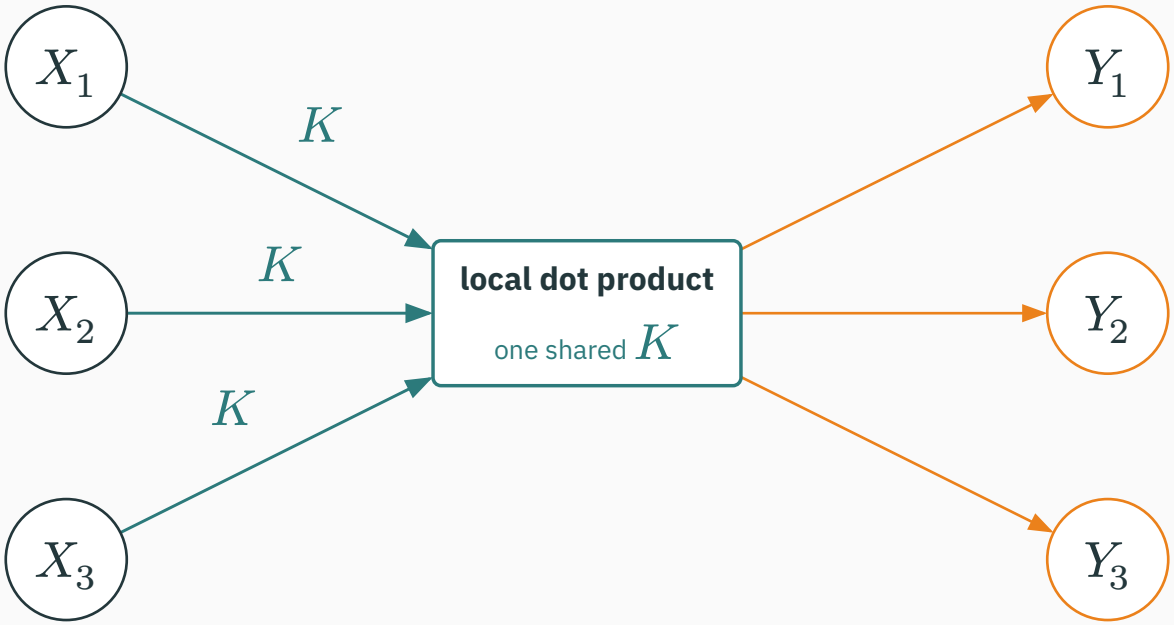
$Y: 3 \times 3$

−3	−3	−3
−3	−3	−3
3	3	3

$$Y = \begin{pmatrix} -3 & -3 & -3 \\ -3 & -3 & -3 \\ 3 & 3 & 3 \end{pmatrix}$$ **: the same detector, nine locations.**

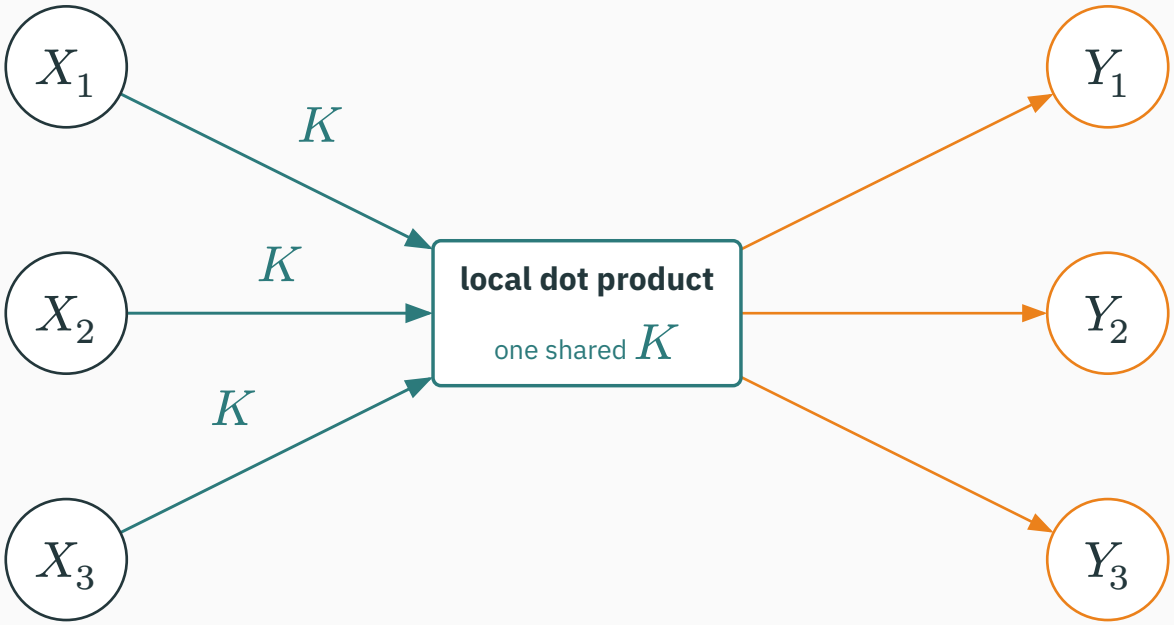
Weight sharing is visible in the computation graph

V VISUAL



Weight sharing is visible in the computation graph

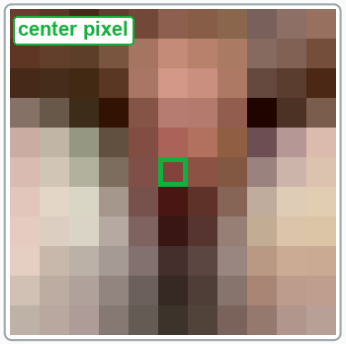
V VISUAL



Outputs are numerous; learned questions are few. Backprop must therefore sum evidence from every use of K.

One actual RGB pixel still spans every input channel

D DERIVATION



REAL PIXEL · photo coordinate $(373, 137)$

The observed 8-bit value is $(131, 66, 60)$, so

$$x = \frac{131,66,60}{255} \approx (0.514, 0.259, 0.235).$$

Let one illustrative 1×1 kernel be $k = (0.5, -1, 0.4)$.

$$y = 0.514(0.5) - 0.259 + 0.235(0.4) \approx 0.257 - 0.259 + 0.094 = 0.092.$$

1×1 fixes one observed location; $C_{\text{in}} = 3$ spans all RGB channels; the declared teaching kernel returns one response. A 3×3 RGB kernel has 27 weights per output channel.

Tensor shapes name every axis before we count

D DERIVATION

Board notation keeps channels last:

$$X \in \mathbb{R}^{H \times W \times C_{\text{in}}}, \quad K \in \mathbb{R}^{k_h \times k_w \times C_{\text{in}} \times C_{\text{out}}}, \quad Y \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}.$$

Tensor shapes name every axis before we count

D DERIVATION

Board notation keeps channels last:

$$X \in \mathbb{R}^{H \times W \times C_{\text{in}}}, \quad K \in \mathbb{R}^{k_h \times k_w \times C_{\text{in}} \times C_{\text{out}}}, \quad Y \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}.$$

ONE OUTPUT CHANNEL

one kernel spans all C_{in} input channels

C_{OUT} CHANNELS

a bank of C_{out} kernels produces C_{out} feature maps

Tensor shapes name every axis before we count

D DERIVATION

Board notation keeps channels last:

$$X \in \mathbb{R}^{H \times W \times C_{\text{in}}}, \quad K \in \mathbb{R}^{k_h \times k_w \times C_{\text{in}} \times C_{\text{out}}}, \quad Y \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}.$$

ONE OUTPUT CHANNEL

one kernel spans all C_{in} input channels

C_{OUT} CHANNELS

a bank of C_{out} kernels produces C_{out} feature maps

$$\text{parameters} = k_h k_w C_{\text{in}} C_{\text{out}} + C_{\text{out}}$$ **when each output channel has one bias.**

Checkpoint: count a kernel bank

Q QUESTION

A 3×3 convolution maps 64 input channels to 128 output channels and uses one bias per output channel.

How many learned parameters does it have?

A. $3 \cdot 3 \cdot 64 + 128$

B. $3 \cdot 3 \cdot 64 \cdot 128$

C. $3 \cdot 3 \cdot 64 \cdot 128 + 128$

D. $H'W' \cdot 3 \cdot 3 \cdot 64 \cdot 128$

Checkpoint: count a kernel bank

Q QUESTION

A 3×3 convolution maps 64 input channels to 128 output channels and uses one bias per output channel.

How many learned parameters does it have?

A. $3 \cdot 3 \cdot 64 + 128$

B. $3 \cdot 3 \cdot 64 \cdot 128$

C. $3 \cdot 3 \cdot 64 \cdot 128 + 128$

D. $H'W' \cdot 3 \cdot 3 \cdot 64 \cdot 128$

> Commit before calculating. Output height and width do not belong in the parameter formula.

Answer: kernels connect channel pairs; biases index outputs

A ANSWER

Correct: C — $3 \cdot 3 \cdot 64 \cdot 128 + 128 = 73{,}856$

Why: Each of 128 output channels owns a $3 \times 3 \times 64$ kernel and one bias. Spatial reuse adds computations, not parameters.

Answer: kernels connect channel pairs; biases index outputs

A ANSWER

Correct: C — $3 \cdot 3 \cdot 64 \cdot 128 + 128 = 73{,}856$

Why: Each of 128 output channels owns a $3 \times 3 \times 64$ kernel and one bias. Spatial reuse adds computations, not parameters.

Parameters depend on kernel and channel counts—not on image resolution.

Companion notebook: predict, calculate, run, inspect, explain

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Open the exact Colab notebook. First predict $Y[0, 0]$ and the full 3×3 map; then run the NumPy loops and compare every value.

Companion notebook: predict, calculate, run, inspect, explain

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Open the exact Colab notebook. First predict $Y[0, 0]$ and the full 3×3 map; then run the NumPy loops and compare every value.

PREDICT	RUN	EXPLAIN
shape, sign, and legal starts	from-scratch cross-correlation	where sharing, padding, and stride occur

Companion notebook: predict, calculate, run, inspect, explain

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Open the exact Colab notebook. First predict $Y[0, 0]$ and the full 3×3 map; then run the NumPy loops and compare every value.

PREDICT	RUN	EXPLAIN
shape, sign, and legal starts	from-scratch cross-correlation	where sharing, padding, and stride occur

The notebook retains outputs and later verifies the same gradients with PyTorch and finite differences.

Padding and stride count legal starts

Derive output size by counting kernel starts

D DERIVATION

For one spatial dimension with input length H, symmetric padding p, kernel size k, stride s, and dilation 1:

Derive output size by counting kernel starts

D DERIVATION

For one spatial dimension with input length H, symmetric padding p, kernel size k, stride s, and dilation 1:

PADDED LENGTH	LAST LEGAL START	STARTS
$H + 2p$	$H + 2p - k$	$0, s, 2s, \ldots$ up to that value

Derive output size by counting kernel starts

D DERIVATION

For one spatial dimension with input length H, symmetric padding p, kernel size k, stride s, and dilation 1:

PADDED LENGTH	LAST LEGAL START	STARTS
$H + 2p$	$H + 2p - k$	$0, s, 2s, \ldots$ up to that value

$$H_{\text{out}} = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1.$$

Derive output size by counting kernel starts

D DERIVATION

For one spatial dimension with input length H, symmetric padding p, kernel size k, stride s, and dilation 1:

PADDED LENGTH	LAST LEGAL START	STARTS
$H + 2p$	$H + 2p - k$	$0, s, 2s, \ldots$ up to that value

$$H_{\text{out}} = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1.$$

The floor discards an incomplete final jump; it is not a rounding convention.

The running image separates padding from stride

D DERIVATION

RUNNING CASE · GEOMETRY Same X and K; only p and s change.

$p = 0, s = 1 \cdot 3 \times 3$

The running image separates padding from stride

D DERIVATION

RUNNING CASE · GEOMETRY Same X and K; only p and s change.

$p = 0, s = 1 \cdot 3 \times 3$

−3	−3	−3
−3	−3	−3
3	3	3

$p = 1, s = 1 \cdot 5 \times 5$

0	0	0	0	0
−2	−3	−3	−3	−2
−2	−3	−3	−3	−2
2	3	3	3	2
2	3	3	3	2

The running image separates padding from stride

D DERIVATION

RUNNING CASE · GEOMETRY Same X and K; only p and s change.

$p = 0, s = 1 \cdot 3 \times 3$

−3	−3	−3
−3	−3	−3
3	3	3

$p = 1, s = 1 \cdot 5 \times 5$

0	0	0	0	0
−2	−3	−3	−3	−2
−2	−3	−3	−3	−2
2	3	3	3	2
2	3	3	3	2

$p = 1, s = 2 \cdot 3 \times 3$

0	0	0
−2	−3	−2
2	3	2

Padding changes which border patches are legal. Stride keeps only every s-th legal start.

“Same” padding is a restricted promise

D DERIVATION

For odd k and stride $s = 1$, symmetric

“Same” padding is a restricted promise

D DERIVATION

For odd k and stride $s = 1$, symmetric

$$p = \frac{k-1}{2} \quad \rightarrow \quad H_{\text{out}} = H.$$

“Same” padding is a restricted promise

D DERIVATION

For odd k and stride $s = 1$, symmetric

$$p = \frac{k-1}{2} \quad \rightarrow \quad H_{\text{out}} = H.$$

EXACT SIMPLE CASE

odd kernel, unit stride, equal padding on both sides

FRAMEWORK GENERALIZATION

even kernels or $s > 1$ may require asymmetric padding and a stated output convention

“Same” padding is a restricted promise

D DERIVATION

For odd k and stride $s = 1$, symmetric

$$p = \frac{k-1}{2} \quad \rightarrow \quad H_{\text{out}} = H.$$

EXACT SIMPLE CASE

odd kernel, unit stride, equal padding on both sides

FRAMEWORK GENERALIZATION

even kernels or $s > 1$ may require asymmetric padding and a stated output convention

Padding preserves shape; it does not create new scene evidence. Near a border, some receptive-field entries are padding.

Floor is not optional

Q QUESTION

Let $H = 8, k = 3, p = 0$, and $s = 2$.

What are the legal starts and output length?

A. $0, 2, 4, 6$ and $H_{\text{out}} = 4$

B. $0, 2, 4$ and $H_{\text{out}} = 3$

C. $0, 1, 2, 3, 4, 5$ and $H_{\text{out}} = 6$

D. $0, 3, 6$ and $H_{\text{out}} = 3$

Floor is not optional

Q QUESTION

Let $H = 8, k = 3, p = 0$, and $s = 2$.

What are the legal starts and output length?

A. $0, 2, 4, 6$ and $H_{\text{out}} = 4$

B. $0, 2, 4$ and $H_{\text{out}} = 3$

C. $0, 1, 2, 3, 4, 5$ and $H_{\text{out}} = 6$

D. $0, 3, 6$ and $H_{\text{out}} = 3$

> Check whether the length-3 kernel fits completely at each proposed start.

Answer: start 6 would require an index outside the input

A ANSWER

Correct: B — legal starts $0, 2, 4$; $H_{\text{out}} = 3$

Why: $\left\lfloor \frac{8-3}{2} \right\rfloor + 1 = \lfloor 2.5 \rfloor + 1 = 3$. A start at 6 would use indices $6, 7, 8$, but index 8 is outside a length-8 input.

Answer: start 6 would require an index outside the input

A ANSWER

Correct: B — legal starts $0, 2, 4$; $H_{\text{out}} = 3$

Why: $\left\lfloor \frac{8-3}{2} \right\rfloor + 1 = \lfloor 2.5 \rfloor + 1 = 3$. A start at 6 would use indices $6, 7, 8$, but index 8 is outside a length-8 input.

List legal starts when the floor formula feels opaque.

Shape arithmetic and parameter arithmetic answer different questions

D DERIVATION

question	depends on	does not depend on
output shape	$H, W, k, p, s, C_{\text{out}}$	number of training examples
parameter count	$k_h, k_w, C_{\text{in}}, C_{\text{out}}$ and bias choice	H, W, p, s
convolution MACs	output positions and kernel volume	bias count in our convention

Shape arithmetic and parameter arithmetic answer different questions

D DERIVATION

question	depends on	does not depend on
output shape	$H, W, k, p, s, C_{\text{out}}$	number of training examples
parameter count	$k_h, k_w, C_{\text{in}}, C_{\text{out}}$ and bias choice	H, W, p, s
convolution MACs	output positions and kernel volume	bias count in our convention

> Our MAC ledger counts one multiply-accumulate per kernel weight per output position and excludes bias adds, pooling, and activation costs.

Sharing moves responses; pooling discards location

Predict what a one-step shift does

Q QUESTION

Use $x = (0, 1, 1, 0, 0, 0)$ and $k = (-1, 0, 1)$ with valid cross-correlation.

x

k

Predict what a one-step shift does

Q QUESTION

Use $x = (0, 1, 1, 0, 0, 0)$ and $k = (-1, 0, 1)$ with valid cross-correlation.

x

k

First compute $f(x)$. Then prepend zero and drop the last input value to obtain $T_1 x$.

Predict what a one-step shift does

Q QUESTION

Use $x = (0, 1, 1, 0, 0, 0)$ and $k = (-1, 0, 1)$ with valid cross-correlation.

x

k

First compute $f(x)$. Then prepend zero and drop the last input value to obtain $T_1 x$.

> Will $f(T_1 x)$ equal the zero-filled shift $T_1 f(x)$ at all four displayed output positions?

Answer: the common interior shifts; the cropped boundary differs

A ANSWER

quantity	exact values
x	$0, 1, 1, 0, 0, 0$
$f(x)$	$1, -1, -1, 0$
$T_1 x$	$0, 0, 1, 1, 0, 0$
$f(T_1 x)$	$1, 1, -1, -1$
$T_1 f(x)$	$0, 1, -1, -1$

Answer: the common interior shifts; the cropped boundary differs

A ANSWER

quantity	exact values
x	$0, 1, 1, 0, 0, 0$
$f(x)$	$1, -1, -1, 0$
$T_1 x$	$0, 0, 1, 1, 0, 0$
$f(T_1 x)$	$1, 1, -1, -1$
$T_1 f(x)$	$0, 1, -1, -1$

Positions 1–3 agree: $(1, -1, -1)$. Position 0 differs because valid cropping removed the output that would have shifted in from the left.

Answer: the common interior shifts; the cropped boundary differs

A ANSWER

quantity	exact values
x	$0, 1, 1, 0, 0, 0$
$f(x)$	$1, -1, -1, 0$
$T_1 x$	$0, 0, 1, 1, 0, 0$
$f(T_1 x)$	$1, 1, -1, -1$
$T_1 f(x)$	$0, 1, -1, -1$

Positions 1–3 agree: $(1, -1, -1)$. Position 0 differs because valid cropping removed the output that would have shifted in from the left.

The numerical evidence demonstrates interior equivariance and the finite-boundary caveat—both at once.

State the equivariance claim with its conditions

D DERIVATION

On an infinite grid, shared stride-1 cross-correlation commutes with integer translation:

State the equivariance claim with its conditions

D DERIVATION

On an infinite grid, shared stride-1 cross-correlation commutes with integer translation:

$$f(T_\Delta X) = T_\Delta f(X).$$

State the equivariance claim with its conditions

D DERIVATION

On an infinite grid, shared stride-1 cross-correlation commutes with integer translation:

$$f(T_\Delta X) = T_\Delta f(X).$$

BOUNDARY	STRIDE	POINTWISE RELU
crop and padding change what enters from outside	$s > 1$ samples one phase; a one-pixel shift changes phase	preserves equivariance away from the same boundaries

State the equivariance claim with its conditions

D DERIVATION

On an infinite grid, shared stride-1 cross-correlation commutes with integer translation:

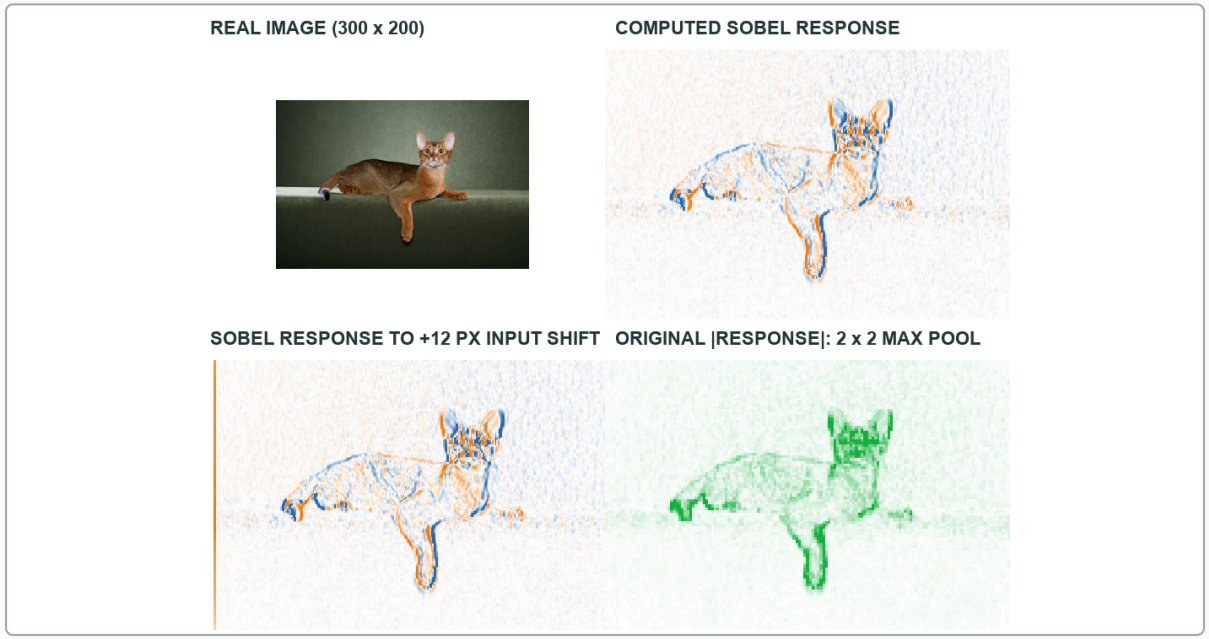
$$f(T_\Delta X) = T_\Delta f(X).$$

BOUNDARY	STRIDE	POINTWISE RELU
crop and padding change what enters from outside	$s > 1$ samples one phase; a one-pixel shift changes phase	preserves equivariance away from the same boundaries

> A stack with total stride j is naturally aligned to translations by multiples of j; arbitrary one-pixel shifts need not commute exactly.

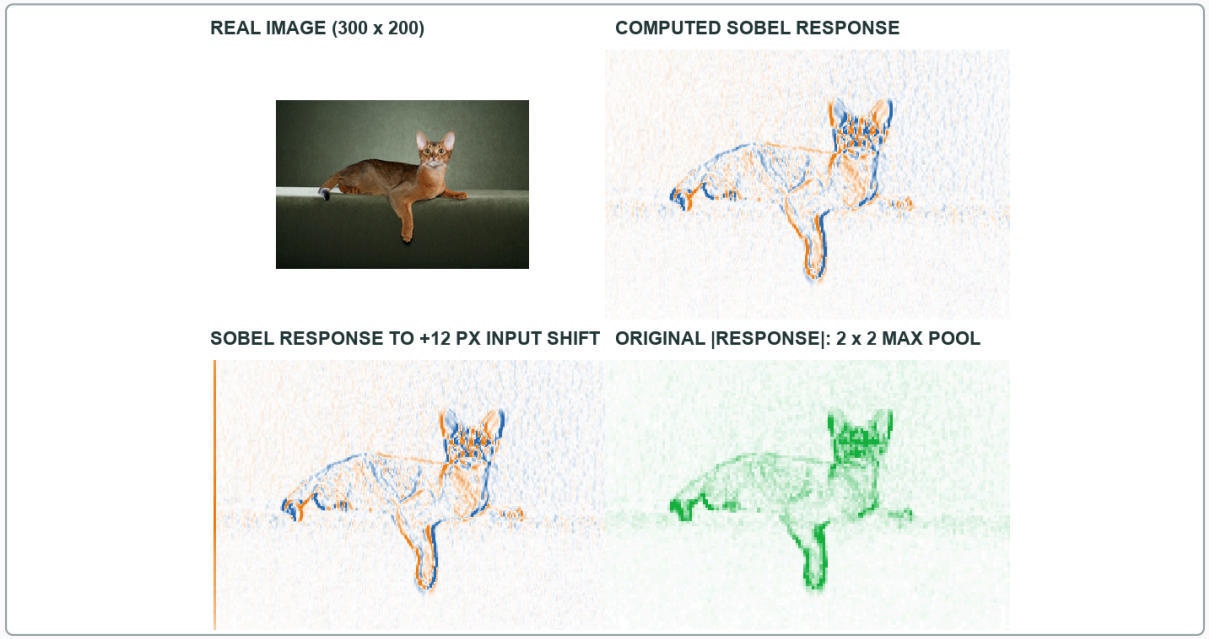
The same claims are visible on computed real-image responses

V VISUAL



The same claims are visible on computed real-image responses

V VISUAL



Two branches share the same observed image: Sobel(original) versus Sobel(zero-filled $+12$ px shifted input); separately, pool $|\mathrm{Sobel}(\mathrm{original})|$ with 2×2, stride 2. Blue = negative, white = zero, orange = positive on one shared clipped p99 scale; green shows pooled magnitude on a clipped p1–p99 scale. Deterministic computations—not learned features or predictions.

Pool the same response map

D DERIVATION

RUNNING CASE · POOLING Apply 2×2 max pooling with stride 1 to the exact Y already computed.

$Y : 3 \times 3$

Pool the same response map

D DERIVATION

RUNNING CASE · POOLING Apply 2×2 max pooling with stride 1 to the exact Y already computed.

$Y: 3 \times 3$

−3	−3	−3
−3	−3	−3
3	3	3

$\rightarrow$

max pool: 2×2

−3	−3
3	3

$$P = \begin{pmatrix} -3 & -3 \\ 3 & 3 \end{pmatrix}$$

Pool the same response map

D DERIVATION

RUNNING CASE · POOLING Apply 2×2 max pooling with stride 1 to the exact Y already computed.

$Y: 3 \times 3$

−3	−3	−3
−3	−3	−3
3	3	3

→

max pool: 2×2

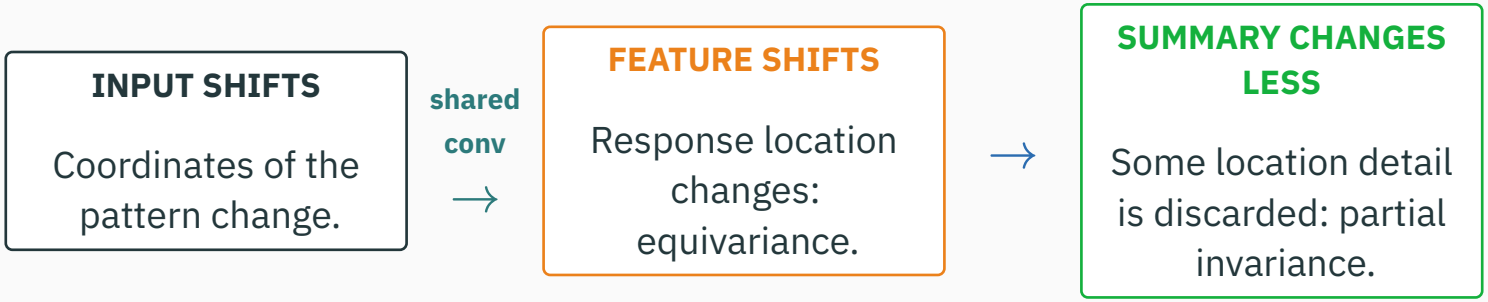
−3	−3
3	3

$$P = \begin{pmatrix} -3 & -3 \\ 3 & 3 \end{pmatrix}$$

Pooling learns no weights: it summarizes a window and deliberately loses some spatial detail.

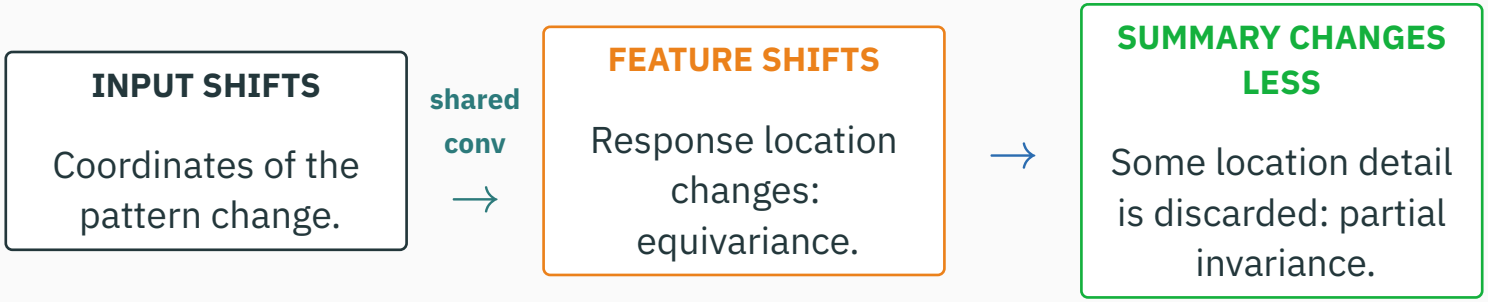
Equivariance and invariance are different promises

V VISUAL



Equivariance and invariance are different promises

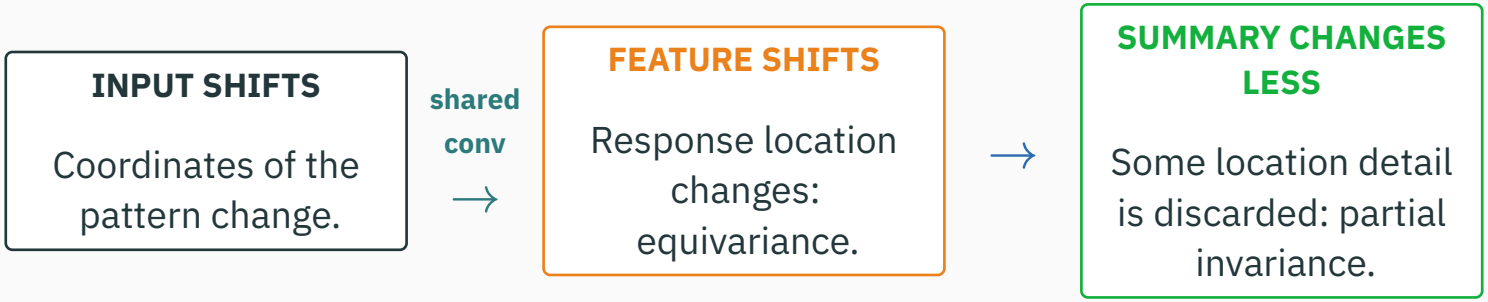
V VISUAL



Global average pooling maps $H \times W \times C \to C$. For our pooled anchor, $\frac{-3-3+3+3}{4} = 0$.

Equivariance and invariance are different promises

V VISUAL



Global average pooling maps $H \times W \times C \to C$. For our pooled anchor, $\frac{-3-3+3+3}{4} = 0$.

Pooling and augmentation can promote robustness; neither guarantees invariance to every translation.

Checkpoint: what does max pooling guarantee?

Q QUESTION

Choose the strongest defensible statement.

A. Max pooling makes every CNN exactly translation invariant

B. Max pooling preserves the precise position of every activation

C. A max is unchanged by rearrangements within one fixed window, but window boundaries and stride still matter

D. Pooling learns one detector per window

Checkpoint: what does max pooling guarantee?

Q QUESTION

Choose the strongest defensible statement.

A. Max pooling makes every CNN exactly translation invariant

B. Max pooling preserves the precise position of every activation

C. A max is unchanged by rearrangements within one fixed window, but window boundaries and stride still matter

D. Pooling learns one detector per window

> Commit before the answer. Separate a local window fact from a whole-network guarantee.

Answer: local robustness is conditional

A ANSWER

Correct: C — the max is unchanged by rearrangements within one fixed window

Why: Crossing a window boundary can change two pooled outputs, and strided pooling changes the sampling phase. The layer also discards the winning location.

Answer: local robustness is conditional

A ANSWER

Correct: C — the max is unchanged by rearrangements within one fixed window

Why: Crossing a window boundary can change two pooled outputs, and strided pooling changes the sampling phase. The layer also discards the winning location.

Use “partial invariance” or “local robustness,” not “pooling guarantees invariance.”

Compose local questions into a classifier

Derive the receptive-field recursion

D DERIVATION

Assume dilation 1. A size-k_ℓ kernel spans $k_\ell - 1$ gaps from the previous layer, and each gap is $j_{\ell-1}$ input pixels.

Derive the receptive-field recursion

D DERIVATION

Assume dilation 1. A size-k_ℓ kernel spans $k_\ell - 1$ gaps from the previous layer, and each gap is $j_{\ell-1}$ input pixels.

$$r_\ell = r_{\ell-1} + (k_\ell - 1) j_{\ell-1}.$$

Derive the receptive-field recursion

D DERIVATION

Assume dilation 1. A size-k_ℓ kernel spans $k_\ell - 1$ gaps from the previous layer, and each gap is $j_{\ell-1}$ input pixels.

$$r_\ell = r_{\ell-1} + (k_\ell - 1) j_{\ell-1}.$$

Stride changes the spacing between adjacent new units:

$$j_\ell = j_{\ell-1} s_\ell.$$

Derive the receptive-field recursion

D DERIVATION

Assume dilation 1. A size-k_ℓ kernel spans $k_\ell - 1$ gaps from the previous layer, and each gap is $j_{\ell-1}$ input pixels.

$$r_\ell = r_{\ell-1} + (k_\ell - 1) j_{\ell-1}.$$

Stride changes the spacing between adjacent new units:

$$j_\ell = j_{\ell-1} s_\ell.$$

> With dilation d_ℓ, replace $(k_\ell - 1)$ by $(k_\ell - 1) d_\ell$. Padding changes boundary alignment, not this theoretical size.

The running case reaches $r = 4$ after convolution and pooling

D DERIVATION

RUNNING CASE · RECEPTIVE FIELD The first 3×3 response uses the anchor kernel; a following 2×2, stride-2 pool combines adjacent responses.

layer	k	s	r	j
input	1	1	1	1
shared conv	3	1		
max pool	2	2		
next conv	3	1		

The running case reaches $r = 4$ after convolution and pooling

D DERIVATION

RUNNING CASE · RECEPTIVE FIELD The first 3×3 response uses the anchor kernel; a following 2×2, stride-2 pool combines adjacent responses.

layer	k	s	r	j
input	1	1	1	1
shared conv	3	1	3	1
max pool	2	2		
next conv	3	1		

The running case reaches $r = 4$ after convolution and pooling

D DERIVATION

RUNNING CASE · RECEPTIVE FIELD The first 3×3 response uses the anchor kernel; a following 2×2, stride-2 pool combines adjacent responses.

layer	k	s	r	j
input	1	1	1	1
shared conv	3	1	3	1
max pool	2	2	4	2
next conv	3	1		

The running case reaches $r = 4$ after convolution and pooling

D DERIVATION

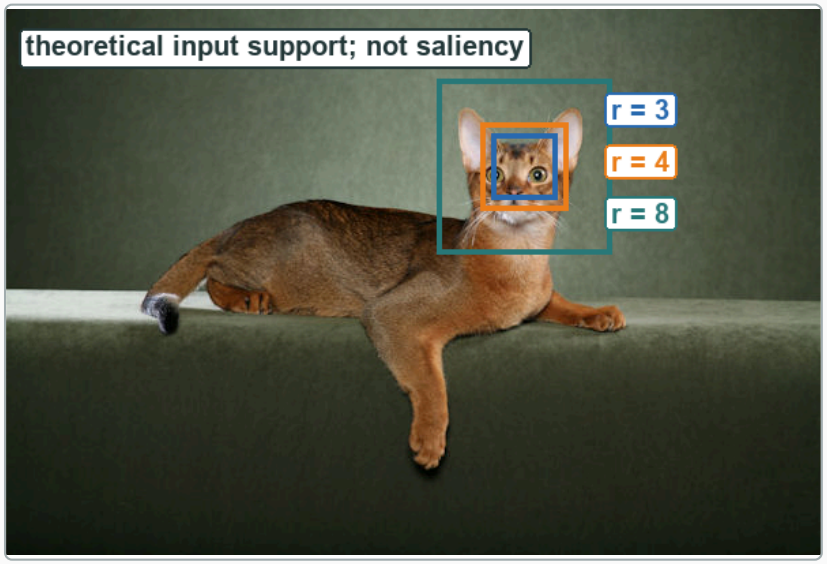
RUNNING CASE · RECEPTIVE FIELD The first 3×3 response uses the anchor kernel; a following 2×2, stride-2 pool combines adjacent responses.

layer	k	s	r	j
input	1	1	1	1
shared conv	3	1	3	1
max pool	2	2	4	2
next conv	3	1	8	2

The first pooled unit can depend on a 4×4 anchor region; the next conv expands context to 8×8.

Receptive-field boxes show support, not importance

V VISUAL



$r = 3$

one local convolution

$r = 4$

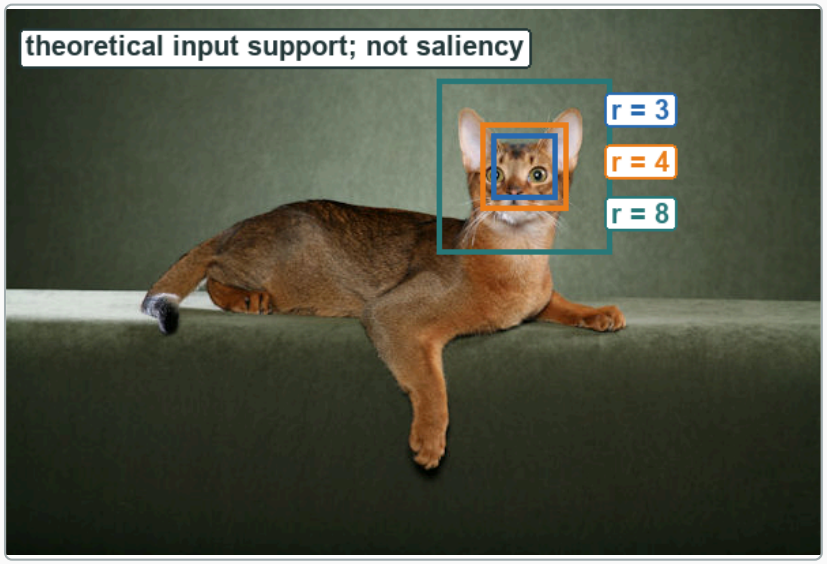
then 2×2 pooling

$r = 8$

then the next 3×3 convolution

Receptive-field boxes show support, not importance

V VISUAL



$r = 3$

one local convolution

$r = 4$

then 2×2 pooling

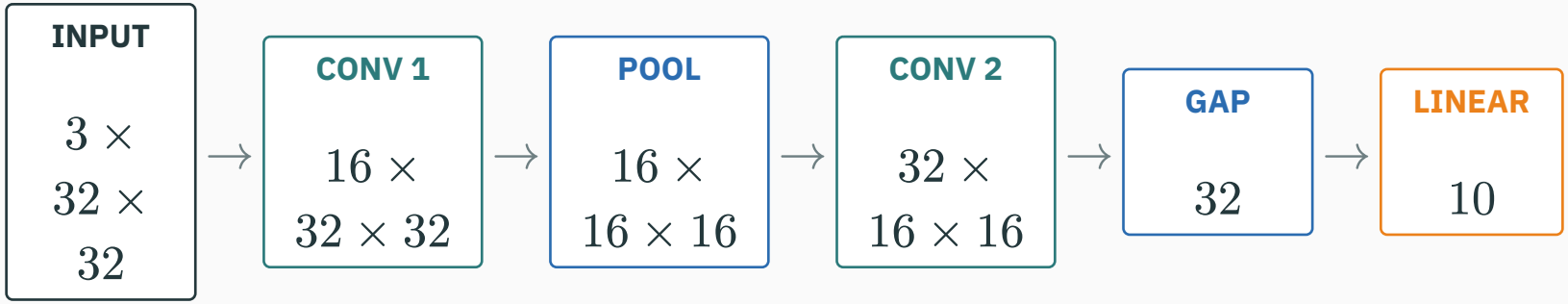
$r = 8$

then the next 3×3 convolution

Track r_ℓ = input-support width and j_ℓ = input-pixel spacing; $r_0 = j_0 = 1$. These boxes are not saliency, attention, or proof the model used the face.

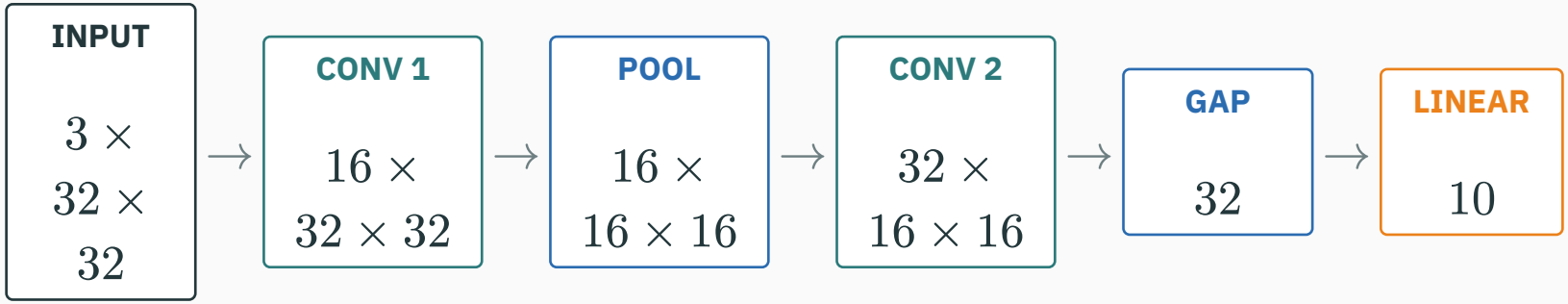
Scale the rule into one 32×32 classifier

V VISUAL



Scale the rule into one 32×32 classifier

V VISUAL



BOARD

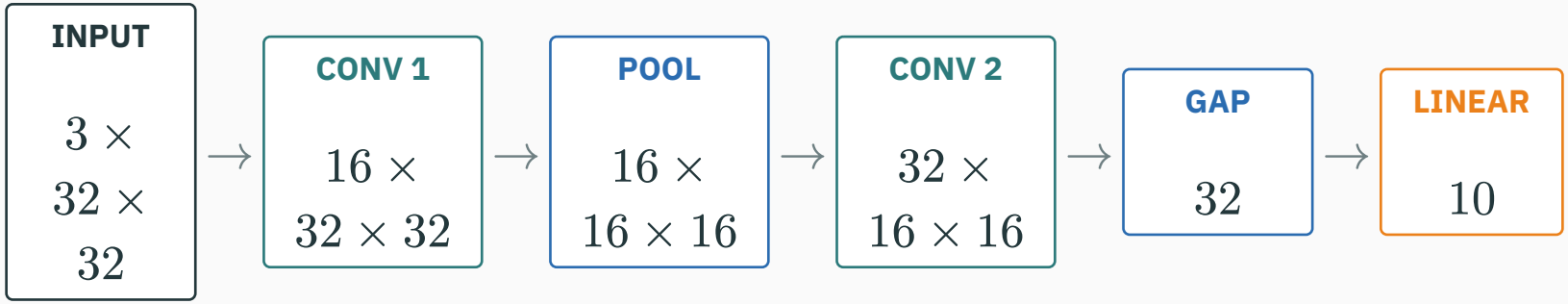
$H \times W \times C$; kernel $k_h \times k_w \times C_{\text{in}} \times C_{\text{out}}$

PYTORCH

batch $N \times C \times H \times W$; weights $C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w$

Scale the rule into one 32×32 classifier

V VISUAL



BOARD

$H \times W \times C$; kernel $k_h \times k_w \times C_{\text{in}} \times C_{\text{out}}$

PYTORCH

batch $N \times C \times H \times W$; weights $C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w$

Axis order changes between notation and code; the represented operation does not.

Ledger row 1: same-padded convolution

D DERIVATION

Input: $N \times 3 \times 32 \times 32$. Layer: $3 \times 3, p = 1, s = 1, C_{\text{out}} = 16$.

Ledger row 1: same-padded convolution

D DERIVATION

Input: $N \times 3 \times 32 \times 32$. Layer: $3 \times 3, p = 1, s = 1, C_{\text{out}} = 16$.

OUTPUT	PARAMETERS	RF / JUMP
$N \times 16 \times 32 \times 32$	$3 \cdot 3 \cdot 3 \cdot 16 + 16 = 448$	$r = 3, j = 1$

Ledger row 1: same-padded convolution

D DERIVATION

Input: $N \times 3 \times 32 \times 32$. Layer: $3 \times 3, p = 1, s = 1, C_{\text{out}} = 16$.

OUTPUT	PARAMETERS	RF / JUMP
$N \times 16 \times 32 \times 32$	$3 \cdot 3 \cdot 3 \cdot 16 + 16 = 448$	$r = 3, j = 1$

$$\text{MACs} = 32 \cdot 32 \cdot 16 \cdot (3 \cdot 3 \cdot 3) = 442{,}368.$$

Ledger row 1: same-padded convolution

D DERIVATION

Input: $N \times 3 \times 32 \times 32$. Layer: $3 \times 3, p = 1, s = 1, C_{\text{out}} = 16$.

OUTPUT	PARAMETERS	RF / JUMP
$N \times 16 \times 32 \times 32$	$3 \cdot 3 \cdot 3 \cdot 16 + 16 = 448$	$r = 3, j = 1$

$$\text{MACs} = 32 \cdot 32 \cdot 16 \cdot (3 \cdot 3 \cdot 3) = 442{,}368.$$

> Each of 16 kernels is reused at all $32 \cdot 32$ output positions.

Ledger rows 2–3: pool, then learn 32 feature types

D DERIVATION

layer	output	parameters	r, j
2×2 max pool, $s = 2$	$N \times 16 \times 16 \times 16$	0	
3×3 conv, $p = 1$	$N \times 32 \times 16 \times 16$		

Ledger rows 2–3: pool, then learn 32 feature types

D DERIVATION

layer	output	parameters	r, j
2×2 max pool, $s = 2$	$N \times 16 \times 16 \times 16$	0	$4, 2$
3×3 conv, $p = 1$	$N \times 32 \times 16 \times 16$		

Ledger rows 2–3: pool, then learn 32 feature types

D DERIVATION

layer	output	parameters	r, j
2×2 max pool, $s = 2$	$N \times 16 \times 16 \times 16$	0	$4, 2$
3×3 conv, $p = 1$	$N \times 32 \times 16 \times 16$	$3 \cdot 3 \cdot 16 \cdot 32 + 32 = 4{,}640$	$8, 2$

$$\text{conv 2 MACs} = 16 \cdot 16 \cdot 32 \cdot (3 \cdot 3 \cdot 16) = 1{,}179{,}648.$$

Ledger rows 2–3: pool, then learn 32 feature types

D DERIVATION

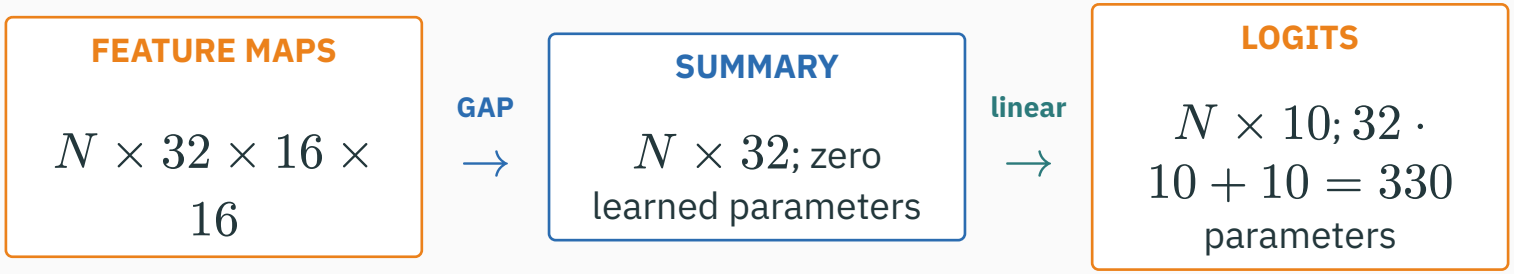
layer	output	parameters	r, j
2×2 max pool, $s = 2$	$N \times 16 \times 16 \times 16$	0	$4, 2$
3×3 conv, $p = 1$	$N \times 32 \times 16 \times 16$	$3 \cdot 3 \cdot 16 \cdot 32 + 32 = 4{,}640$	$8, 2$

$$\text{conv 2 MACs} = 16 \cdot 16 \cdot 32 \cdot (3 \cdot 3 \cdot 16) = 1{,}179{,}648.$$

Stride does not add learned parameters; it changes spatial size, compute, jump, and downstream receptive field.

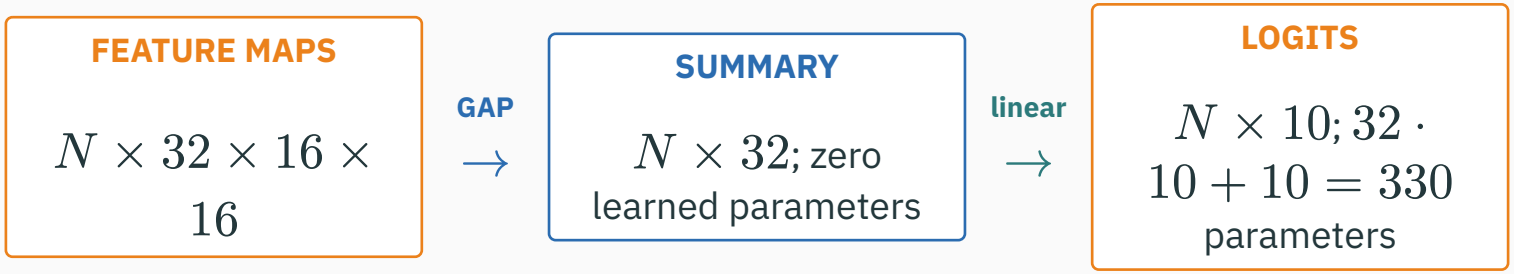
Ledger rows 4–5: global average, then ten logits

D DERIVATION



Ledger rows 4–5: global average, then ten logits

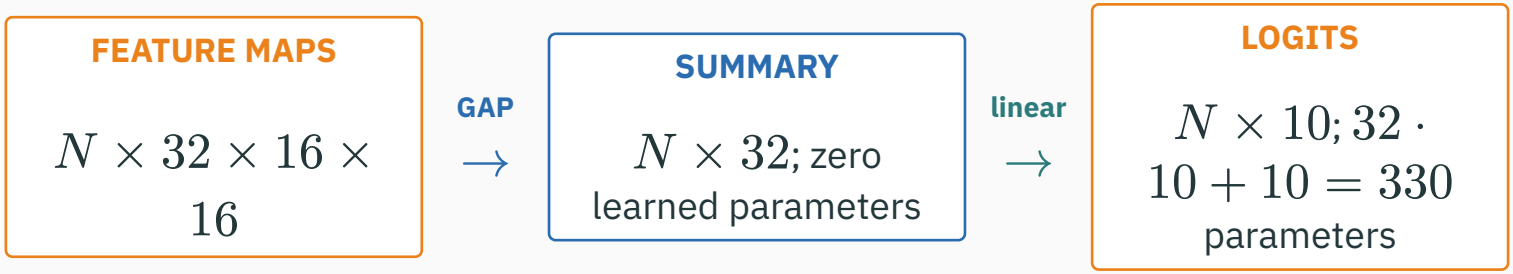
D DERIVATION



$$\text{linear MACs} = 32 \cdot 10 = 320.$$

Ledger rows 4–5: global average, then ten logits

D DERIVATION



$$\text{linear MACs} = 32 \cdot 10 = 320.$$

> Global average pooling removes the fixed-resolution requirement that a flattened dense head would introduce.

The complete ledger separates capacity, compute, and context

D DERIVATION

layer	output	parameters	MACs	r, j
input	$N \times 3 \times 32 \times 32$	0	0	$1, 1$
conv 1	$N \times 16 \times 32 \times 32$	448	$442{,}368$	$3, 1$
pool	$N \times 16 \times 16 \times 16$	0	not counted	$4, 2$
conv 2	$N \times 32 \times 16 \times 16$	$4{,}640$	$1{,}179{,}648$	$8, 2$
GAP	$N \times 32$	0	not counted	global
linear	$N \times 10$	330	320	global
total		$5{,}418$	$1{,}622{,}336$	

The complete ledger separates capacity, compute, and context

D DERIVATION

layer	output	parameters	MACs	r, j
input	$N \times 3 \times 32 \times 32$	0	0	$1, 1$
conv 1	$N \times 16 \times 32 \times 32$	448	$442{,}368$	$3, 1$
pool	$N \times 16 \times 16 \times 16$	0	not counted	$4, 2$
conv 2	$N \times 32 \times 16 \times 16$	$4{,}640$	$1{,}179{,}648$	$8, 2$
GAP	$N \times 32$	0	not counted	global
linear	$N \times 10$	330	320	global
total		$5{,}418$	$1{,}622{,}336$	

MAC convention: kernel multiplies only. Bias adds, activations, pooling, and GAP are excluded and stated explicitly.

Checkpoint: double height and width

Q QUESTION

Run the same fully convolutional classifier on 64×64 RGB input.

What necessarily changes?

A. Convolution parameters quadruple; MACs stay fixed

B. Parameters stay fixed; convolution activations and MACs grow by about $4\times$

C. The number of logits becomes 40

D. The pre-GAP receptive-field width doubles automatically

Checkpoint: double height and width

Q QUESTION

Run the same fully convolutional classifier on 64×64 RGB input.

What necessarily changes?

A. Convolution parameters quadruple; MACs stay fixed

B. Parameters stay fixed; convolution activations and MACs grow by about $4\times$

C. The number of logits becomes 40

D. The pre-GAP receptive-field width doubles automatically

> Hold kernel sizes, channels, padding, and stride fixed.

Answer: reuse keeps parameters fixed, not compute

A ANSWER

Correct: B — parameters stay fixed; convolution activations and MACs grow by about $4\times$

Why: Doubling both spatial axes creates four times as many output positions. Kernel banks and the GAP-to-linear head are unchanged.

Answer: reuse keeps parameters fixed, not compute

A ANSWER

Correct: B — parameters stay fixed; convolution activations and MACs grow by about $4\times$

Why: Doubling both spatial axes creates four times as many output positions. Kernel banks and the GAP-to-linear head are unchanged.

exact MACs: $4(442{,}368 + 1{,}179{,}648) + 320 = 6{,}488{,}384.$

Answer: reuse keeps parameters fixed, not compute

A ANSWER

Correct: B — parameters stay fixed; convolution activations and MACs grow by about $4\times$

Why: Doubling both spatial axes creates four times as many output positions. Kernel banks and the GAP-to-linear head are unchanged.

exact MACs: $4(442{,}368 + 1{,}179{,}648) + 320 = 6{,}488{,}384.$

Parameter count measures stored capacity; MAC count measures work at a chosen resolution.

Shared gradients and implementation checks

Forward example: one kernel is reused twice

D DERIVATION

Let $x = (2, 1, 3)$, $k = (0.5, -1)$, and $b = 0$.

Forward example: one kernel is reused twice

D DERIVATION

Let $x = (2, 1, 3)$, $k = (0.5, -1)$, and $b = 0$.

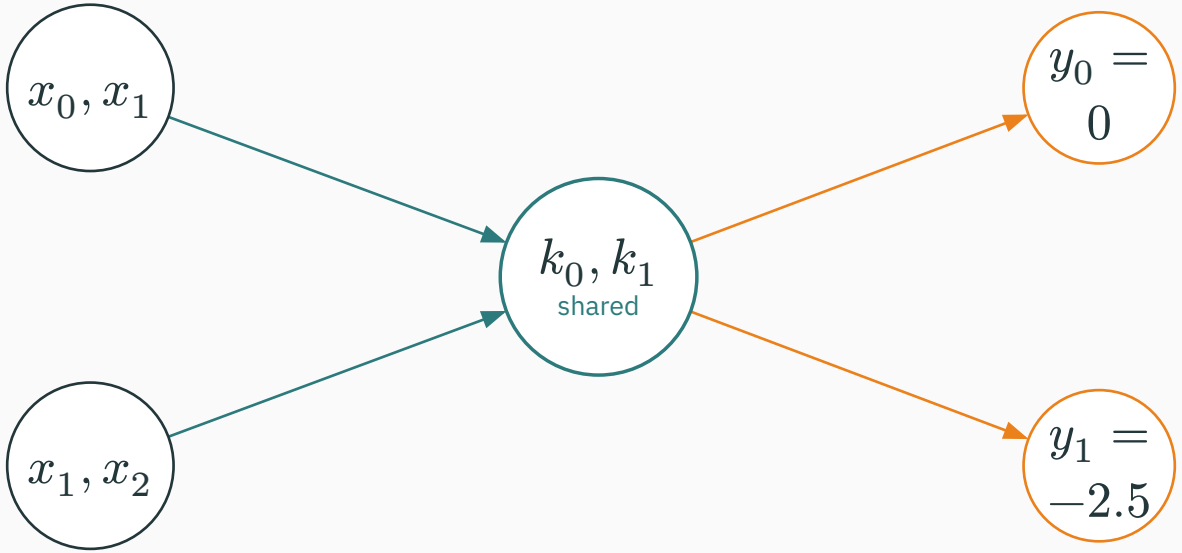
$$y_0 = 0.5(2) - 1(1) = 0, \quad y_1 = 0.5(1) - 1(3) = -2.5.$$

Forward example: one kernel is reused twice

D DERIVATION

Let $x = (2, 1, 3)$, $k = (0.5, -1)$, and $b = 0$.

$$y_0 = 0.5(2) - 1(1) = 0, \quad y_1 = 0.5(1) - 1(3) = -2.5.$$

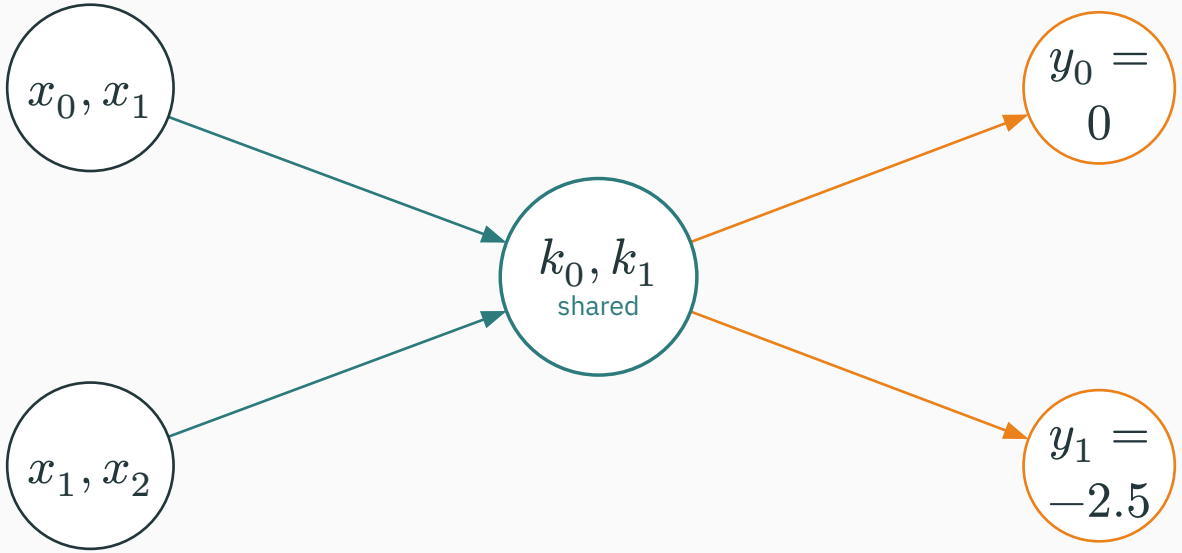


Forward example: one kernel is reused twice

D DERIVATION

Let $x = (2, 1, 3)$, $k = (0.5, -1)$, and $b = 0$.

$$y_0 = 0.5(2) - 1(1) = 0, \quad y_1 = 0.5(1) - 1(3) = -2.5.$$



One parameter value participates in multiple output equations.

Predict the shared kernel gradient

Q QUESTION

An upstream gradient $g = \left(\partial \frac{\mathcal{L}}{\partial} y_0, \partial \frac{\mathcal{L}}{\partial} y_1\right) = (1, 2)$ arrives.

Does $\partial \frac{\mathcal{L}}{\partial} k_0$ use the first patch, the second patch, or both?

Predict the shared kernel gradient

Q QUESTION

An upstream gradient $g = \left(\partial \frac{\mathcal{L}}{\partial} y_0, \partial \frac{\mathcal{L}}{\partial} y_1\right) = (1, 2)$ arrives.

Does $\partial \frac{\mathcal{L}}{\partial} k_0$ use the first patch, the second patch, or both?

k_0 USES	k_1 USES
x_0 in y_0 and x_1 in y_1	x_1 in y_0 and x_2 in y_1

Predict the shared kernel gradient

Q QUESTION

An upstream gradient $g = \left(\partial \frac{\mathcal{L}}{\partial} y_0, \partial \frac{\mathcal{L}}{\partial} y_1\right) = (1, 2)$ arrives.

Does $\partial \frac{\mathcal{L}}{\partial} k_0$ use the first patch, the second patch, or both?

k_0 USES	k_1 USES
x_0 in y_0 and x_1 in y_1	x_1 in y_0 and x_2 in y_1

> Write both sums before substituting values.

Answer: every spatial use contributes

A ANSWER

Answer: every spatial use contributes

A ANSWER

$$\partial \frac{\mathcal{L}}{\partial} k_0 = g_0 x_0 + g_1 x_1 = 1(2) + 2(1) = 4.$$

Answer: every spatial use contributes

A ANSWER

$$\partial \frac{\mathcal{L}}{\partial} k_0 = g_0 x_0 + g_1 x_1 = 1(2) + 2(1) = 4.$$

$$\partial \frac{\mathcal{L}}{\partial} k_1 = g_0 x_1 + g_1 x_2 = 1(1) + 2(3) = 7.$$

$\nabla_k \mathcal{L} = (4, 7)$**: sharing ties the uses together, so their evidence accumulates.**

Overlapping patches also accumulate input gradients

D DERIVATION

Overlapping patches also accumulate input gradients

D DERIVATION

$$\partial \frac{\mathcal{L}}{\partial} x_0 = g_0 k_0 = 0.5.$$

Overlapping patches also accumulate input gradients

D DERIVATION

$$\partial \frac{\mathcal{L}}{\partial} x_0 = g_0 k_0 = 0.5.$$

$$\partial \frac{\mathcal{L}}{\partial} x_1 = g_0 k_1 + g_1 k_0 = -1 + 1 = 0.$$

Overlapping patches also accumulate input gradients

D DERIVATION

$$\partial \frac{\mathcal{L}}{\partial} x_0 = g_0 k_0 = 0.5.$$

$$\partial \frac{\mathcal{L}}{\partial} x_1 = g_0 k_1 + g_1 k_0 = -1 + 1 = 0.$$

$$\partial \frac{\mathcal{L}}{\partial} x_2 = g_1 k_1 = -2, \quad \partial \frac{\mathcal{L}}{\partial} b = g_0 + g_1 = 3.$$

$\nabla_x \mathcal{L} = (0.5, 0, -2)$ **and** $\nabla_b \mathcal{L} = 3$**. Shared use and patch overlap are different sums.**

The 2-D kernel gradient is the same sum over locations

D DERIVATION

For a batch, input channel c, output channel o, and stride s:

The 2-D kernel gradient is the same sum over locations

D DERIVATION

For a batch, input channel c, output channel o, and stride s:

$$\partial \frac{\mathcal{L}}{\partial} K[u, v, c, o] = \sum_n \sum_i \sum_j G[n, i, j, o] X[n, is + u - p, js + v - p, c].$$

The 2-D kernel gradient is the same sum over locations

D DERIVATION

For a batch, input channel c, output channel o, and stride s:

$$\partial \frac{\mathcal{L}}{\partial} K[u, v, c, o] = \sum_n \sum_i \sum_j G[n, i, j, o] X[n, is + u - p, js + v - p, c].$$

BATCH	SPACE	SHARING
sum evidence over examples n	sum every output location (i, j)	all terms update the same $K[u, v, c, o]$

The 2-D kernel gradient is the same sum over locations

D DERIVATION

For a batch, input channel c, output channel o, and stride s:

$$\partial \frac{\mathcal{L}}{\partial} K[u, v, c, o] = \sum_{n} \sum_{i} \sum_{j} G[n, i, j, o] X[n, is + u - p, js + v - p, c].$$

BATCH	SPACE	SHARING
sum evidence over examples n	sum every output location (i, j)	all terms update the same $K[u, v, c, o]$

Framework autodiff performs these accumulations; the notebook checks them against both PyTorch and finite differences.

Training state has three separate contracts

D DERIVATION

mechanism	training	validation / inference
input standardization	estimate mean/std on training data	reuse those fixed values unchanged
augmentation	random, task-valid transforms	deterministic preprocessing
Dropout	active	off
BatchNorm	batch statistics + running-stat updates	stored running statistics
autograd	record graph for updates	usually disable graph separately

Training state has three separate contracts

D DERIVATION

mechanism	training	validation / inference
input standardization	estimate mean/std on training data	reuse those fixed values unchanged
augmentation	random, task-valid transforms	deterministic preprocessing
Dropout	active	off
BatchNorm	batch statistics + running-stat updates	stored running statistics
autograd	record graph for updates	usually disable graph separately

> model.eval() changes Dropout and BatchNorm behavior; it does not itself disable gradient recording.

Diagnose CNN failures with one targeted test

V VISUAL

symptom	suspect	smallest decisive test
shape error	HWC/NCHW order or wrong C_{in}	print/assert N, C, H, W after every layer
map collapses too early	accumulated stride/pooling	extend the shape + r, j ledger
loss does not fall	data, loss, gradient, or update bug	overfit one tiny batch
same eval input changes	Dropout active or BatchNorm state moving	call eval; run two no-grad passes
train good, validation poor	generalization, split, or mode issue	return to L7: verify split, then change one lever

Diagnose CNN failures with one targeted test

V VISUAL

symptom	suspect	smallest decisive test
shape error	HWC/NCHW order or wrong C_{in}	print/assert N, C, H, W after every layer
map collapses too early	accumulated stride/pooling	extend the shape + r, j ledger
loss does not fall	data, loss, gradient, or update bug	overfit one tiny batch
same eval input changes	Dropout active or BatchNorm state moving	call eval; run two no-grad passes
train good, validation poor	generalization, split, or mode issue	return to L7: verify split, then change one lever

Do not tune the optimizer until the tensor, state, and one-batch contracts pass.

Notebook choreography: one trace, six predictions

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Use the single L08 notebook from top to bottom; do not skip directly to the PyTorch model.

Notebook choreography: one trace, six predictions

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Use the single L08 notebook from top to bottom; do not skip directly to the PyTorch model.

step	predict before run	evidence after run
1	$Y[0,0]$ and full Y	-3 and the exact 3×3 feature map
2	legal starts for each (p, s)	$3, 5, 4, 3$ output lengths
3	parameters, MACs, r, j	$5{,}418$; $1{,}622{,}336$; final $(8, 2)$
4	which shifted positions agree	interior equality; boundary mismatch
5	$\nabla_k, \nabla_x, \nabla_b$	PyTorch and finite difference agree
6	every NCHW activation shape	final $(1, 10)$ logits

Notebook choreography: one trace, six predictions

I INTERACTIVE

INTERACTIVE

↗ colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L08/01_convolution_shapes_params_from_scratch.ipynb

Use the single L08 notebook from top to bottom; do not skip directly to the PyTorch model.

step	predict before run	evidence after run
1	$Y[0,0]$ and full Y	-3 and the exact 3×3 feature map
2	legal starts for each (p, s)	$3, 5, 4, 3$ output lengths
3	parameters, MACs, r, j	$5{,}418$; $1{,}622{,}336$; final $(8, 2)$
4	which shifted positions agree	interior equality; boundary mismatch
5	$\nabla_k, \nabla_x, \nabla_b$	PyTorch and finite difference agree
6	every NCHW activation shape	final $(1, 10)$ logits

The notebook verifies one tiny SHA-checked companion image plus the exact slide numerics; it trains no model.

Close the loop

Revisit the opening commitment

A ANSWER

The position-specific dense map used $1{,}049{,}600$ parameters; the shared 3×3 map used 10.

Revisit the opening commitment

A ANSWER

The position-specific dense map used $1{,}049{,}600$ parameters; the shared 3×3 map used 10.

HOW	GAIN	GIVE UP
reuse the same nine weights and one bias at all 1024 output locations	locality, translation structure, and $104,960\times$ fewer parameters	arbitrary position-specific connectivity in this layer

Revisit the opening commitment

A ANSWER

The position-specific dense map used $1{,}049{,}600$ parameters; the shared 3×3 map used 10.

HOW	GAIN	GIVE UP
reuse the same nine weights and one bias at all 1024 output locations	locality, translation structure, and $104,960\times$ fewer parameters	arbitrary position-specific connectivity in this layer

Opening answer: B. With $p = 1, s = 1$, ten shared parameters can produce 1,024 responses because output count and parameter count are different quantities.

Retrieval: which audit is internally consistent?

Q QUESTION

Choose the only fully consistent statement.

A. Stride changes parameter count because it changes output size

B. A shared convolution is exactly translation invariant on every finite image

C. A 3×3 conv can keep shape with $p = 1, s = 1$; its parameters ignore H, W; its MACs do not

D. Global average pooling learns one weight per pixel

Retrieval: which audit is internally consistent?

Q QUESTION

Choose the only fully consistent statement.

A. Stride changes parameter count because it changes output size

B. A shared convolution is exactly translation invariant on every finite image

C. A 3×3 conv can keep shape with $p = 1, s = 1$; its parameters ignore H, W; its MACs do not

D. Global average pooling learns one weight per pixel

> Commit, then justify your choice with one formula and one caveat.

Answer: geometry, capacity, and compute stay separate

A ANSWER

Correct: C — shape can stay fixed while parameters ignore resolution and MACs scale with it

Why: $H_{\text{out}} = \left\lfloor \frac{H+2p-k}{s} \right\rfloor + 1$ controls geometry; $k_h k_w C_{\text{in}} C_{\text{out}} + C_{\text{out}}$ controls parameters; output positions multiply MACs.

Answer: geometry, capacity, and compute stay separate

A ANSWER

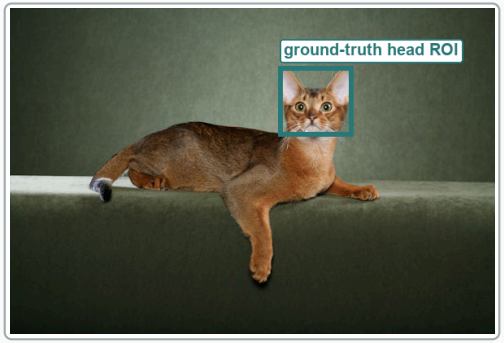
Correct: C — shape can stay fixed while parameters ignore resolution and MACs scale with it

Why: $H_{\text{out}} = \left\lfloor \frac{H+2p-k}{s} \right\rfloor + 1$ controls geometry; $k_h k_w C_{\text{in}} C_{\text{out}} + C_{\text{out}}$ controls parameters; output positions multiply MACs.

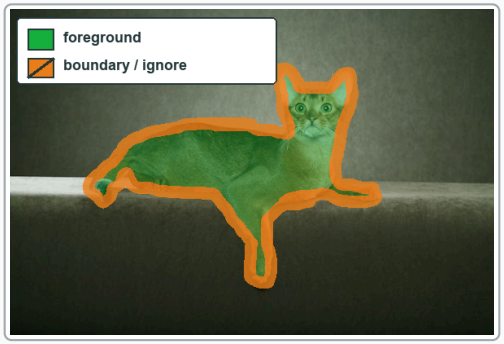
Translation equivariance is conditional; pooling is fixed; stride changes sampling, not learned parameter count.

L9 handoff: make the local pipeline deep, efficient, and reusable

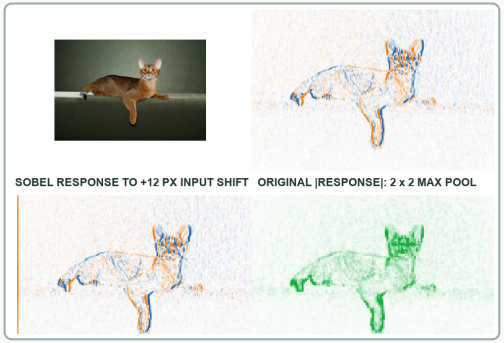
V VISUAL



GT ANNOTATION · head ROI



GT TRIMAP · foreground + boundary/ ignore

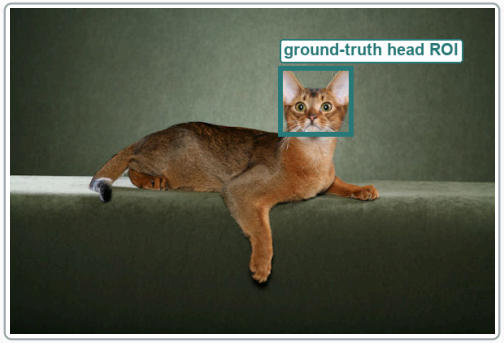


COMPUTED · declared operators

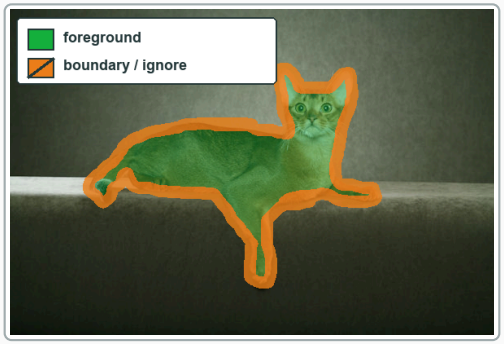
TRIMAP KEY · green = foreground · orange hatched = boundary/ignore · uncolored = background

L9 handoff: make the local pipeline deep, efficient, and reusable

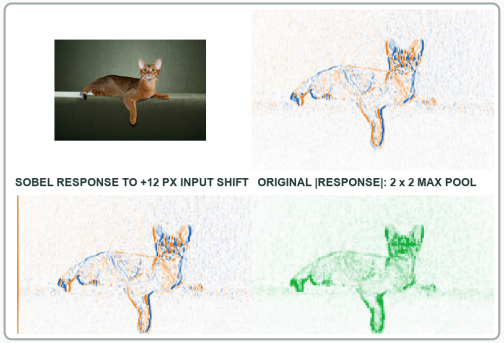
V VISUAL



GT ANNOTATION · head ROI



GT TRIMAP · foreground + boundary/ ignore



COMPUTED · declared operators

TRIMAP KEY · green = foreground · orange hatched = boundary/ignore · uncolored = background

L8 · TODAY

distinguish input, annotation, construction, and computation

NOT YET

no trained feature, saliency, detector, or classifier output

L9 · NEXT

compose and reuse learned backbones; demand measured evidence

A convolution is a linear layer that confesses its assumptions.

Local connections decide what can interact; shared weights ask the same learned question everywhere.

Efficiency and provenance

Two 3×3 layers versus one 5×5 layer

At constant width C and ignoring biases:

Two 3×3 layers versus one 5×5 layer

At constant width C and ignoring biases:

TWO 3×3

receptive field: 5×5
parameters: $2(3 \cdot 3C^2) = 18C^2$
two nonlinearities

ONE 5×5

receptive field: 5×5
parameters: $25C^2$
one nonlinearity

Two 3×3 layers versus one 5×5 layer

At constant width C and ignoring biases:

TWO 3×3

receptive field: 5×5
parameters: $2(3 \cdot 3C^2) = 18C^2$
two nonlinearities

ONE 5×5

receptive field: 5×5
parameters: $25C^2$
one nonlinearity

> The receptive-field sizes match, but the function classes are not identical: the two-layer path factorizes computation and inserts a nonlinearity.

im2col exposes matrix multiplication; sources fix conventions

For $x = (2, 1, 3)$ and kernel $k = (0.5, -1)$,

im2col exposes matrix multiplication; sources fix conventions

For $x = (2, 1, 3)$ and kernel $k = (0.5, -1)$,

$$\underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}}_{\text{patch matrix}} \quad \underbrace{\begin{pmatrix} 0.5 \\ -1 \end{pmatrix}}_{\text{shared kernel}} = \underbrace{\begin{pmatrix} 0 \\ -2.5 \end{pmatrix}}_{\text{responses}}.$$

im2col exposes matrix multiplication; sources fix conventions

For $x = (2, 1, 3)$ and kernel $k = (0.5, -1)$,

$$\underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}}_{\text{patch matrix}} \quad \underbrace{\begin{pmatrix} 0.5 \\ -1 \end{pmatrix}}_{\text{shared kernel}} = \underbrace{\begin{pmatrix} 0 \\ -2.5 \end{pmatrix}}_{\text{responses}}.$$

> im2col duplicates overlapping input values in a temporary view; it does not create distinct learned kernel copies.

im2col exposes matrix multiplication; sources fix conventions

For $x = (2, 1, 3)$ and kernel $k = (0.5, -1)$,

$$\underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}}_{\text{patch matrix}} \quad \underbrace{\begin{pmatrix} 0.5 \\ -1 \end{pmatrix}}_{\text{shared kernel}} = \underbrace{\begin{pmatrix} 0 \\ -2.5 \end{pmatrix}}_{\text{responses}}.$$

> im2col duplicates overlapping input values in a temporary view; it does not create distinct learned kernel copies.

- PyTorch Conv2d documentation — operator, shapes, groups, dilation, and framework convention.
- Dumoulin & Visin (2016), A guide to convolution arithmetic — output geometry.
- LeCun et al. (1998), Gradient-based learning applied to document recognition — classic shared-weight CNN.
- Oxford-IIIT Pet dataset — the real photo, head ROI, trimap, and CC BY-SA 4.0 provenance used throughout the CV module.
- Exact L08 Colab notebook — deterministic slide numerics, gradients, and NCHW model.

Real-image derivatives are rebuilt by shared/vision-evidence/oxford-iiit-pet/build_evidence.py; all charts, equations, grids, and diagrams remain native/vector.