Advanced Convolutional Neural Networks

CS344
Deep Learning

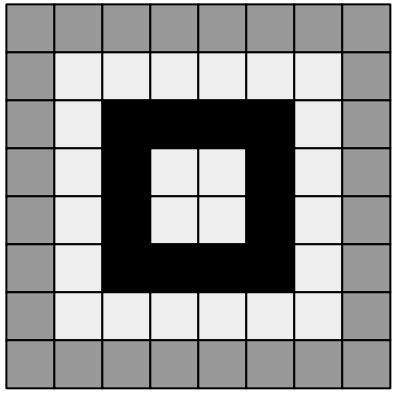
WELLESLEY

CNN

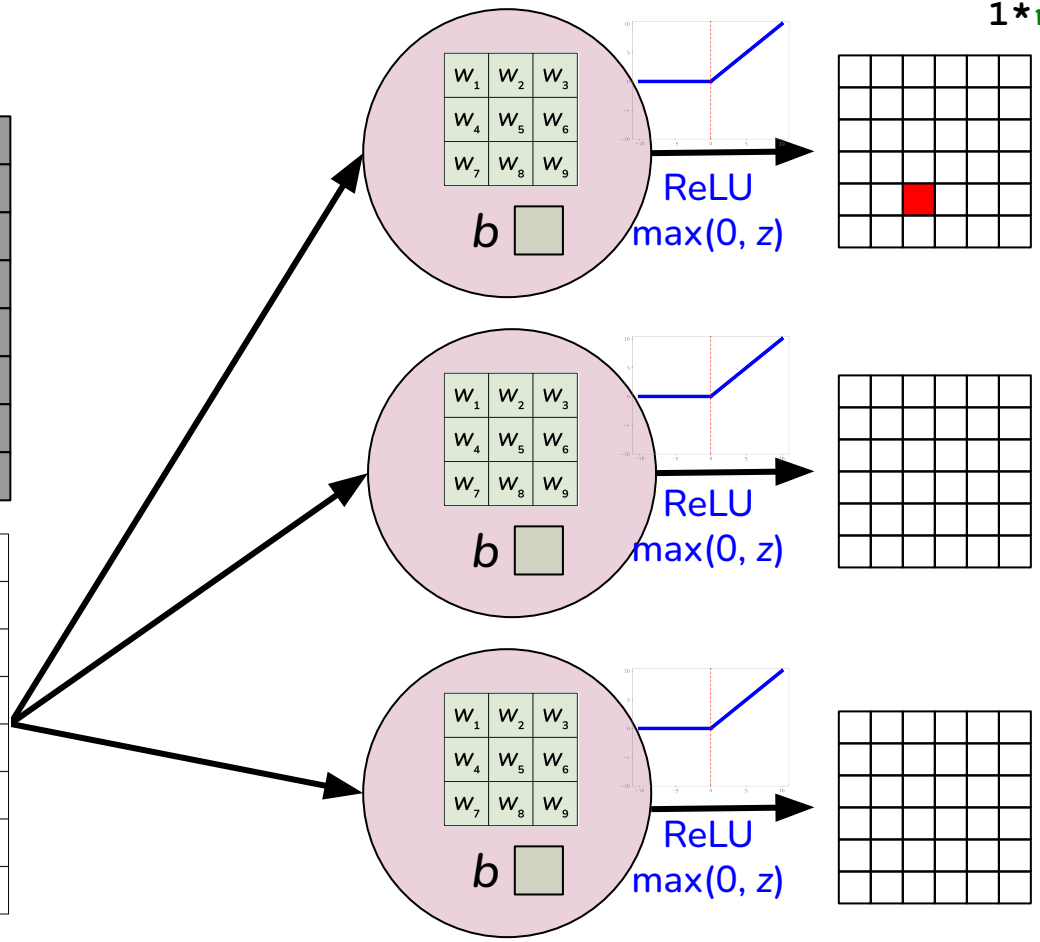
Convolutional Layer

$$z = 0*w_1 + 1*w_2 + 1*w_3 + 0*w_4 + 0*w_5 + 0*w_6 + 1*w_7 + 1*w_8 + 1*w_9 + b$$

$$a = \max(0, z)$$

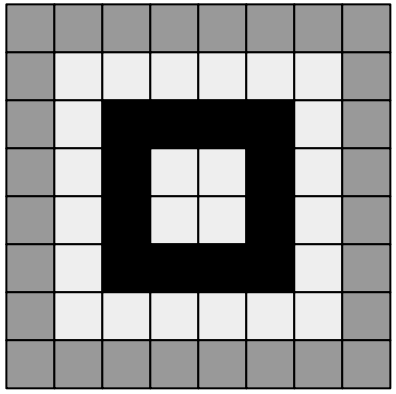


0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	1	1	1	1	1	1	0.5
0.5	1	0	0	0	0	1	0.5
0.5	1	0	1	1	0	1	0.5
0.5	1	0	1	1	0	1	0.5
0.5	1	0	0	0	0	1	0.5
0.5	1	1	1	1	1	1	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

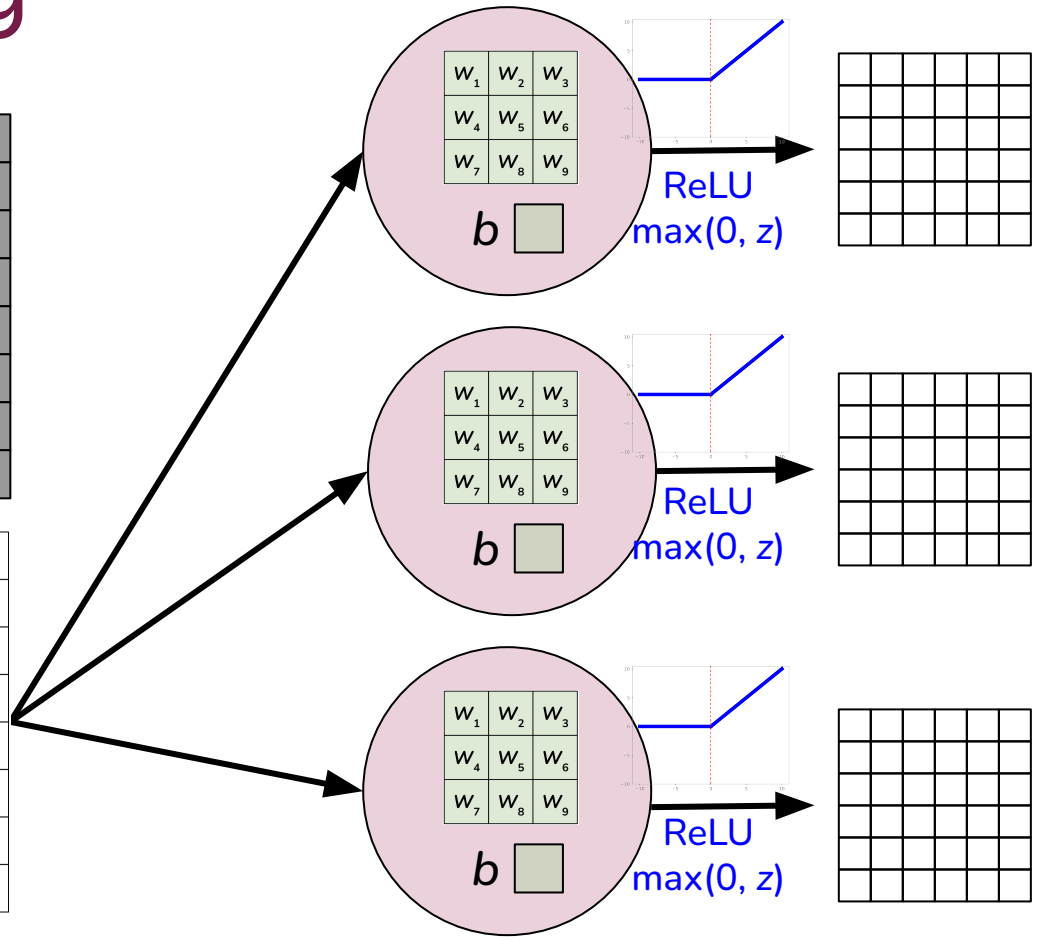


Training

Convolutional Layer



0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	1	1	1	1	1	1	0.5
0.5	1	0	0	0	0	1	0.5
0.5	1	0	1	1	0	1	0.5
0.5	1	0	1	1	0	1	0.5
0.5	1	0	0	0	0	1	0.5
0.5	1	1	1	1	1	1	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5



- Choose number of units (filters) in layer
- Initialize parameters in each unit
- Perform gradient descent to update parameters

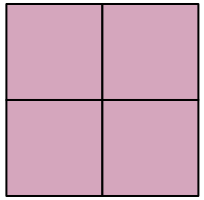
Max Pooling

Stride of 2

3	7	9	1	0	4
8	1	3	3	3	0
7	4	6	4	4	6
4	4	1	9	3	6
9	8	4	9	4	3
7	5	2	3	3	1

No parameters to learn!

max



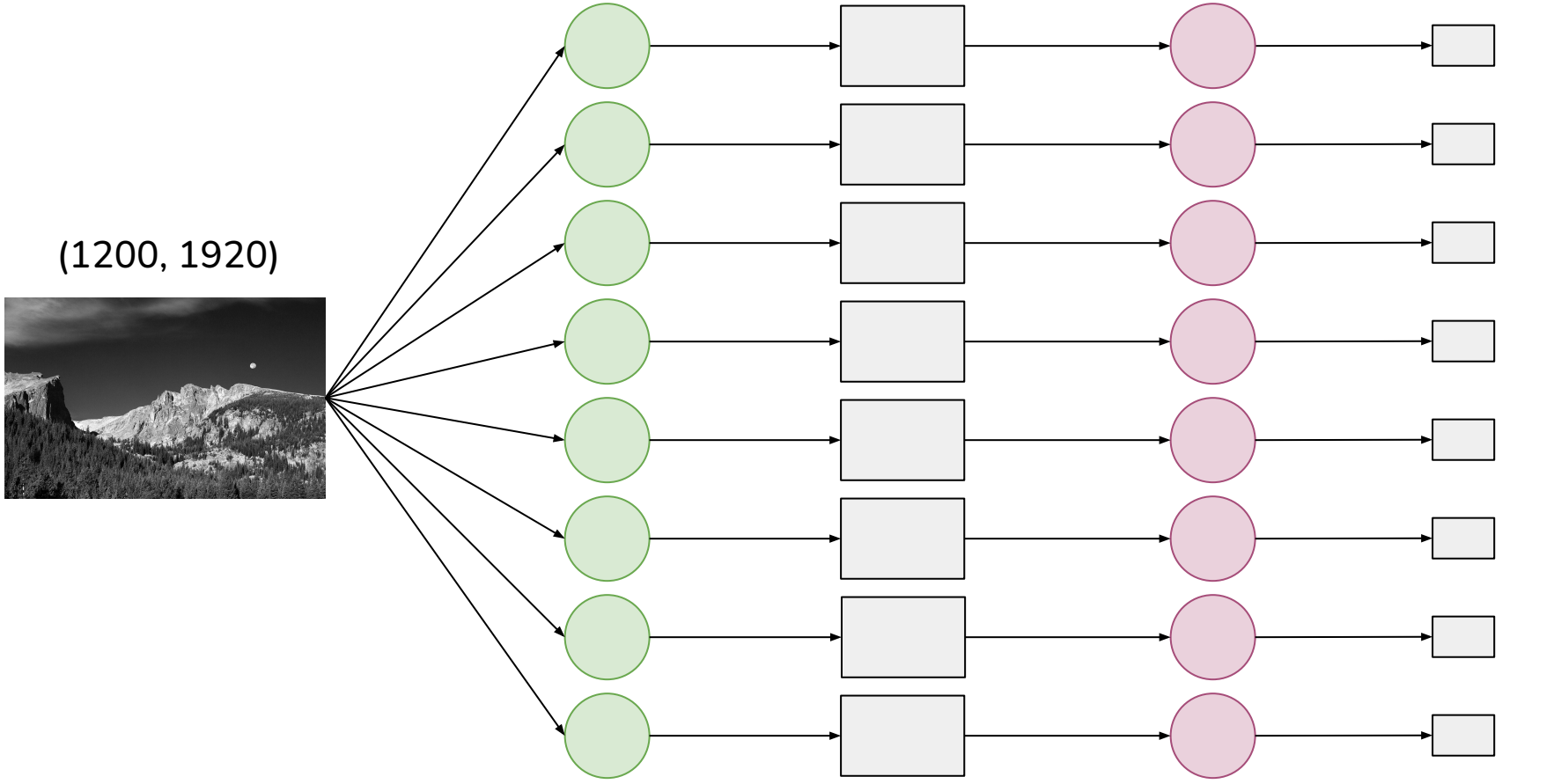
2 x 2

=

8	9	4
7	9	6
9	9	4

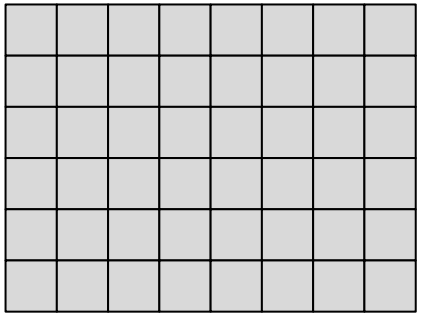
Output has half the height and half the width of input

Layers

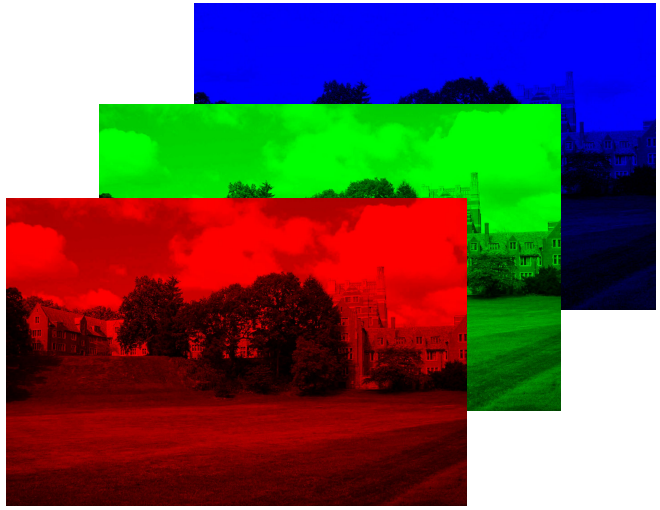


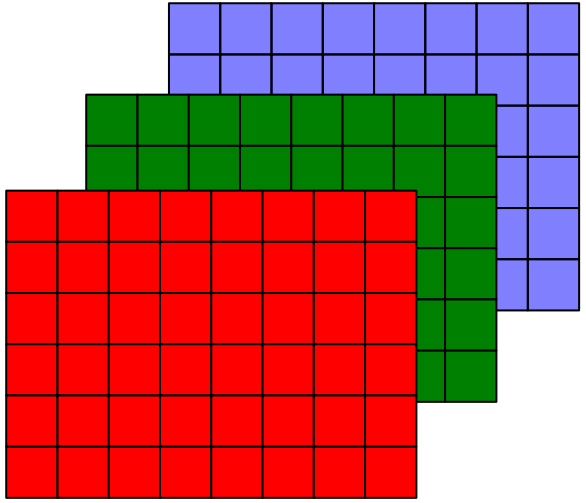
RGB (Color)

(1200, 1920)

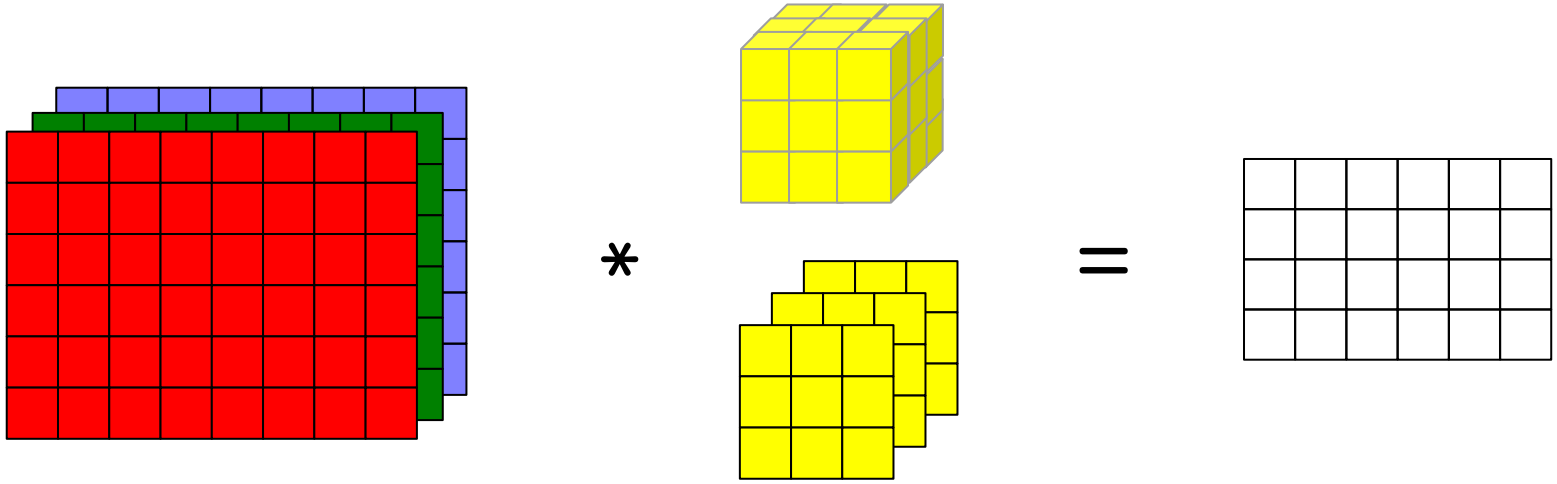


(1200, 1920, 3)

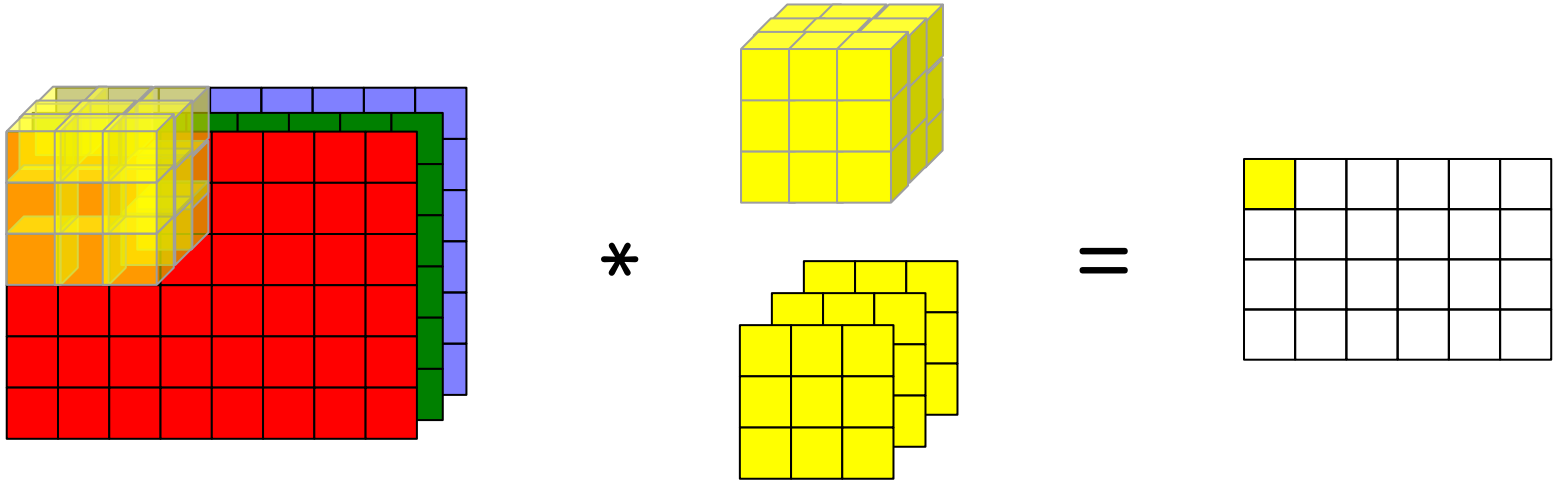




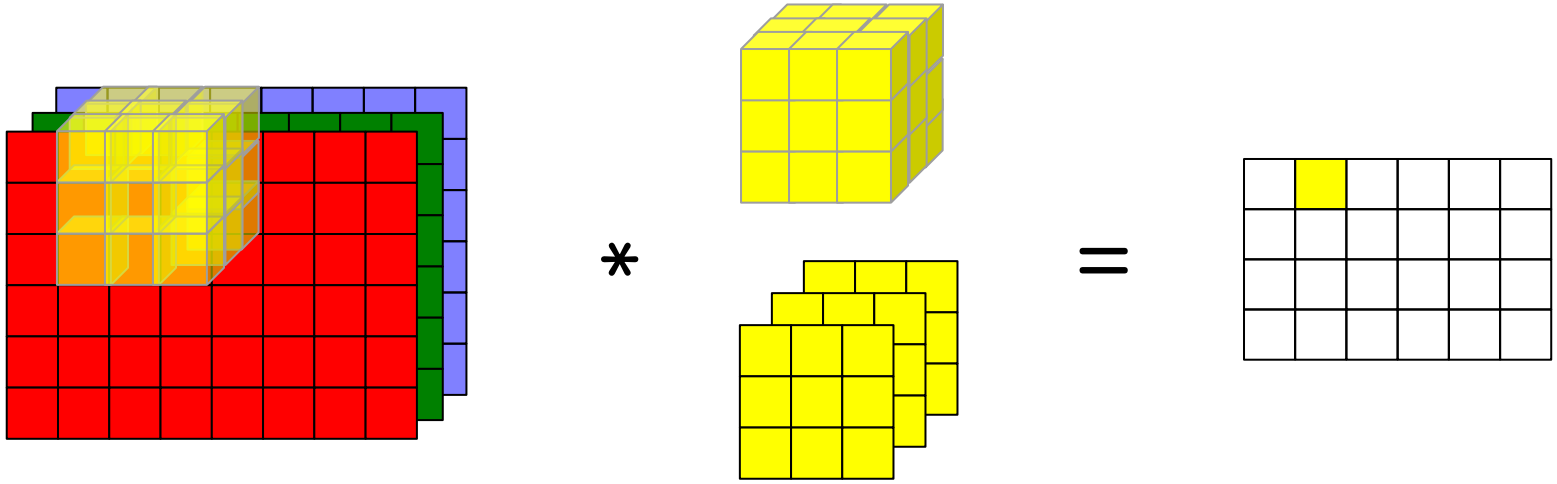
Convolution on Volume



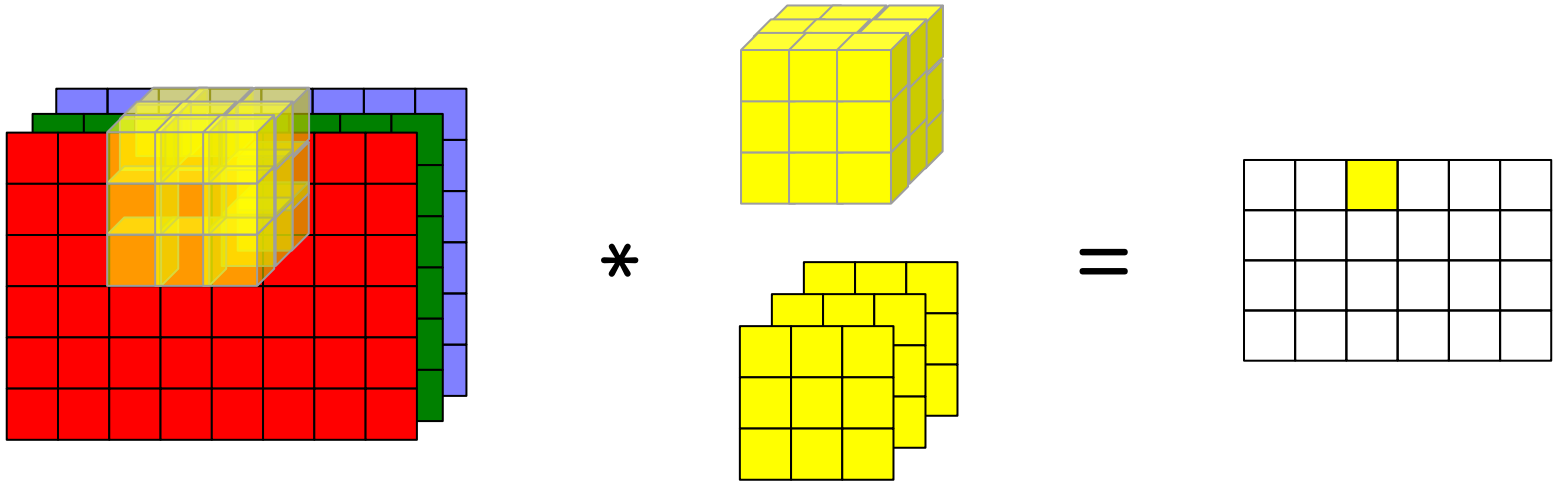
Convolution on Volume



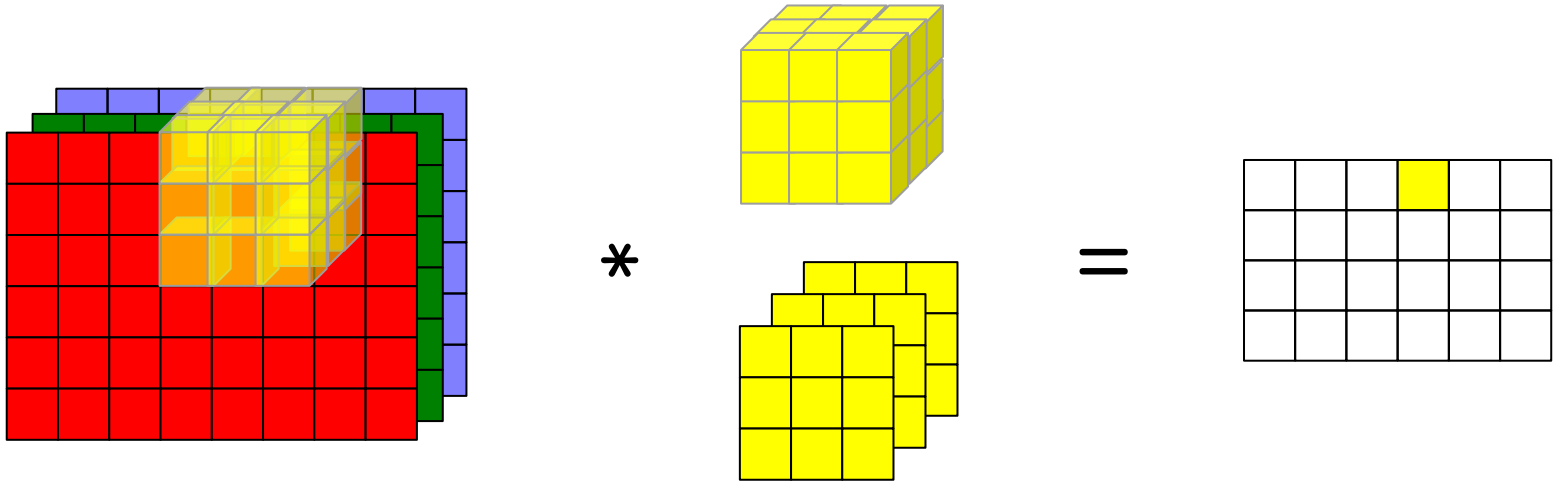
Convolution on Volume



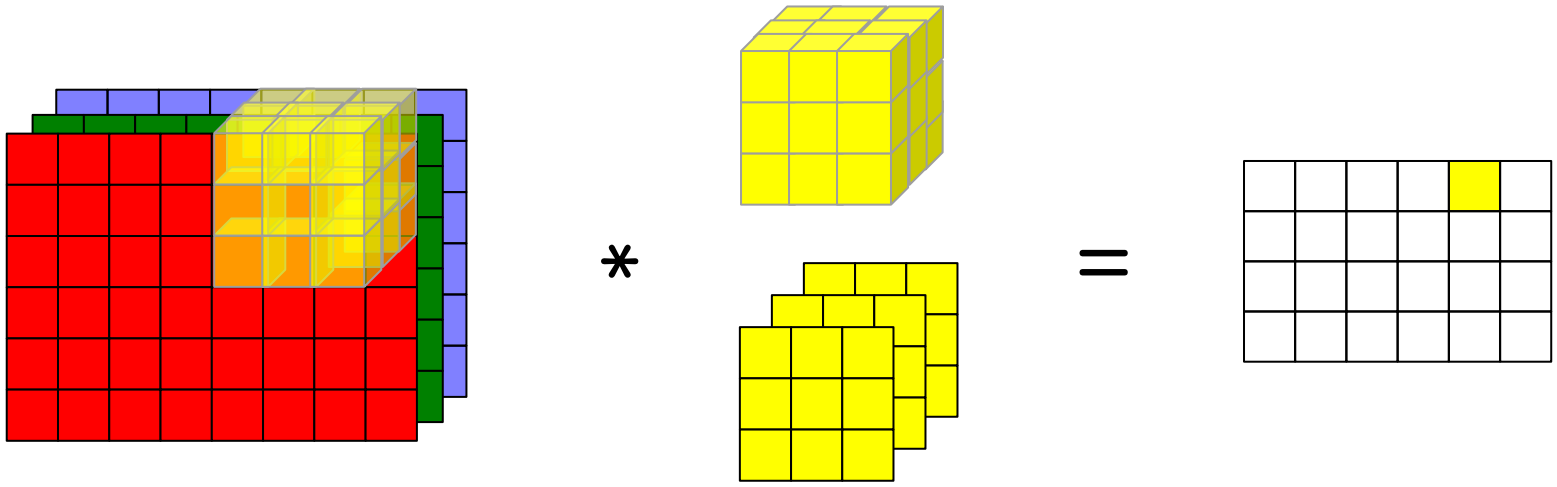
Convolution on Volume



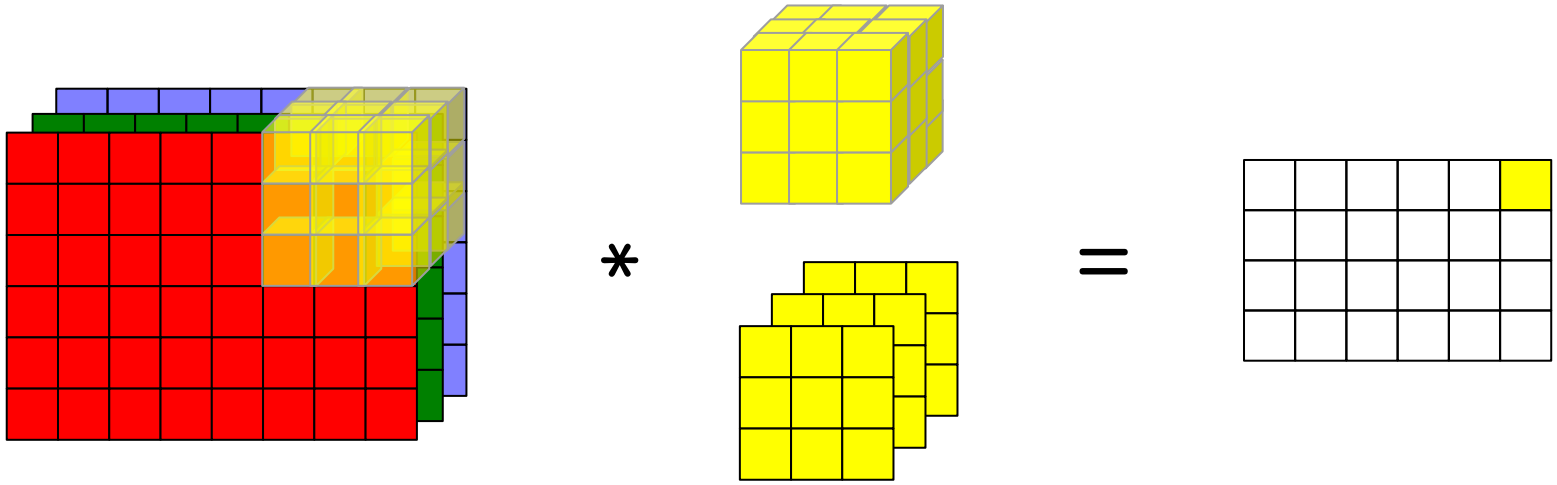
Convolution on Volume



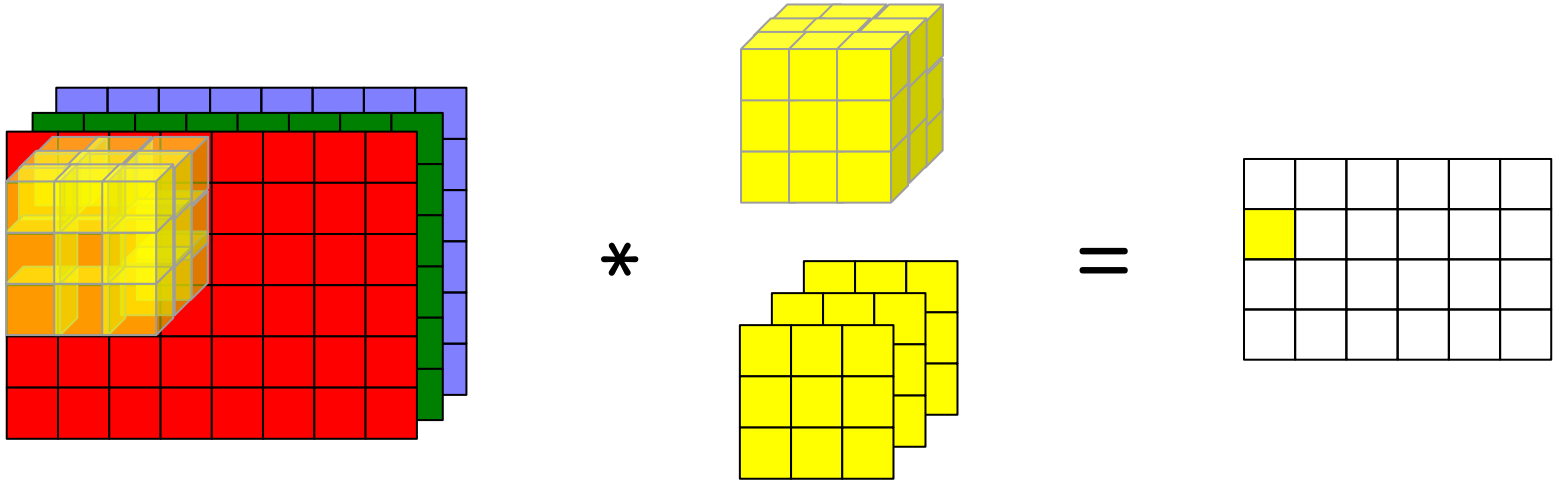
Convolution on Volume



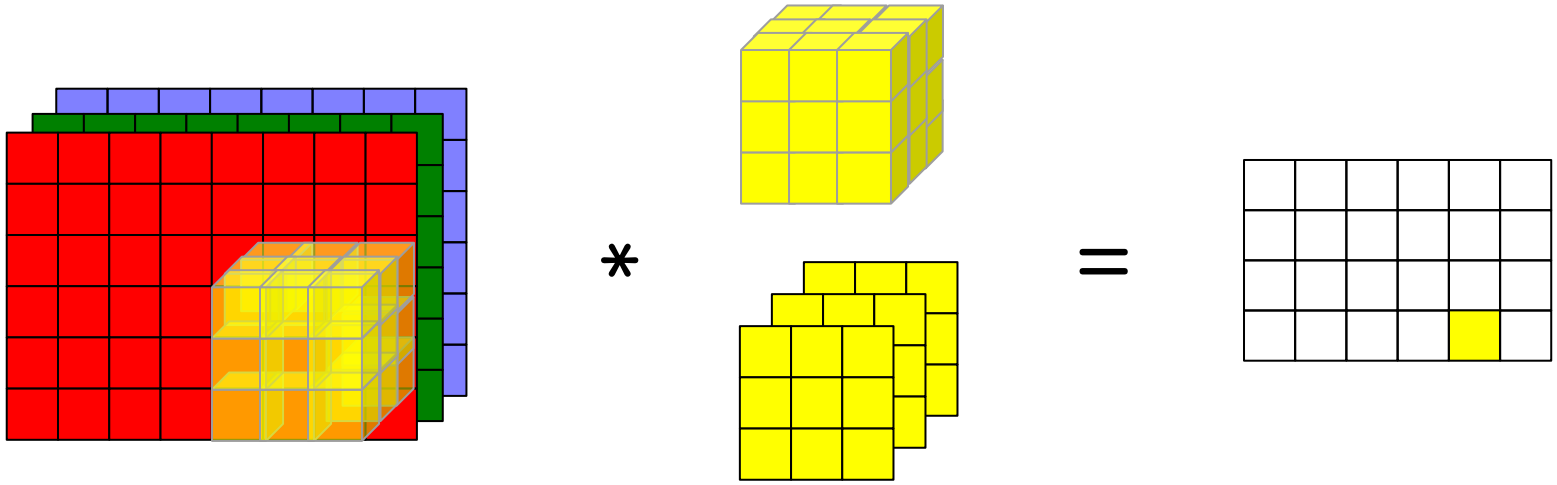
Convolution on Volume



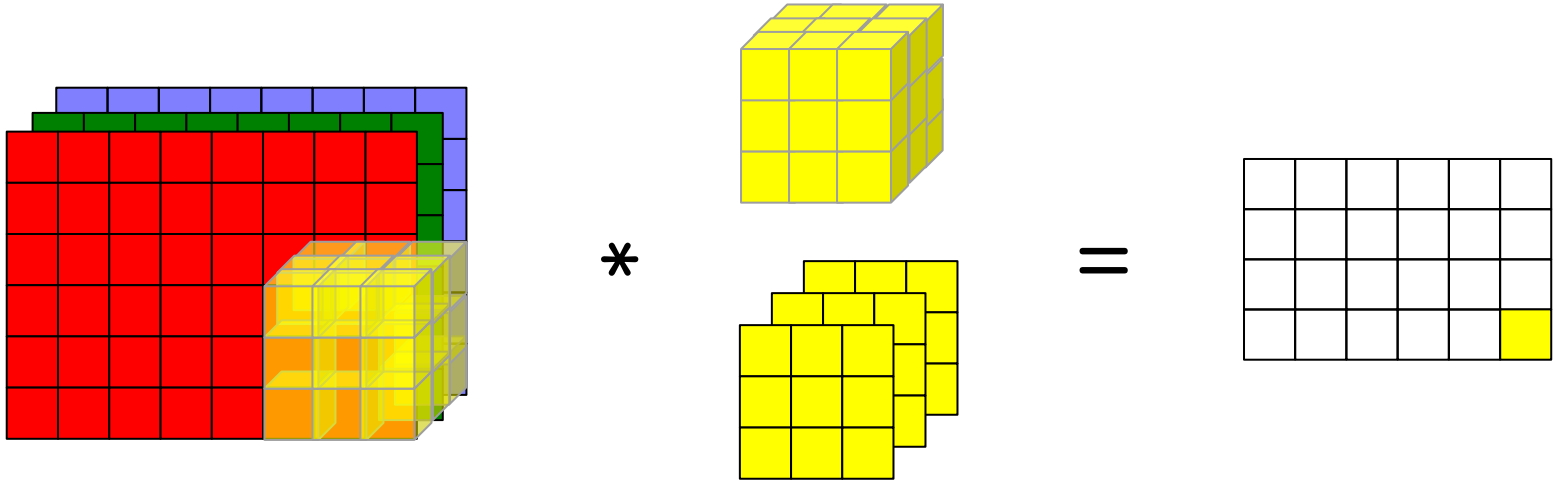
Convolution on Volume



Convolution on Volume

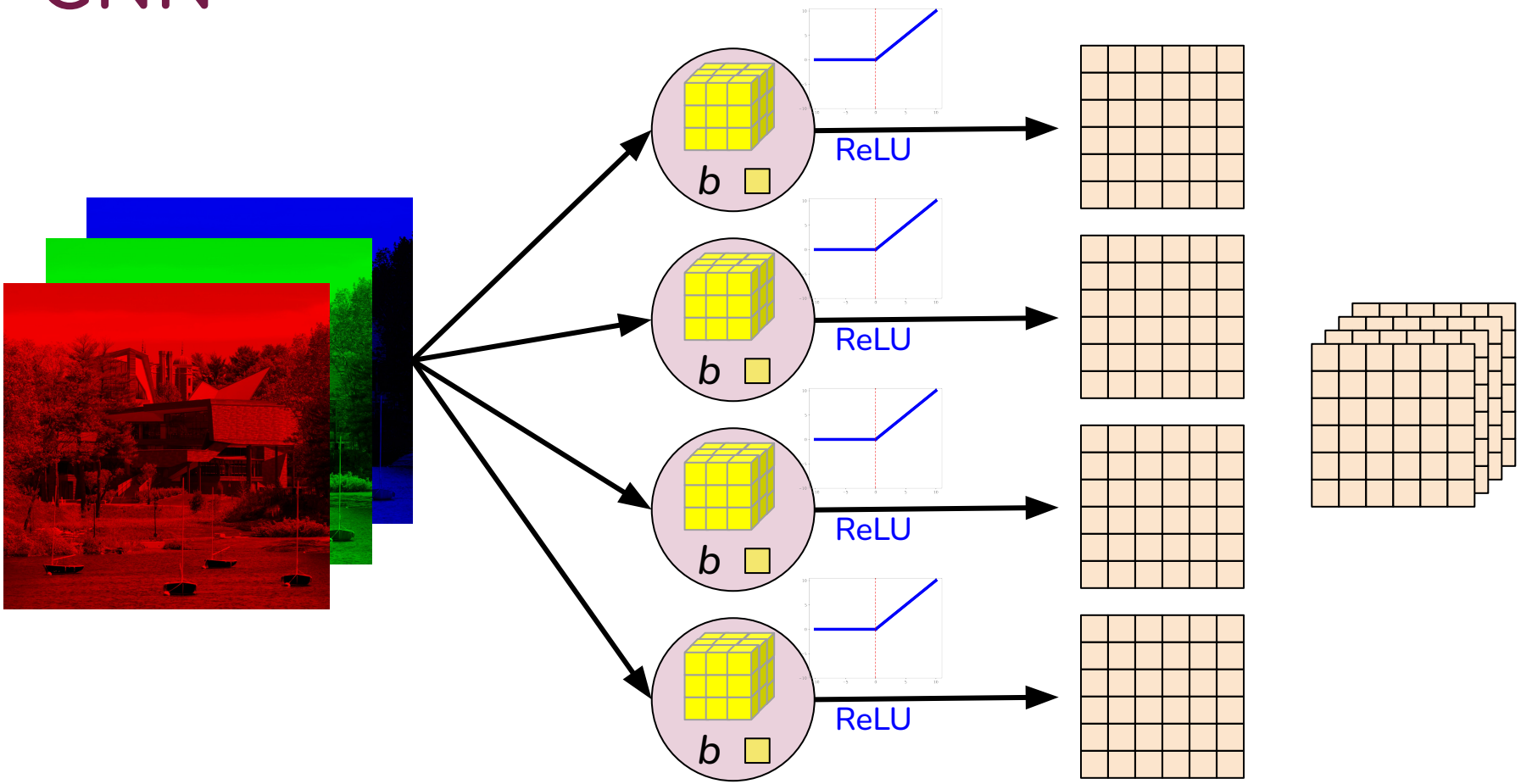


Convolution on Volume



CNN

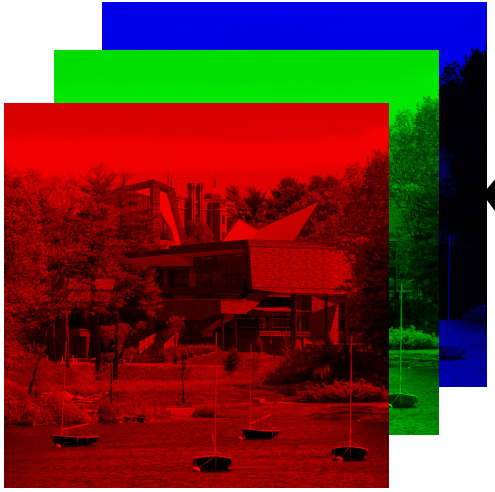
Convolutional Layer



CNN

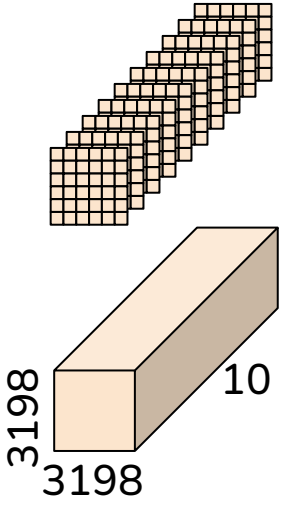
Convolutional Layer

(3200, 3200, 3)



280
parameters

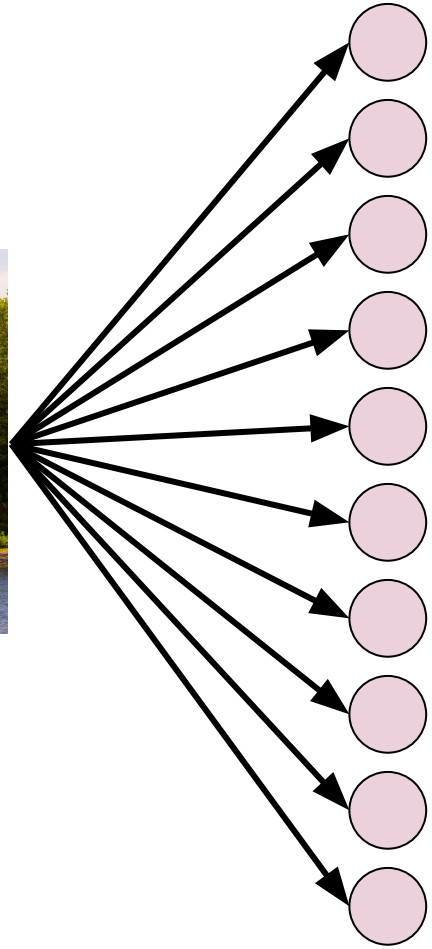
(3198, 3198, 10)



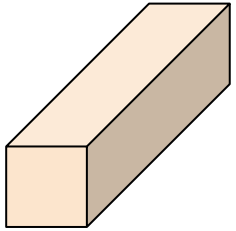
CNN

Convolutional Layer

(3200, 3200, 3)



(3198, 3198, 10)

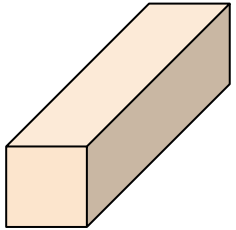


CNN

Convolutional Layer

(3200, 3200, 3)

(3198, 3198, 10)



CNN

Conv Layer

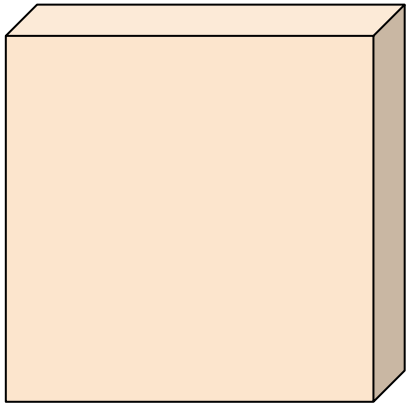
Conv Layer

(3200, 3200, 3)

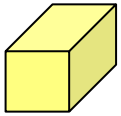
(3, 3, 3)

Suppose we use 10 units (filters or channels)

(3198, 3198, 10)

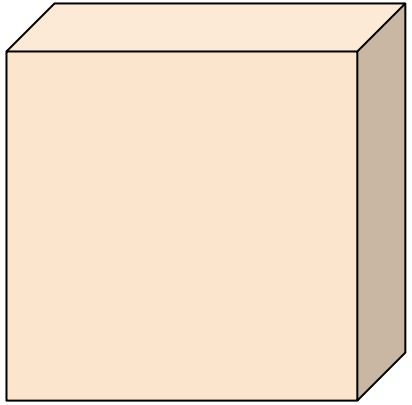


(3, 3, 10)

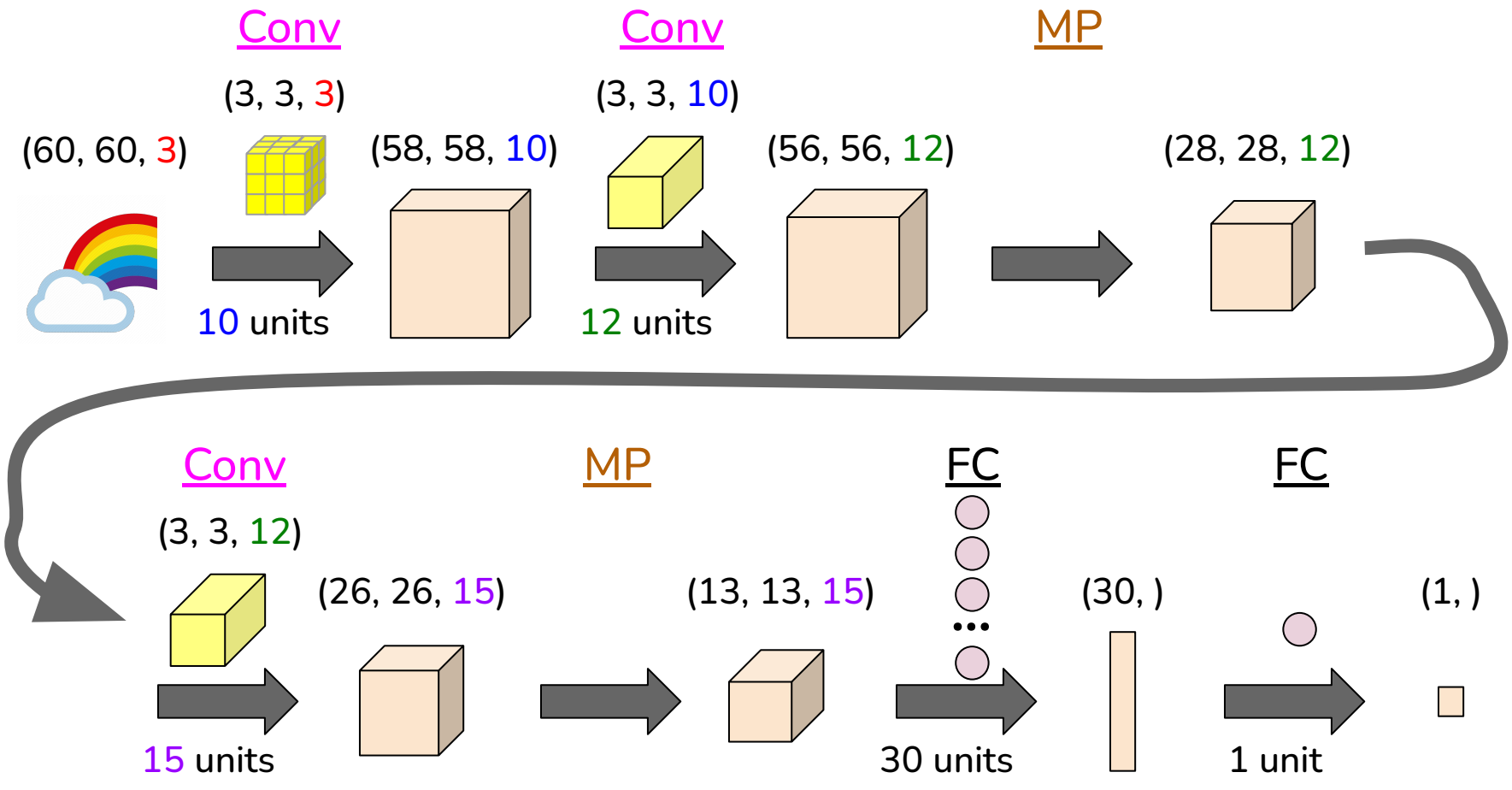


Suppose we use 15 units (filters or channels)

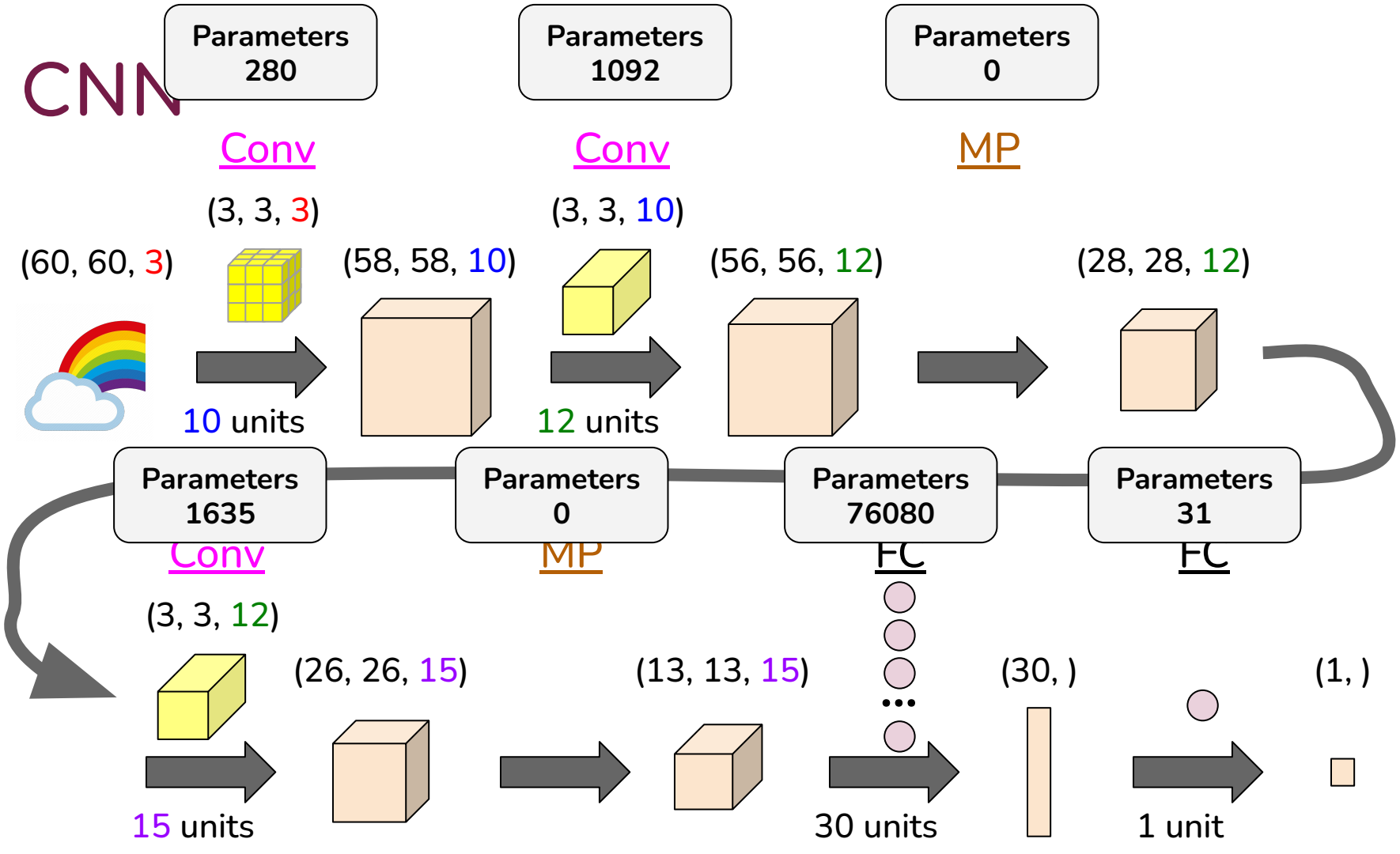
(3196, 3196, 15)



CNN


Conv
(3, 3, 3)
Conv
(3, 3, 10)
MP
(60, 60, 3)
(58, 58, 10)
(56, 56, 12)
(28, 28, 12)
10 units
12 units
Conv
(3, 3, 12)
MP
FC
FC
(26, 26, 15)
(13, 13, 15)
(30,)
(1,)
15 units
30 units
1 unit

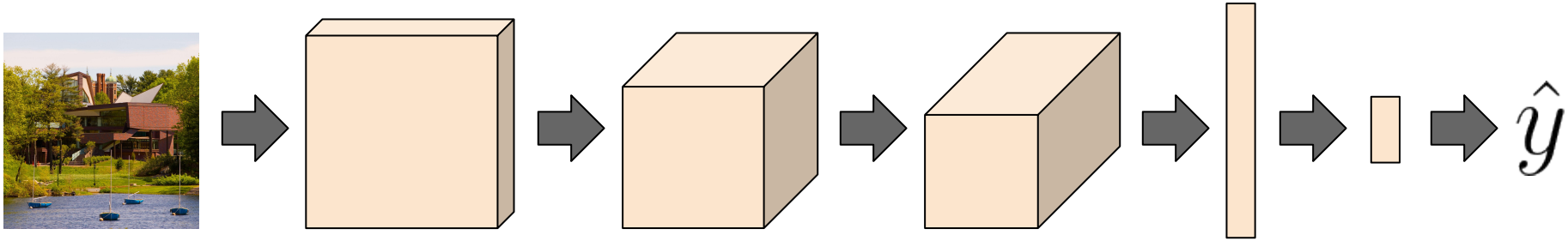
CNN


Parameters
280
Parameters
1092
Parameters
0
Conv
Conv
MP
(3, 3, 3)
(3, 3, 10)
(60, 60, 3)
(58, 58, 10)
(56, 56, 12)
(28, 28, 12)
10 units
12 units
Parameters
1635
Parameters
0
Parameters
76080
Parameters
31
Conv
MP
FC
FC
(3, 3, 12)
(26, 26, 15)
(13, 13, 15)
(30,)
(1,)
15 units
30 units
1 unit

Neural Network Architecture

Layer	Activation Shape	Activation Size	Activation Function	Number of Parameters
Input	(60, 60, 3)	10,800	N/A	0
Conv	(58, 58, 10)	33,640	ReLU	280
Conv	(56, 56, 12)	37,632	ReLU	1,092
MP	(28, 28, 12)	9,408	ReLU	0
Conv	(26, 26, 15)	10,140	ReLU	1,635
MP	(13, 13, 15)	2,535	ReLU	0
FC	(30, 1)	30	ReLU	76,080
FC	(1, 1)	1	Sigmoid	31

Training



$$\text{Cost}\, J = \frac{1}{m} \sum_{i=1}^{m} Loss$$

Use gradient descent to determine parameters that minimize cost

VGG-19

Each convolutional layer uses (3, 3, ?) filters, stride=1, and “same” convolution (padding=1)

Softmax

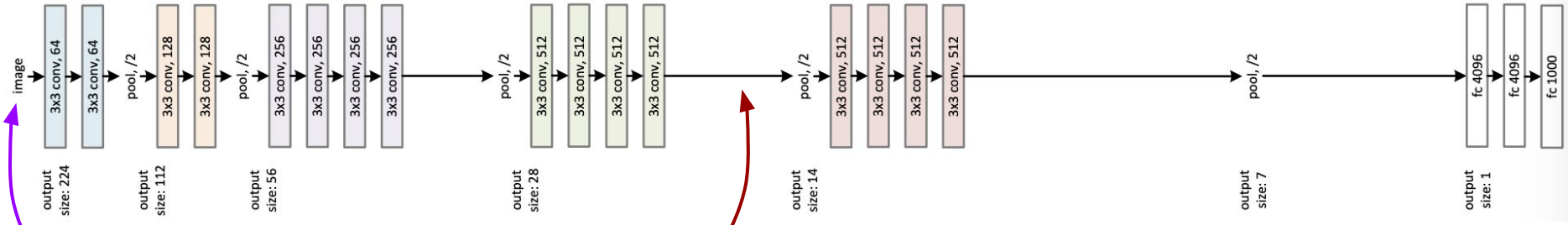


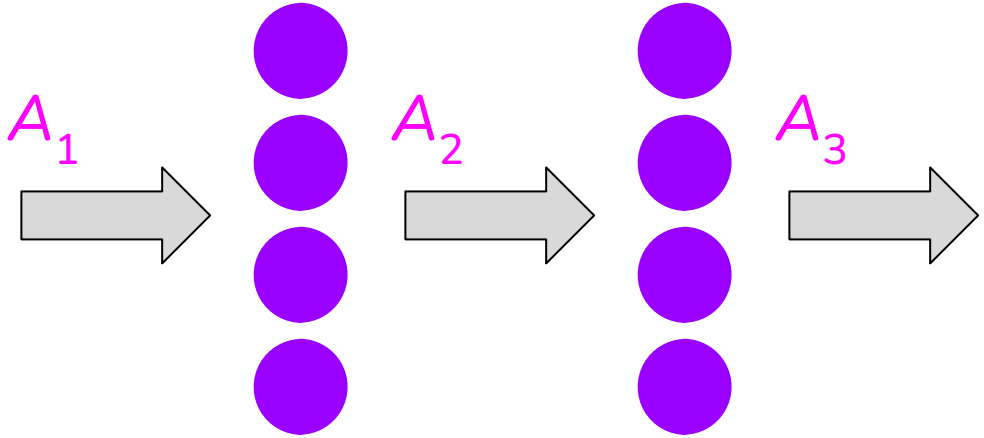
image (224, 224, 3)

Max pooling uses (2, 2) filters with stride=2

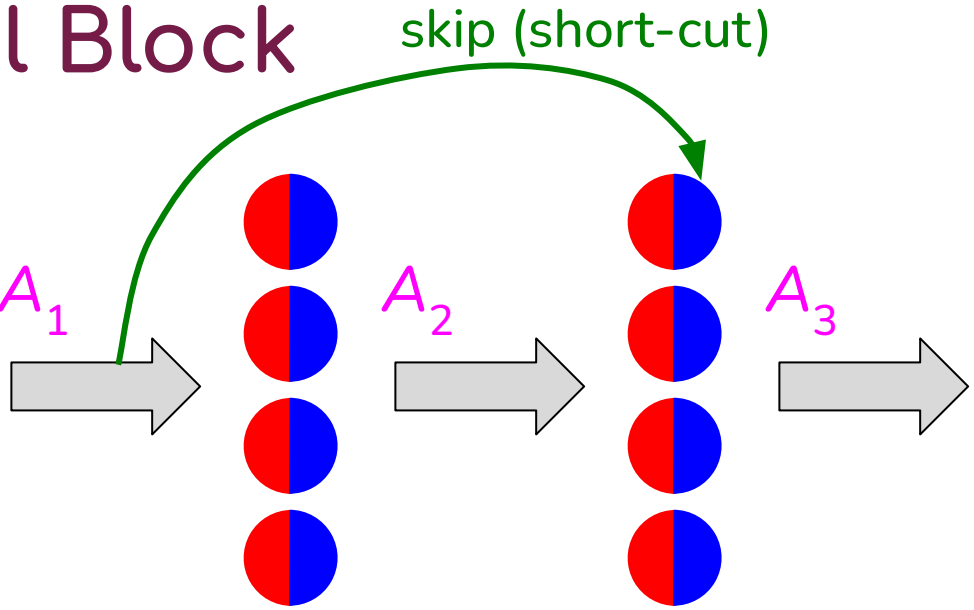
- 19 layers with parameters to learn
- Sizes halve with pooling
- Channels double with units (filters)
- 138 million parameters
- Trained on early version of ImageNet, 1.2 million images belonging to 1000 classes

Plain Network (34-layer)

Residual Block



Residual Block



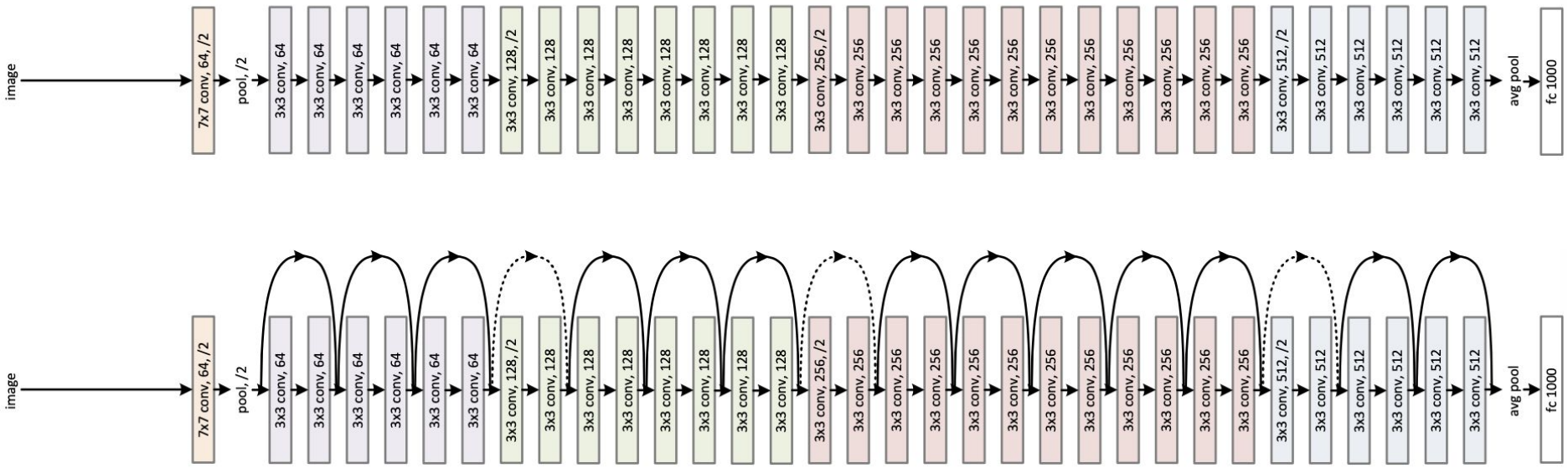
A_1 → Linear → Activation → A_2 → Linear → Activation → A_3

$Z_2 = A_1 \cdot W_2 + b_2$ $\quad A_2 = g(Z_2)$ $\quad Z_3 = A_2 \cdot W_3 + b_3$

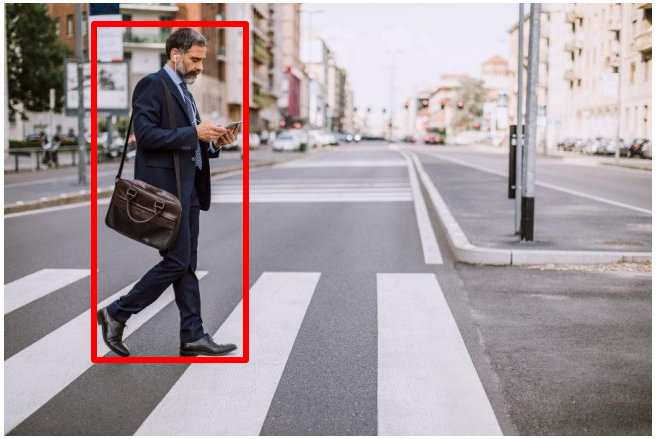
$A_3 = g(Z_3 + A_1)$

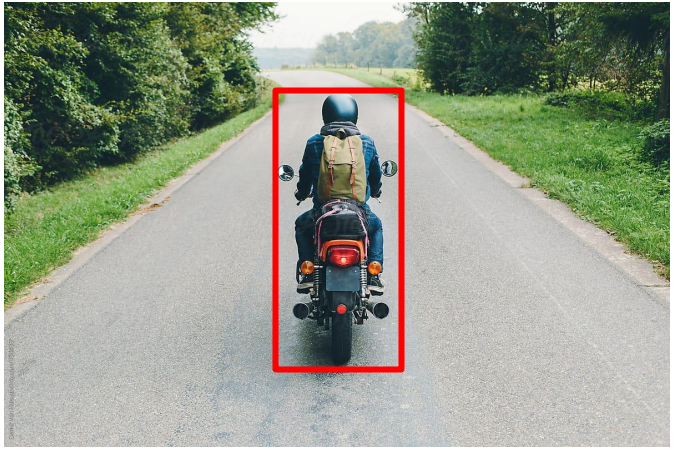
~~$A_3 = g(Z_3)$~~

ResNet (Residual Network)



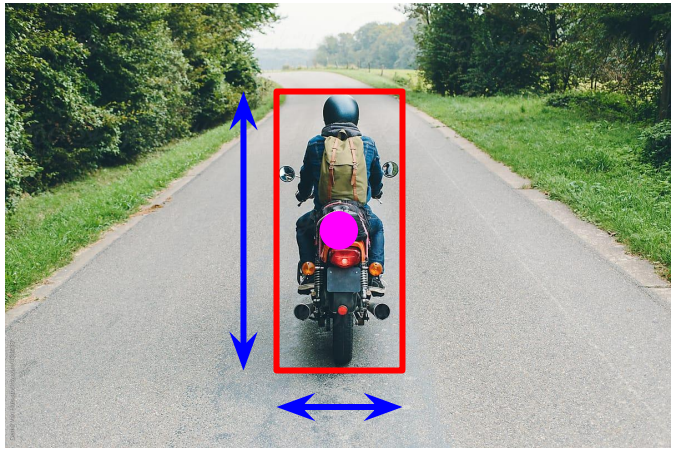
Localization





Localization

- Center coordinate
- Width and Height

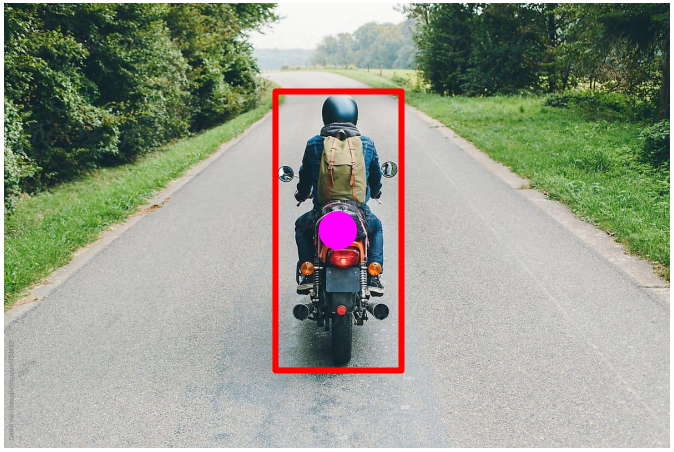


(0.5, 0.5, 0.2, 0.7)

Use MSE
loss function
on (1, 4) output

Localization

(0.3, 0.4, 0.3, 0.8)



(0.5, 0.5, 0.2, 0.7)

(0.5, 0.8, 0.1, 0.1)