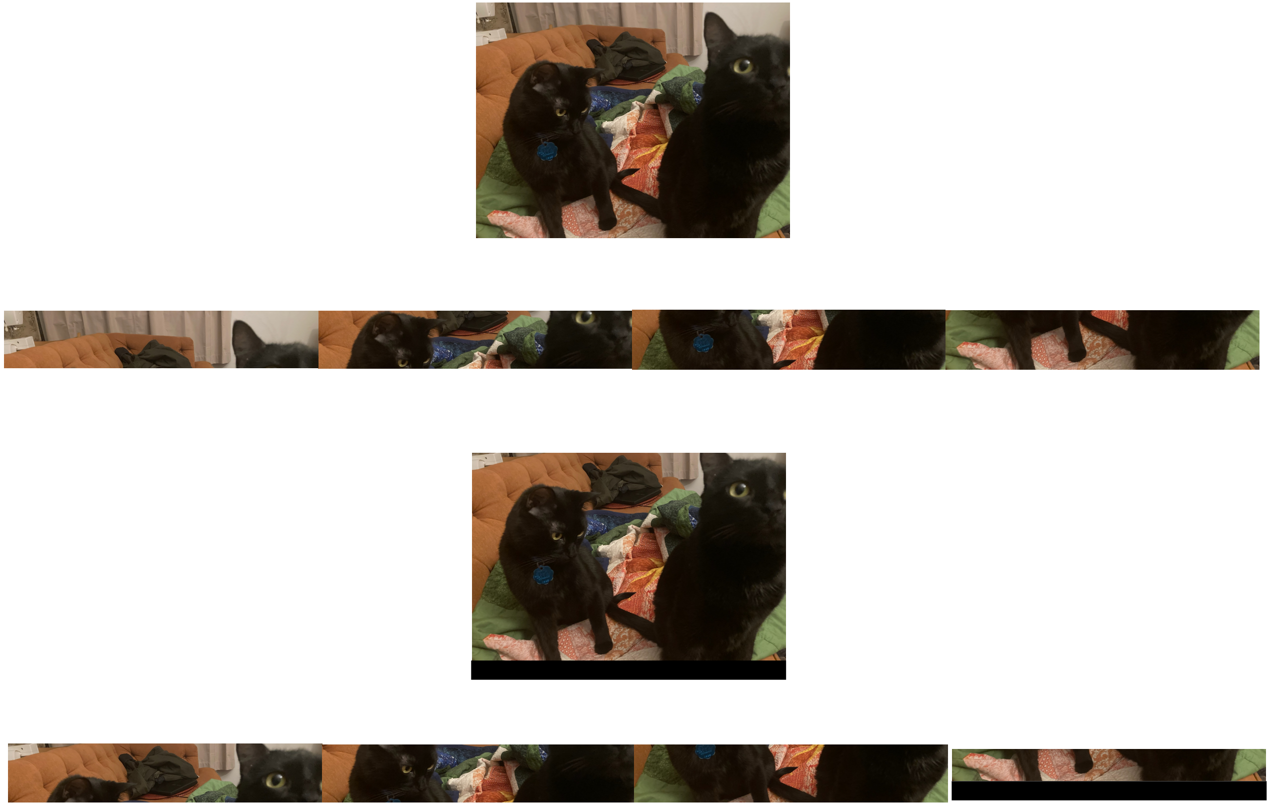
Convolution and Pooling

Image structure



Computational Inefficiency

- 128x128 image with 3 color channels: ~50,000 values

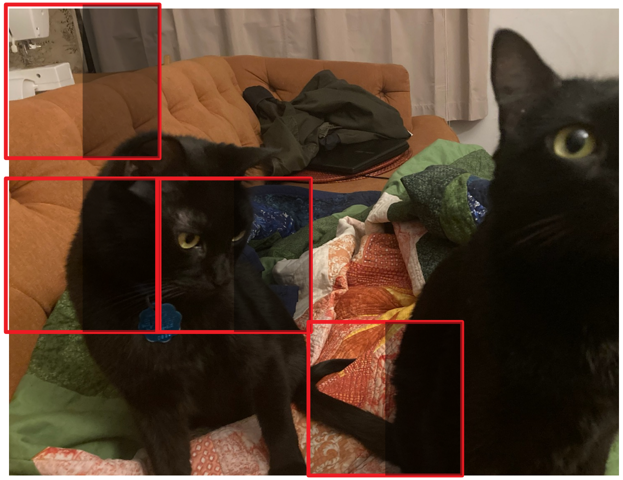
$$f(x) = \mathrm{W}\,\mathrm{x} + \mathrm{b}$$

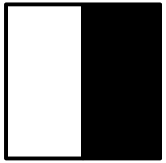
$$\mathrm{W} \in \mathbb{R}^{n \times m} \qquad \mathrm{b} \in \mathbb{R}^{n}$$

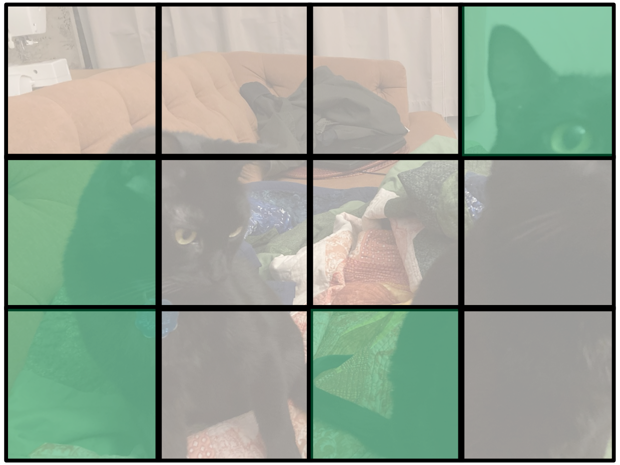
$n \times (m+1)$ parameters

For an output size of 1000, we already have 50M parameters.

Local Patterns



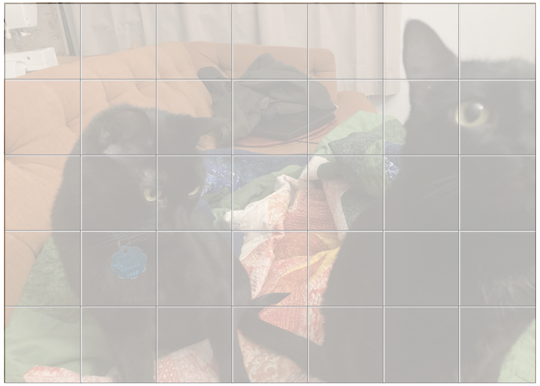




Convolution

Sliding linear transformation

Input



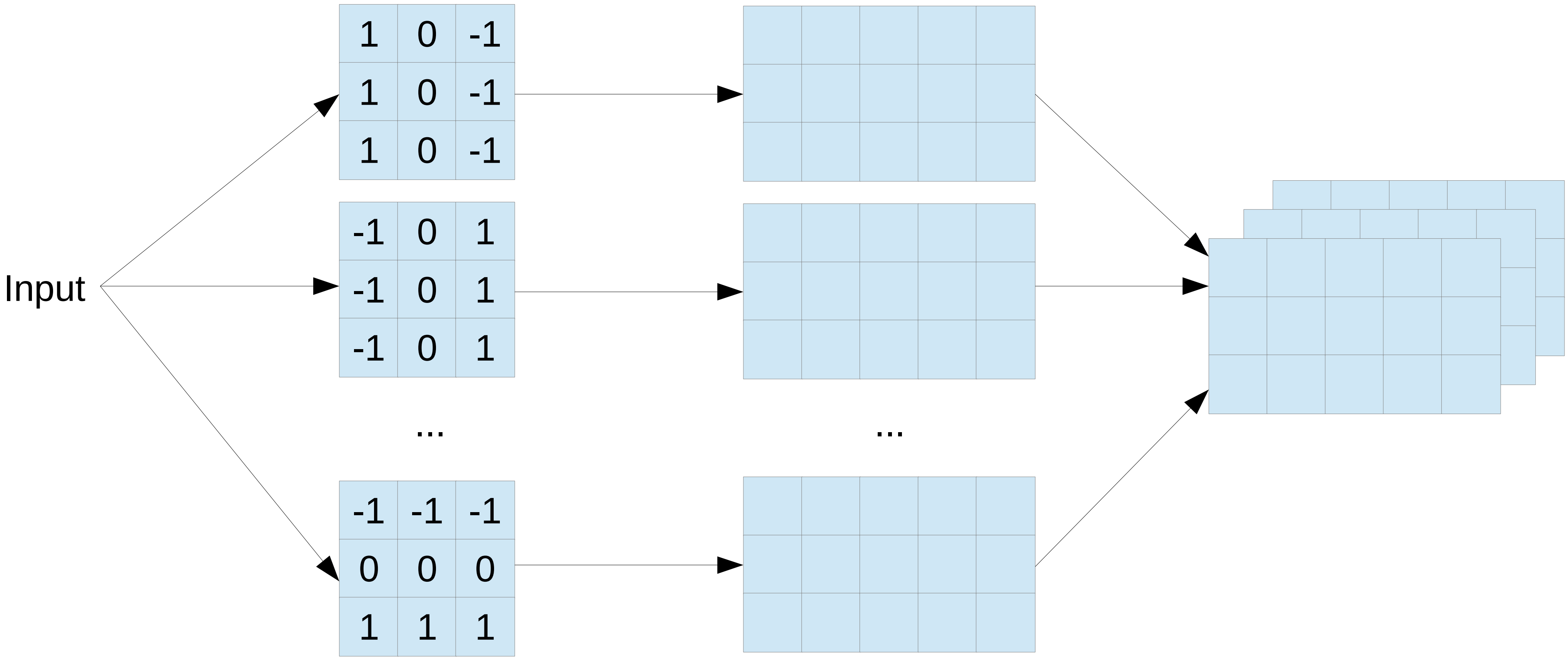
Kernel / Filter

a	b	c
d	e	f
g	h	i

1	0	-1
1	0	-1
1	0	-1

Output

Convolutional Layer



1	0	-1
1	0	-1
1	0	-1

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Convolutional Layers

Input $X \in \mathbb{R}^{C_i \times H \times W}$

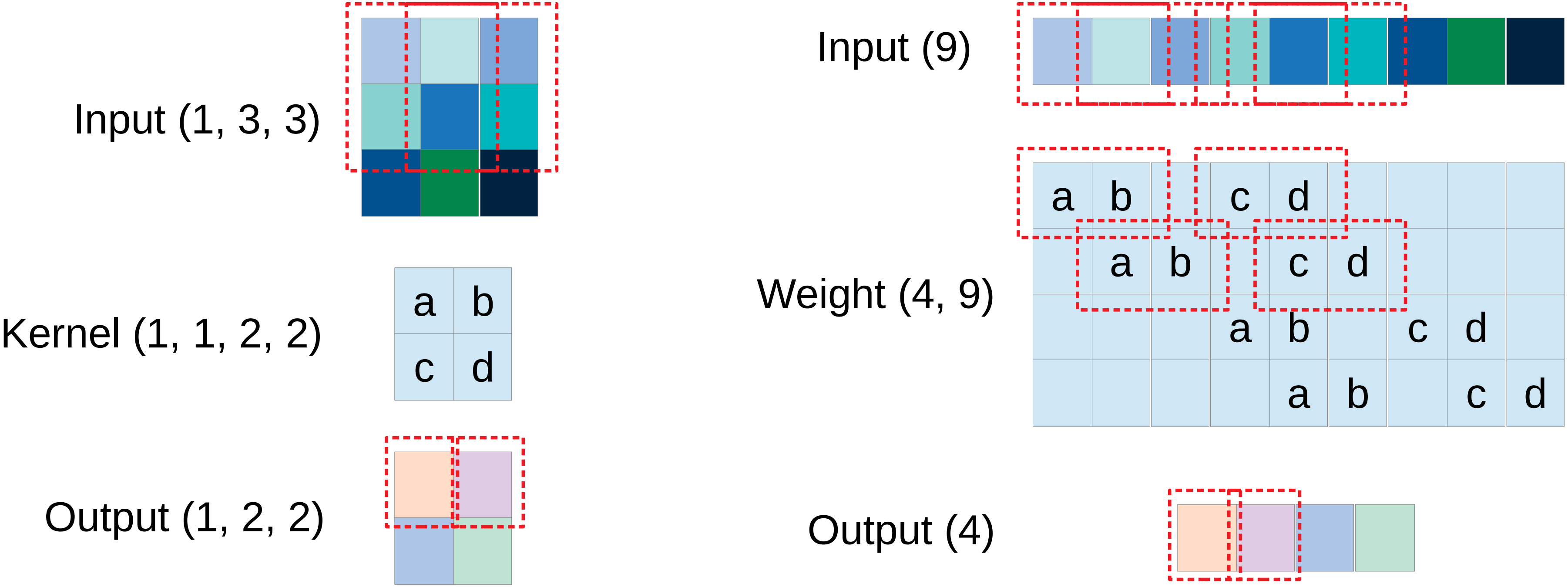
Kernels $W \in \mathbb{R}^{C_o \times \boxed{C_i} \times h \times w}$

Bias $b \in \mathbb{R}^{C_o}$

Output $Y \in \mathbb{R}^{C_o \times (H-h+1) \times (W-w+1)}$

$$Y_{a,b,c} = b_a + \boxed{\sum_{i=0}^{C_i}} \sum_{j=0}^{h} \sum_{k=0}^{w} X_{i,b+j,c+k} W_{a,i,j,k}$$

Convolution as a Linear Layer



Input (1, 3, 3)

Kernel (1, 1, 2, 2)

a	b
c	d

Output (1, 2, 2)

Input (9)

Weight (4, 9)

a	b		c	d				
	a	b		c	d			
			a	b		c	d	
				a	b		c	d

Output (4)

Practical Issues

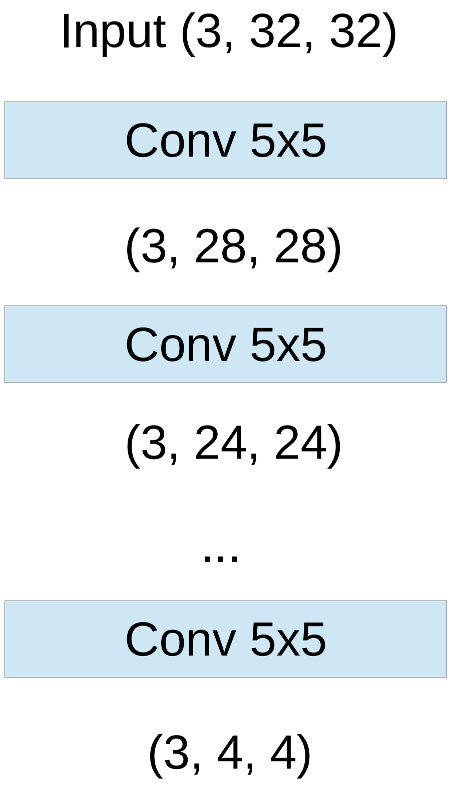
Output Size

Input $X \in \mathbb{R}^{C_i \times H \times W}$

Kernels $W \in \mathbb{R}^{C_o \times C_i \times h \times w}$

Bias $b \in \mathbb{R}^{C_o}$

Output $Y \in \mathbb{R}^{C_o \times \boxed{(H-h+1) \times (W-w+1)}}$



Input (3, 32, 32)

Conv 5x5

(3, 28, 28)

Conv 5x5

(3, 24, 24)

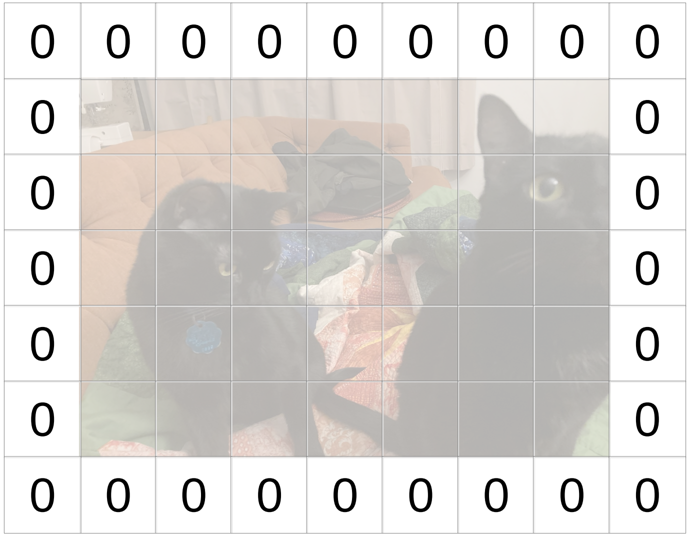
...

Conv 5x5

(3, 4, 4)

Padding

Input: (3, 5, 7)



0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Padded Input: (3, 7, 9)

Conv 3x3

Output: (3, 3, 5)

Padding p_w, p_h

Output $Y \in \mathbb{R}^{C_o \times (H+2p_h-h+1) \times (W+2p_w-w+1)}$

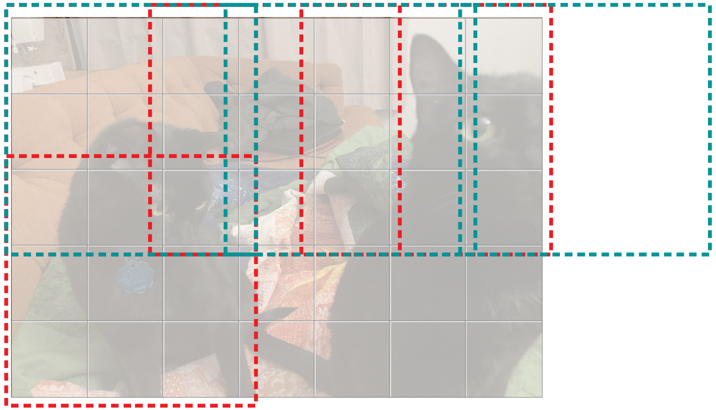
```
torch.nn.Conv2d: padding=’same’
```

Conv 3x3

Output: (3, 5, 7)

Striding

Sometimes sliding over one pixel at a time is unnecessarily expensive

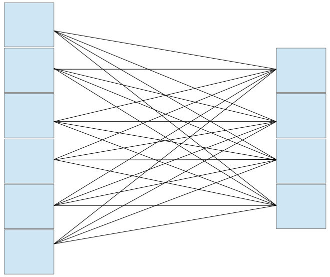


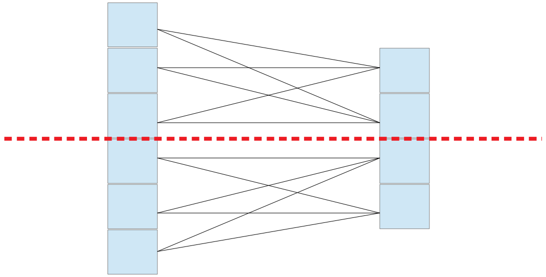
Stride s_w, s_h

Output $Y \in \mathbb{R}^{C_o \times \left(\left\lfloor \frac{H-h+2p_h}{s_h} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{W-w+2p_w}{s_w} \right\rfloor + 1 \right)}$

Grouping

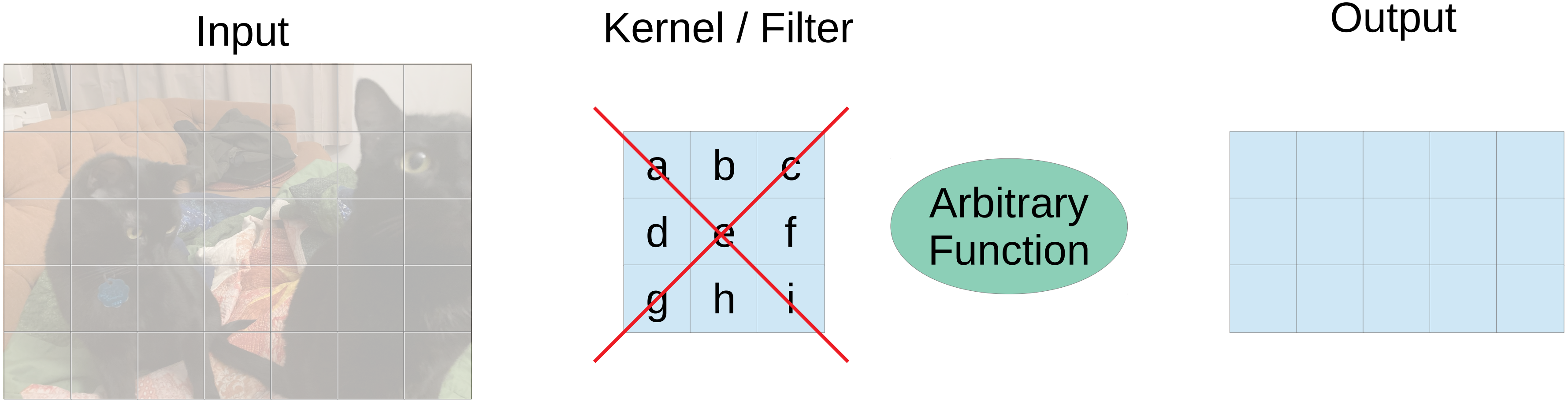
- No grouping: every input channel influences every output channel
  - Kernel size (C_o, C_i, h, w)
  - Kernels are large if there are a lot of channels
- Grouping: Split channels into groups





Convolutional Operators

Convolutional Operators



Input

Kernel / Filter

~~a b c~~
~~d e f~~
~~g h i~~

Arbitrary Function

Output

$$\xcancel{Y_{a,b,c} = b_a + \sum_{i=0}^{C_i} \sum_{j=0}^{h} \sum_{k=0}^{w} X_{i,b+j,c+k} W_{a,i,j,k}}$$

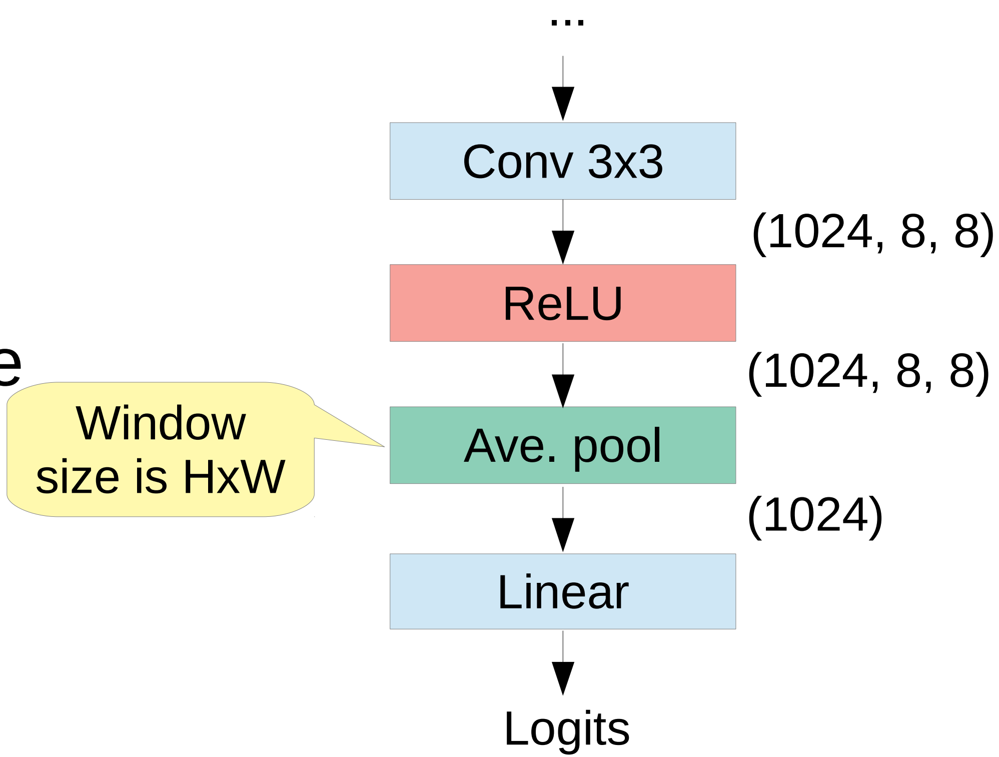
$$Y_{a,b,c} = f\left(X\left[:,\ b:b+h,\ c:c+w\right]\right)$$

Average Pooling

- Average over a small window

$$f(\mathrm{X})_c = \mathrm{mean}_{i,j} \mathrm{X}_{\mathrm{c,i,j}}$$

- Was used with a stride to reduce the size of the data
- No longer used in the middle part of networks
- Global average pooling

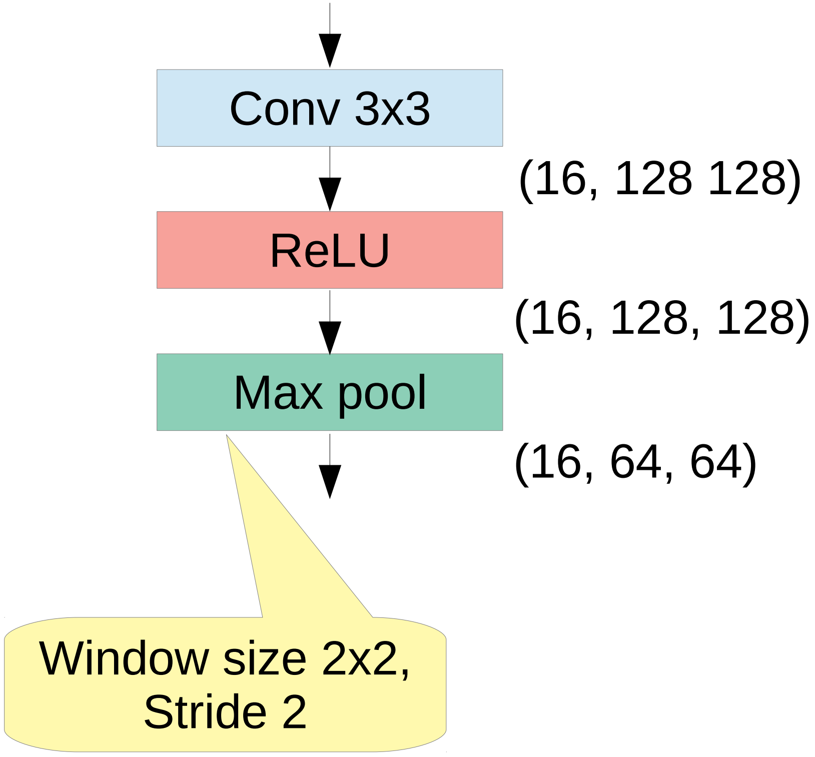


Max Pooling

- Max of a small window

$$f(\mathrm{X})_c = \max_{i,j} \mathrm{X}_{\mathrm{c,i,j}}$$

- Max is nonlinear
- Used to downsample



Conv 3x3

(16, 128 128)

ReLU

(16, 128, 128)

Max pool

(16, 64, 64)

Window size 2x2,
Stride 2