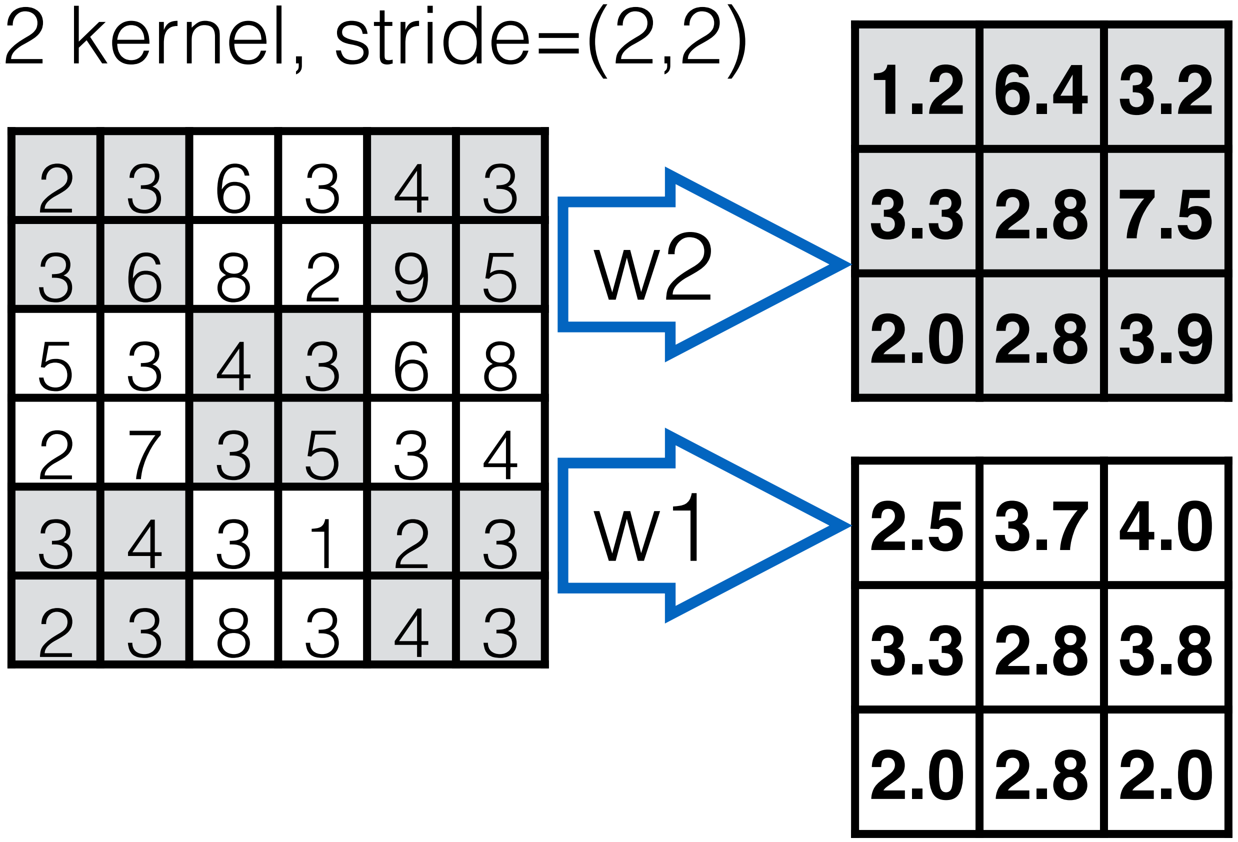
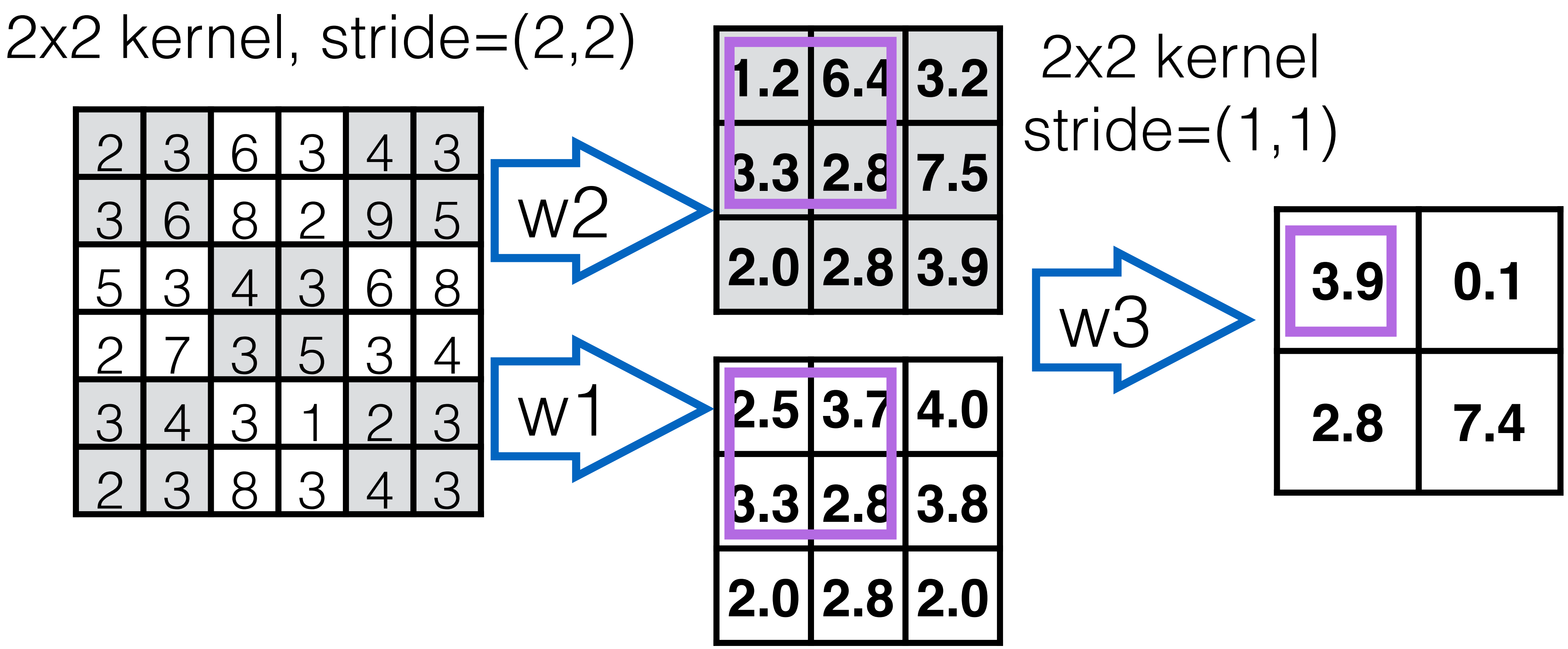
Convolutions with multiple filters

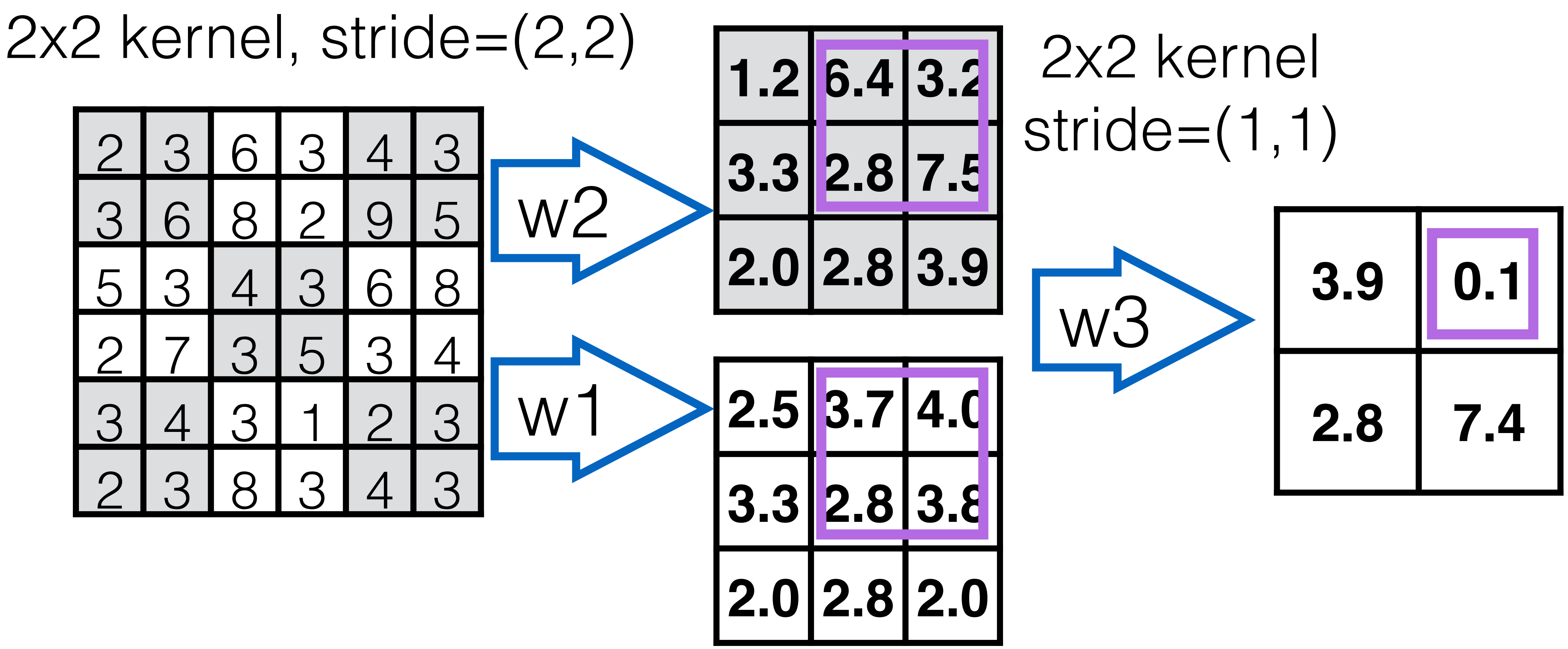
2x2 kernel, stride=(2,2)



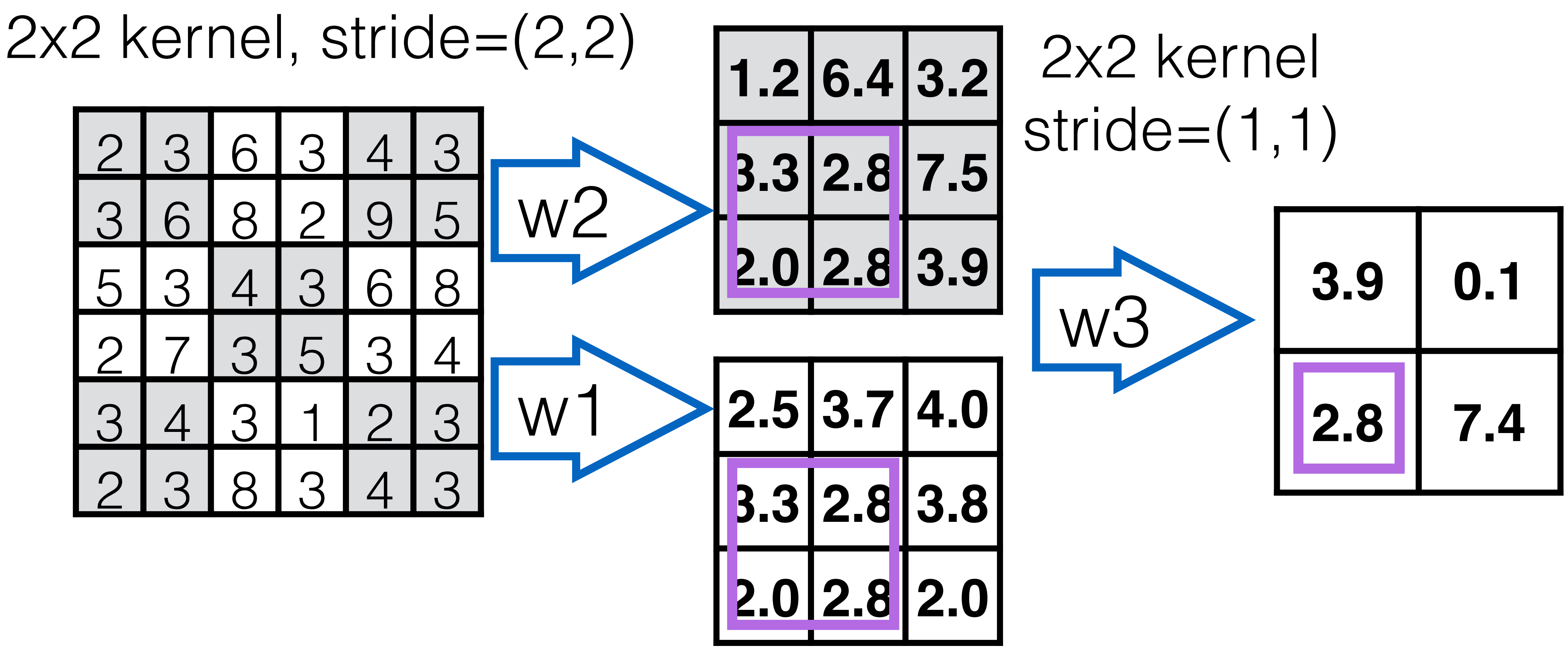
Convolutions with multiple filters



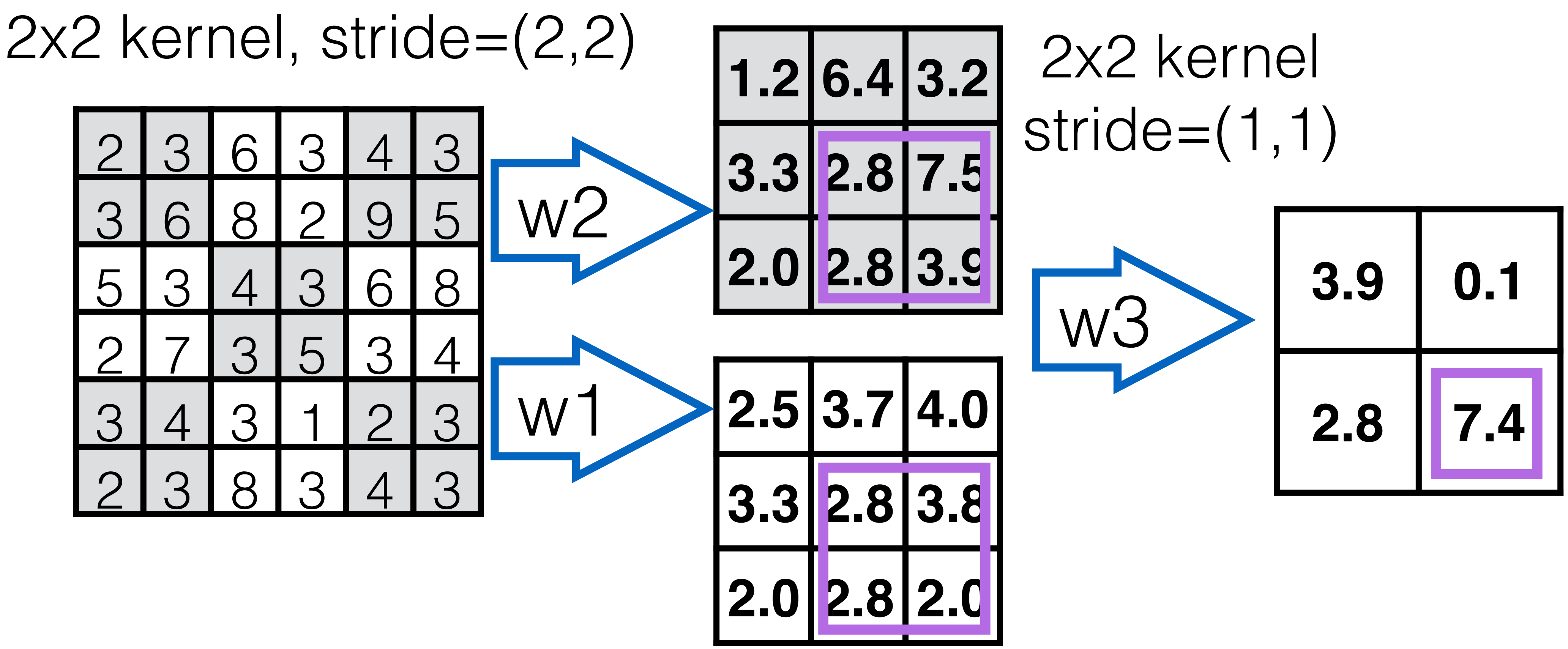
Convolutions with multiple filters



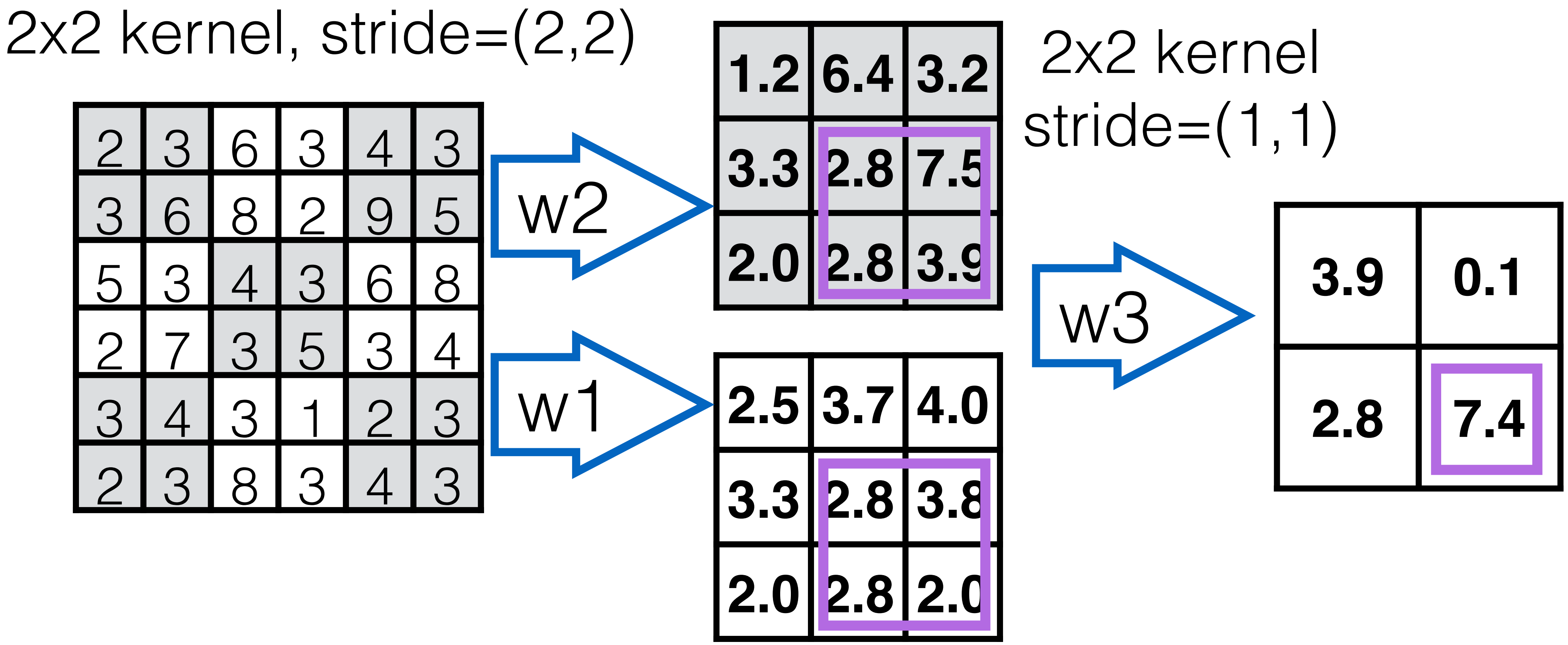
Convolutions with multiple filters



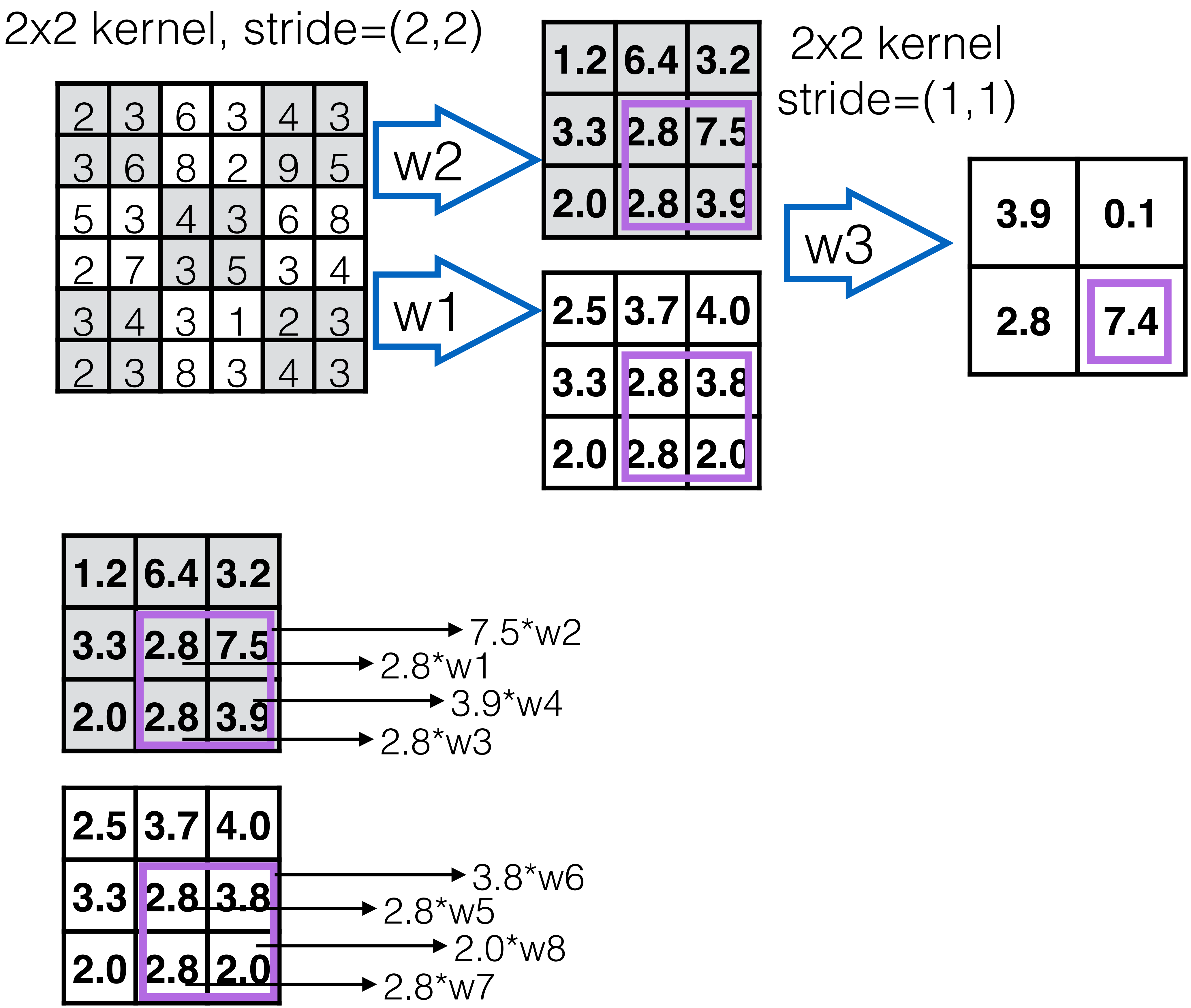
Convolutions with multiple filters



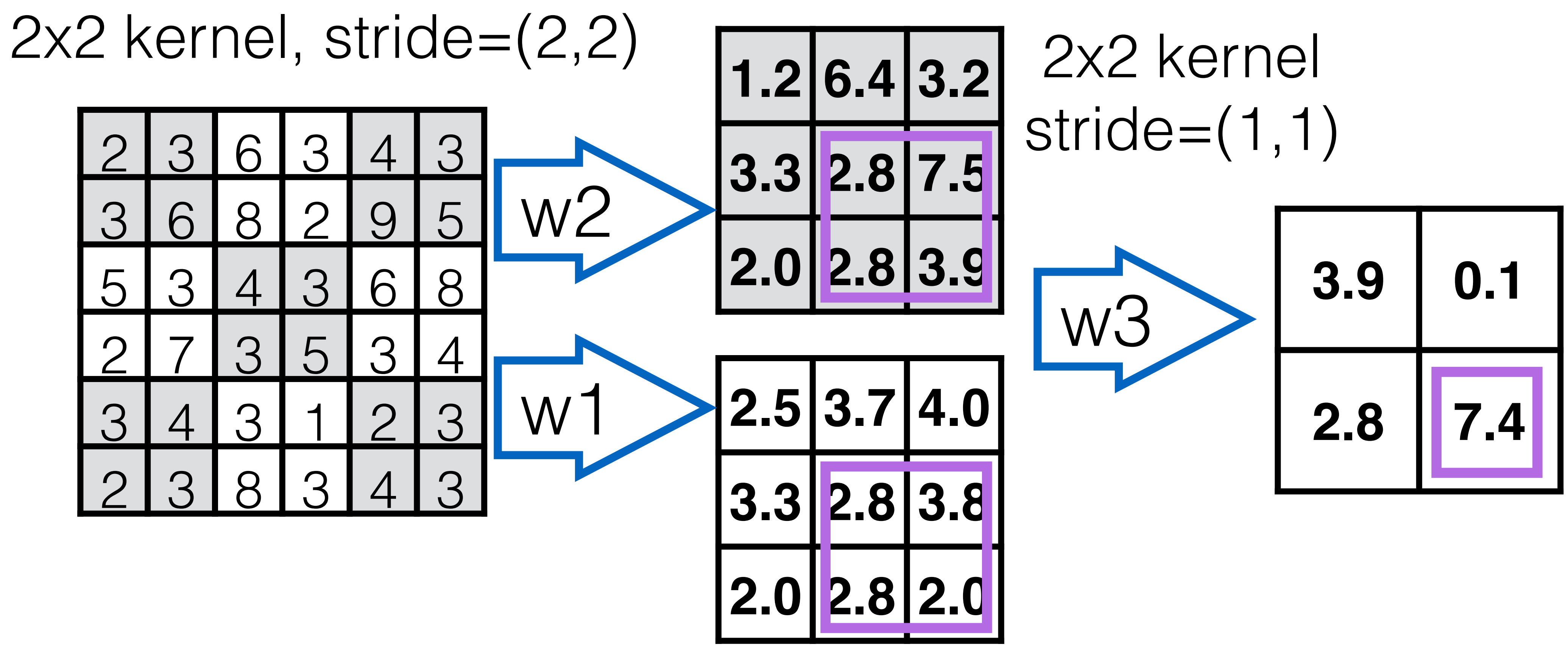
Convolutions with multiple filters

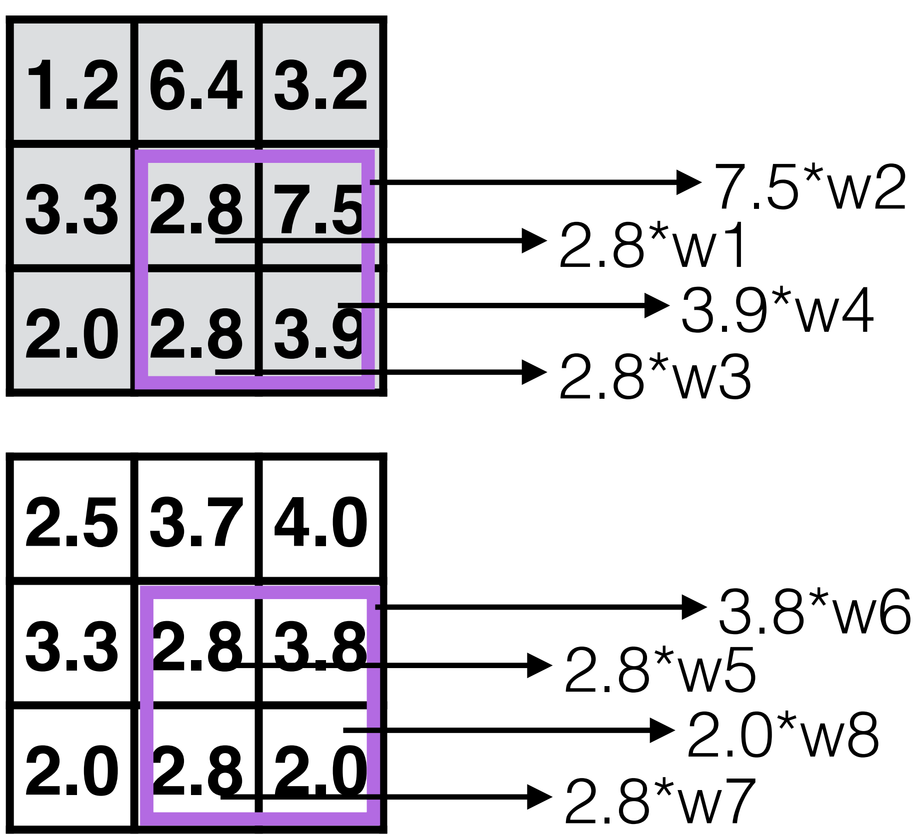


Convolutions with multiple filters



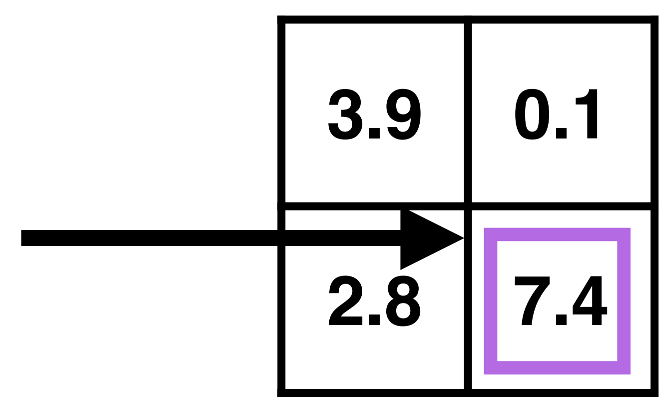
Convolutions with multiple filters





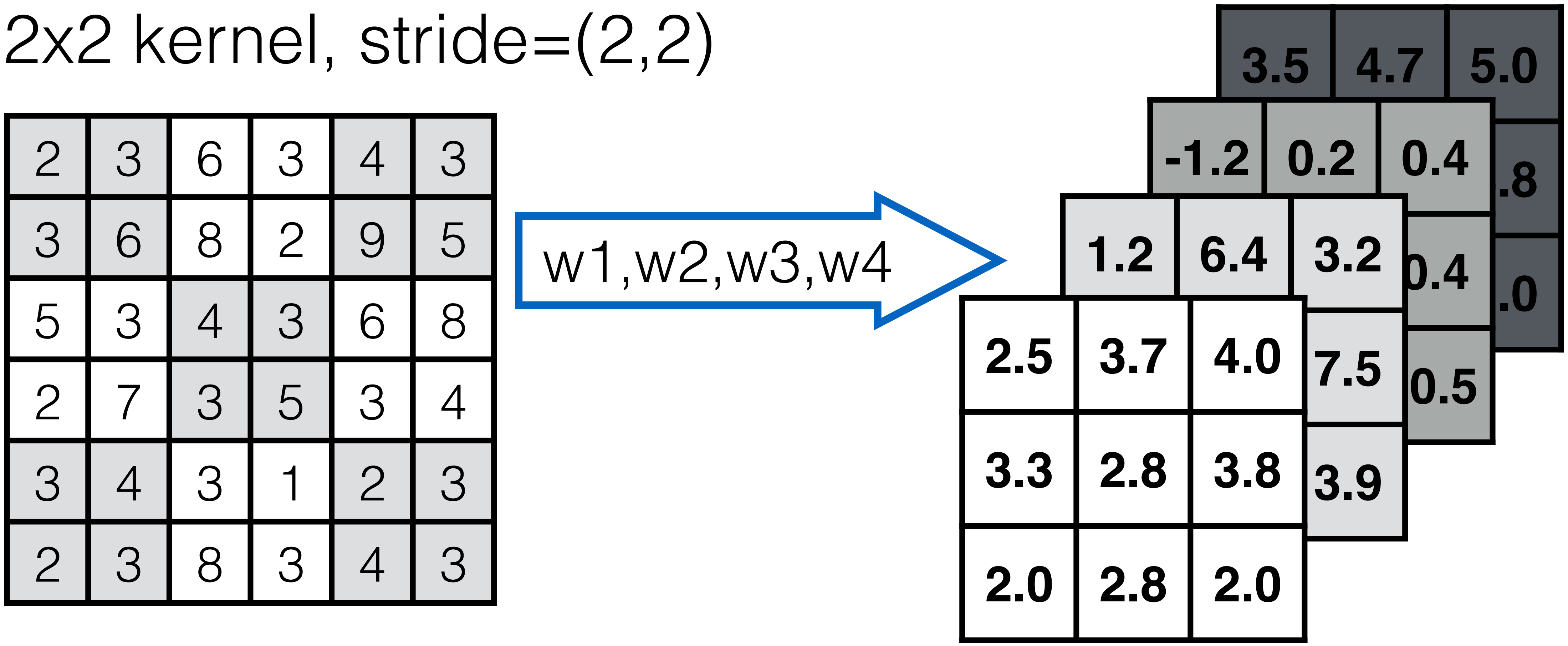
w3 is this:

2.8*w1 + 7.5*w2 +
2.8*w3 + 3.9*w4 +
2.8*w5 + 3.8*w6 +
2.8*w7 + 2.0*w8 + b
= z

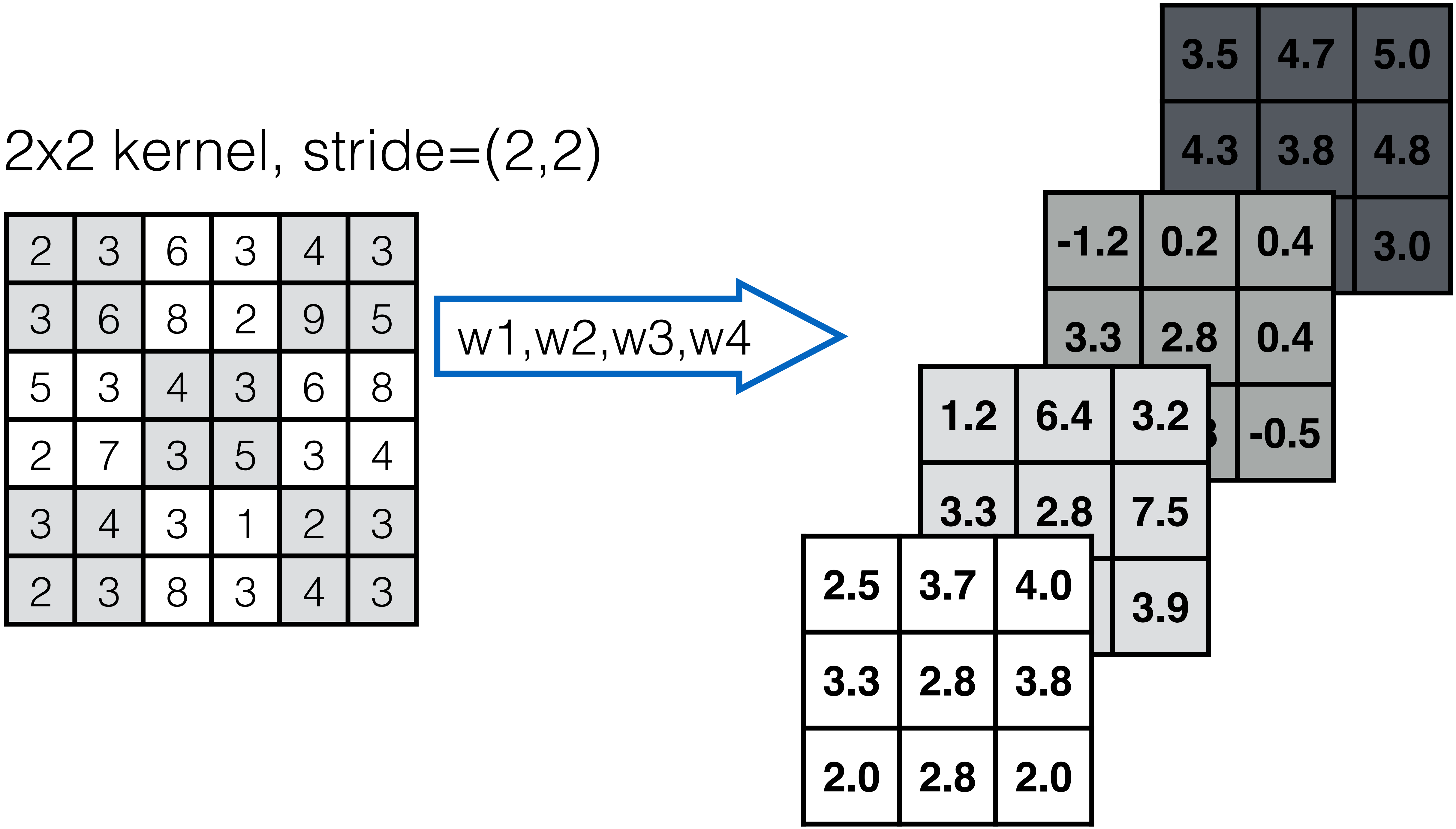


Convolutions with multiple filters

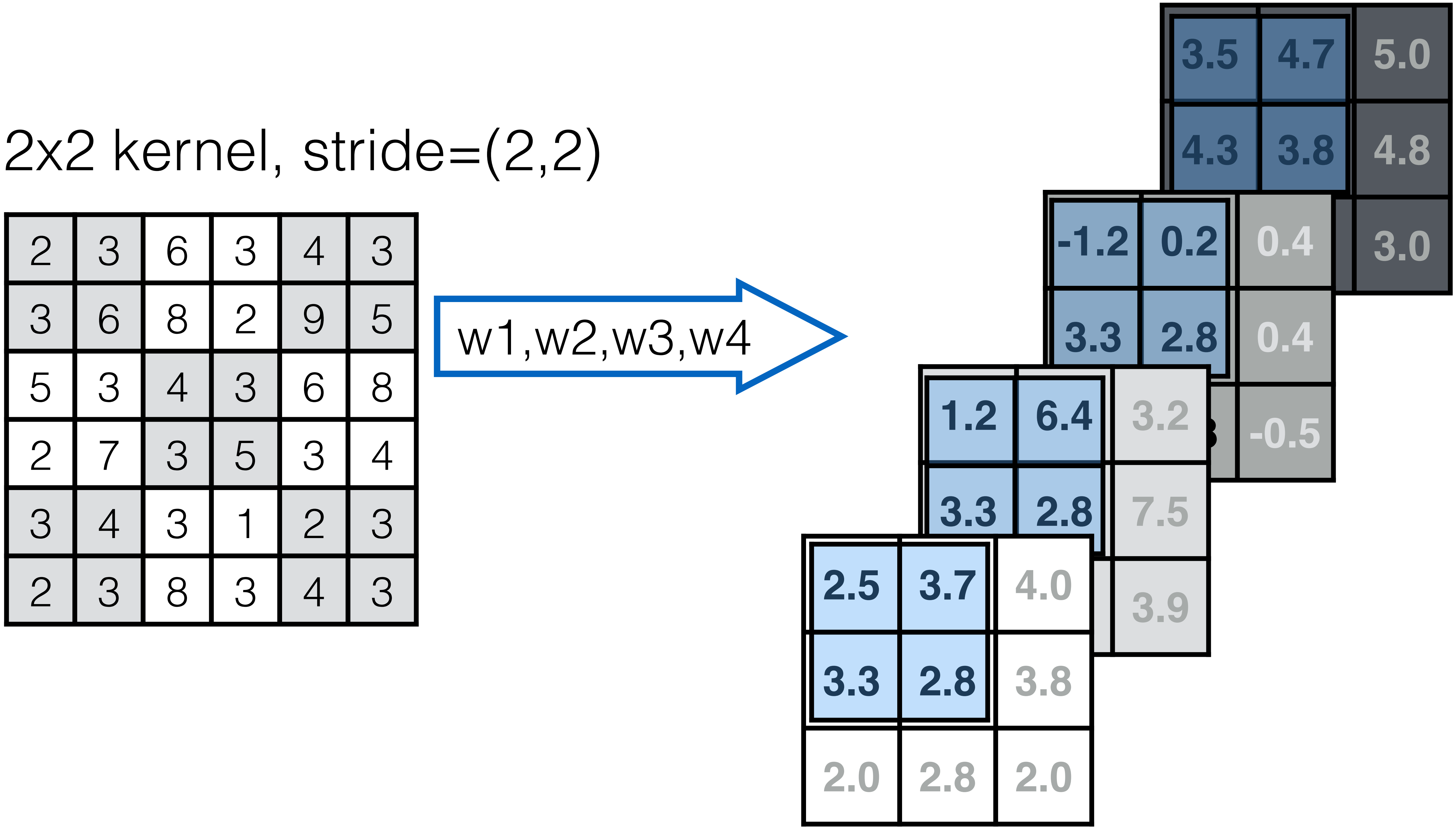
2x2 kernel, stride=(2,2)



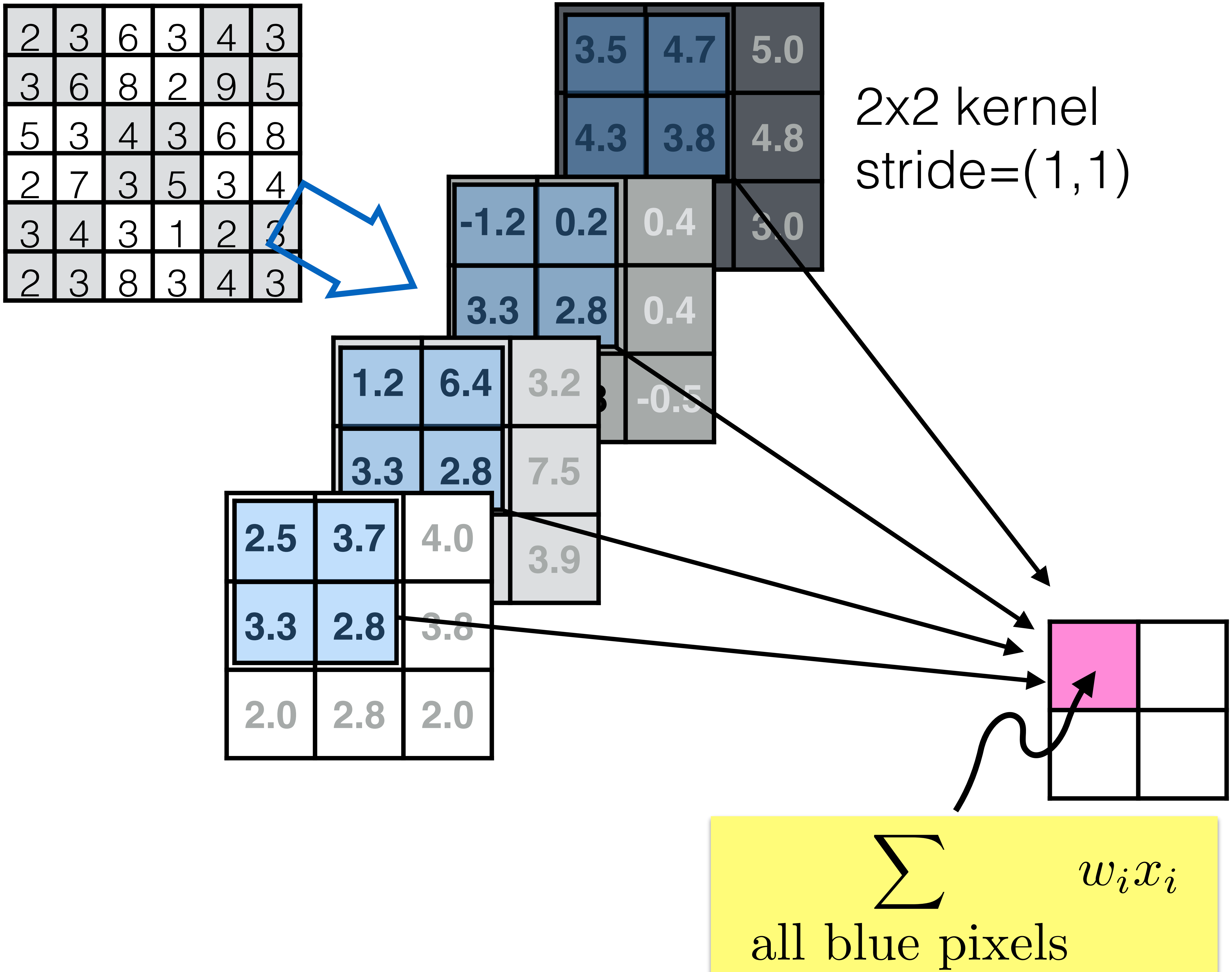
2x2 kernel, stride=(2,2)

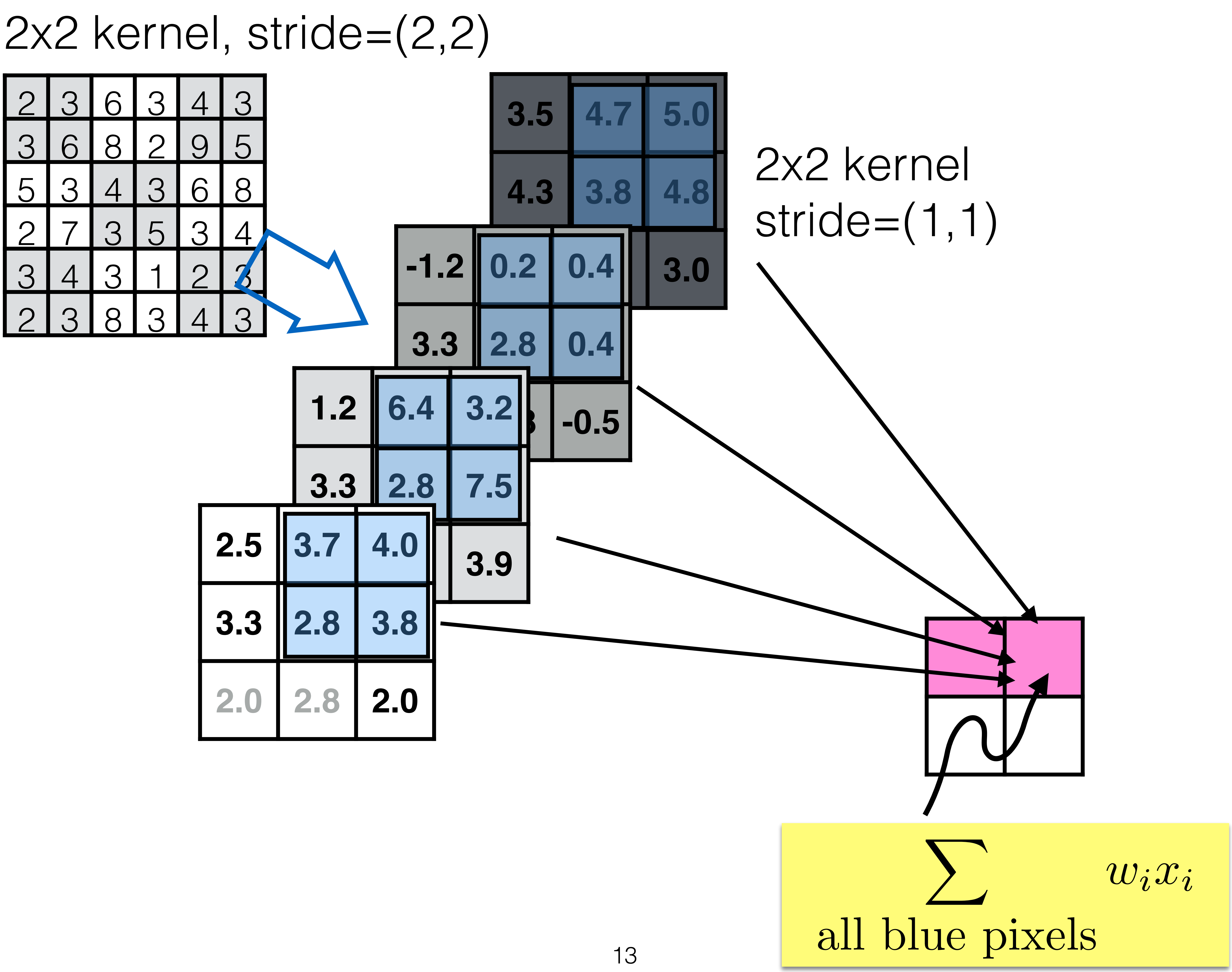


2x2 kernel, stride=(2,2)

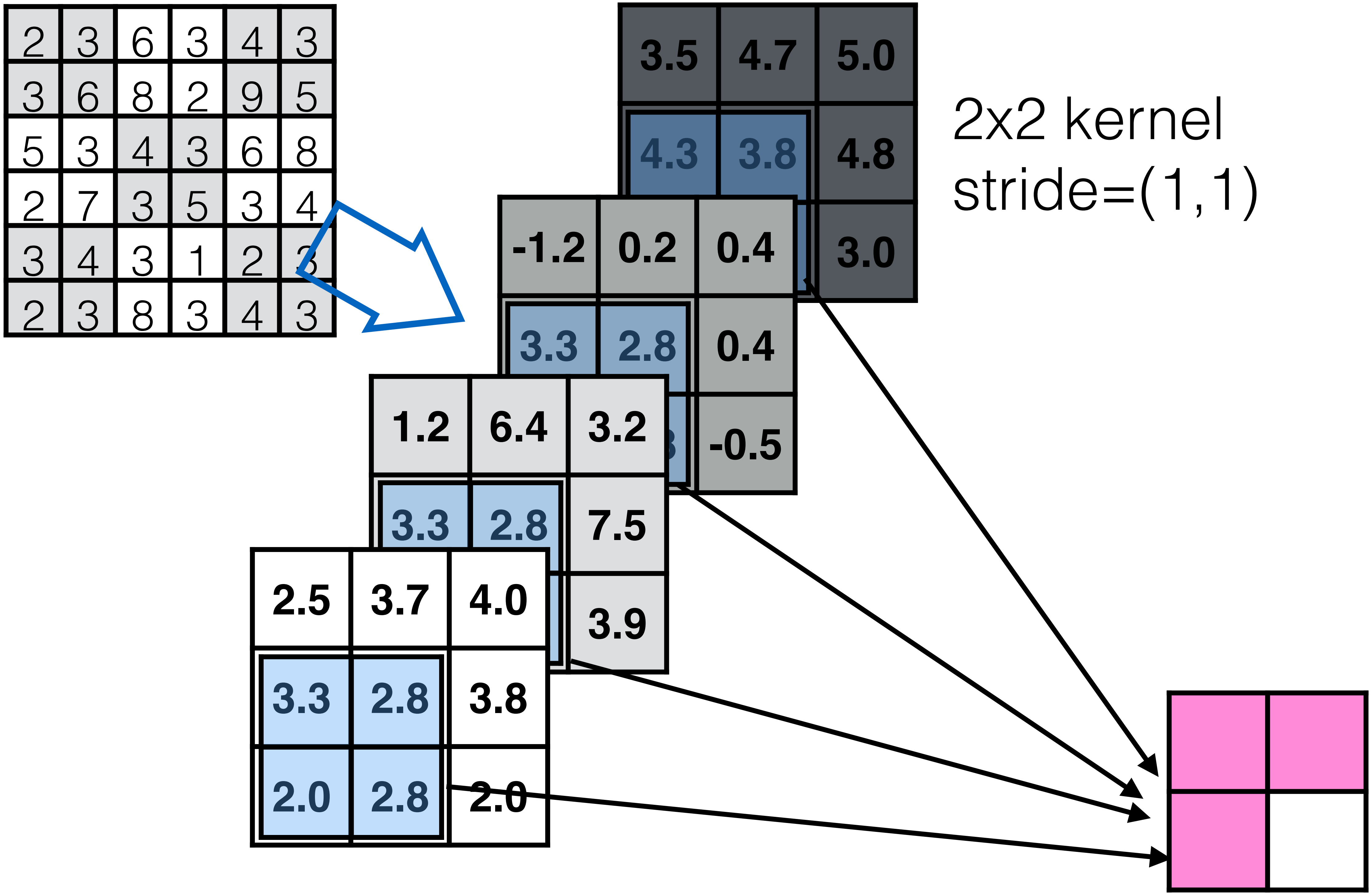


2x2 kernel, stride=(2,2)

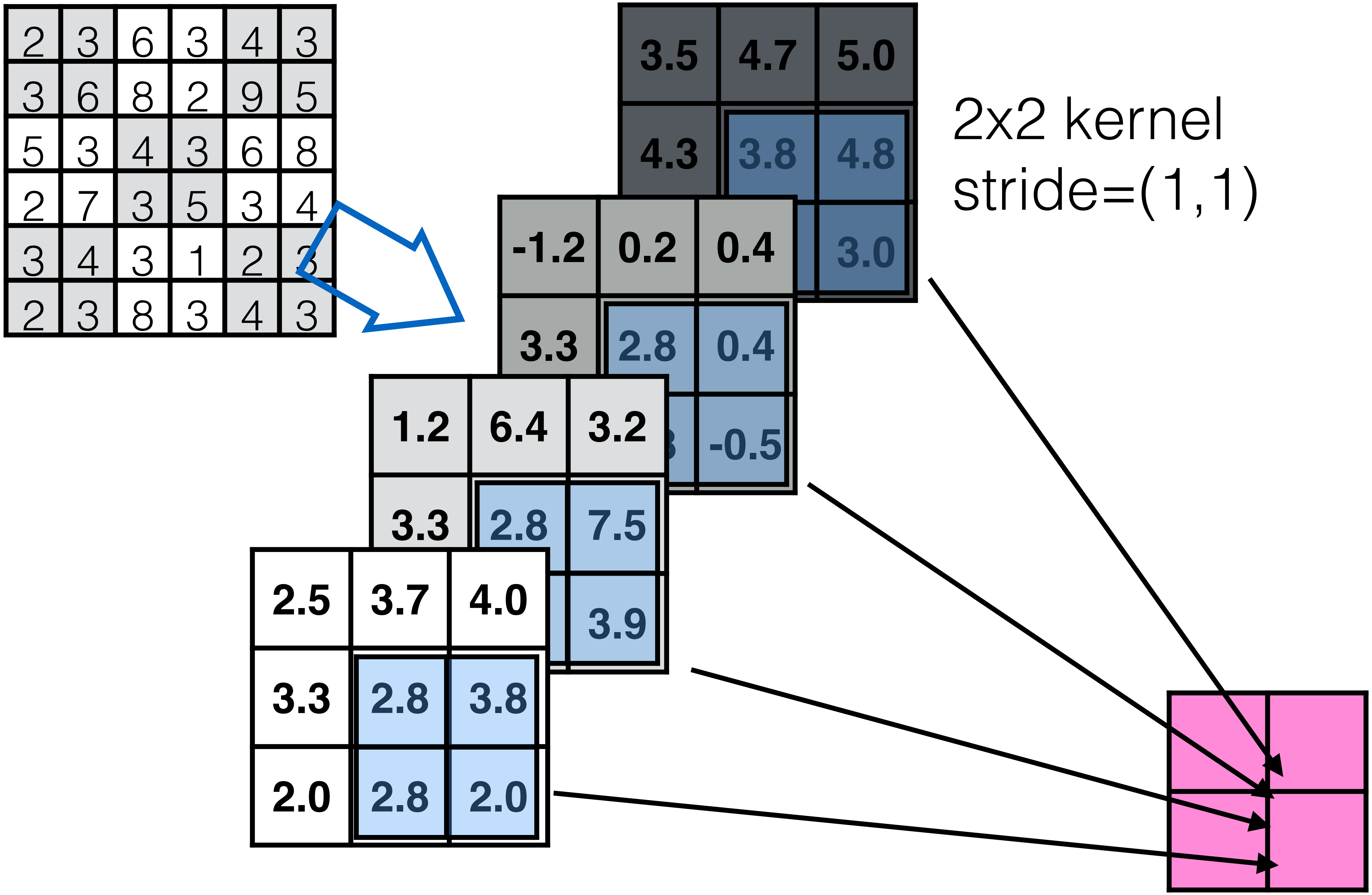



2x2 kernel, stride=(2,2)
2x2 kernel
stride=(1,1)
$\sum_{\text{all blue pixels}} w_i x_i$

2x2 kernel, stride=(2,2)

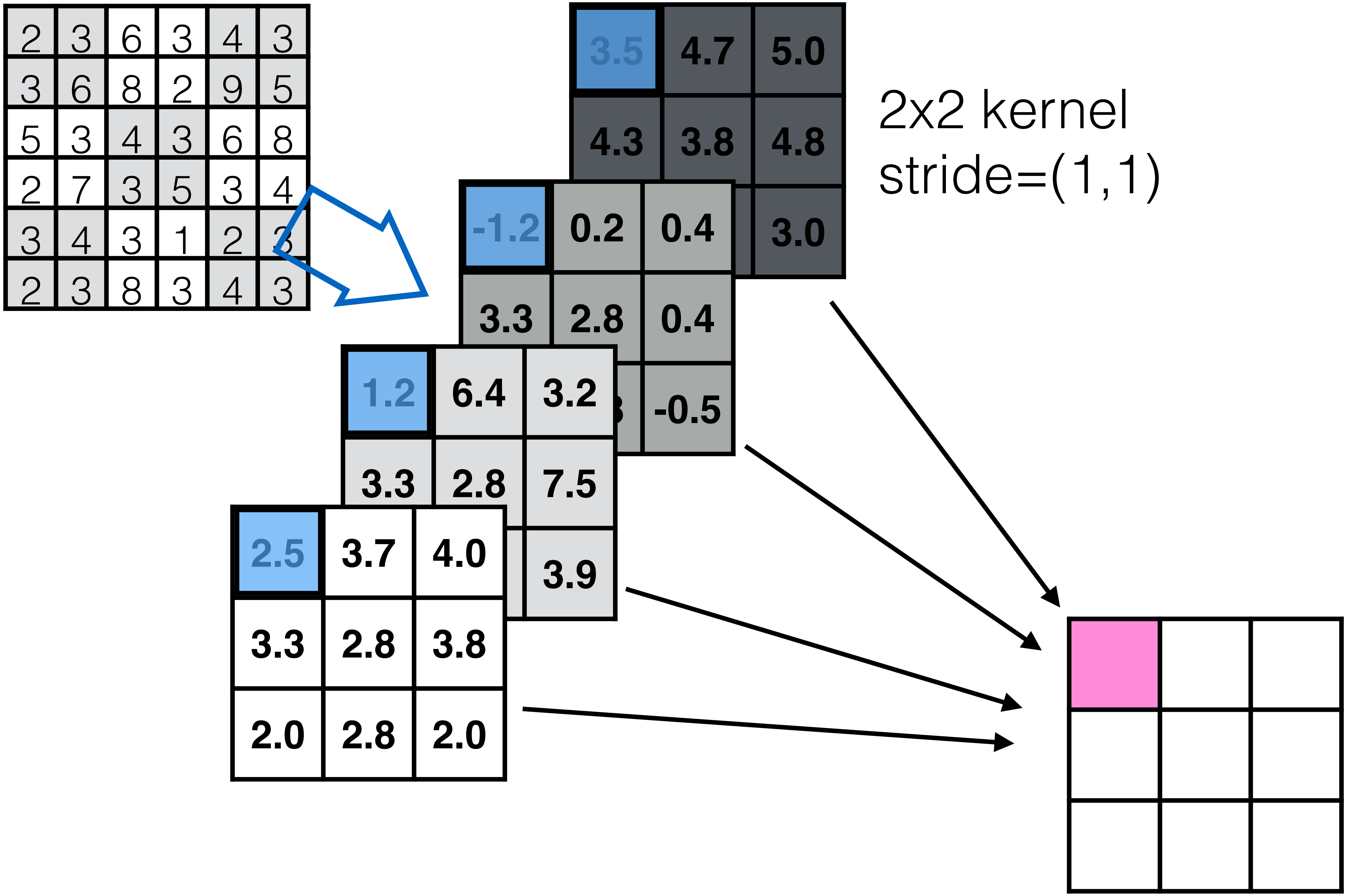


2x2 kernel, stride=(2,2)



1x1 convolutions

2x2 kernel, stride=(2,2)



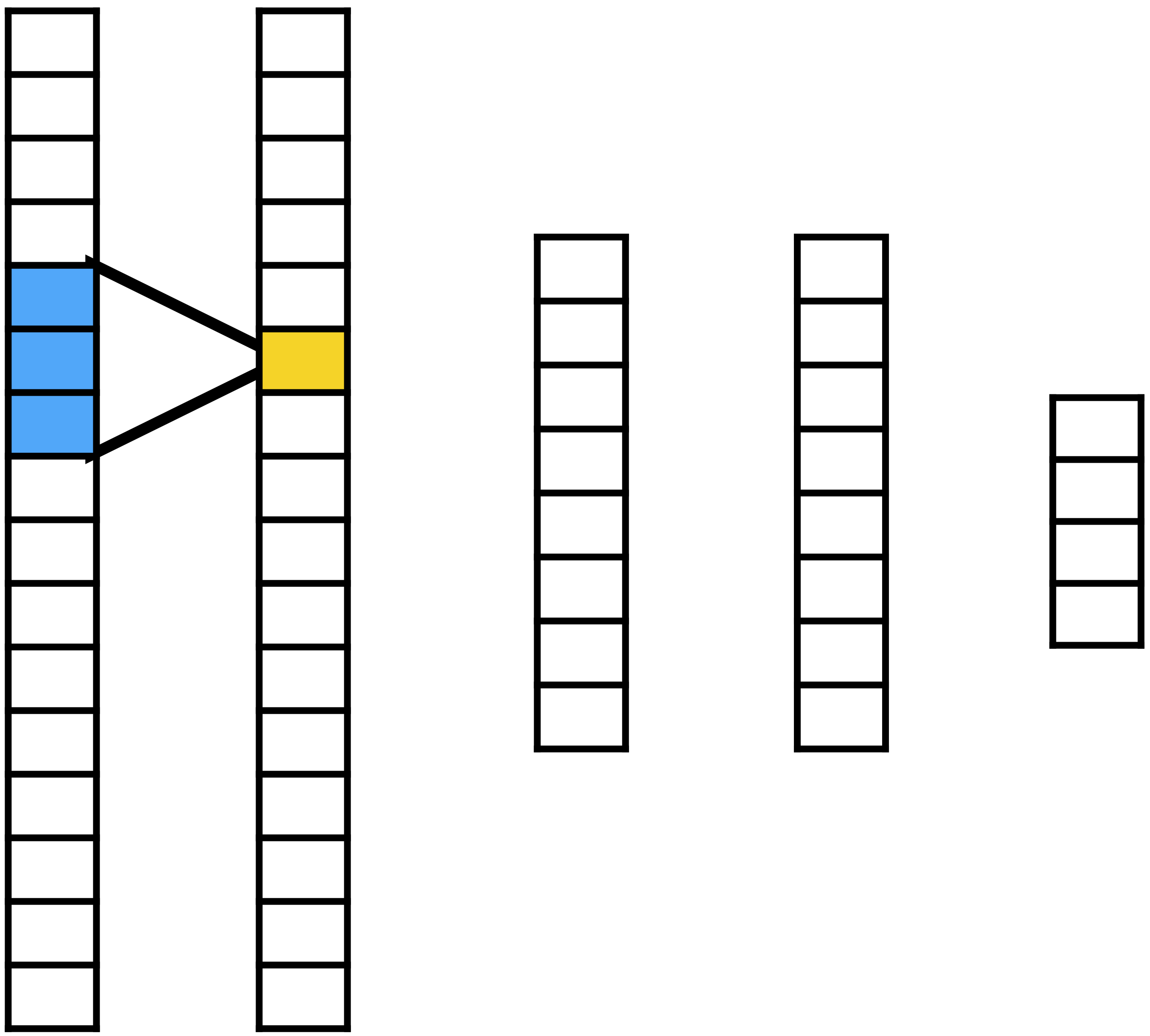
The effect of multiple filters is to do arithmetic with images

add & subtract images

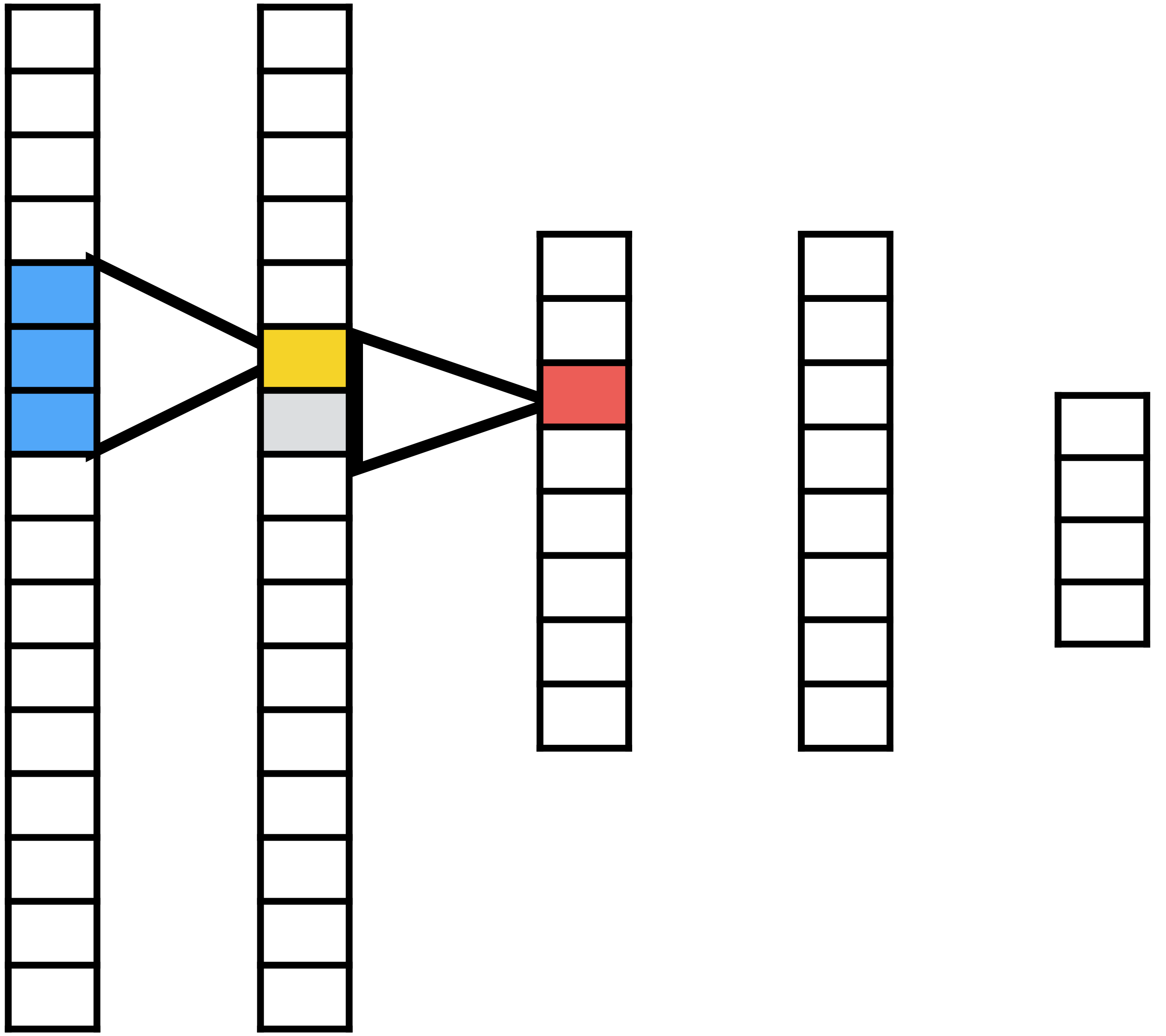
Show Desmos plot for convolution

Receptive field

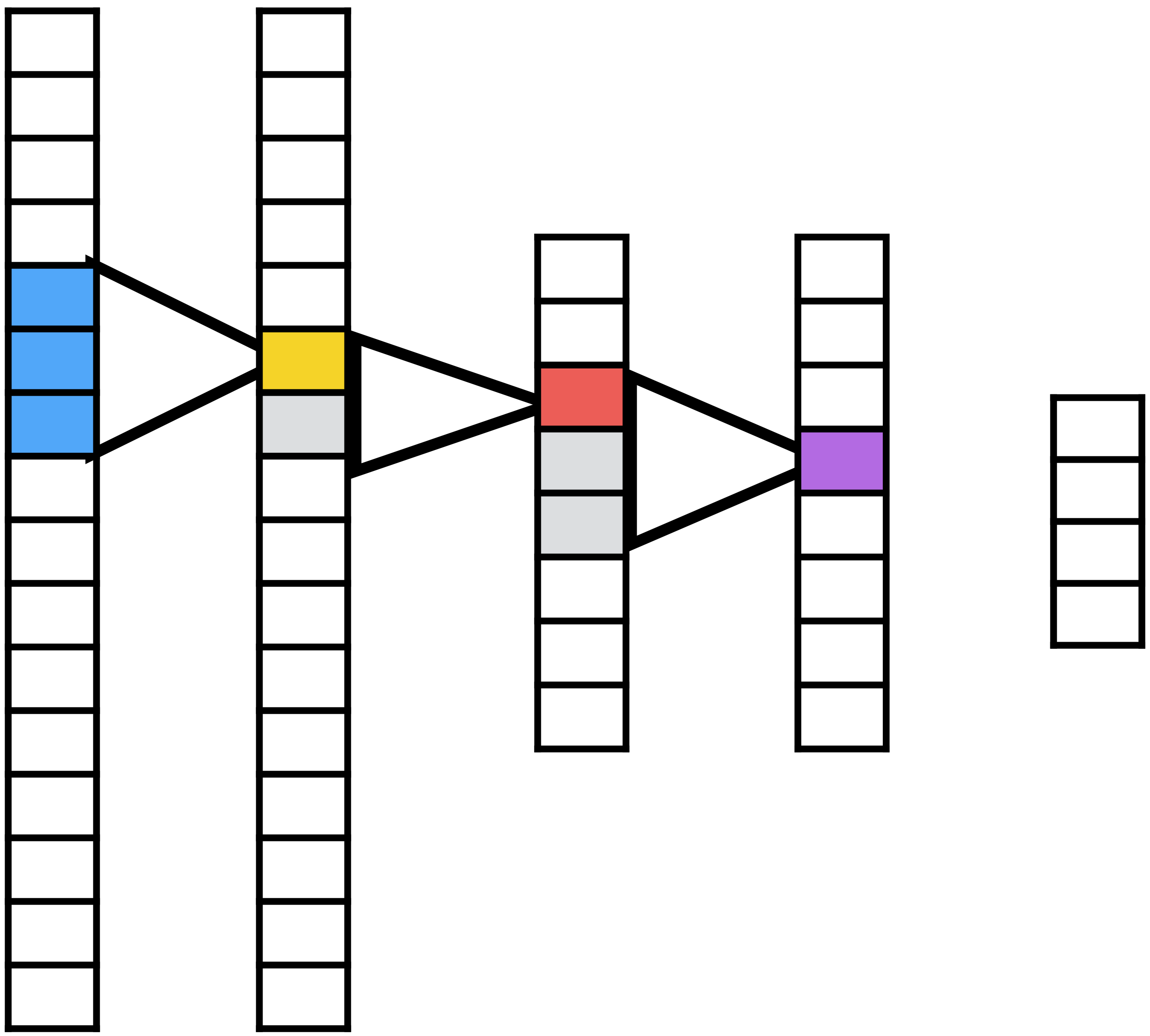
Concept of receptive field



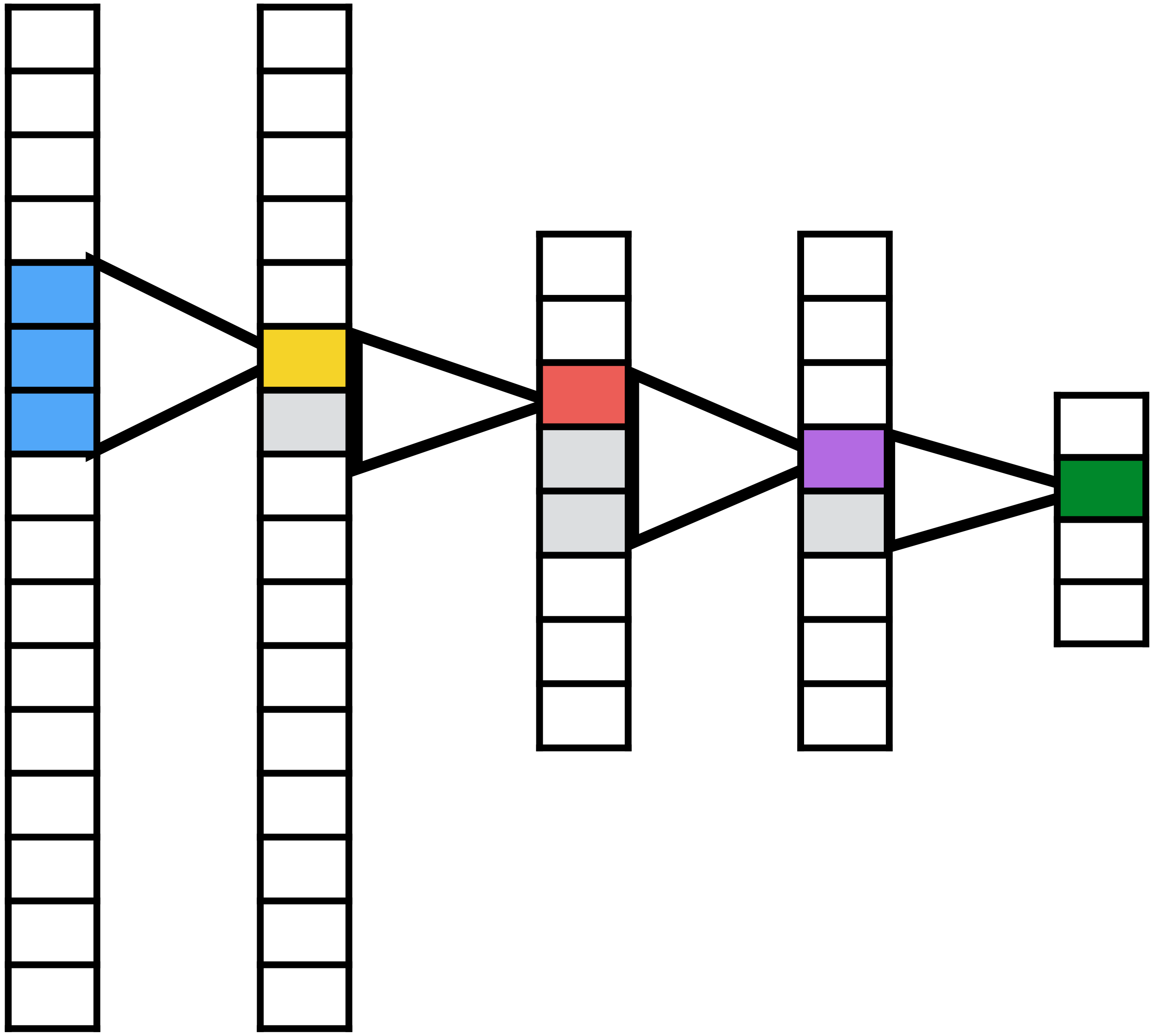
Concept of receptive field



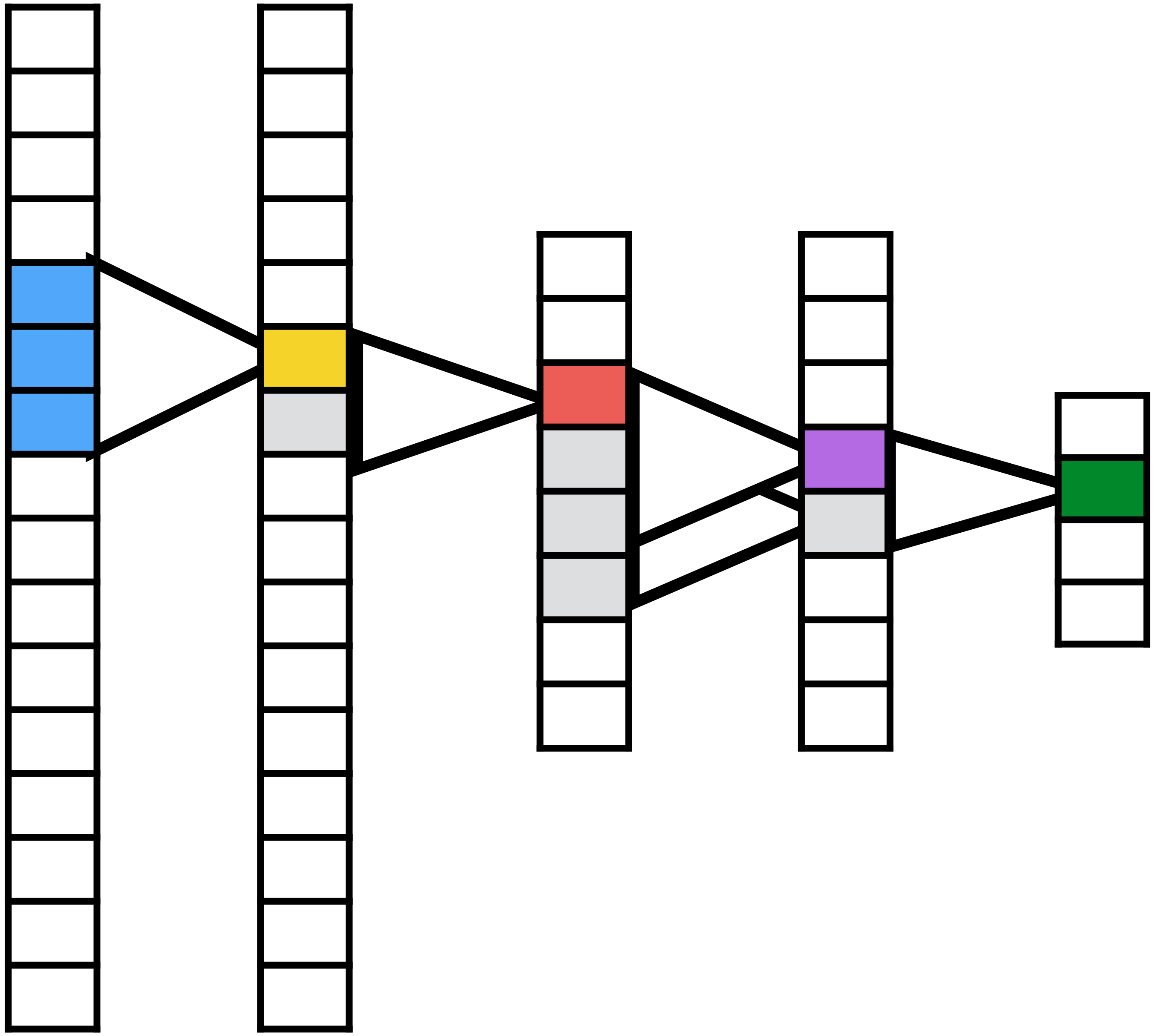
Concept of receptive field



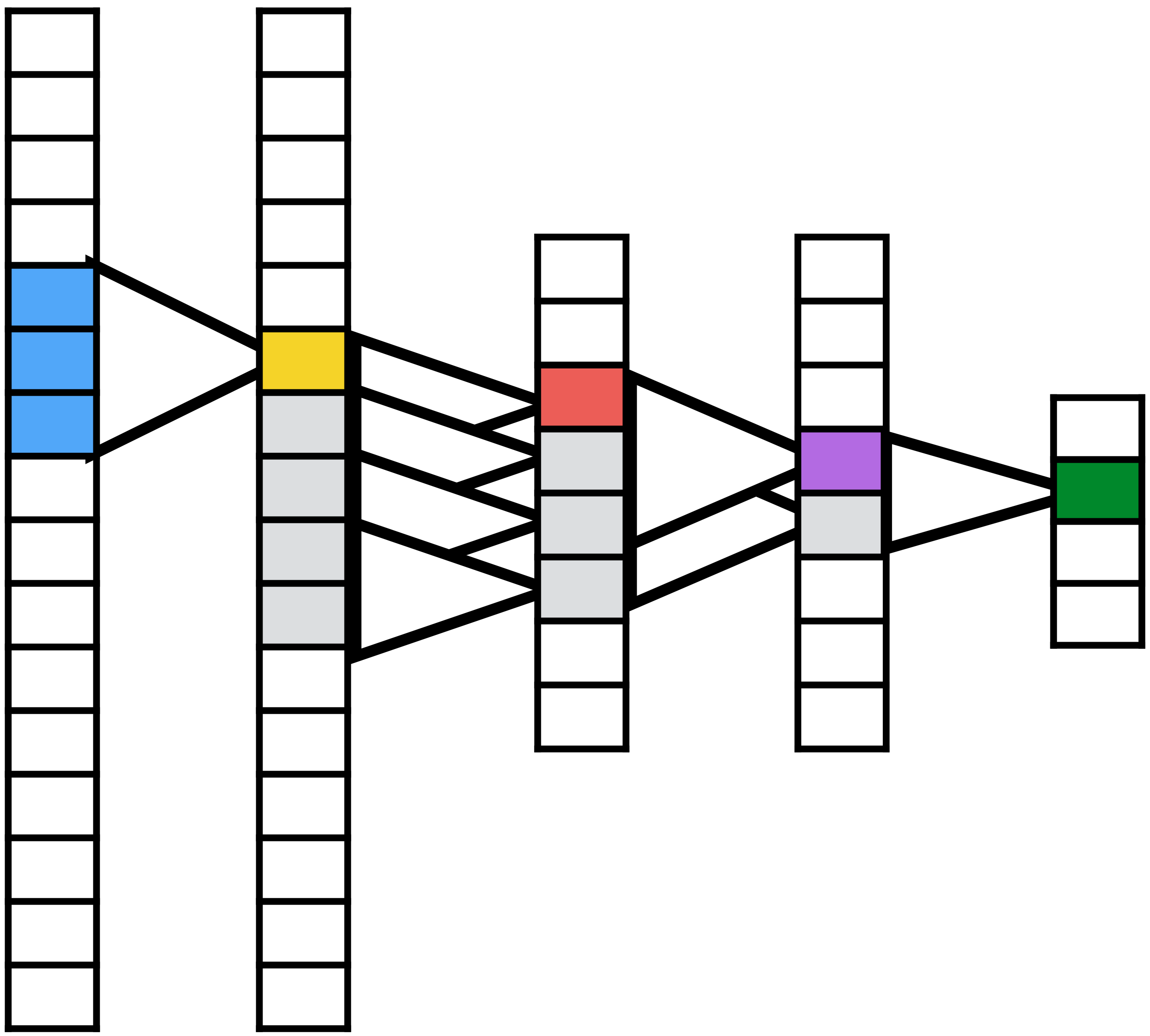
Concept of receptive field



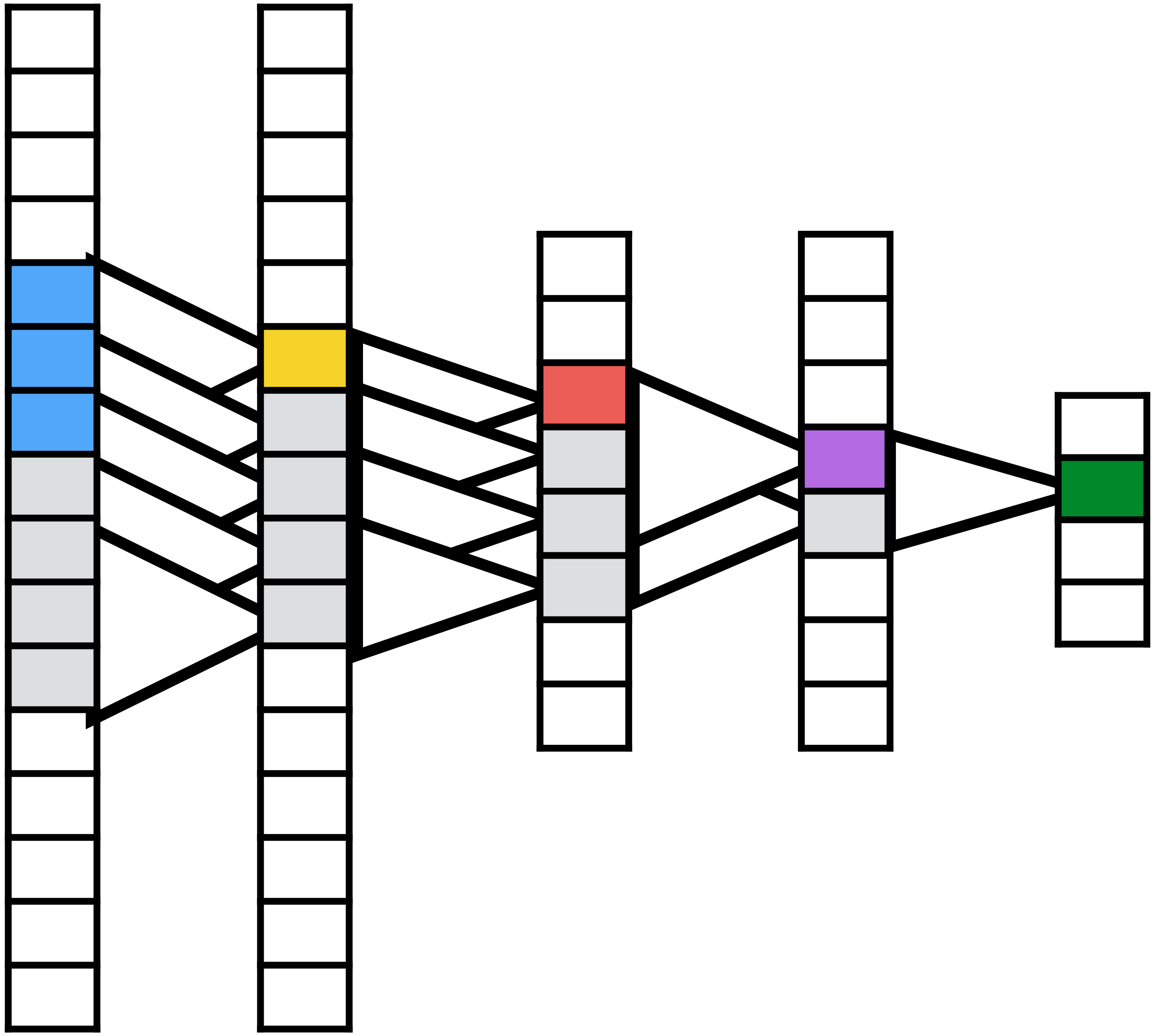
Concept of receptive field



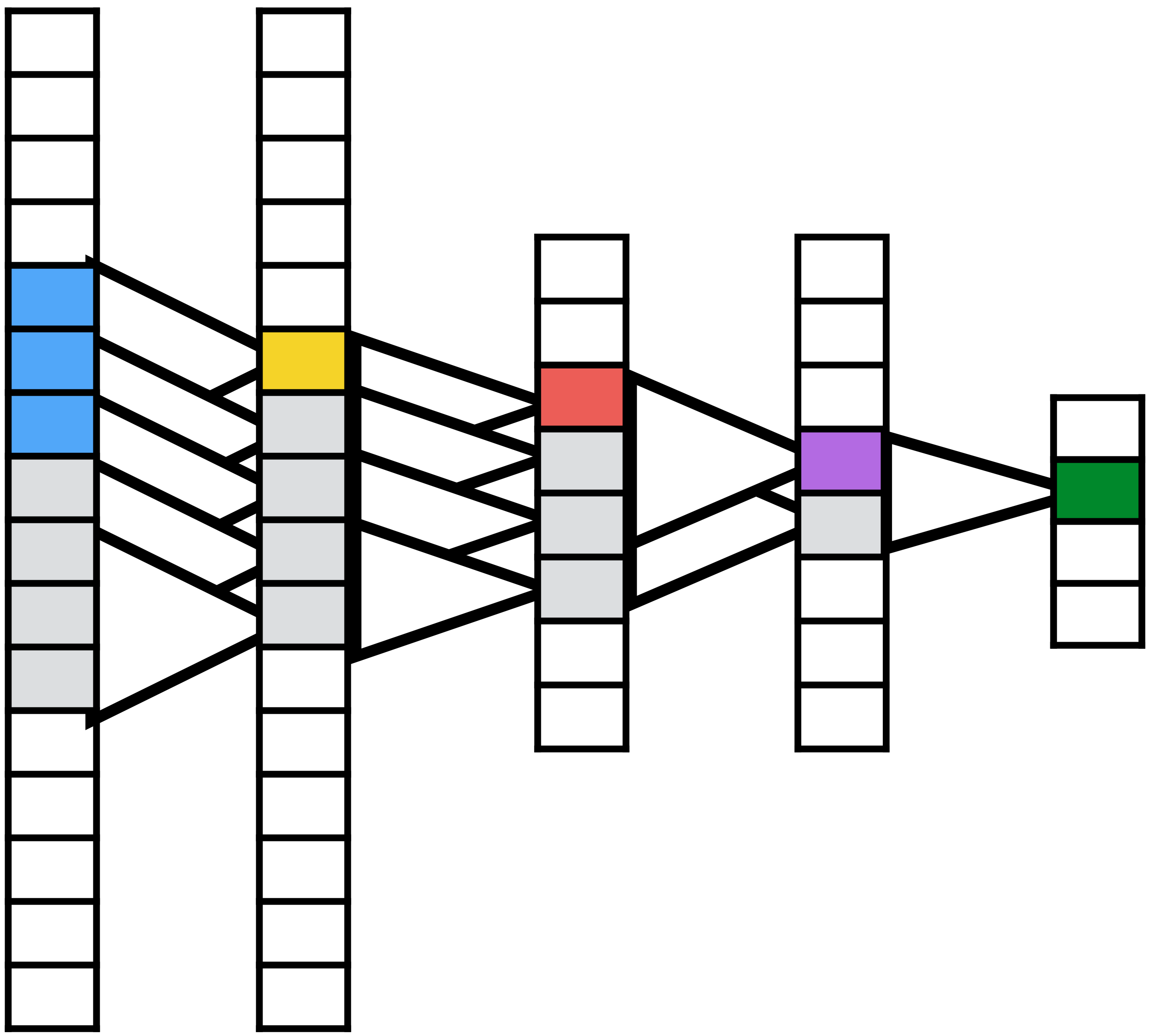
Concept of receptive field

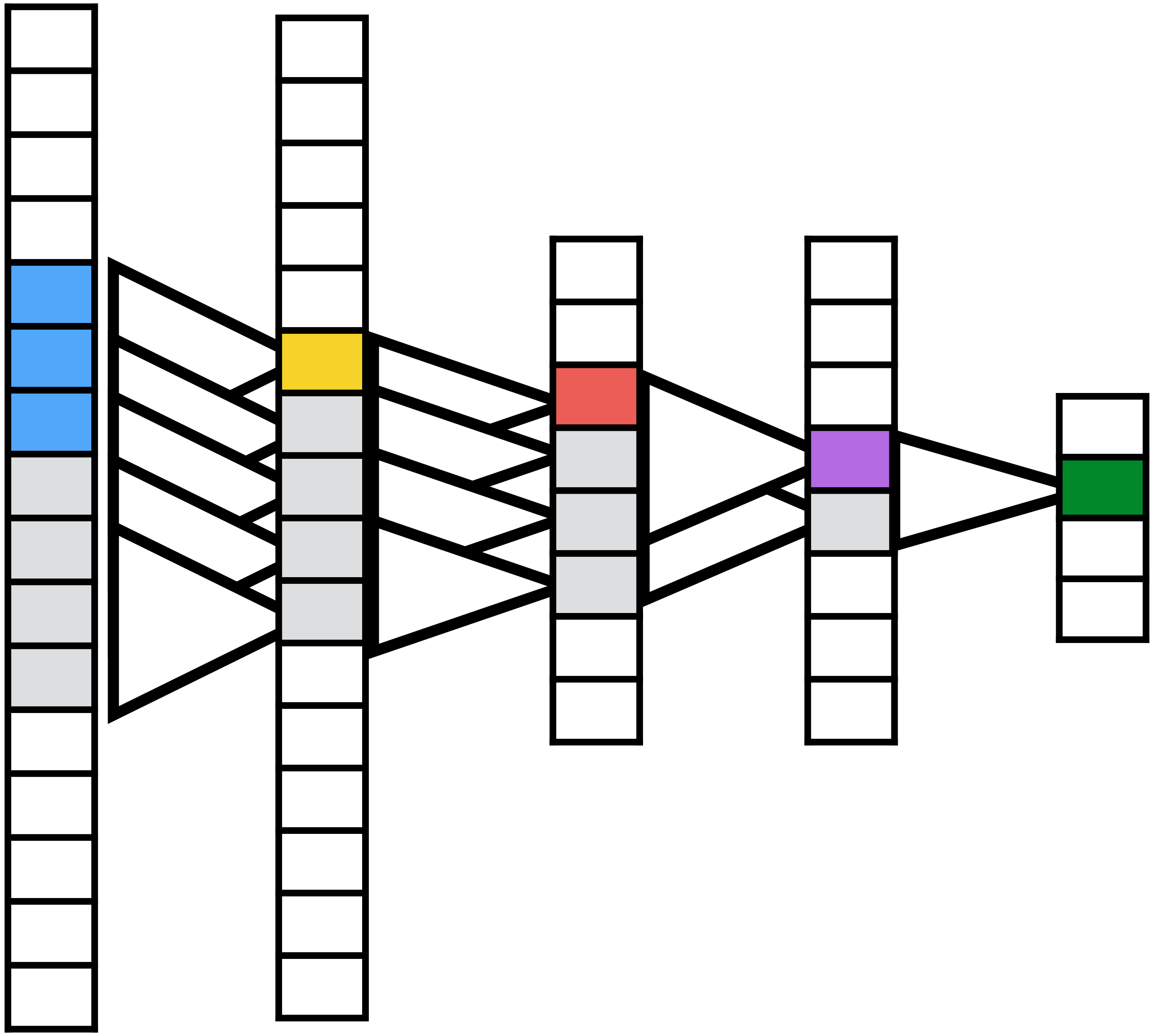


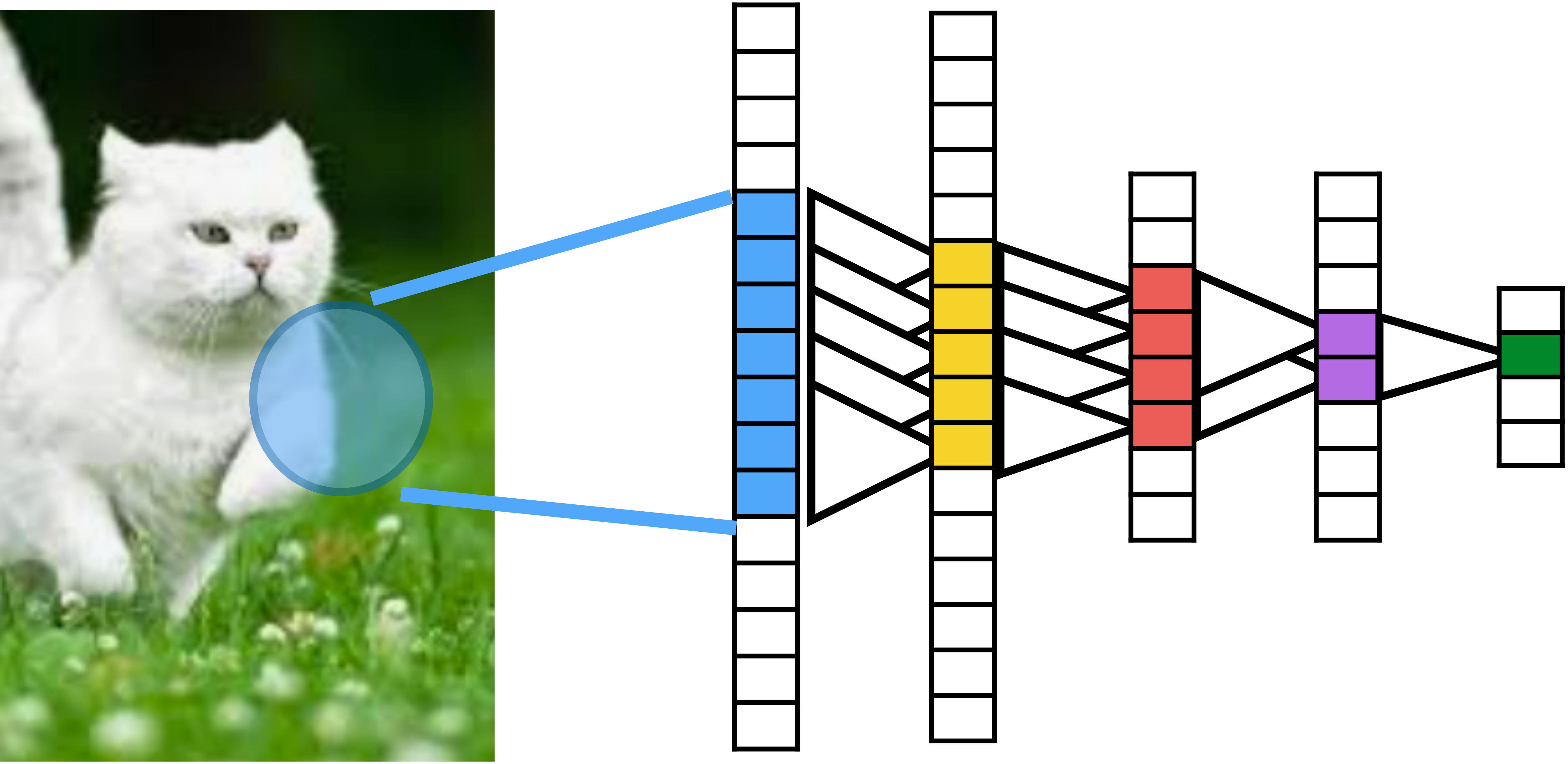
Concept of receptive field

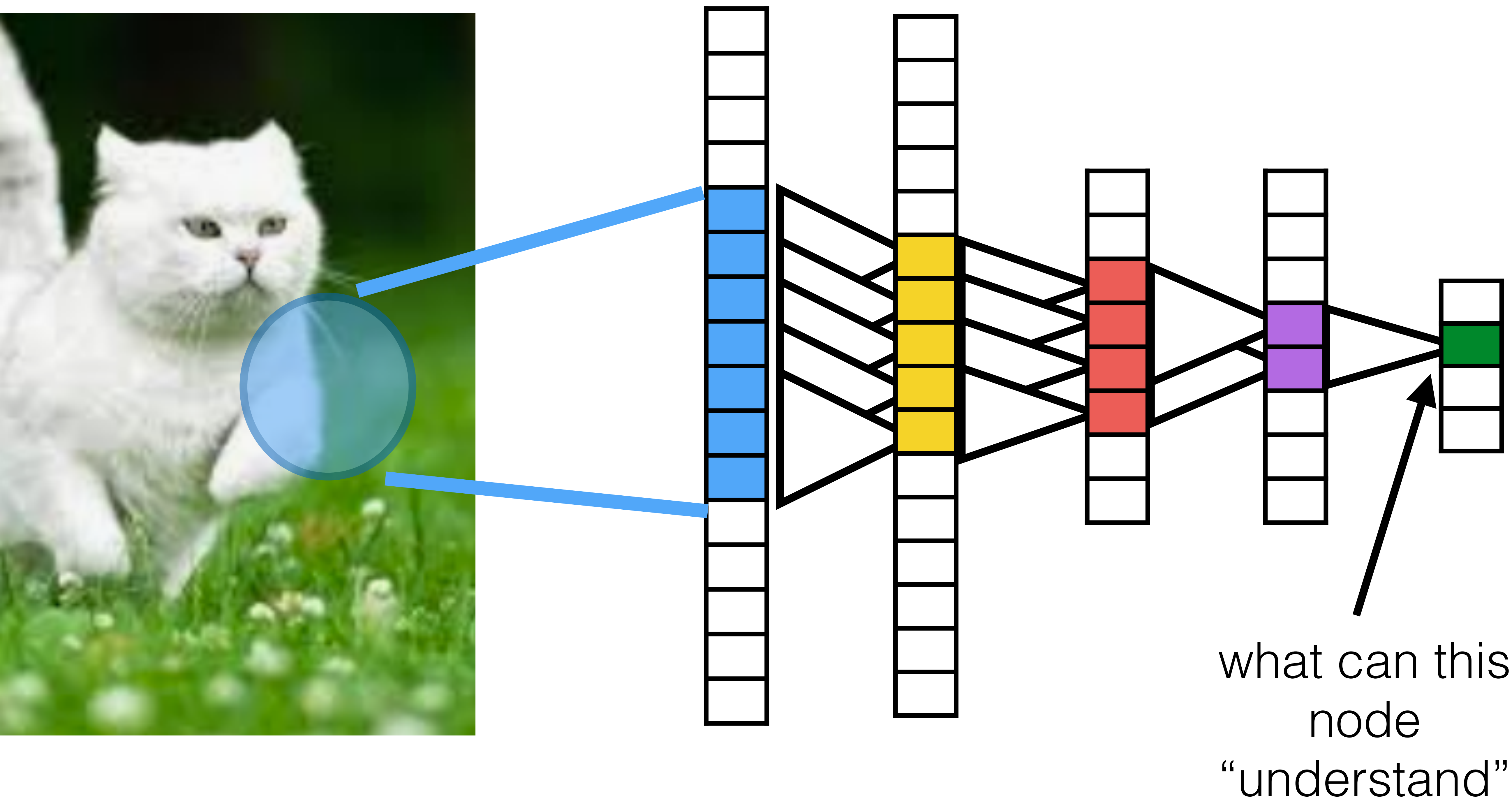


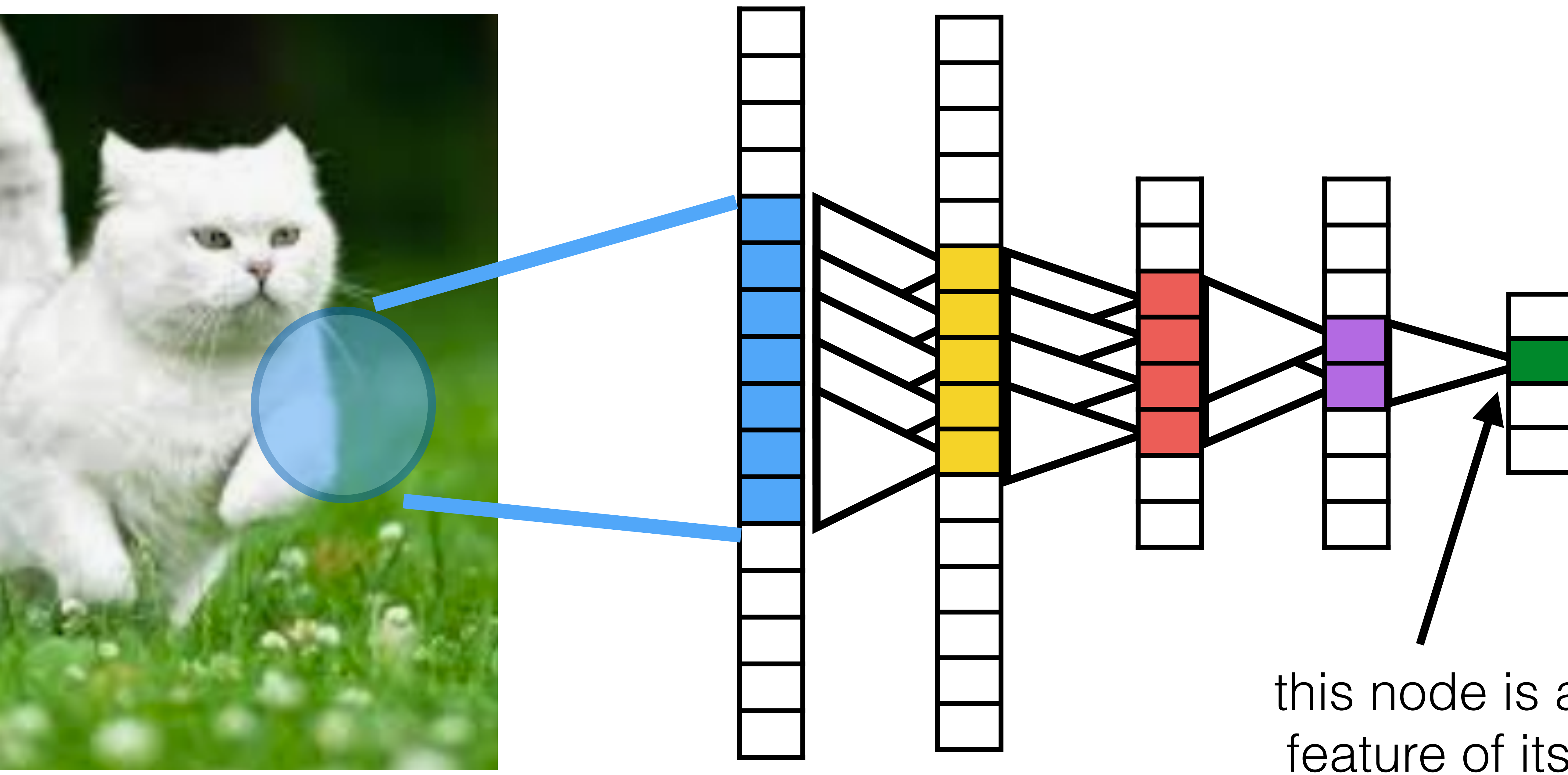
What is the lesson here?





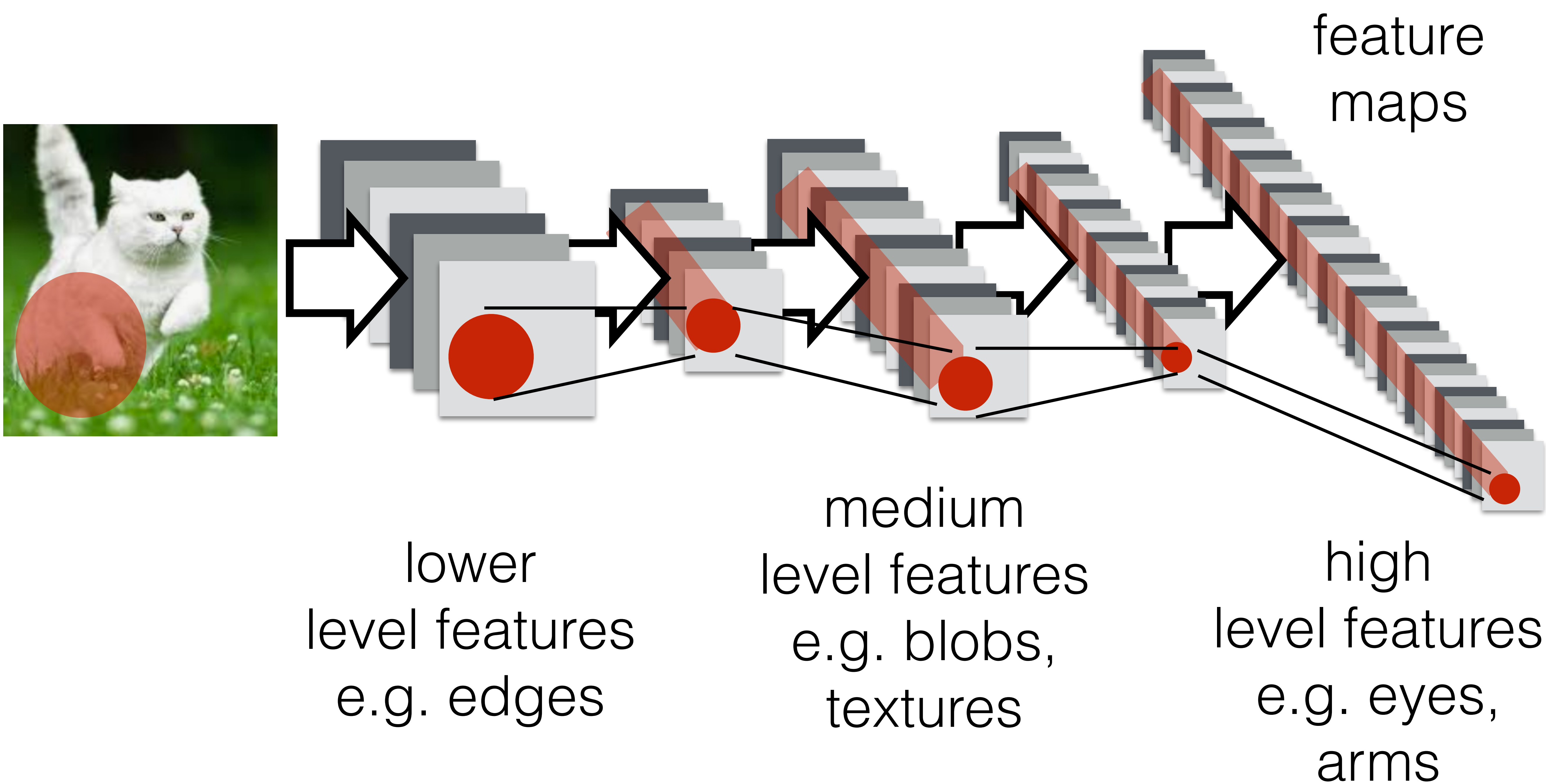



what can this
node
“understand”



this node is a
feature of its
receptive field

CNN part of the network is a image feature extractor for meaningful high level features



Overall architecture of CNN

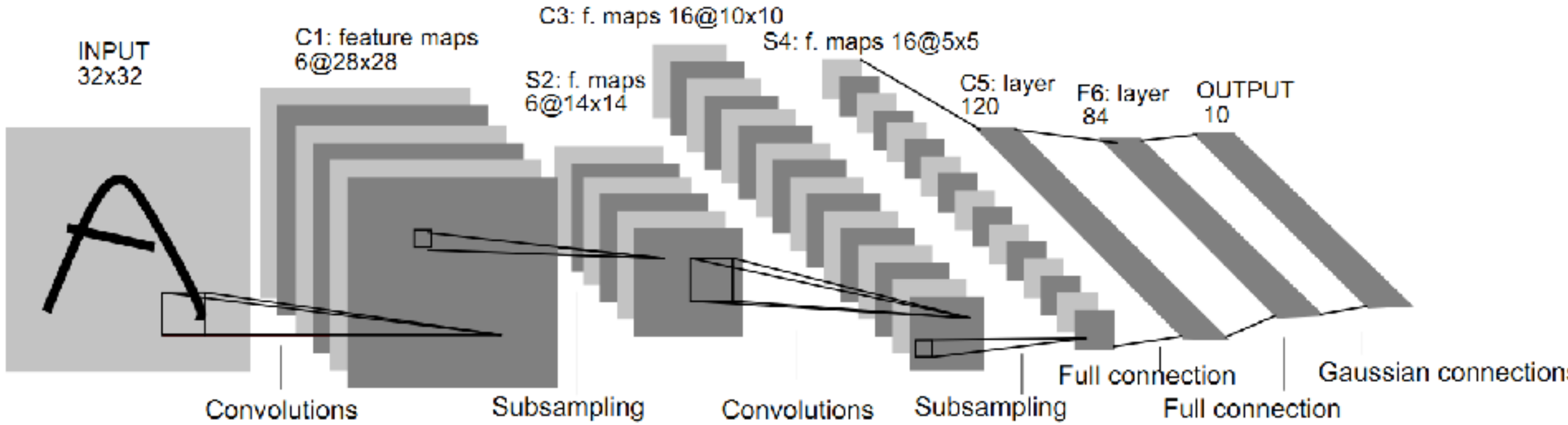
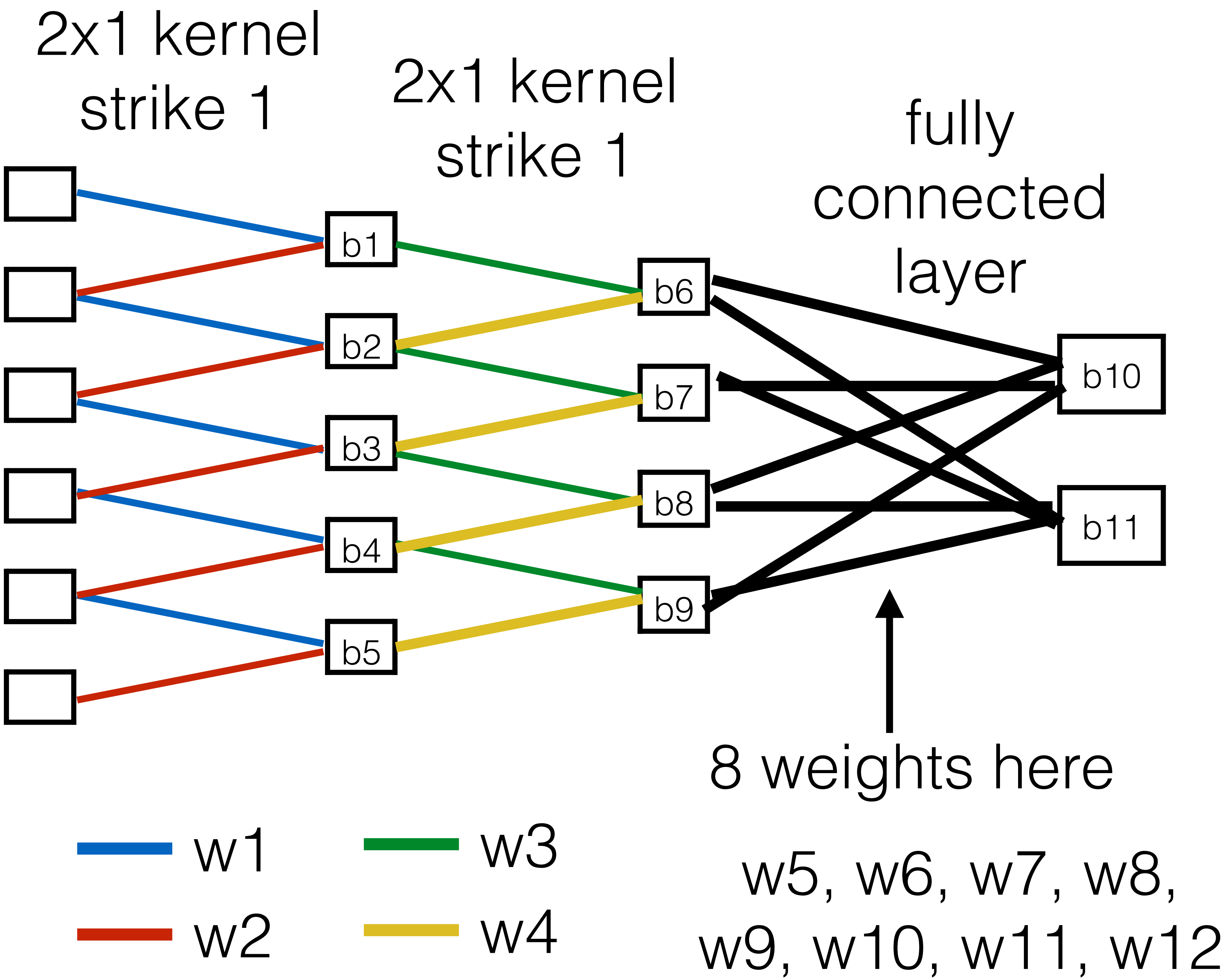
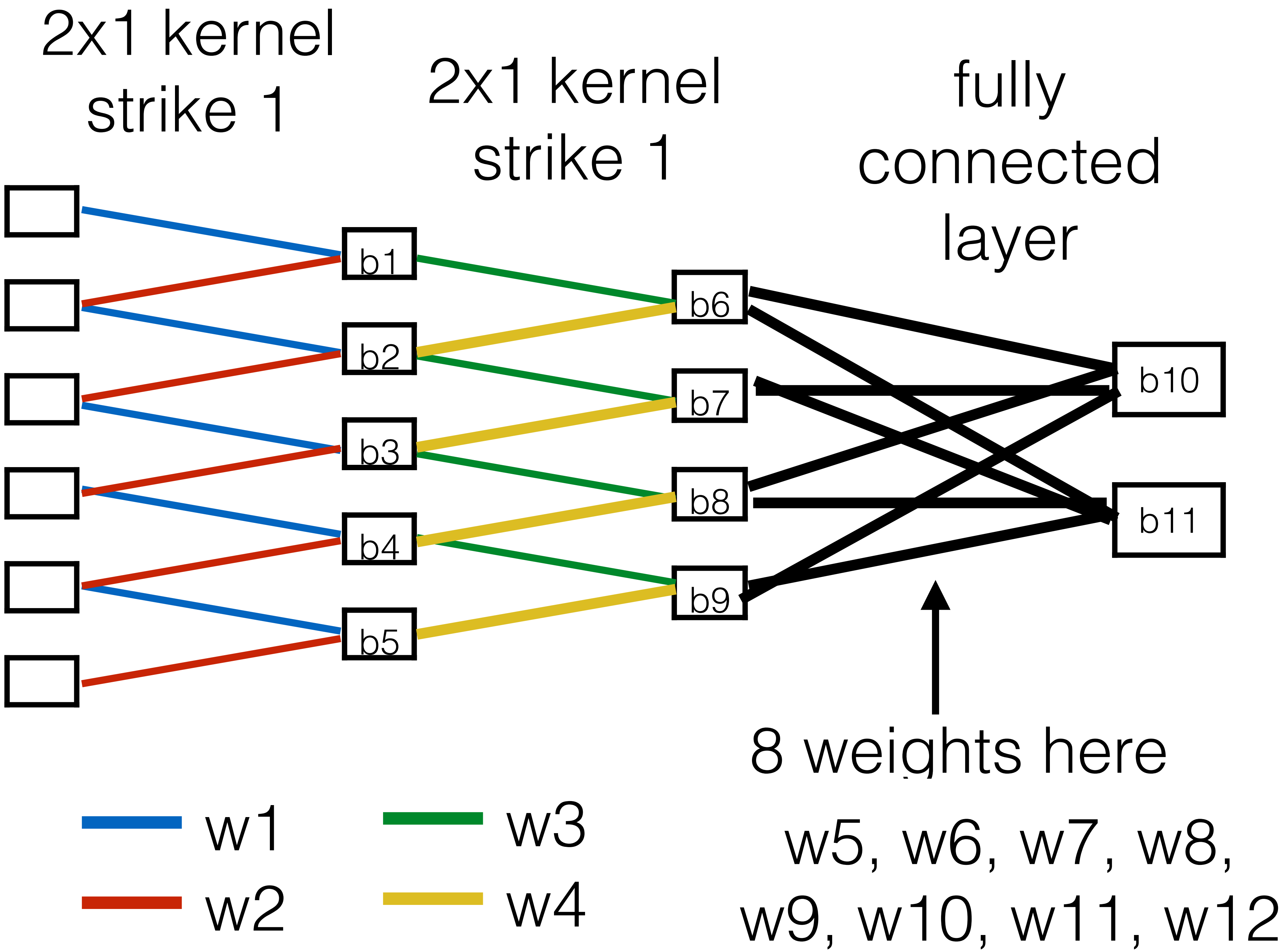


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

https://world4jason.gitbooks.io/research-log/content/deepLearning/CNN/Model%20&%20ImgNet/lenet.html

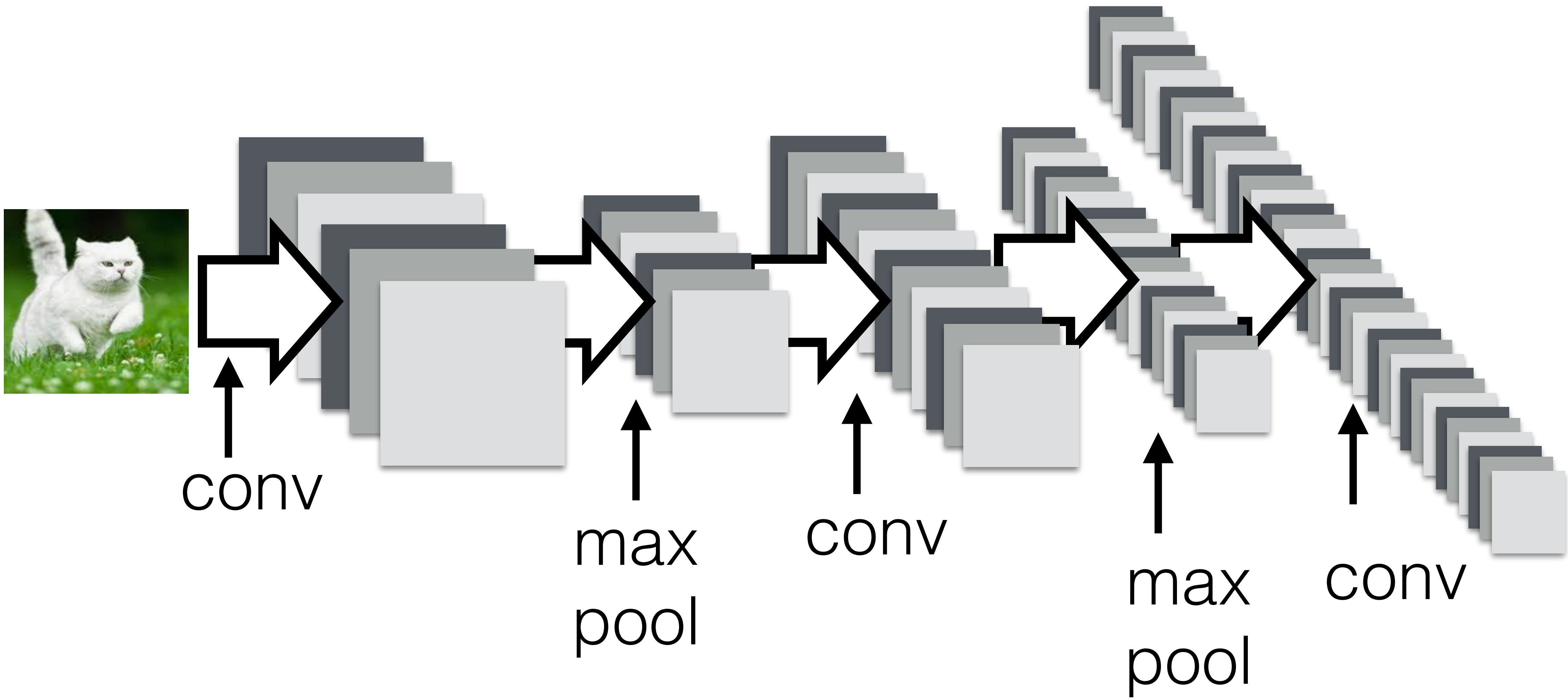
Training of CNN (convolutional neural networks)





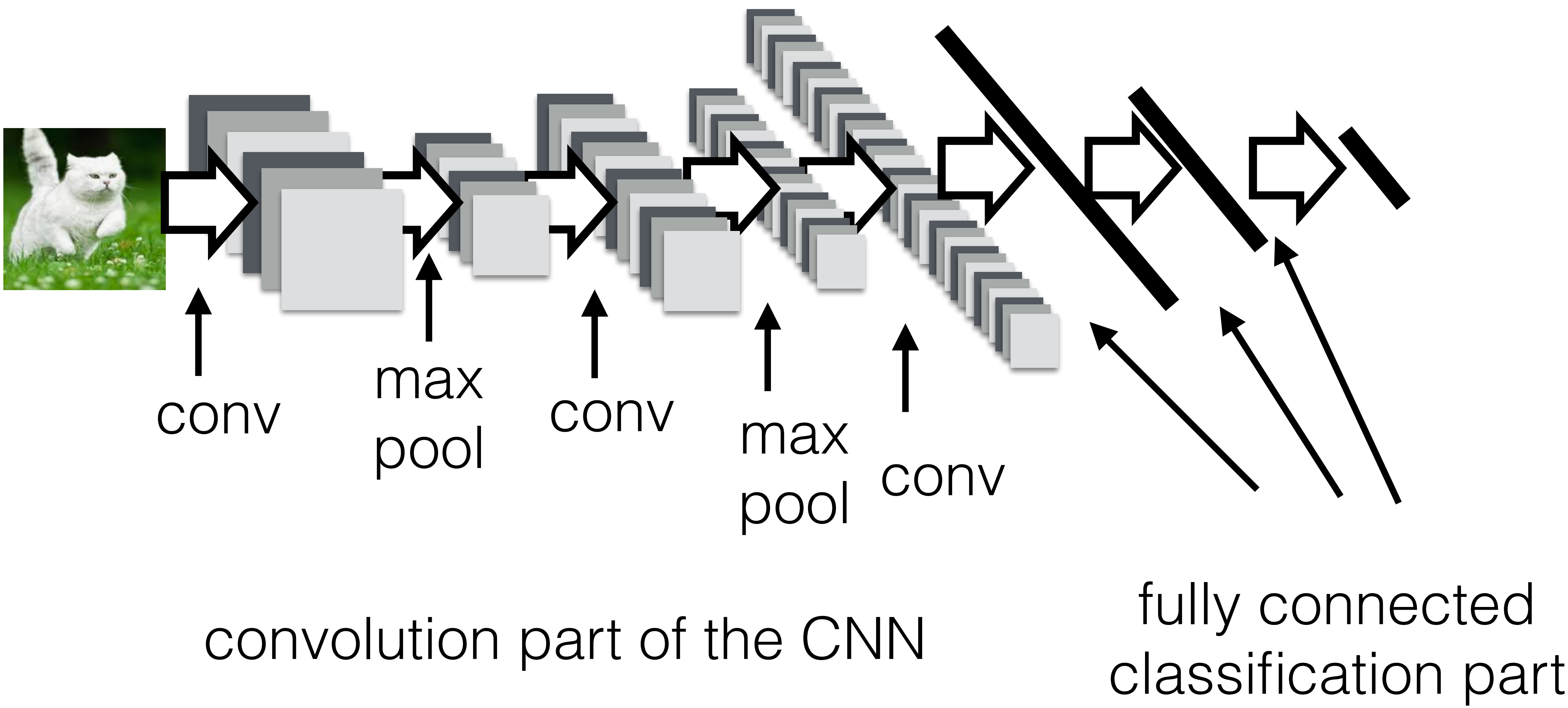
Using back propagation, minimize loss by adjusting
w1, w2, . . . w12 & b1, . . . b11
these are called "Trainable parameters"

What exactly is a CNN?

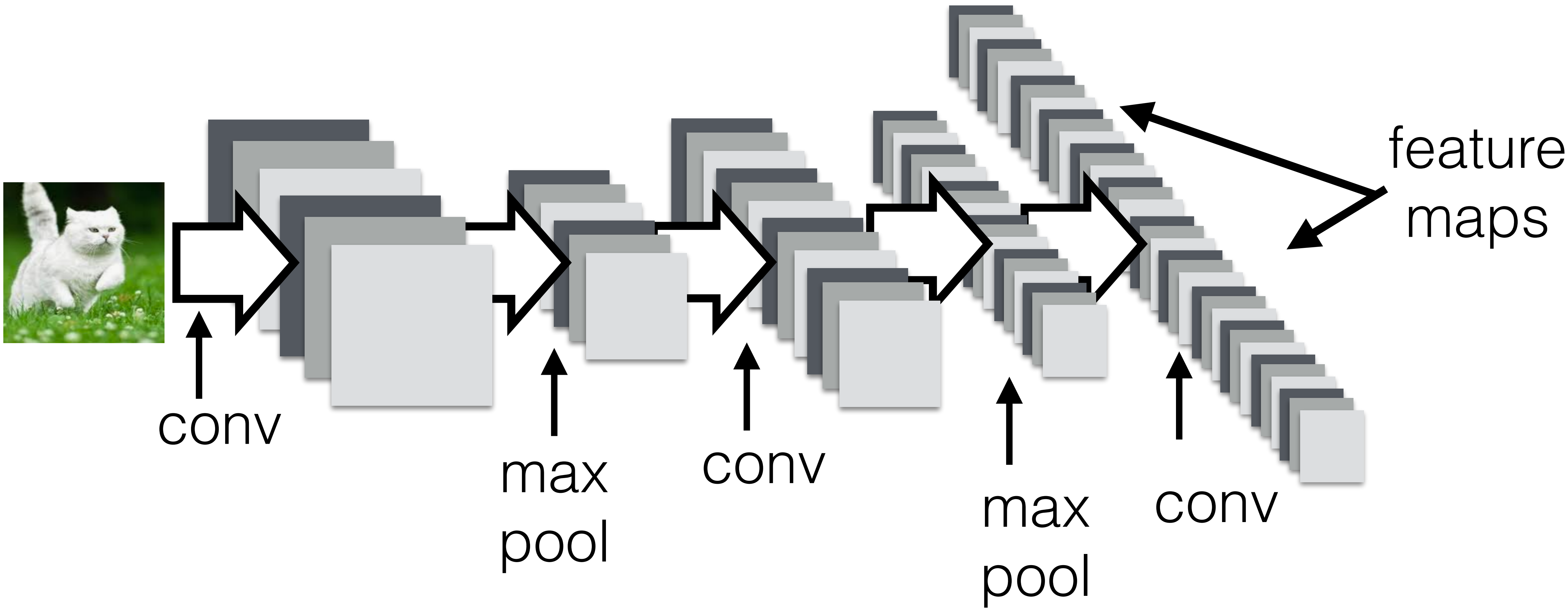


convolution part of the CNN

What exactly is a CNN?

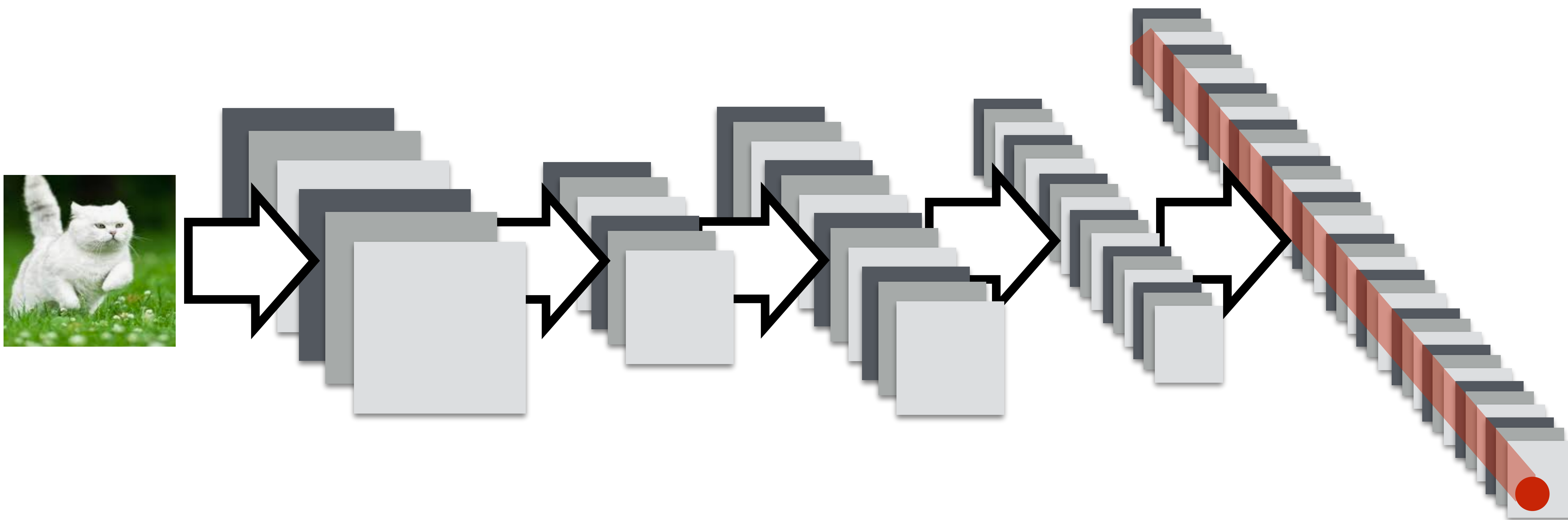


What exactly is a CNN?

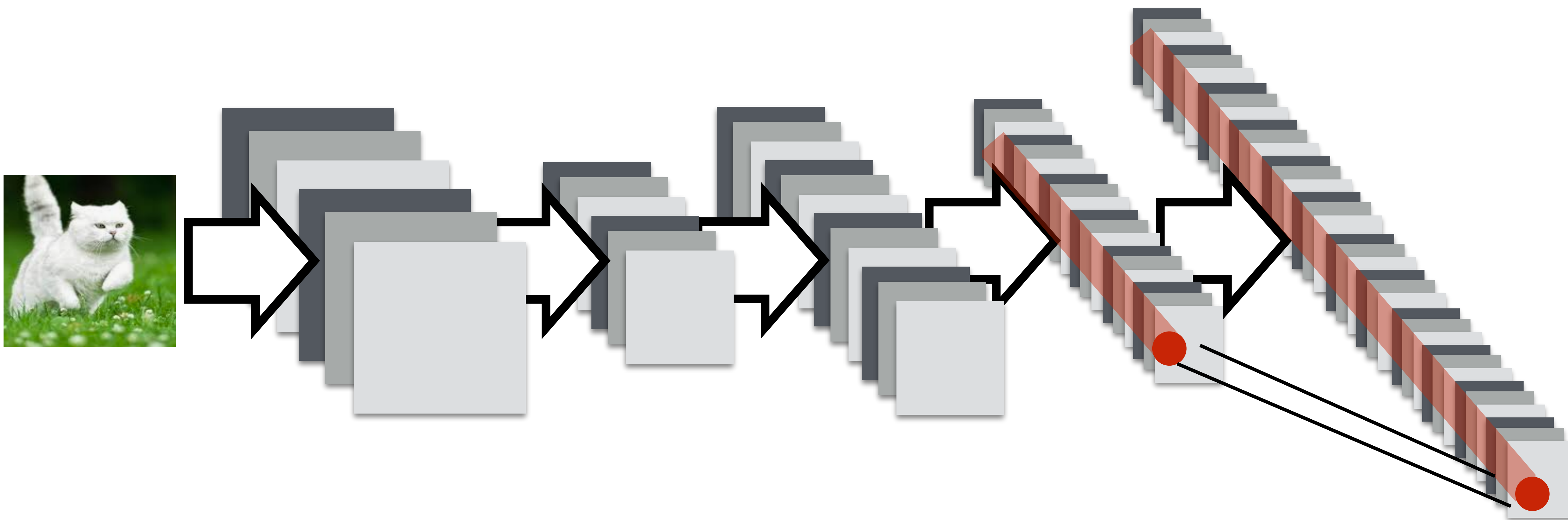


convolution part of the CNN

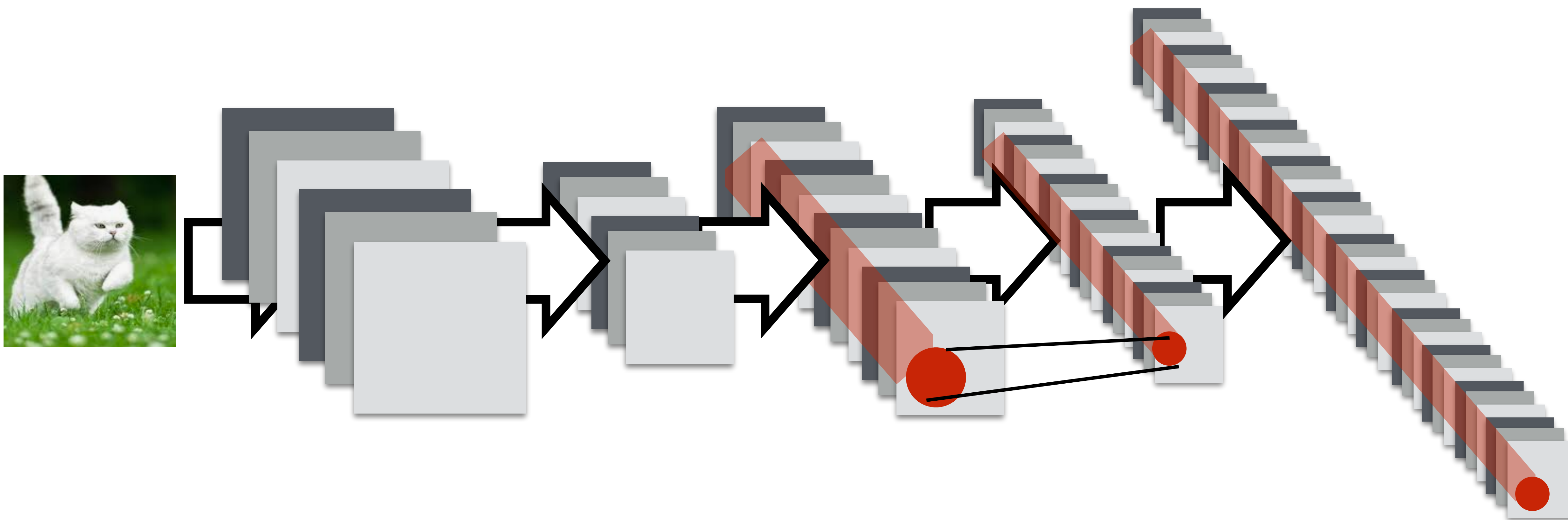
Feature maps and receptive field



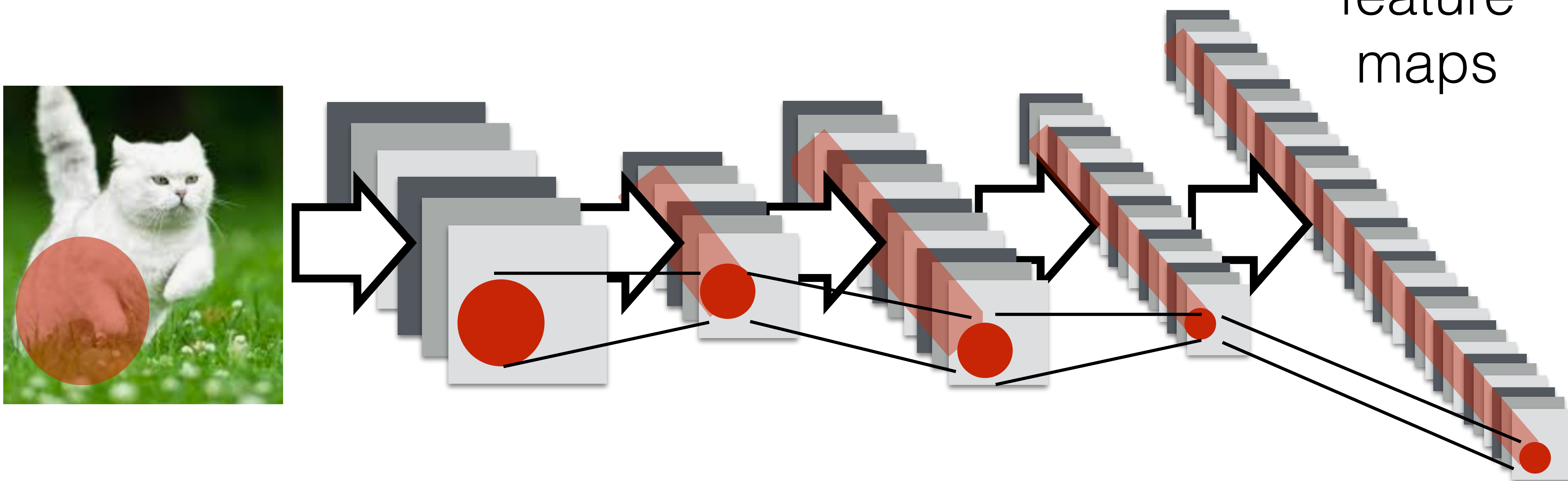
Feature maps and receptive field



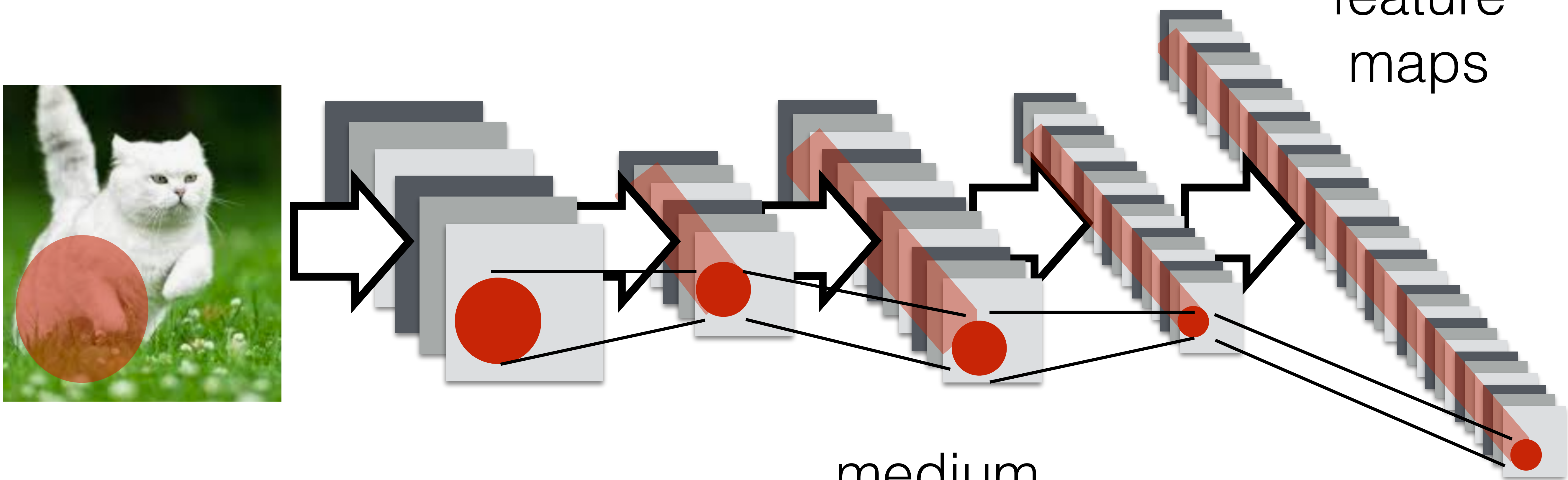
Feature maps and receptive field



Feature maps and receptive field



Feature maps and receptive field



Batch normalization

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

```
keras.layers.BatchNormalization(. . .)
```

suppose we want to use 3 features of a person to do a prediction,

1. height
2. weight
3. hair diameter

We have some issues:

1. units of measurements:
    1. height measure in meters? cm? mm?
    2. weight measure in kg? g? tons?

2. hair feature is going to be a non-factor because hair is so small (in mean and variance) compare to height

use the z score to make all features of equal importance

$$\mu = \frac{1}{n} \sum_i x_i$$

$$\sigma^2 = \frac{1}{n} \sum_i (x_i - \mu)^2$$

$$z_i = \frac{x_i - \mu}{\sigma}$$

https://en.wikipedia.org/wiki/Standard_score use the z score to make all features of equal importance

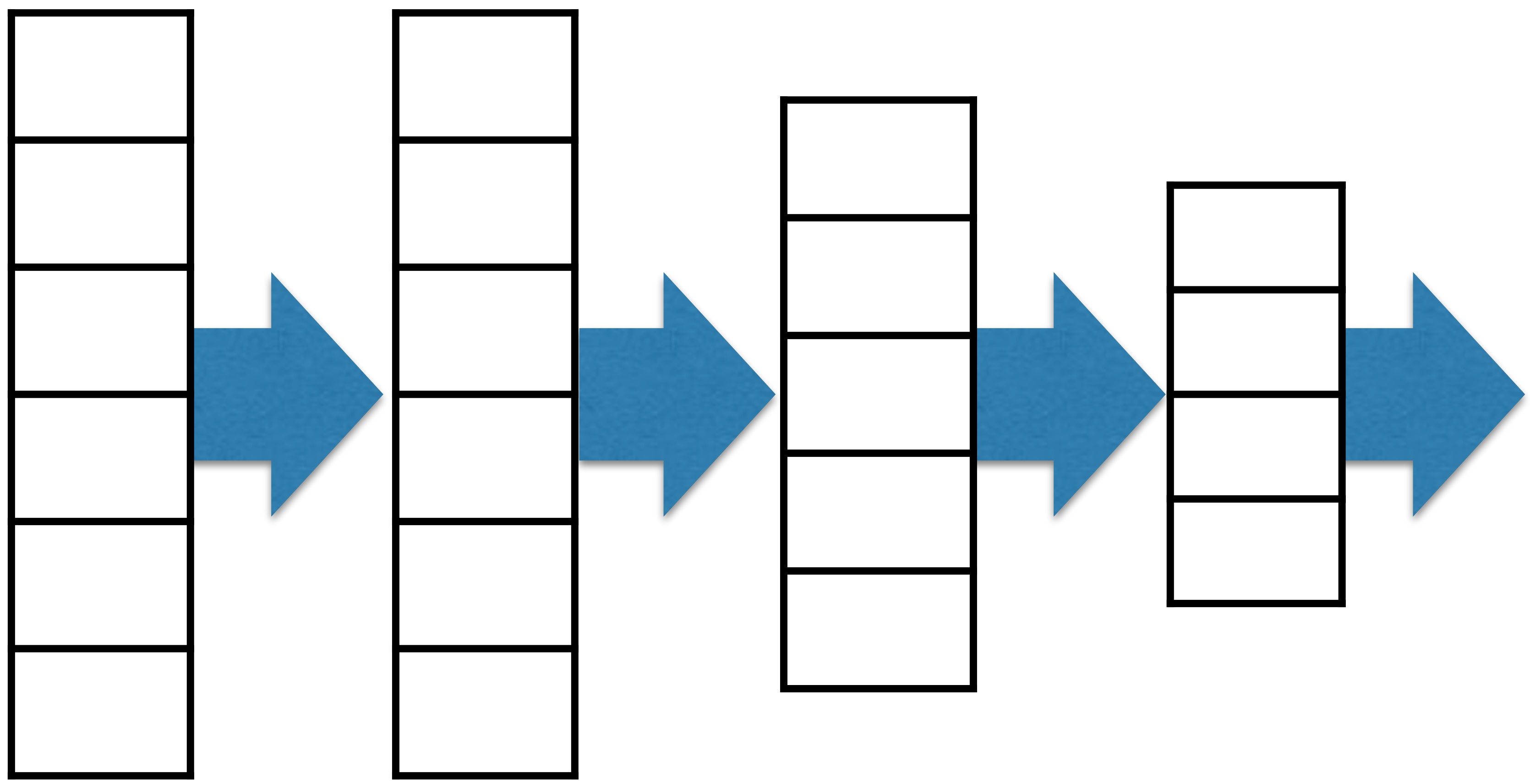
$$\mu = \frac{1}{n}\sum_i x_i$$

$$\sigma^2 = \frac{1}{n}\sum_i (x_i - \mu)^2$$

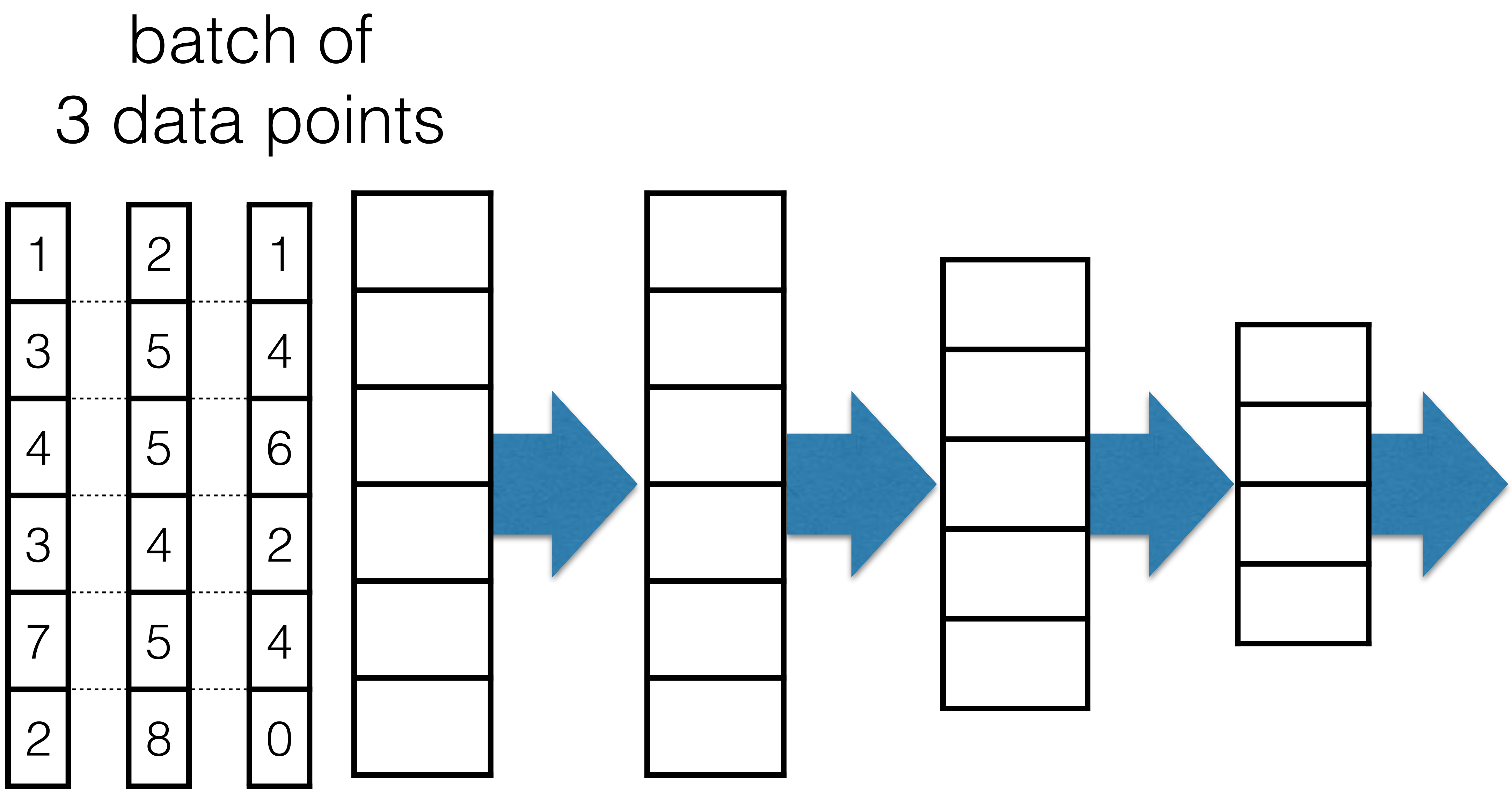
$$z_i = \frac{x_i - \mu}{\sigma}$$

https://en.wikipedia.org/wiki/Standard_score

Feeding in batches - without batch normalization

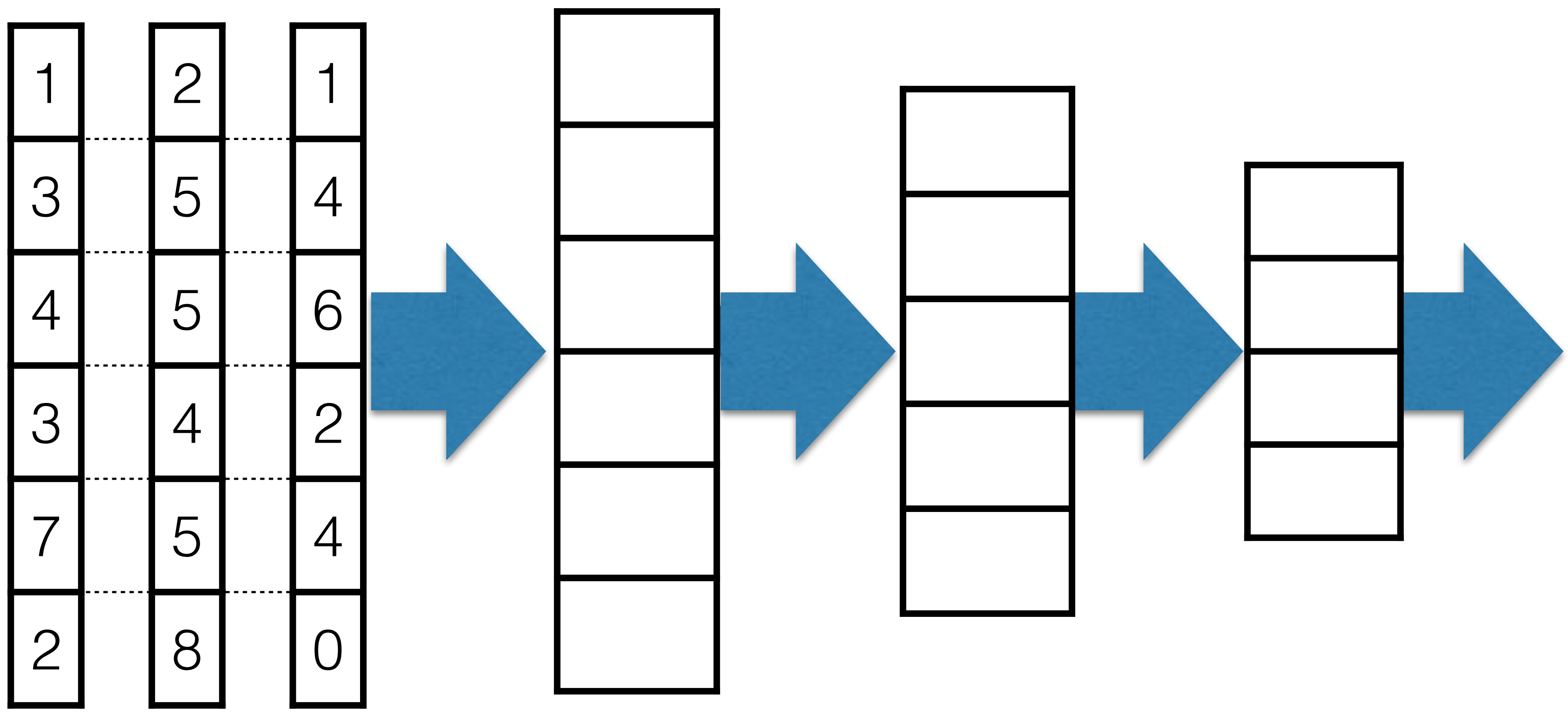


Feeding in batches - without batch normalization

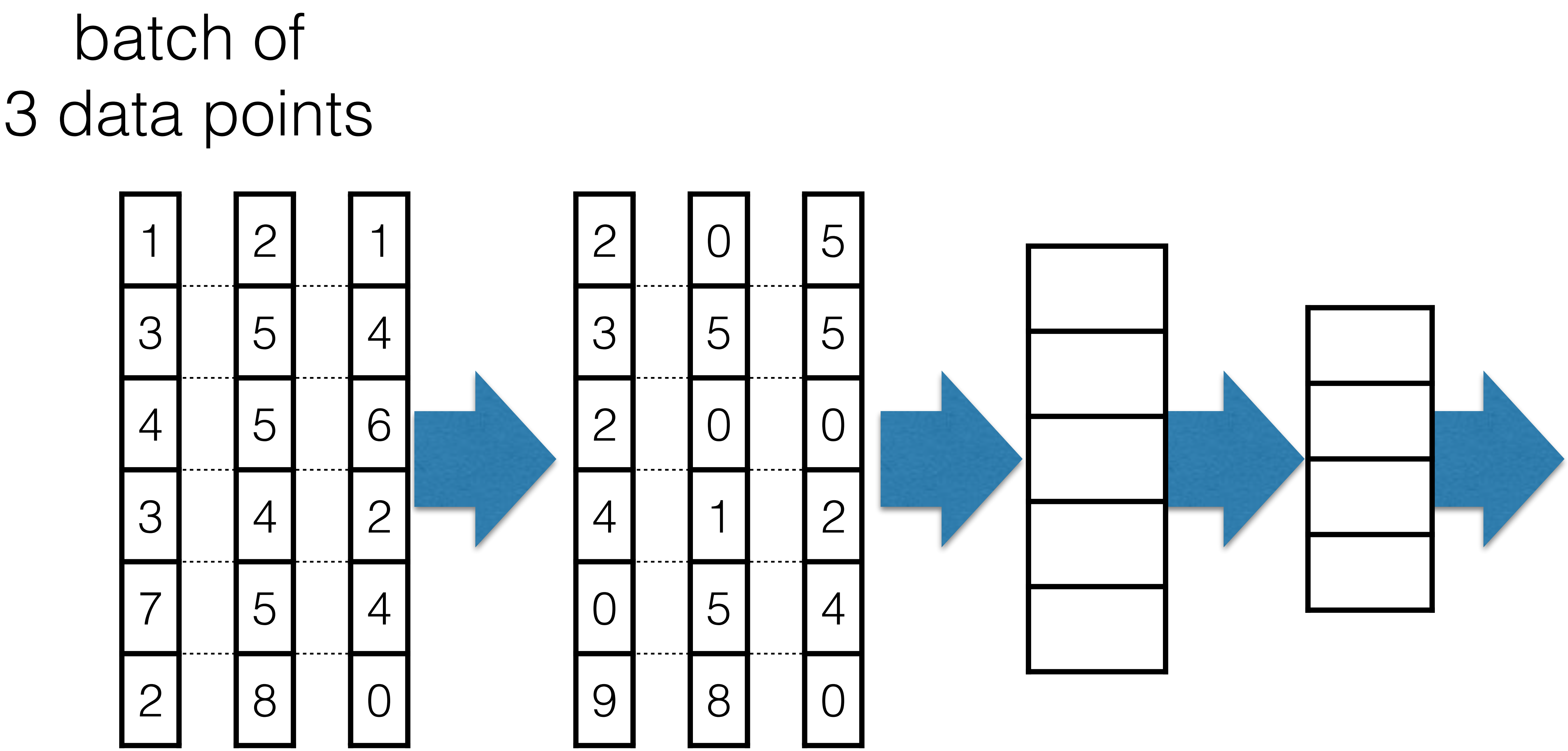


Feeding in batches - without batch normalization

batch of
3 data points

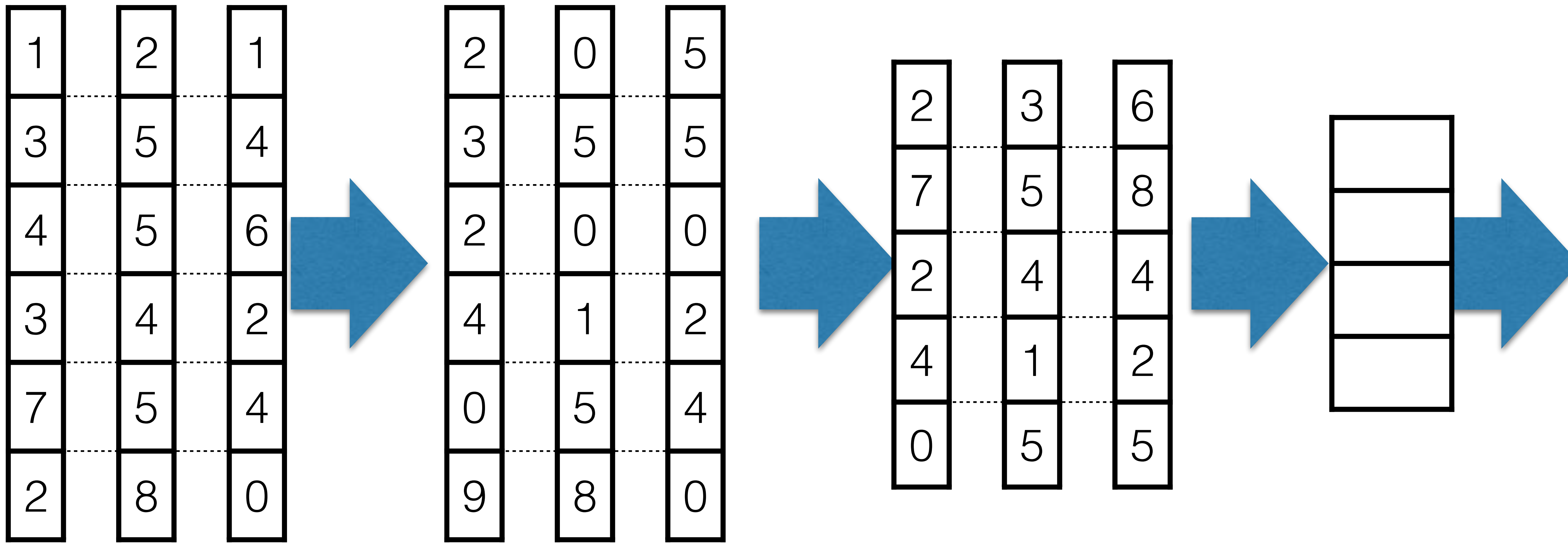


Feeding in batches - without batch normalization



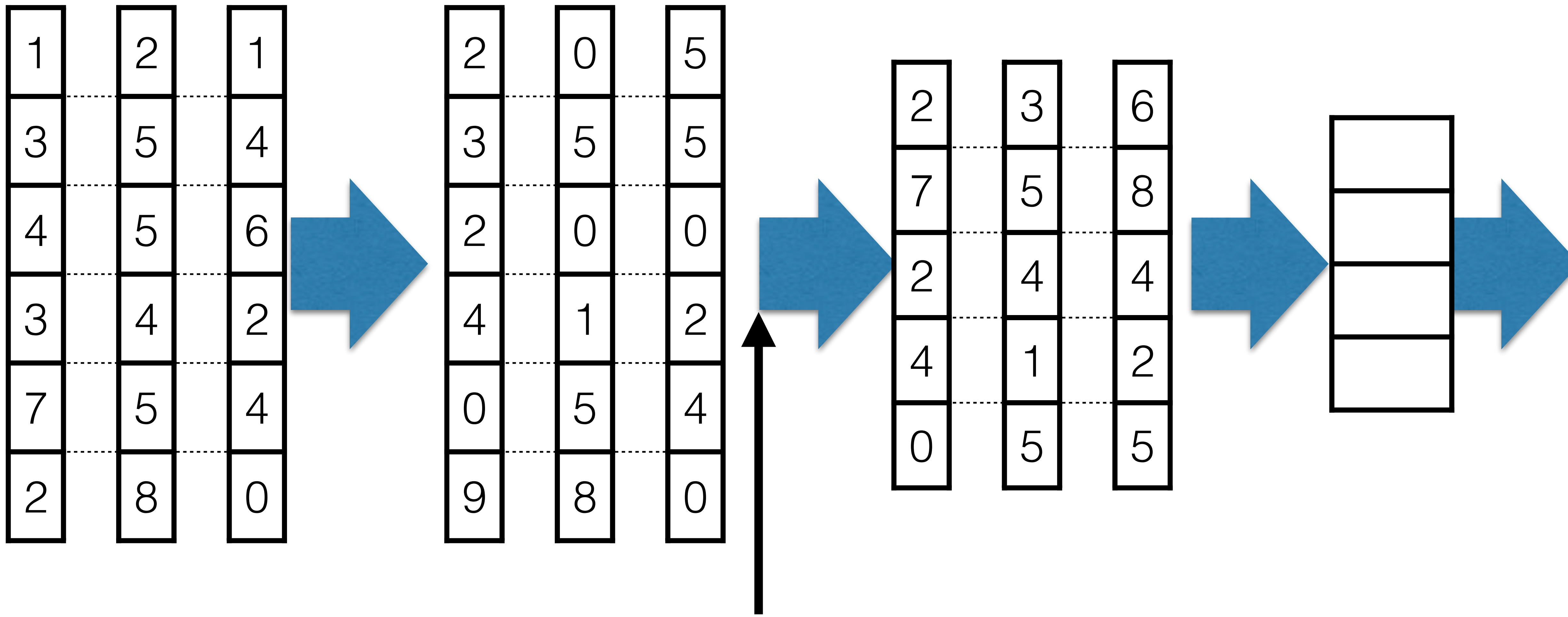
add batch normalization layer

batch of
3 data points



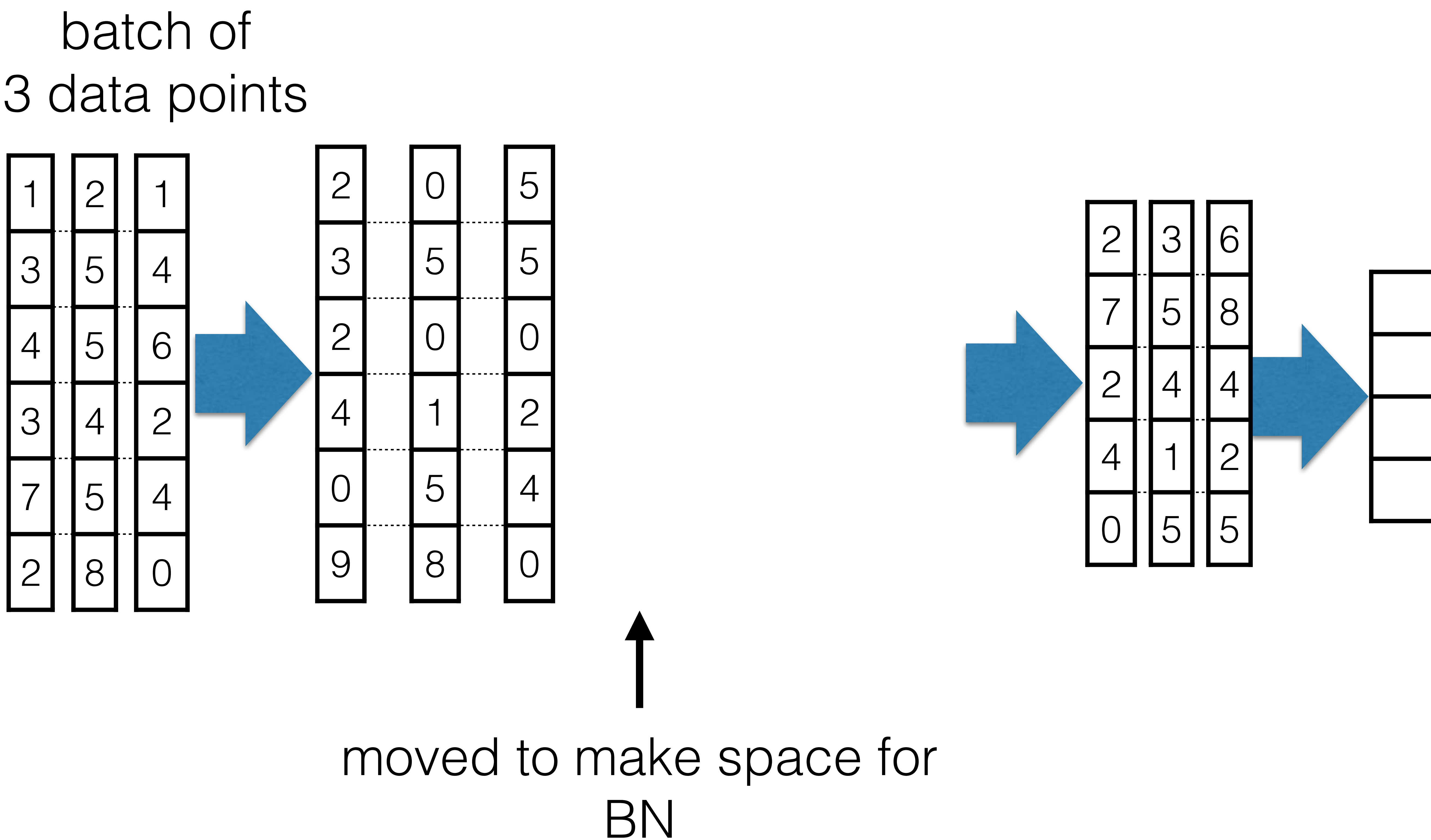
add batch normalization layer

batch of
3 data points



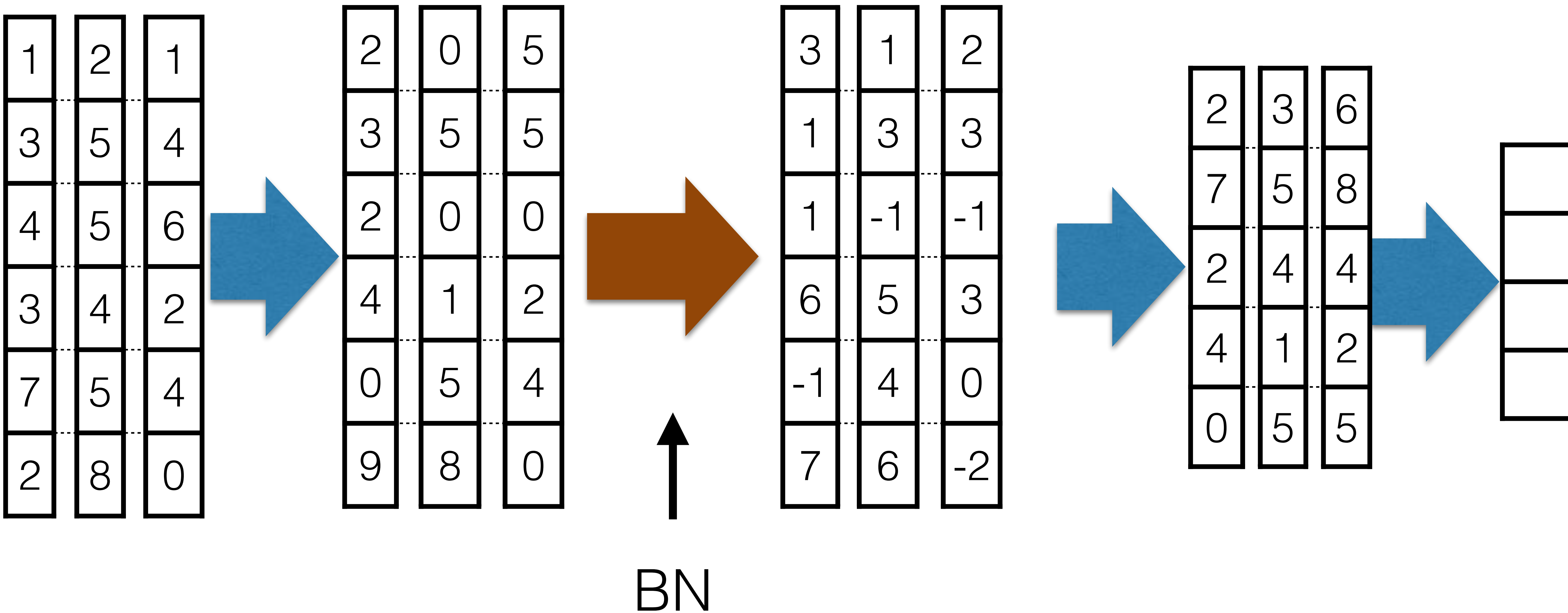
add batch normalization
layer here - note: before the
weights for the next layer

add batch normalization (BN) layer

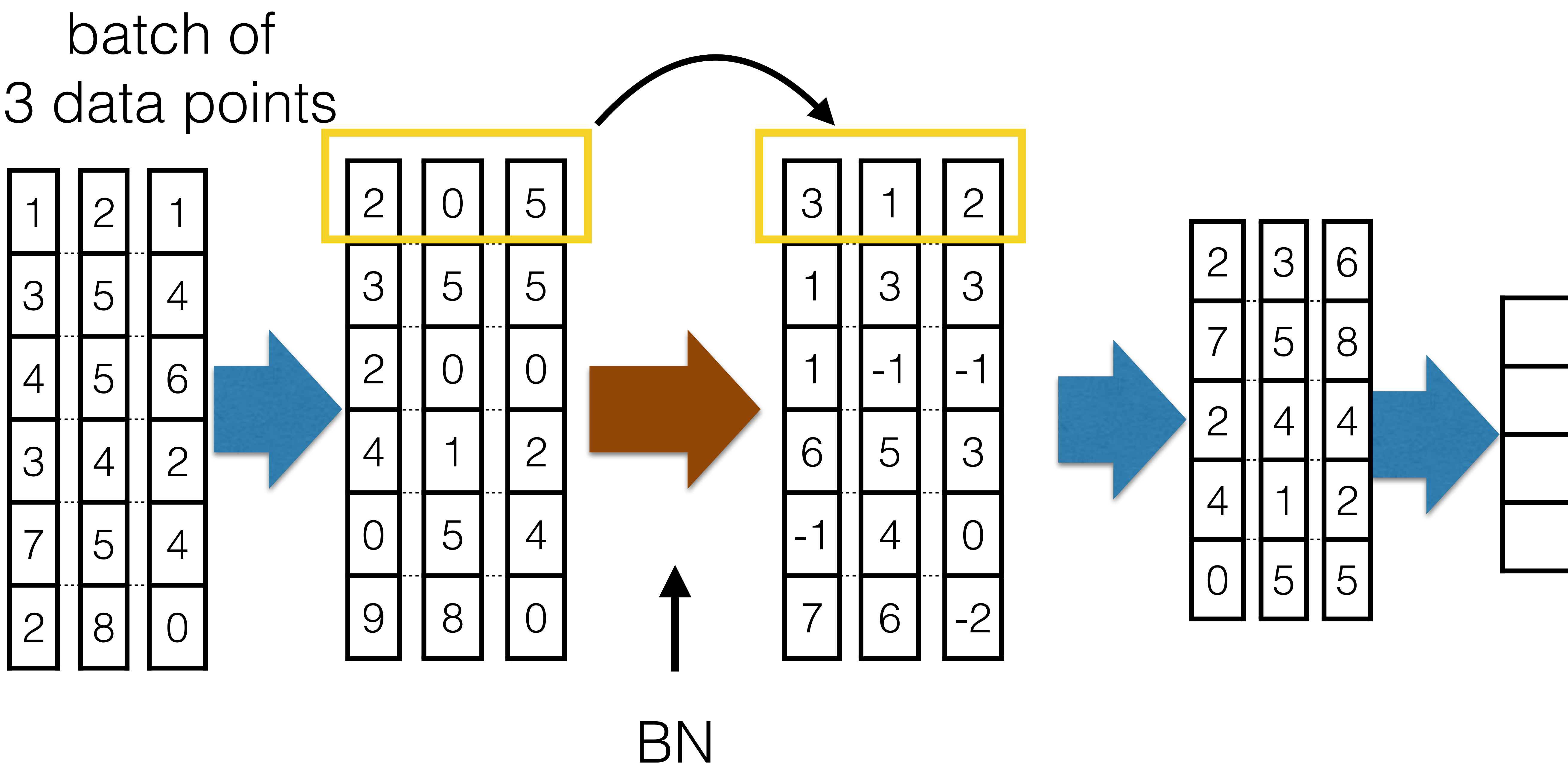


add batch normalization (BN) layer

batch of
3 data points



what happened at BN?



$u_1 = (2+0+5)/3$ — take average

$s_1 = \mathrm{stddev}(2,0,5)$ — take standard deviation

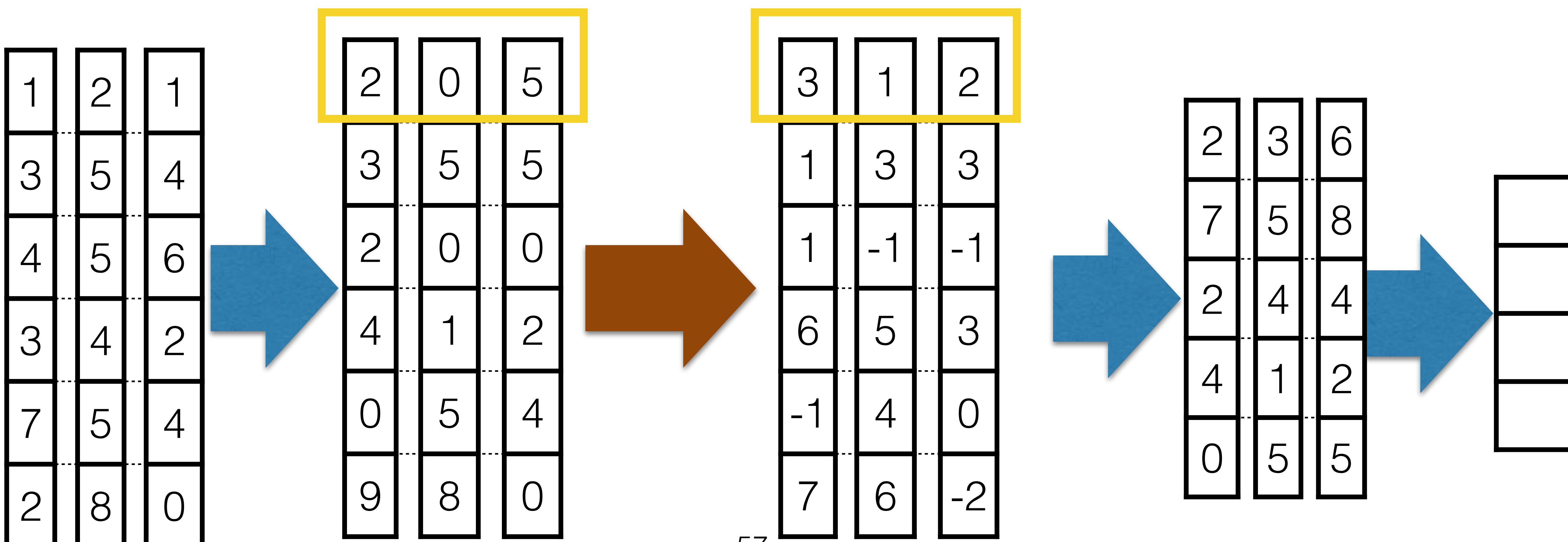
$z_{1,1}=(2-u_1)/s_1$ — transform first data point

$z_{2,1}=(0-u_1)/s_1$ — transform second data point

$z_{3,1}=(5-u_1)/s_1$ — transform third data point

g_1 b_1 are trainable parameters

$$3 = g_1 * z_{1,1} + b_1$$
$$1 = g_1 * z_{2,1} + b_1$$
$$2 = g_1 * z_{3,1} + b_1$$



$u_2 = (3+4+5)/3$ — take average

$s_2 = \text{stddev}(3,5,5)$ — take standard deviation

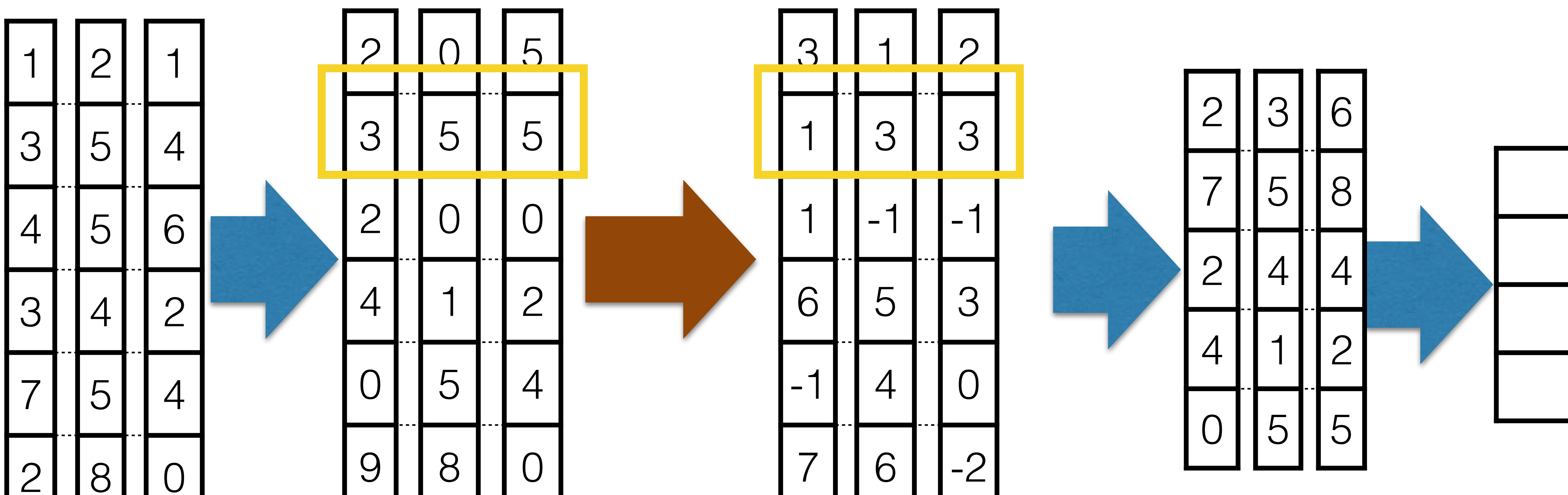
$z_{1,2}=(3-u_2)/s_2$ — transform first data point

$z_{2,2}=(5-u_2)/s_2$ — transform second data point

$z_{3,2}=(5-u_2)/s_2$ — transform third data point

g_2 b_2 are trainable parameters

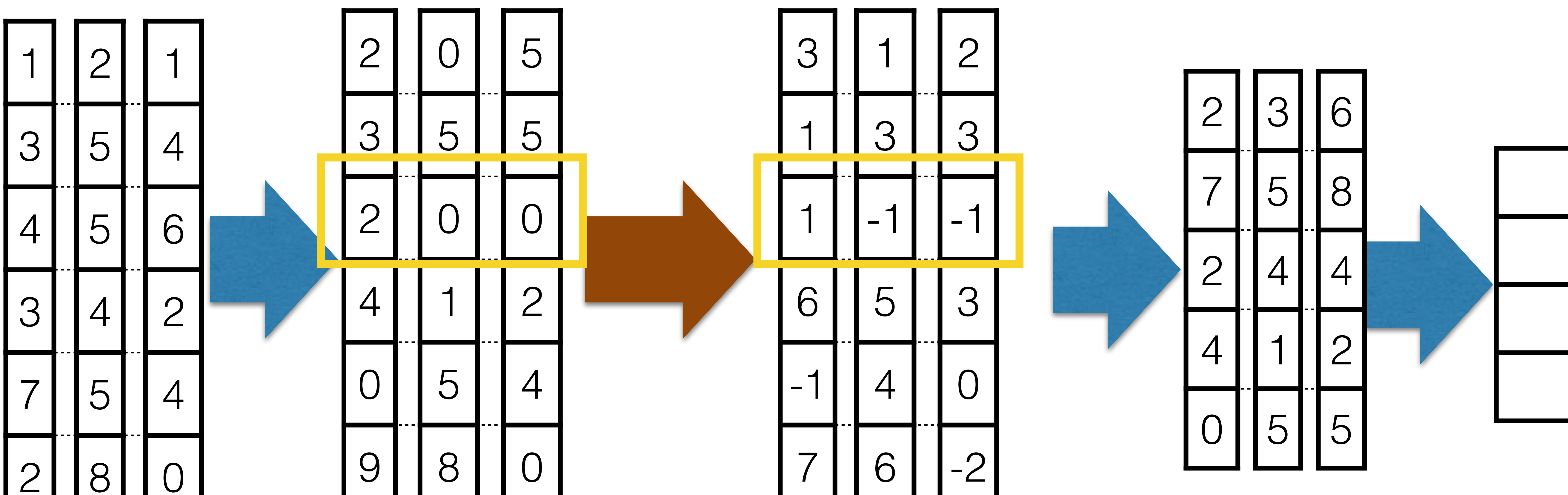
$$1 = g_2 * z_{1,2} + b_2$$
$$3 = g_2 * z_{2,2} + b_2$$
$$3 = g_2 * z_{3,2} + b_2$$



$u_3 = (2+0+0)/3$ — take average
$s_3 = \mathrm{stddev}(2,0,0)$ — take standard deviation
$z_{1,3}=(2-u_3)/s_3$ — transform first data point
$z_{2,3}=(0-u_3)/s_3$ — transform second data point
$z_{3,3}=(0-u_3)/s_3$ — transform third data point

g_3 b_3 are trainable parameters

$$1 = g_3 * z_{1,3} + b_3$$
$$-1 = g_3 * z_{2,3} + b_3$$
$$-1 = g_3 * z_{3,3} + b_3$$

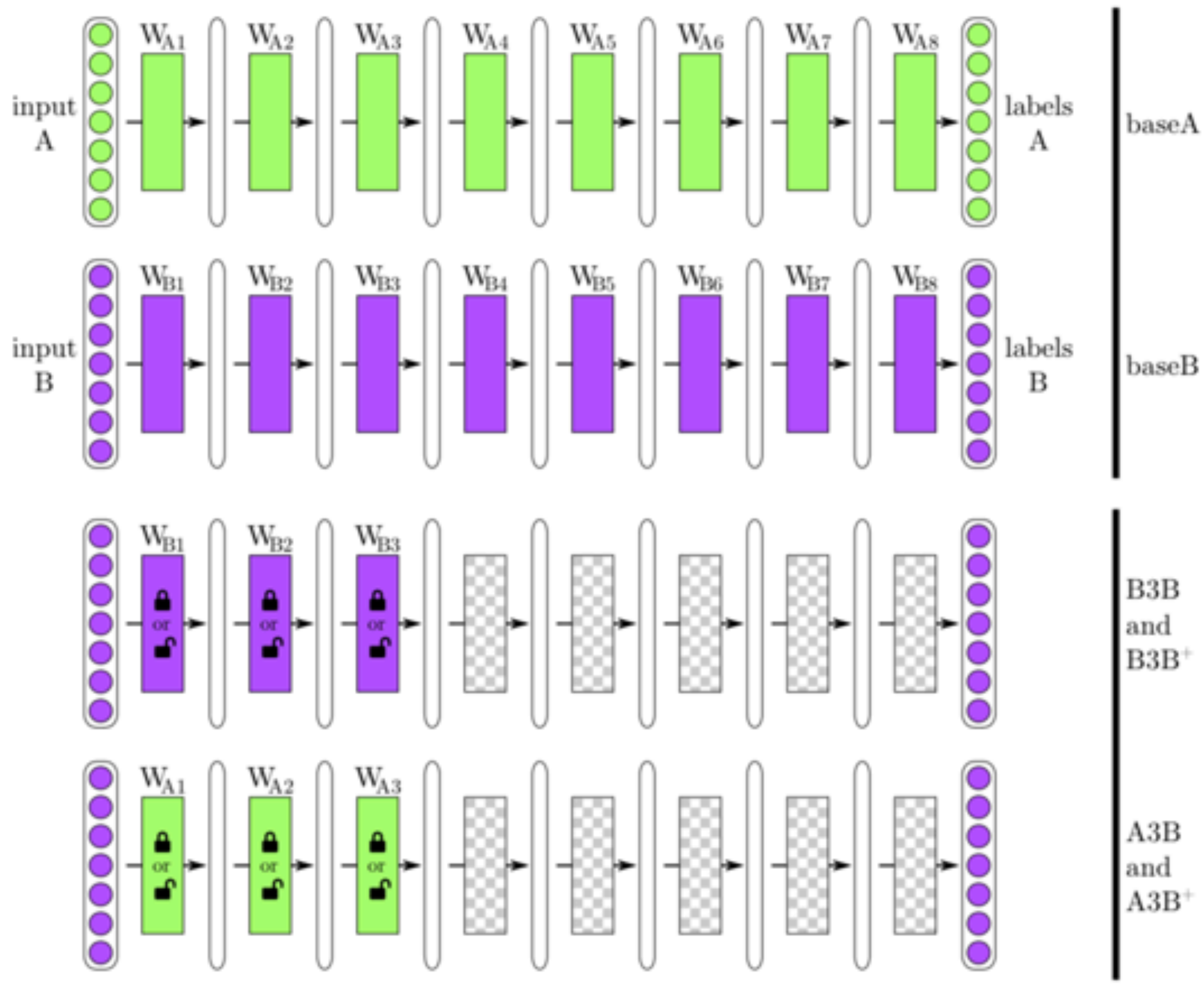


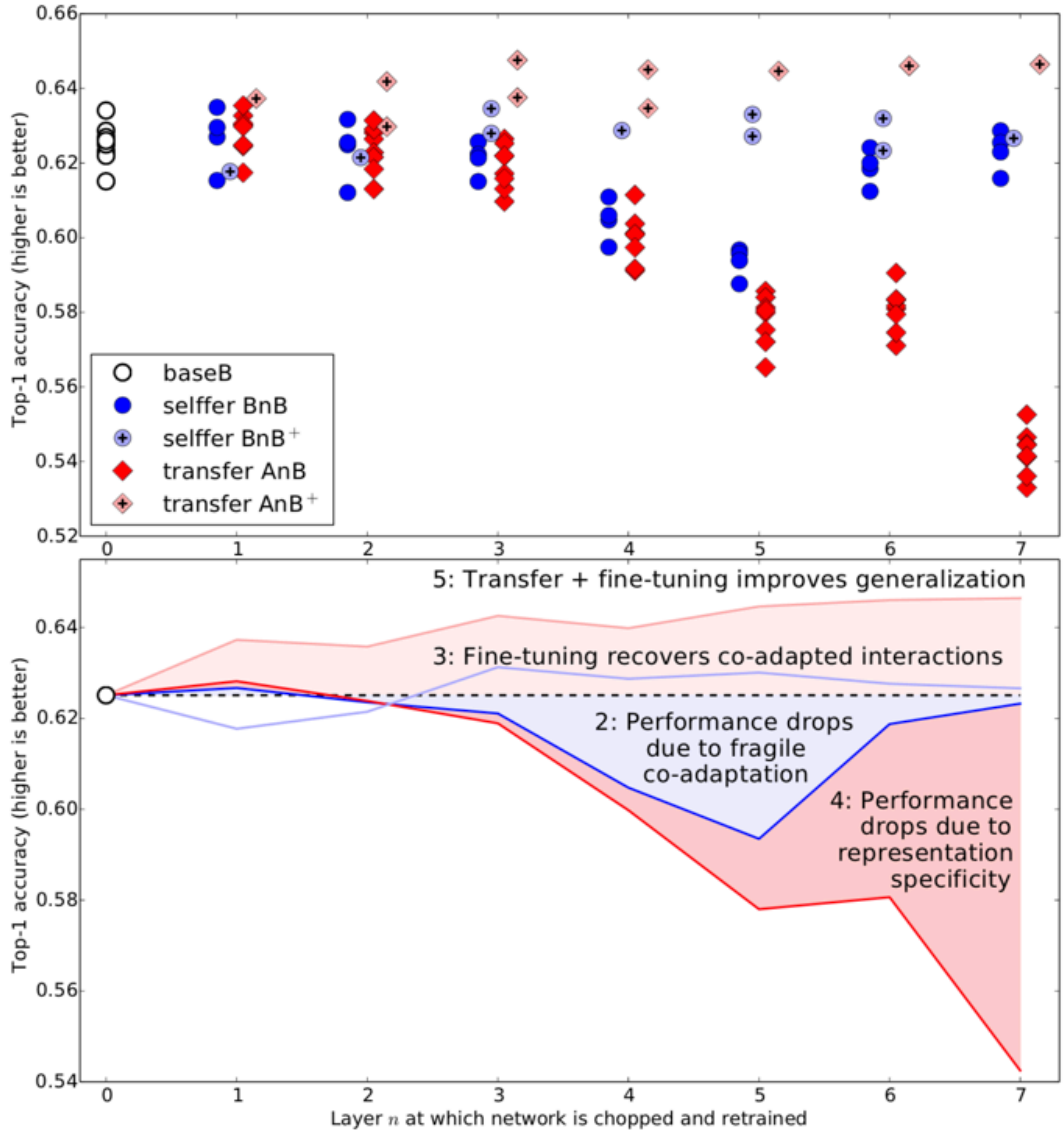
Transfer learning

How transferable are features in deep neural networks?

Jason Yosinski,[1] **Jeff Clune,**[2] **Yoshua Bengio,**[3] **and Hod Lipson**[4]
[1] Dept. Computer Science, Cornell University
[2] Dept. Computer Science, University of Wyoming
[3] Dept. Computer Science & Operations Research, University of Montreal
[4] Dept. Mechanical & Aerospace Engineering, Cornell University

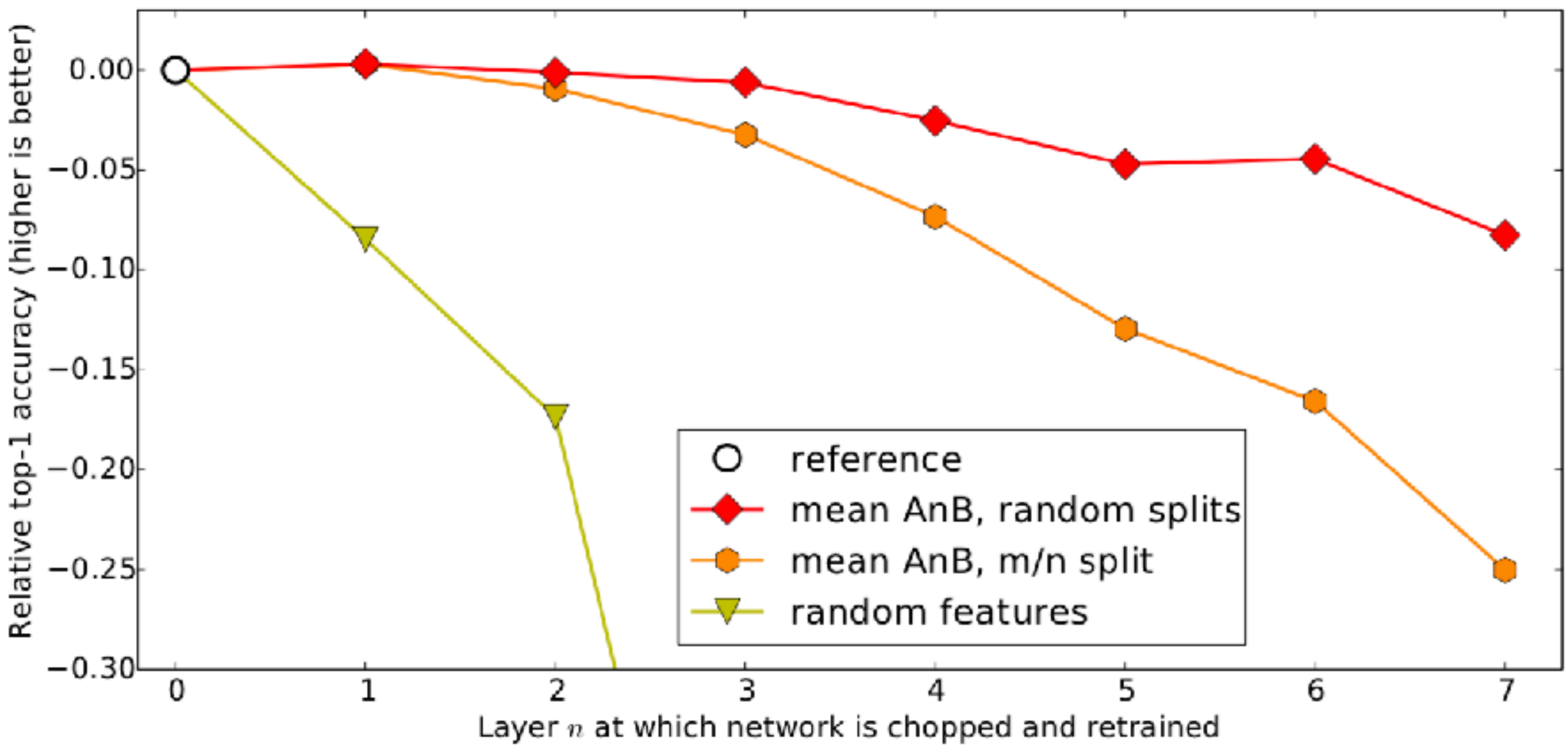
image-net 1000class. randomly split
500 ->A, 500 -> B
train A, transfer to B and vice versa
e.g. A to A, B to B, A to B, B to A





Top-1 accuracy (higher is better)
baseB
selffer BnB
selffer BnB+
transfer AnB
transfer AnB+
5: Transfer + fine-tuning improves generalization
3: Fine-tuning recovers co-adapted interactions
2: Performance drops due to fragile co-adaptation
4: Performance drops due to representation specificity
Layer n at which network is chopped and retrained

split and transfer between man-made and natural images



thinking time
& question time again

when do we use transfer learning?

how do we check if transfer learning is good?

Image data augmentation

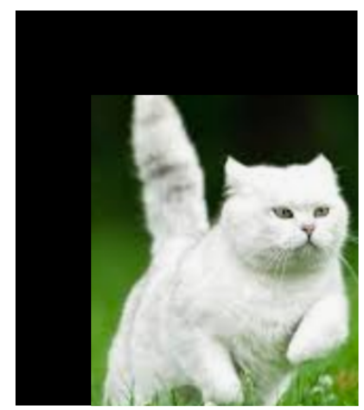
1. to increase the amount of data > 10x data
2. to make the CNN more robust to transformation in images

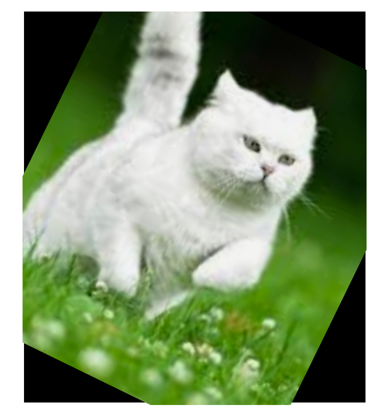
original image

scaling

flipping

translation



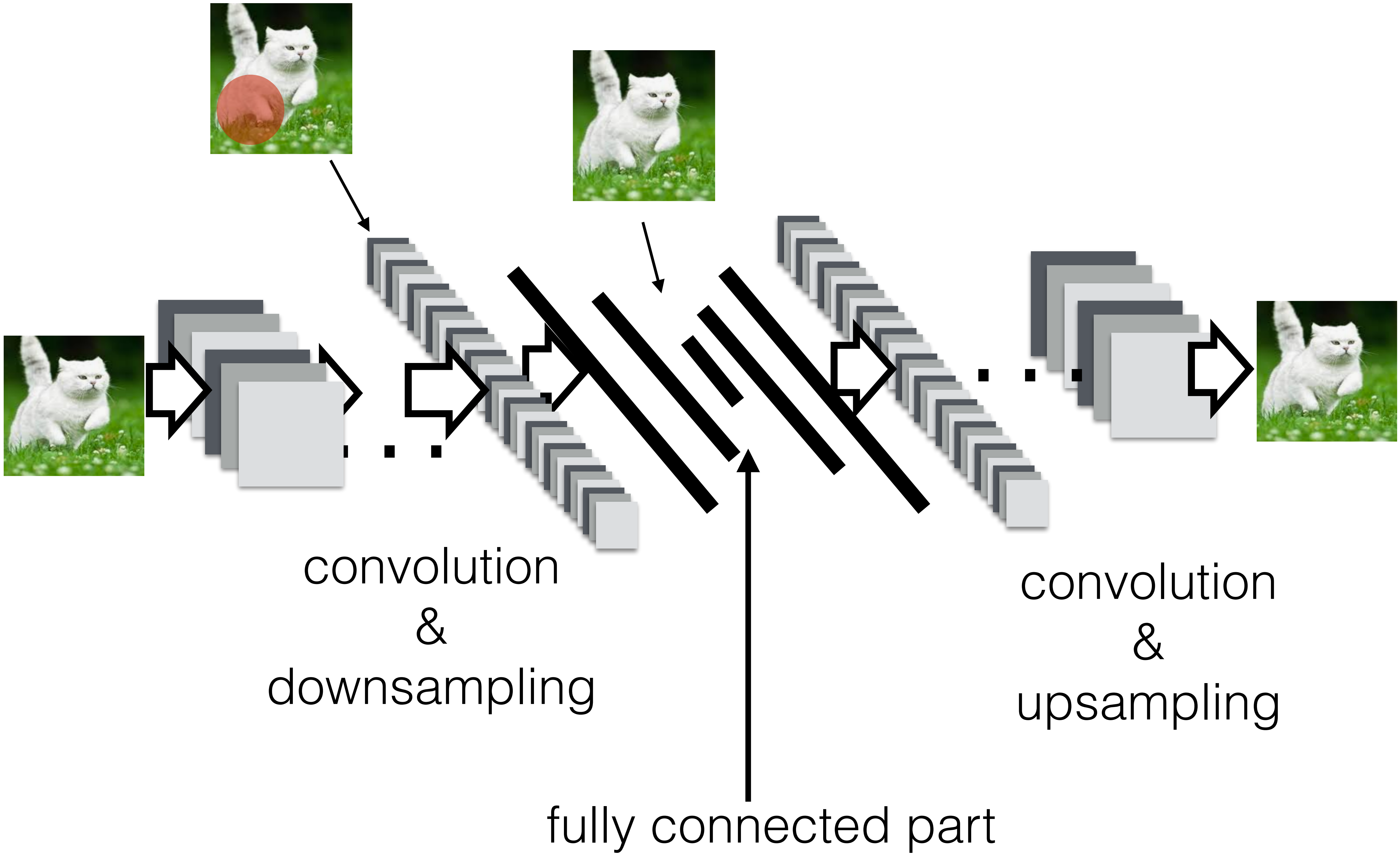


rotation

using another neural network to generate data
Generative Adversarial Networks (GAN)

CNN with autoencoder

CNN auto encoder

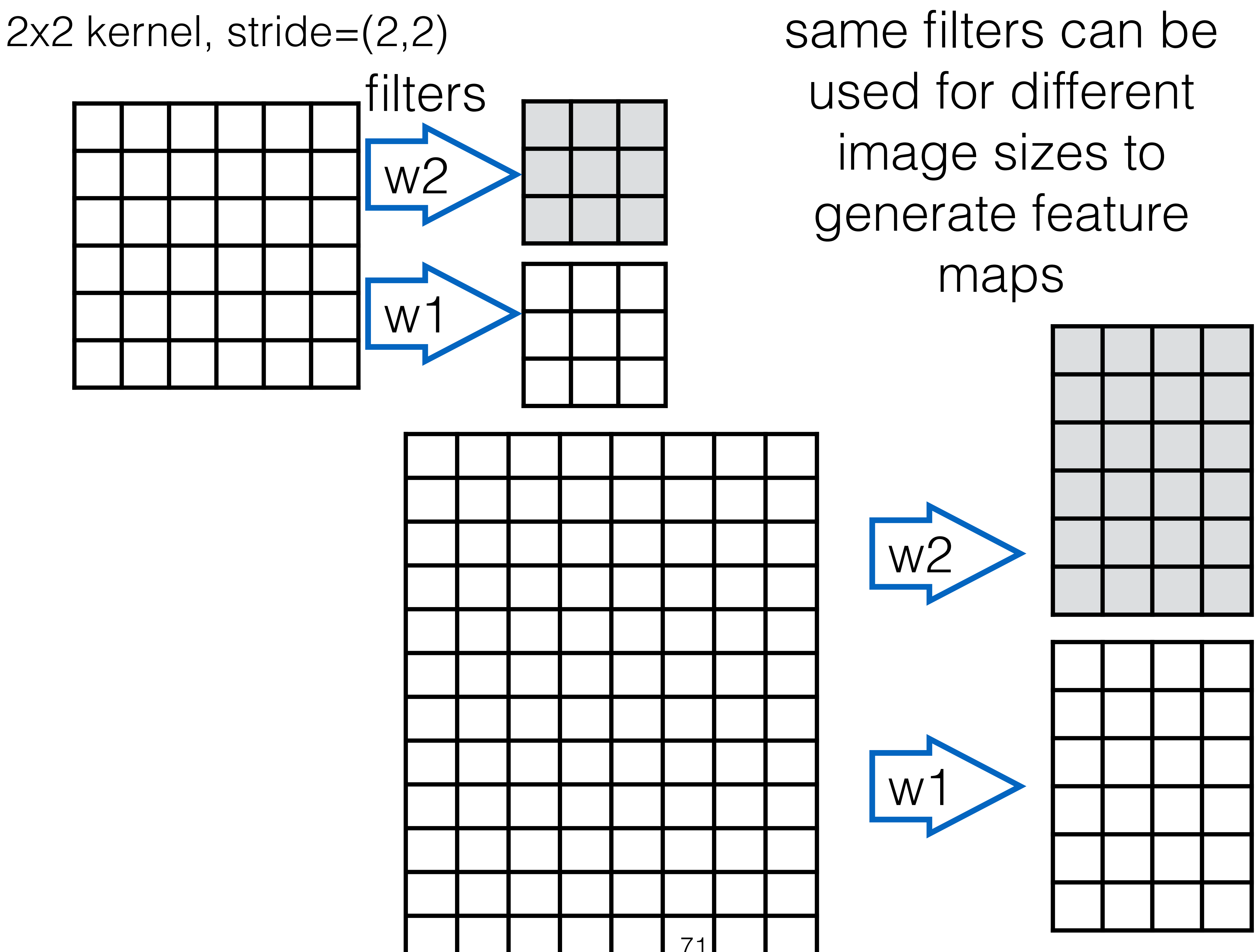


Region Of Interest Pooling

ROI - Pooling

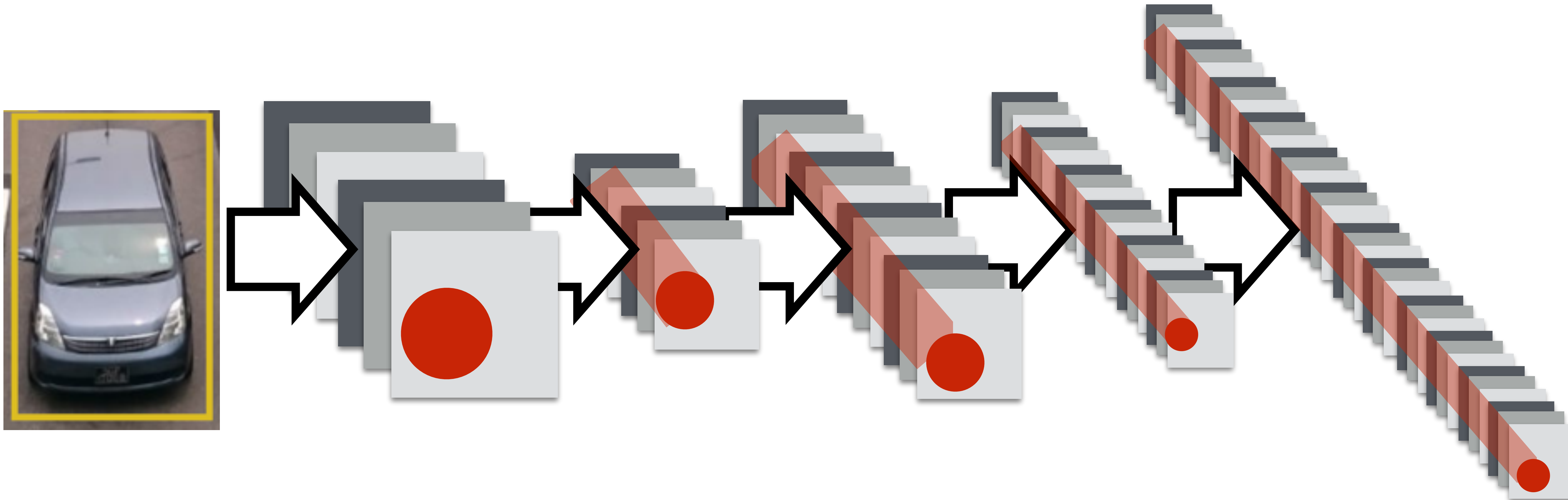
https://deepsense.ai/region-of-interest-pooling-explained/

CNN part of the network can take image of any size



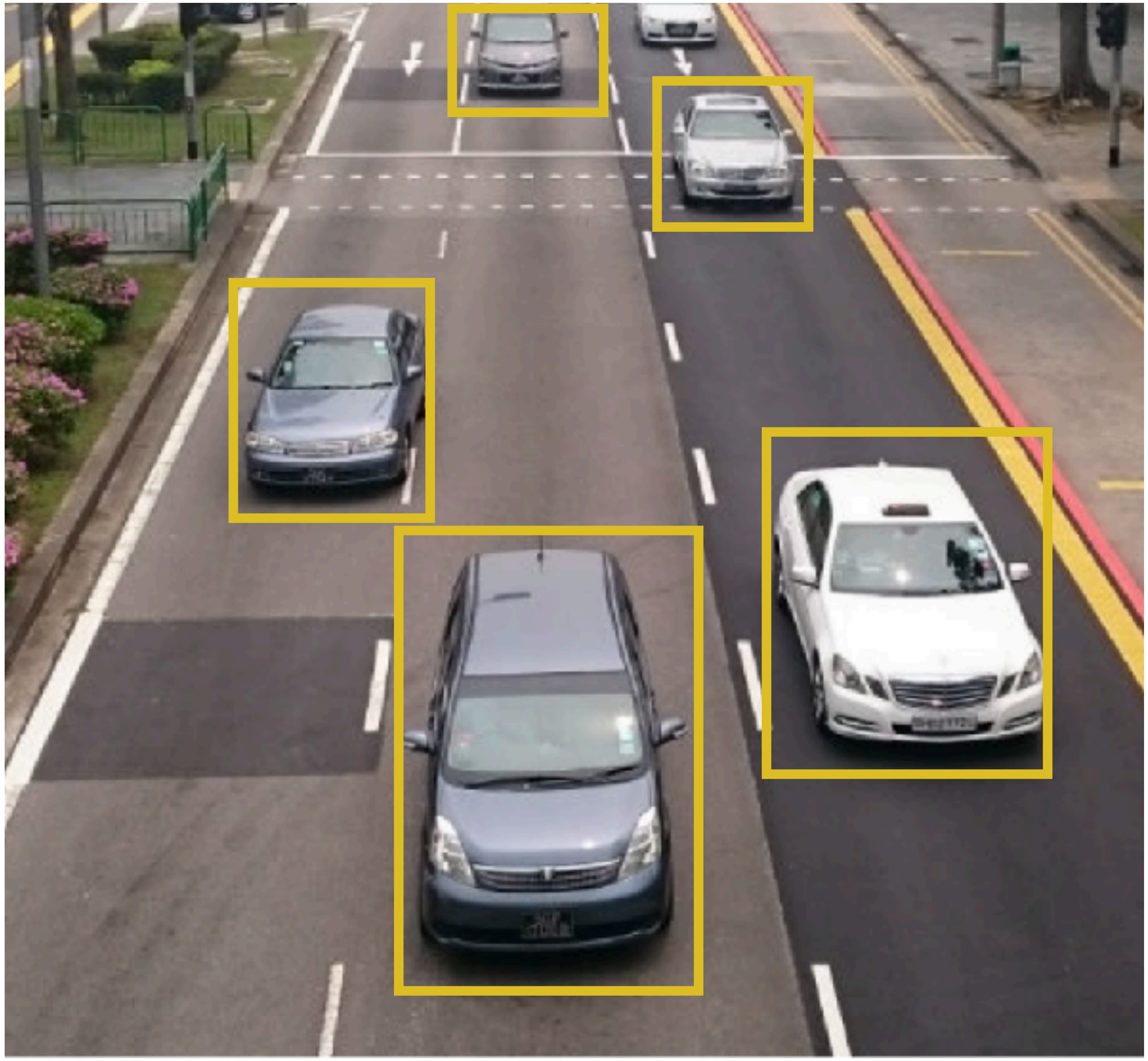
same filters can be used for different image sizes to generate feature maps

CNN to convert images into feature maps

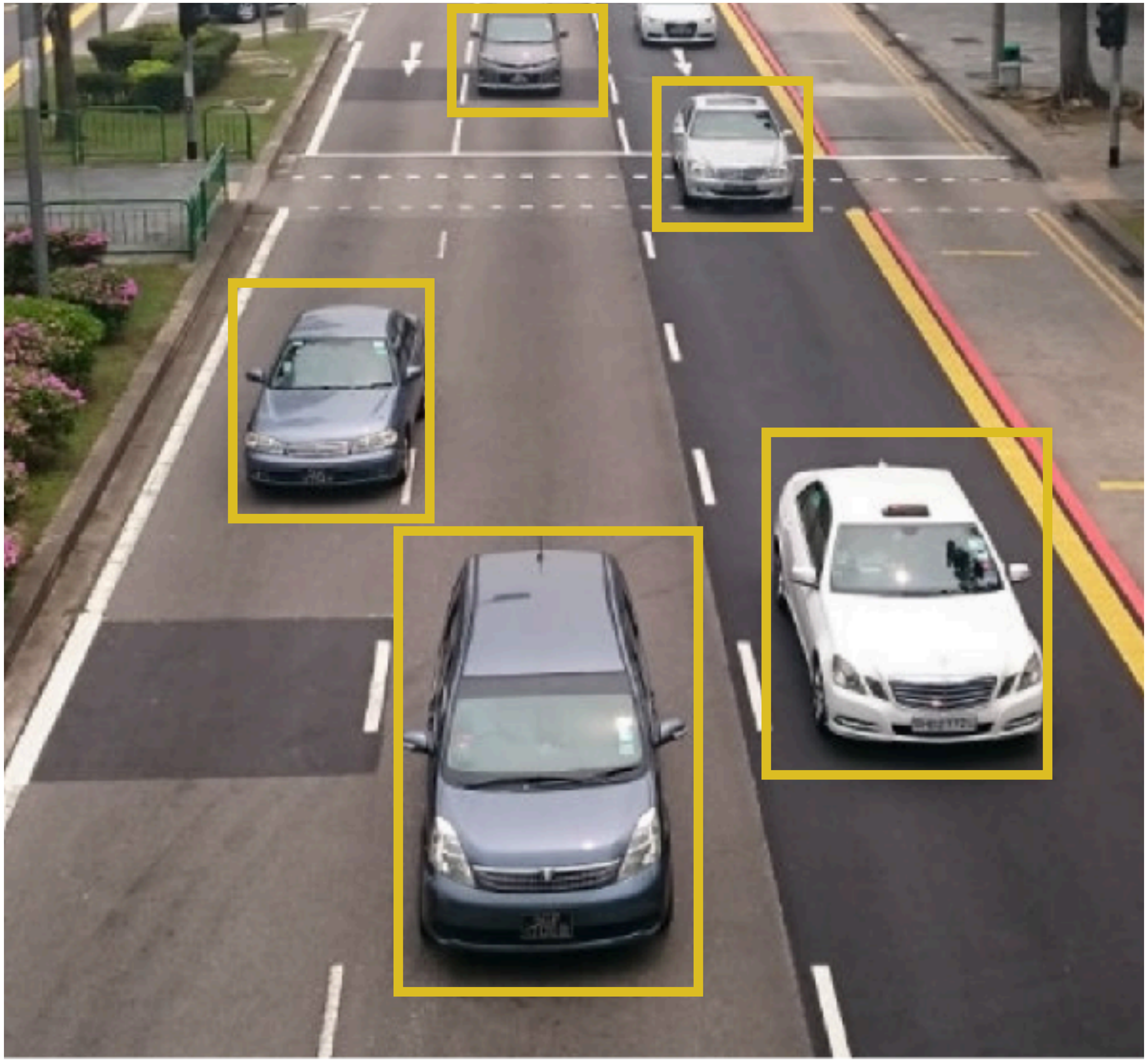


want to detect cars on a road

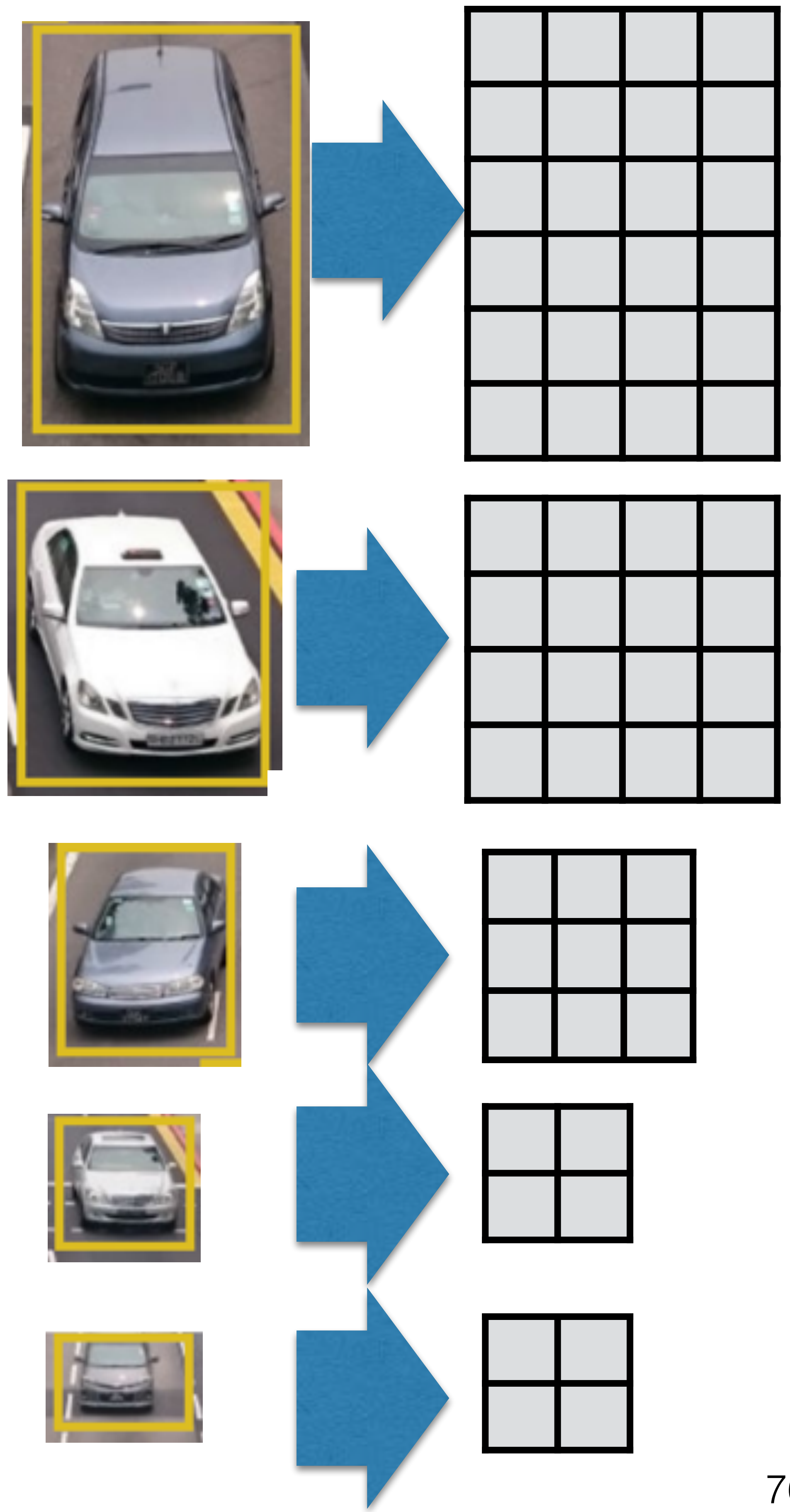
propose different regions to CNN



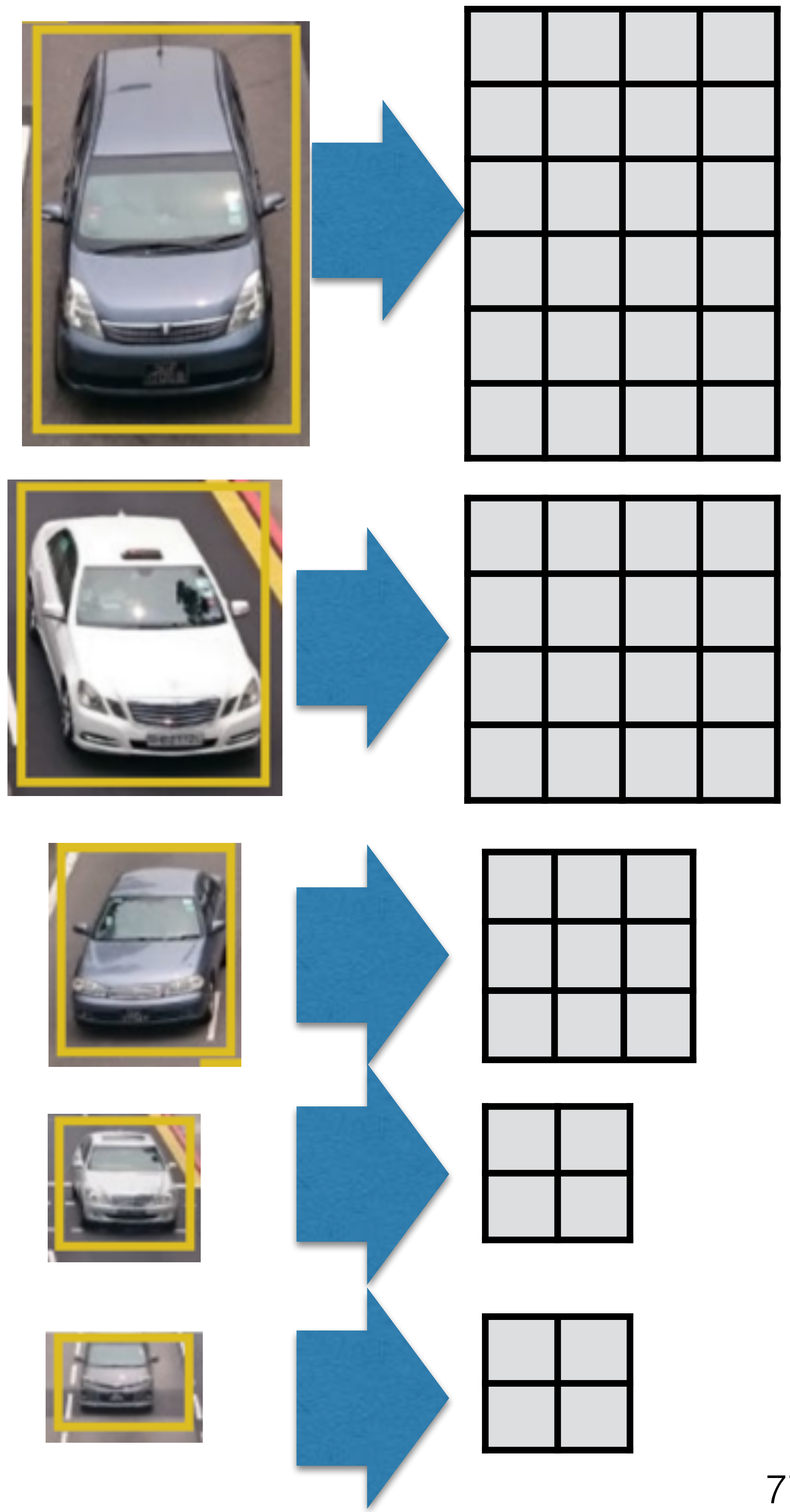
feed these regions into CNN and ask if there is a car in proposed region



use conv part of CNN to convert ROIs into feature maps



feed feature maps into the rest of CNN



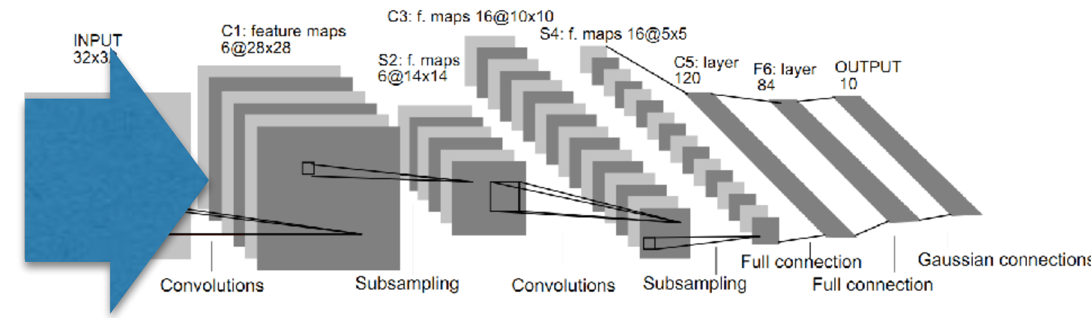


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN

problem is feature maps are of different sizes!

Solutions

1. resize / reshape the feature maps before feeding
2. Region of interest pooling (ROI pooling)

ROI - Pooling example

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

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

reduce these two ROI into 3x2 pixels features

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

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

reduce these two ROI into 3x2 pixels features

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

max pool

7		

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

max pool

8		

reduce these two ROI into 3x2 pixels features

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

max pool

7	6	

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

max pool

8	9	

reduce these two ROI into 3x2 pixels features

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

max pool

7	6	8

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

max pool

8	9	5

reduce these two ROI into 3x2 pixels features

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

7	6	8
9		

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

8	9	5
7		

reduce these two ROI into 3x2 pixels features

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

7	6	8
9	7	

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

8	9	5
7	7	

reduce these two ROI into 3x2 pixels features

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

7	6	8
9	7	9

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

8	9	5
7	7	8

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

7	6	8
9	7	9

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

8	9	5
7	7	8

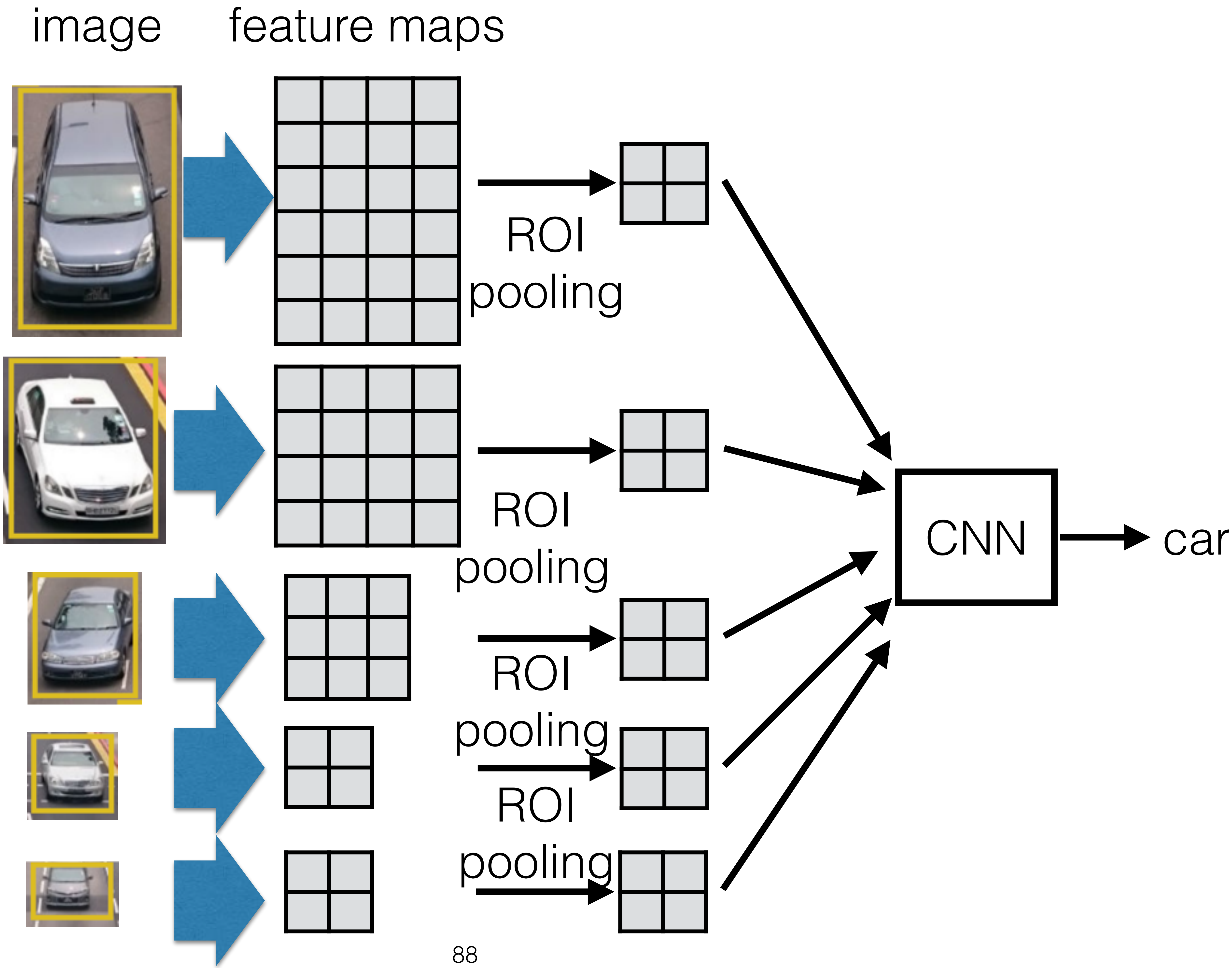


image
feature maps
ROI pooling
ROI pooling
ROI pooling
ROI pooling
CNN
car

To avoid just following what others do without understanding, we need to ask questions!

Any questions?

1. Why do ROI pooling on feature maps and not do ROI pooling on the original image?

2. why not rescale all the image to the same size?

1.Why do ROI pooling on feature maps and not do ROI pooling on the original image?

— taking maximum response of feature makes a lot of sense, taking maximum local intensity may not make sense

2.why not rescale all the image to the same size?

— computational efficiency

— some reasons can become more obvious when we study region proposal network