$$P(Y=+1|x^{\{i\}}) = \frac{P(x^{\{i\}}|Y=+1)\,P(Y=+1)}{P(x^{\{i\}})}$$ Generative

↓

Posterior

$$P(Y=+1|x^{\{i\}}) = \frac{1}{1+\exp(-X\theta)}$$ → Sigmoid

directly

$X\theta = S = U \rightsquigarrow$ LCF

$$\theta^{\{t+1\}} \leftarrow \theta^{\{t\}} - \alpha\,\nabla E_\theta(\theta)$$

$$\theta^{\{t+1\}} \leftarrow \theta^{\{t\}} - \alpha\, X^T(\hat{Y}_p - Y_a)$$

$$P(Y=+1|x^{\{i\}}) = \frac{\exp(S_1)}{\exp(S_1)+\exp(S_2)} \rightsquigarrow \hat{Y}_p \Big]_0^1$$

$$P(Y=2|x^{\{i\}}) = \frac{\exp(S_2)}{=} \rightsquigarrow \hat{Y}_p \Big]_0^1$$

Softmax

$$-\text{avg log likelihood} = \text{cross entropy} = -\sum_{i=1}^{N} y_a^{\{i\}} \log \hat{y}_p^{\{i\}} = E(\theta)$$ ← Minimize

$$P(y|x) = \frac{\exp(s_1)}{\exp(s_1)+\exp(s_2)} = \frac{\frac{\exp(s_1)}{\exp(s_1)}}{1+\exp(s_2-s_1)} = \frac{1}{1+\exp(s_2-s_1)} = \frac{1}{1+\exp(-x\theta)}$$

$$-\left[(s_2-s_1) = x\theta_{cat} - x\theta_{dog} = x\underbrace{(\theta_{cat}-\theta_{dog})}_{\theta}\right] = -x\theta$$

CONVOLUTIONAL NEURAL NETWORK

Mahdi Roozbahani

Georgia Tech

Great visualization tool:
https://poloclub.github.io/cnn-explainer/

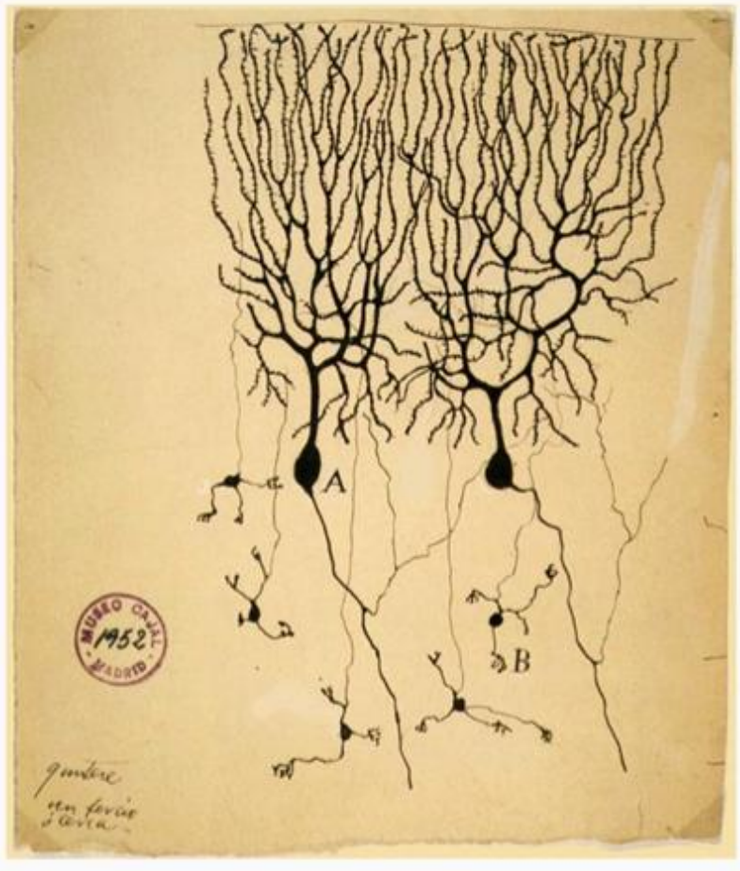
Slides are adopted on Ming Li (University of waterloo – Deep learning part) with some modifications

James Webb Space Telescope

The landscape of “mountains” and “valleys”

A visual grouping of five galaxies

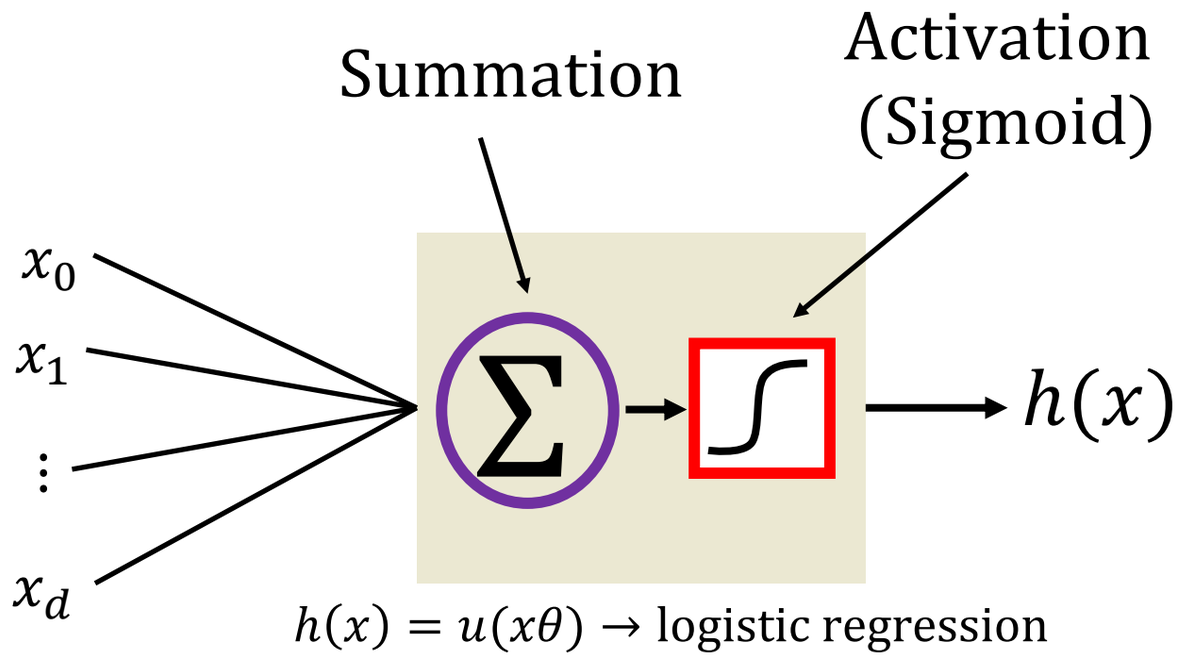
Inspiration from Biological Neurons



The first drawing of a brain cells by Santiago Ramón y Cajal in 1899

Neurons: core components of brain and the nervous system consisting of

1. Dendrites that collect information from other neurons
2. An axon that generates outgoing spikes

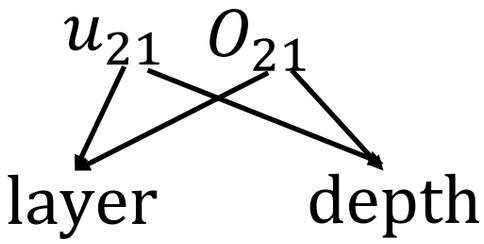


$h(x) = u(x\theta) \rightarrow$ logistic regression

$$output = activation(x\theta + b)$$

Name of the neuron	Activation function: $activation(z)$
Linear unit	$x\theta$
Threshold/sign unit	$sign(x\theta)$
Sigmoid unit	$\frac{1}{1+\exp(x\theta)}$
Rectified linear unit (ReLU)	$\max(0, x\theta)$
Tanh unit	$\tanh(x\theta)$

NN Regression



1st hidden layer

Bias

Neuron or node

θ_0

θ_4

u_{11}

O_{11}

Activation (Sigmoid)

θ_5

θ_2

u_{21}

O_{21}

$f(x)$

Regression

Prediction

θ_1

x_i

feature x_i

θ_3

weight

Summation

u_{12}

O_{12}

θ_6

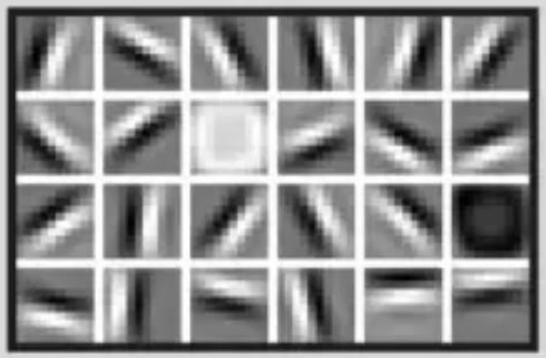
Sigmoid output

Summation output

FACIAL RECOGNITION

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.

Layer 1: The computer identifies pixels of light and dark.



Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.

y_a y_p

$[1 \quad 0]$ $[0.7 \quad 0.3]$

θ_1 θ_d

x_0 x_1 x_2 x_d

$X = \begin{bmatrix} x^{(1)} \\ \vdots \end{bmatrix}$ $y_a = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$

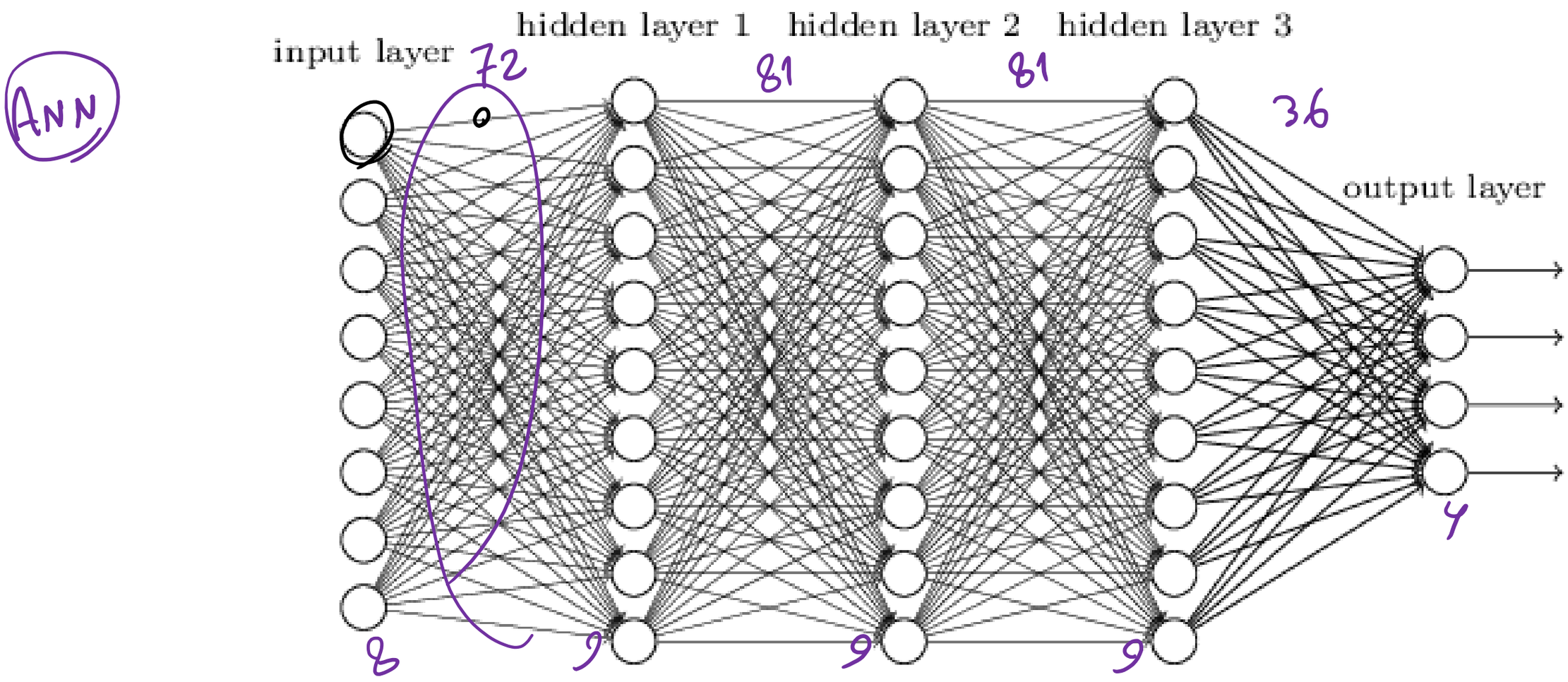
$CE = -\sum y_a \log \hat{y}_p$

$\hat{y}_p = \dfrac{\exp(s_1)}{\exp(s_1) + \exp(s_2)}$ $\Big]_0^1$

$s_1 = X\theta = \theta_1 x_1 + \dots + \theta_d x_d$

Smaller Network: CNN

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

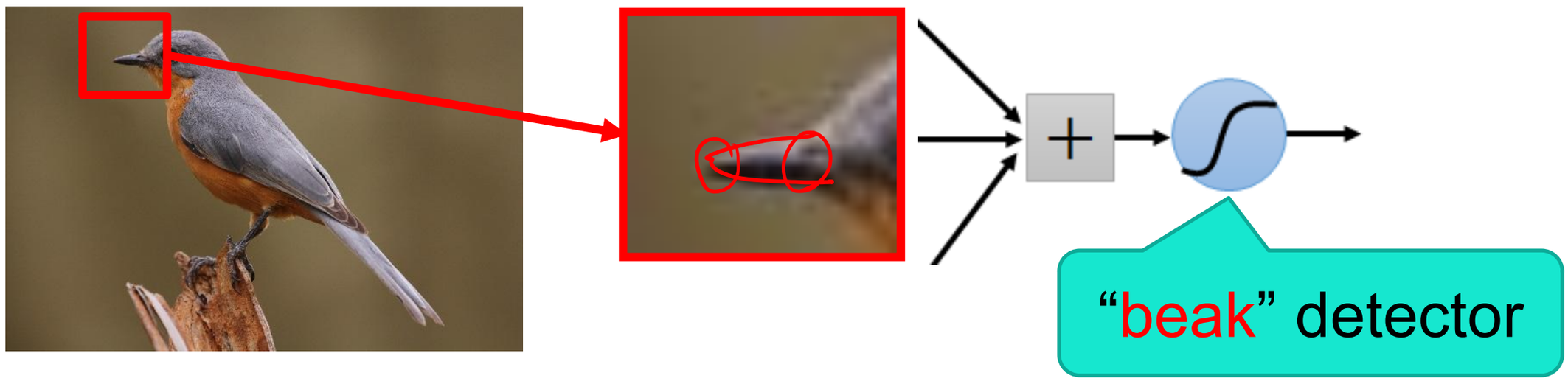
1	3	2	6	0	1
0	1	0	0	1	0
0	7	1	1	0	0
1	0	4	5	1	8
4	1	0	7	2	0
3	0	1	0	1	0

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

Consider learning an image:

- Some patterns are much smaller than the whole image

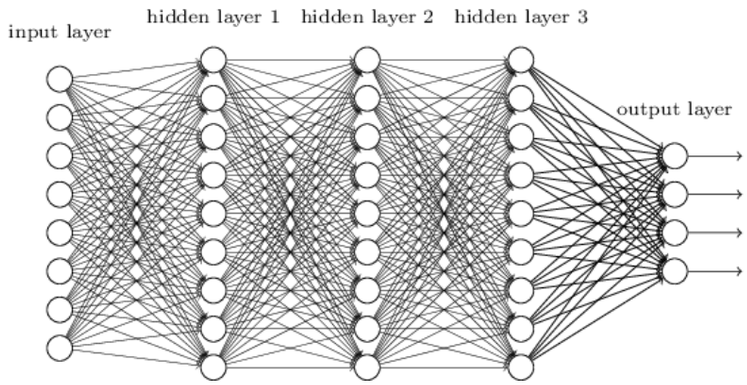
Can represent a small region with fewer parameters

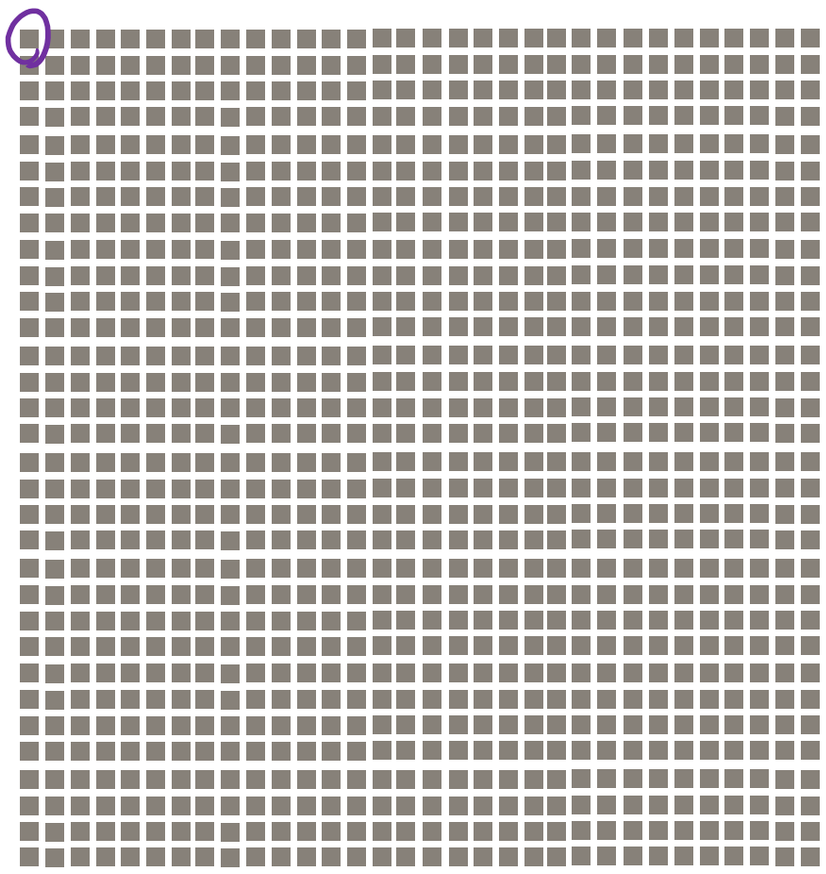


600x600

360,000

360,000

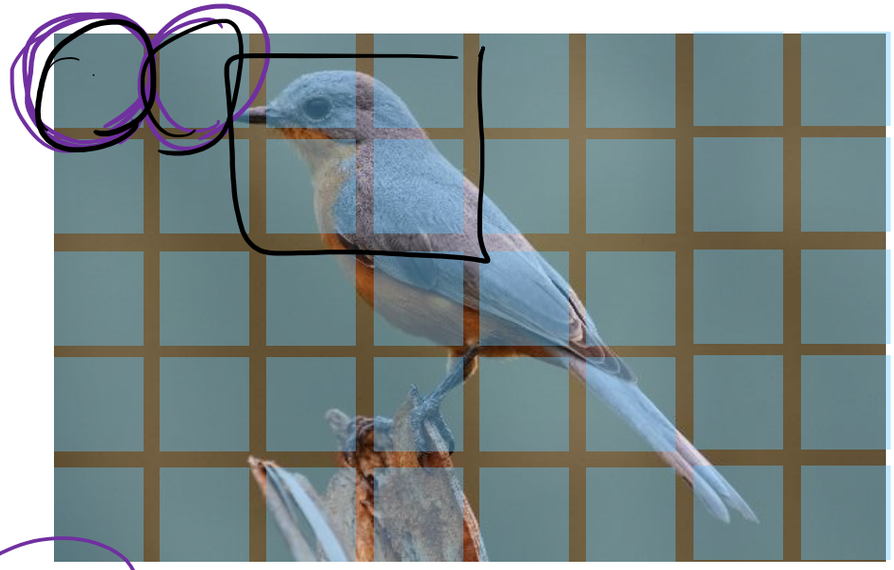


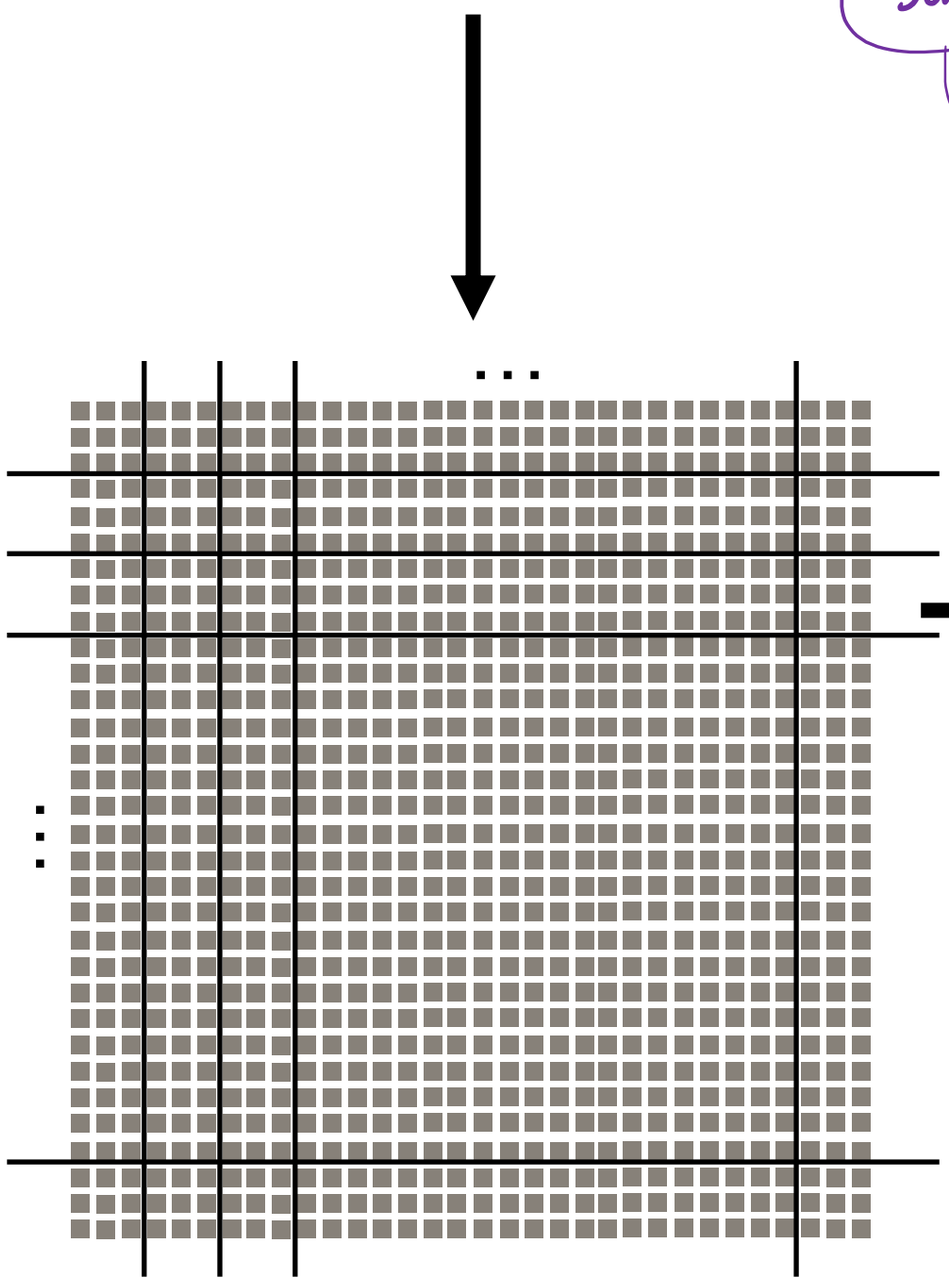


600x600

360,000

360,000 / 9 = 40,000

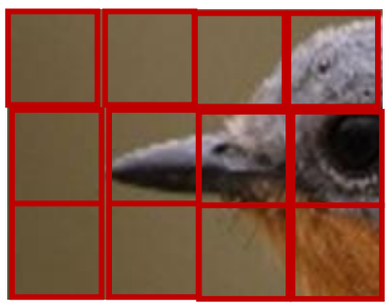




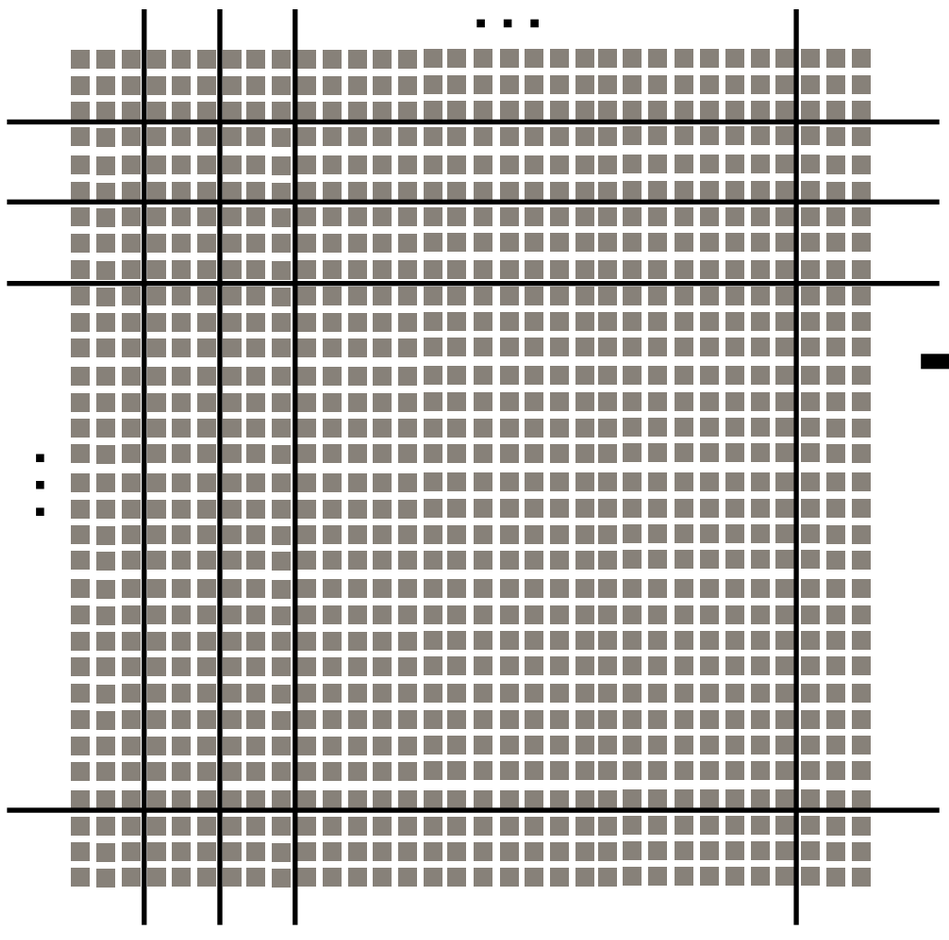
3x3

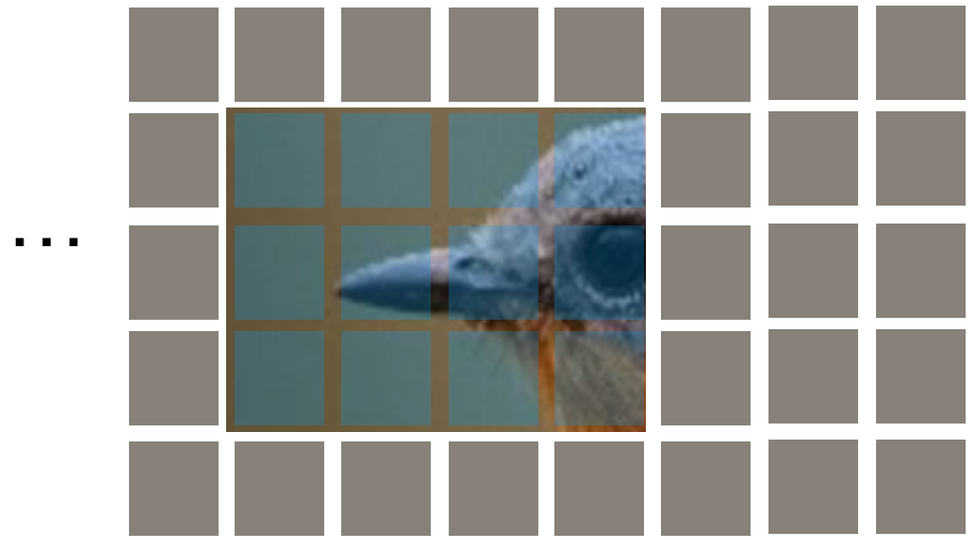
3x3

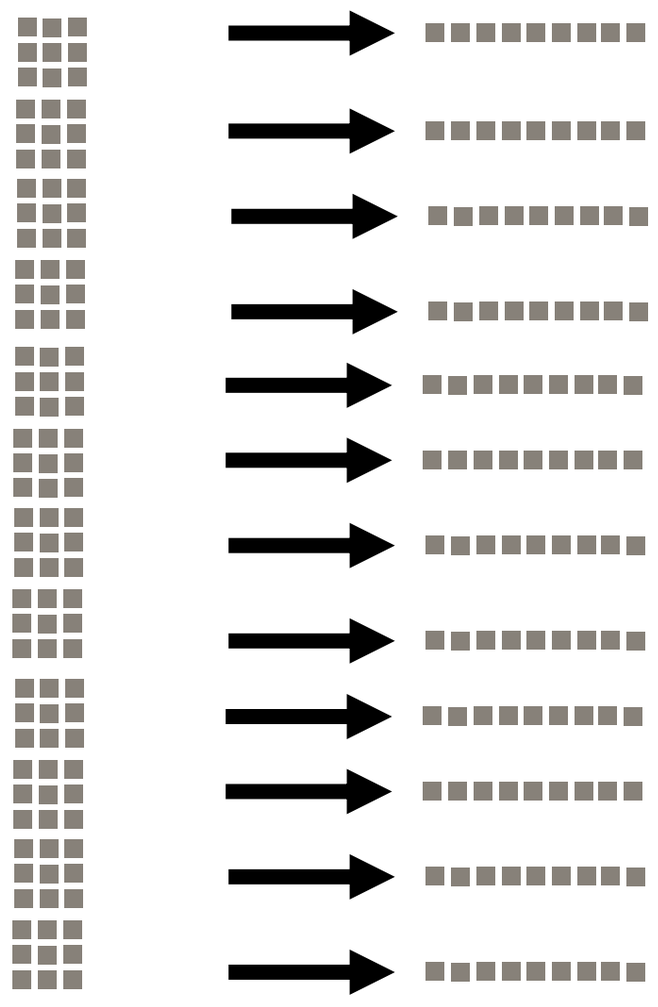
3x3



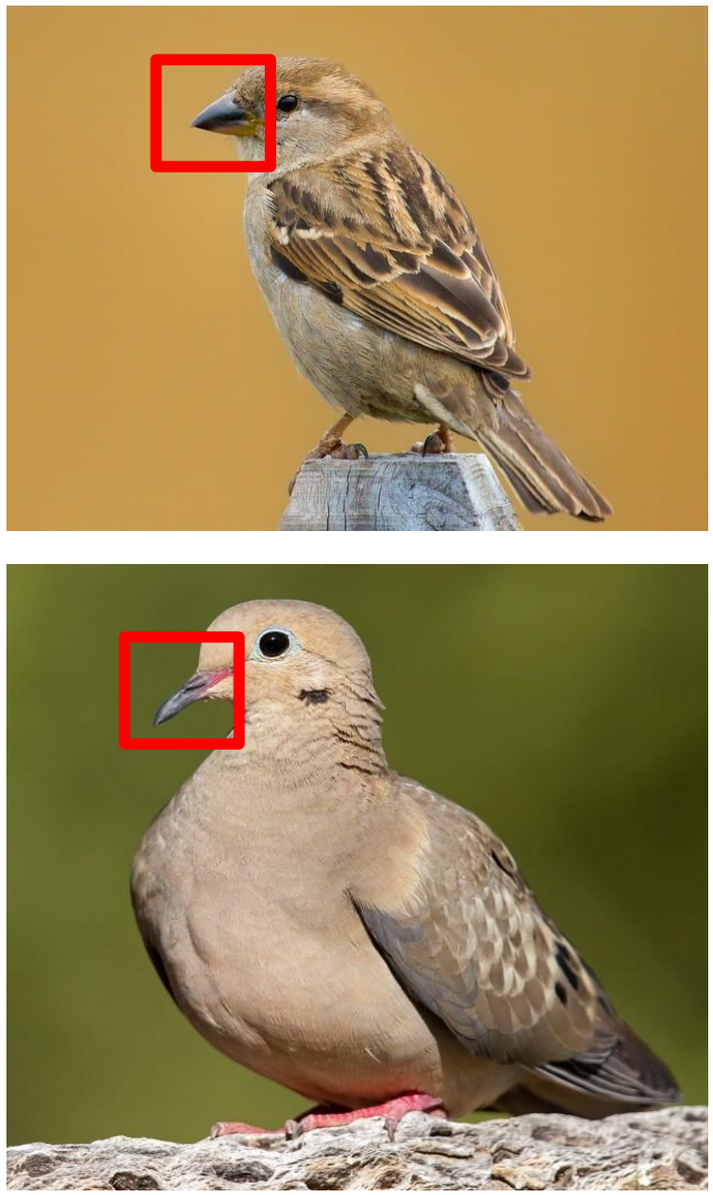
9x12

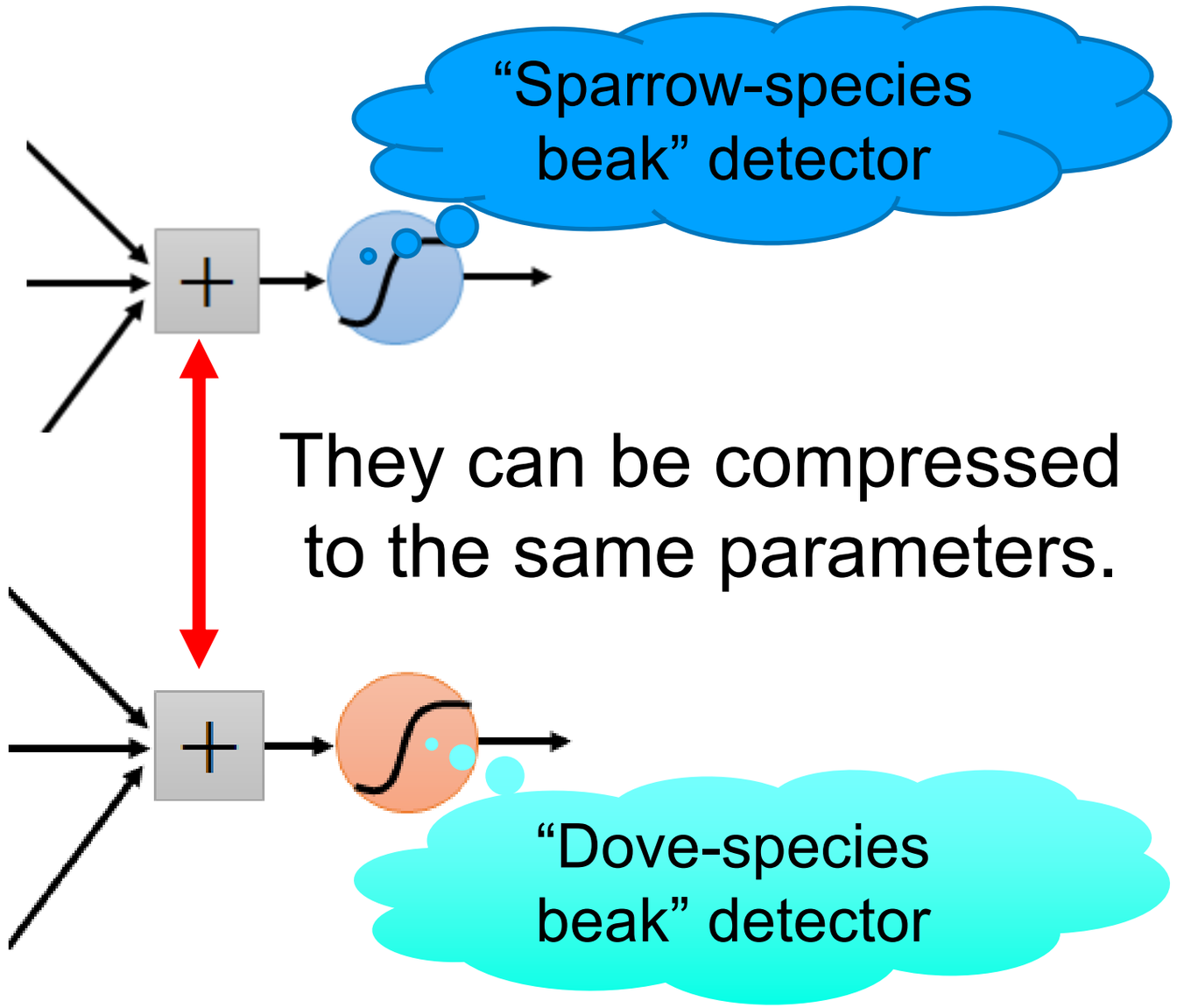






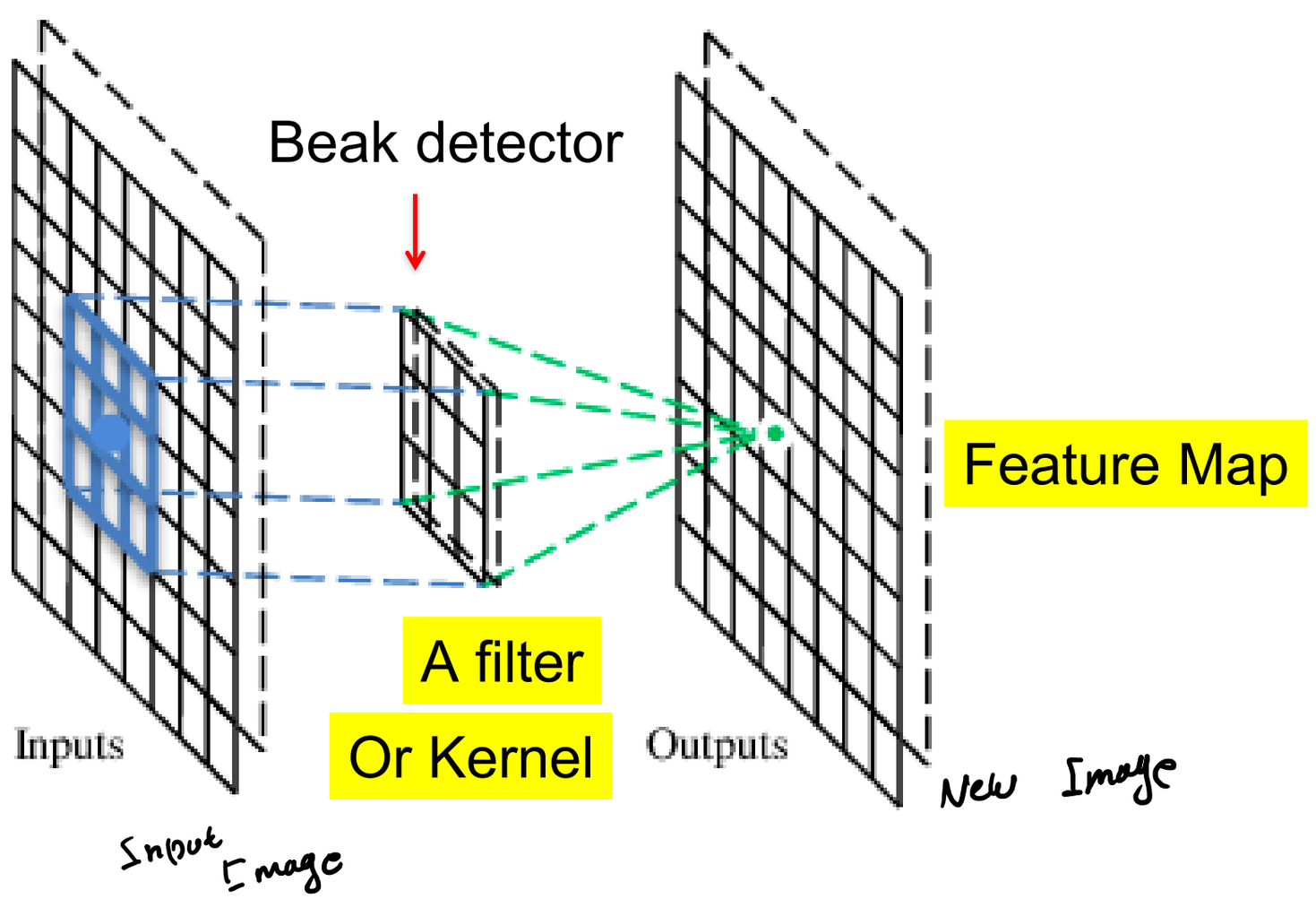
Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector must “move around”.





A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution

These are the network parameters to be learned.

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

$\theta_1 x_1 + \cdots + \theta_9 x_9 = lc$

θ_1	θ_2	θ_3
1	-1	-1
-1	1	-1
-1	-1	$1\theta_9$

Filter 1

stride=1

x_1	x_2	x_3			
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

x_9

6 x 6 image

Dot product →

3 -1

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

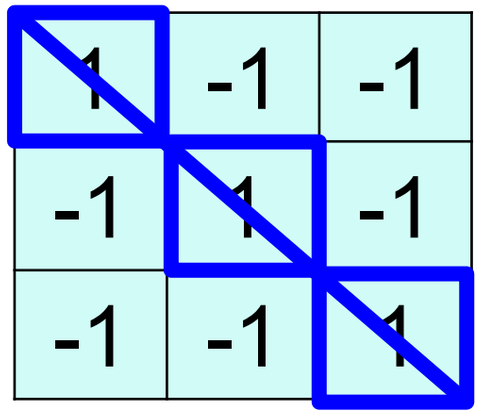
If stride=2

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

6 x 6 image

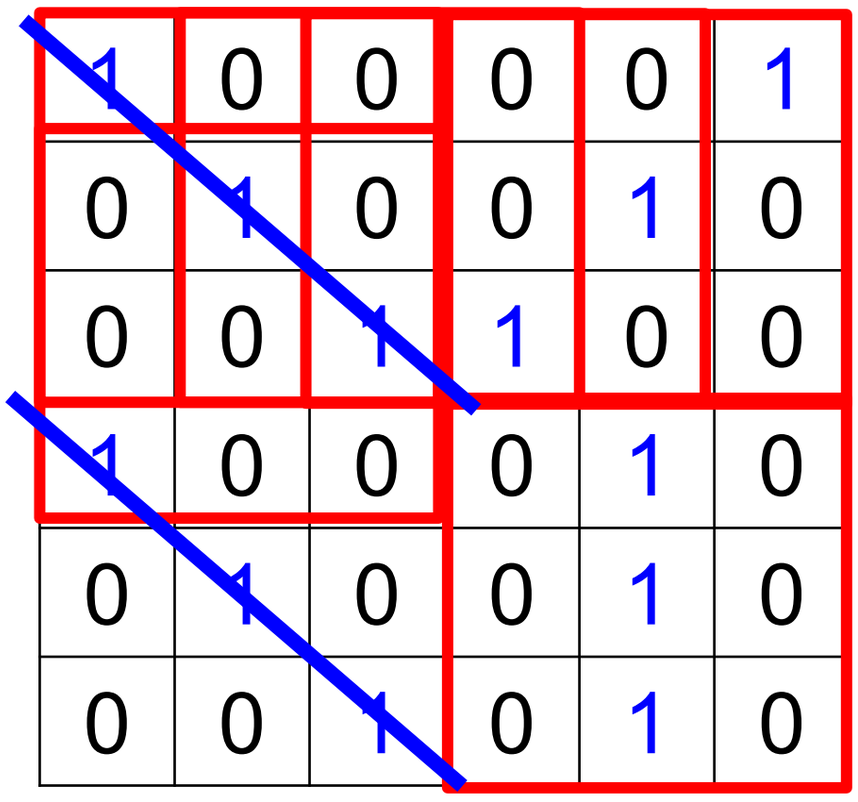
3 -3

Convolution

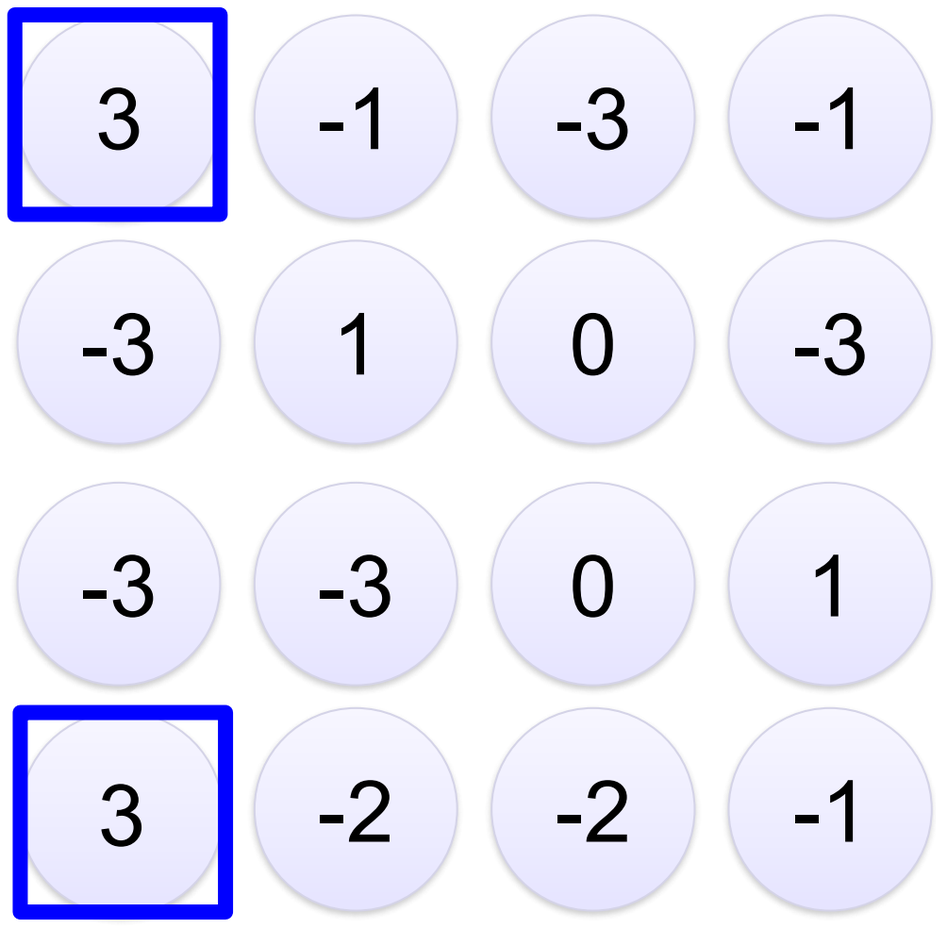


Filter 1

stride=1



6 x 6 image



Convolution

-1	1	-1
-1	1	-1
-1	1	-1

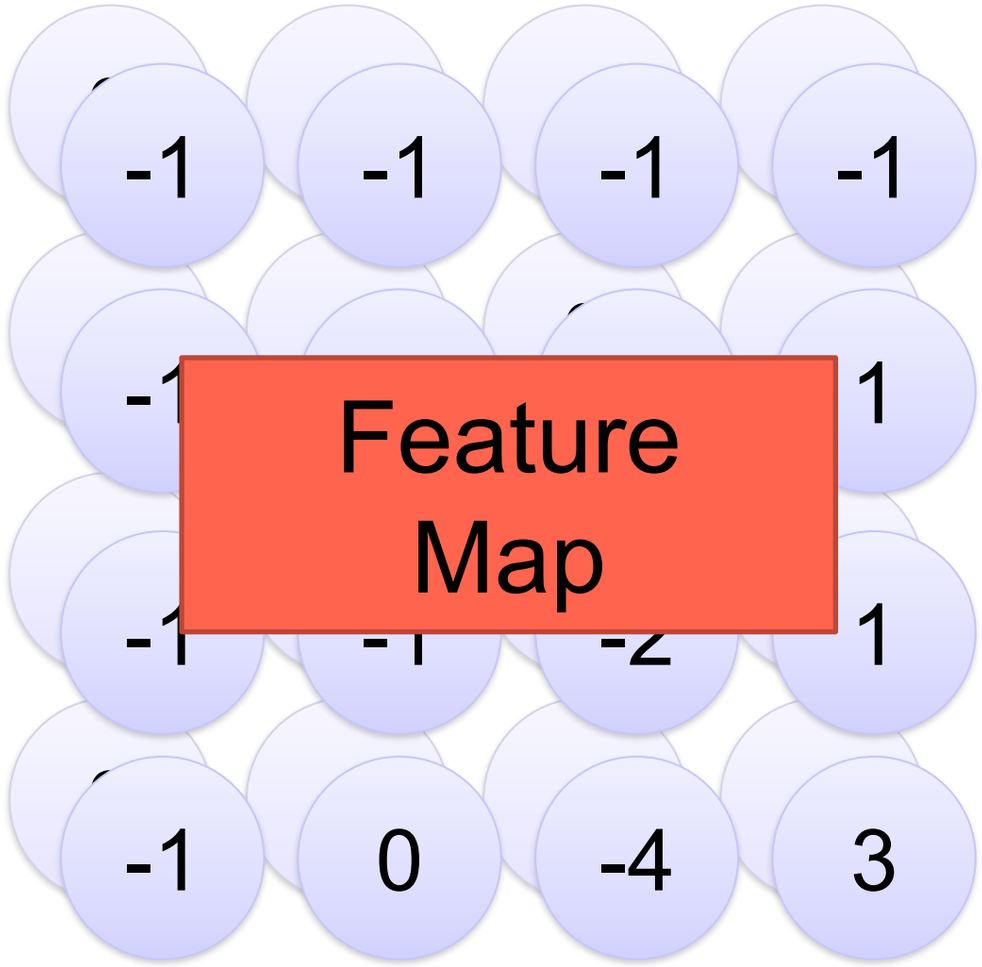
Filter 2

stride=1

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

6 x 6 image

Repeat this for each filter



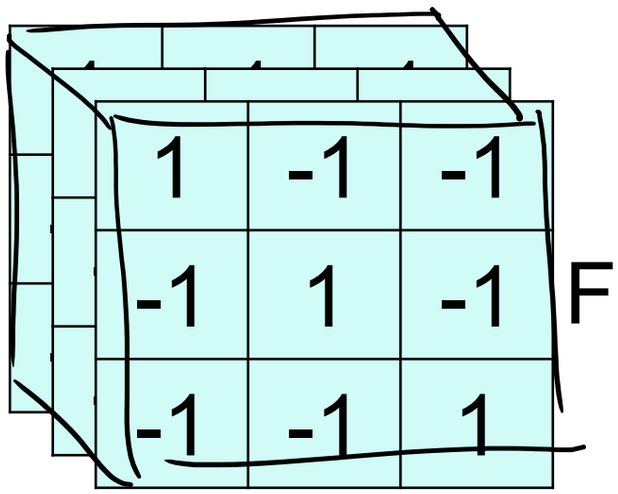
Two 4 x 4 images

Forming 2 x 4 x 4 matrix

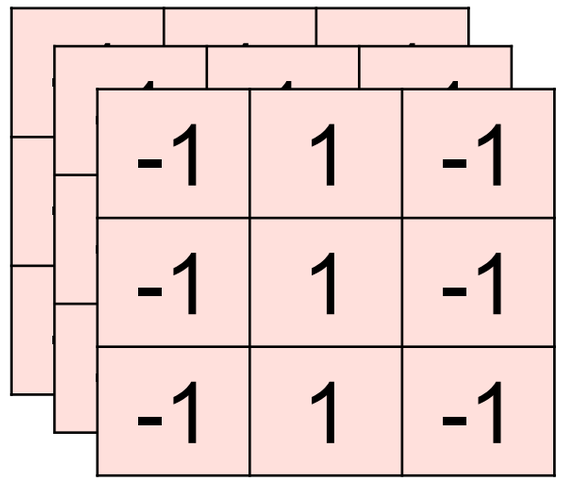
Color image: RGB 3 channels

Sub = [1 0 0 0 1 0 0 0 1 1 .....]$_{1\times 27}$ Sub. $F1 \in \mathbb{R}$

F1 = [1 -1 -1 -1 ...]$_{1\times 27}$

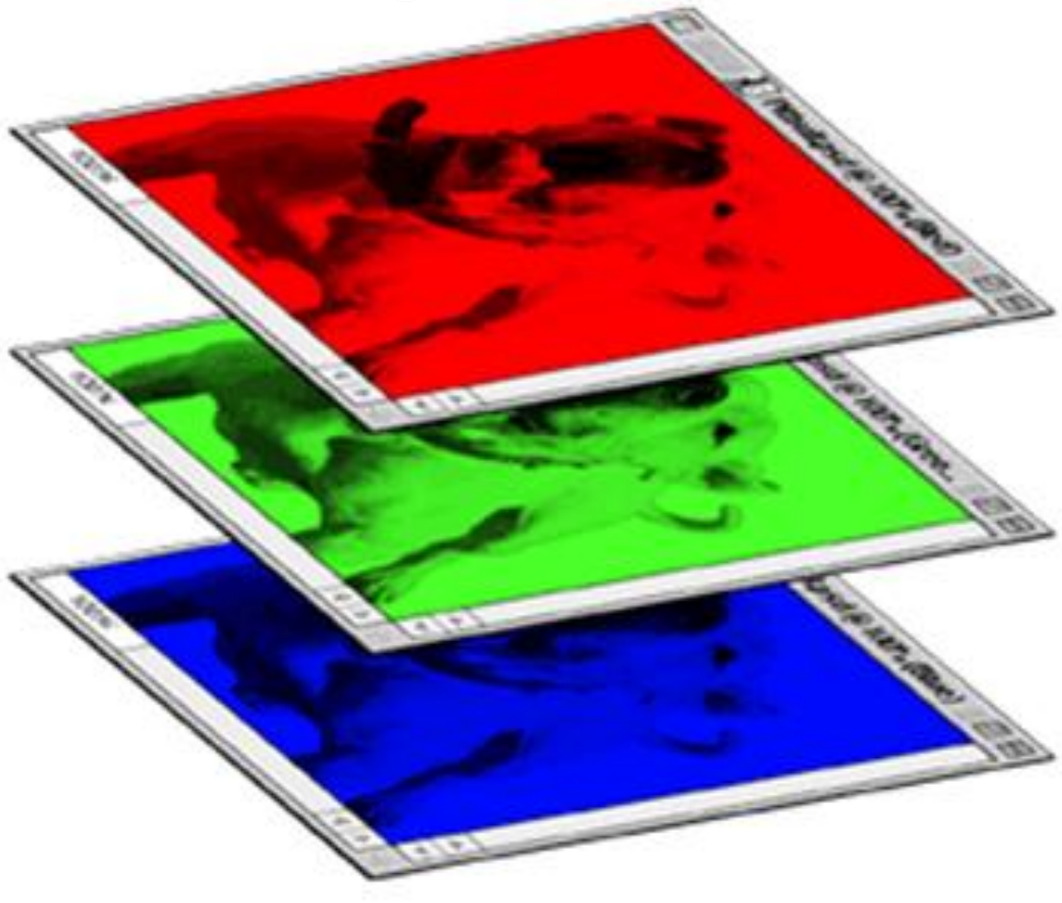


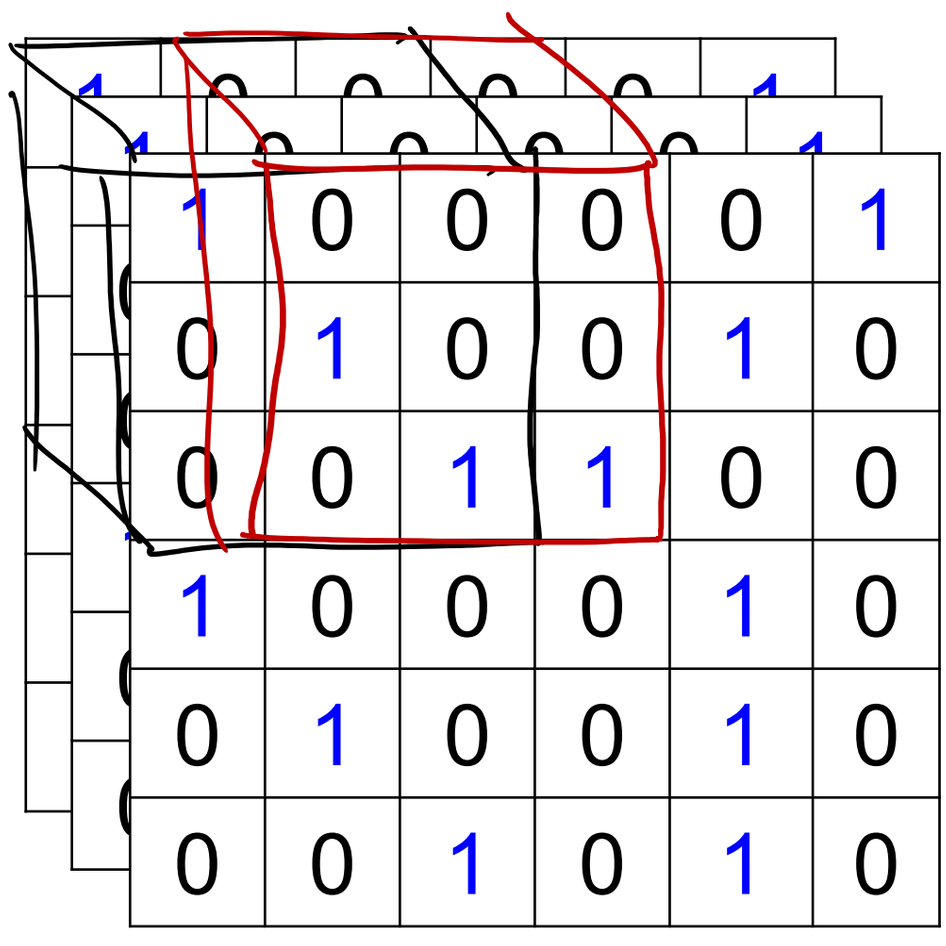
Filter 1



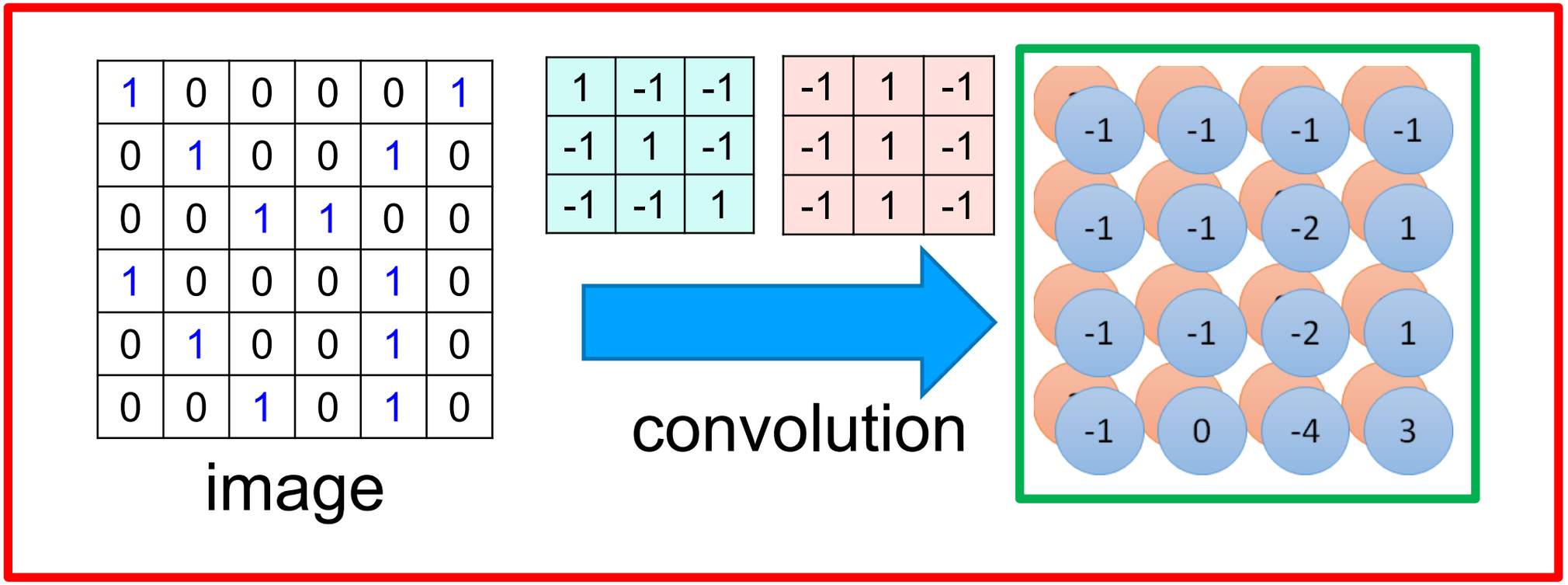
Filter 2

Color image

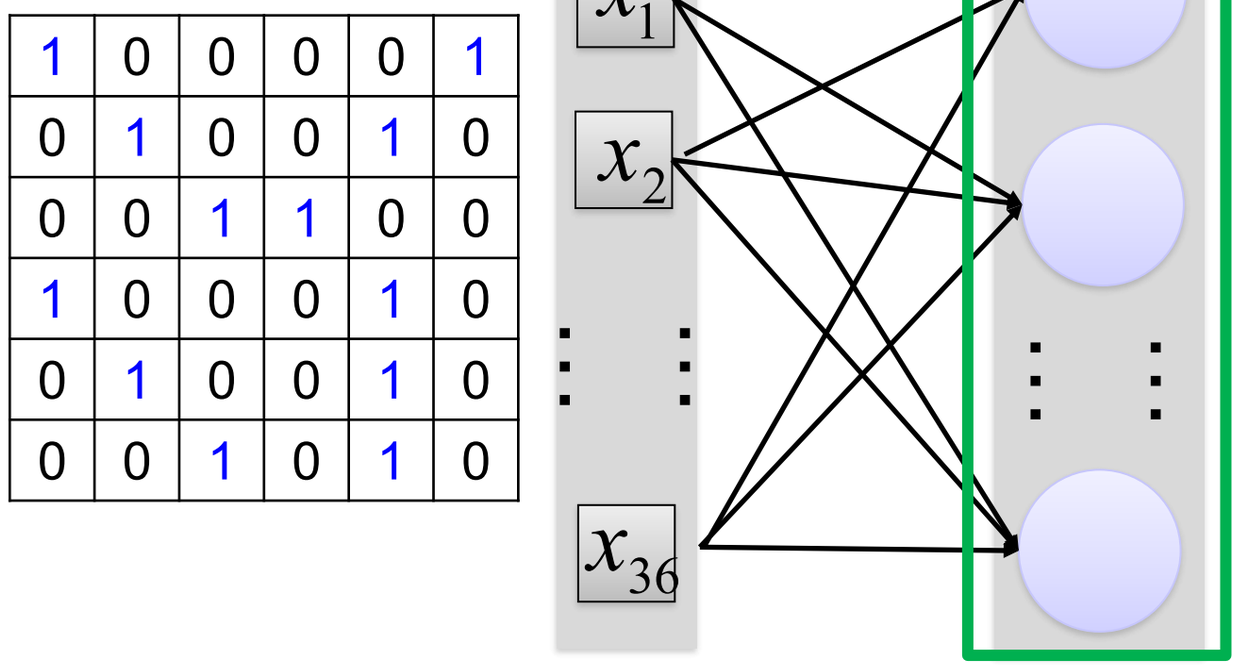




Convolution v.s. Fully Connected



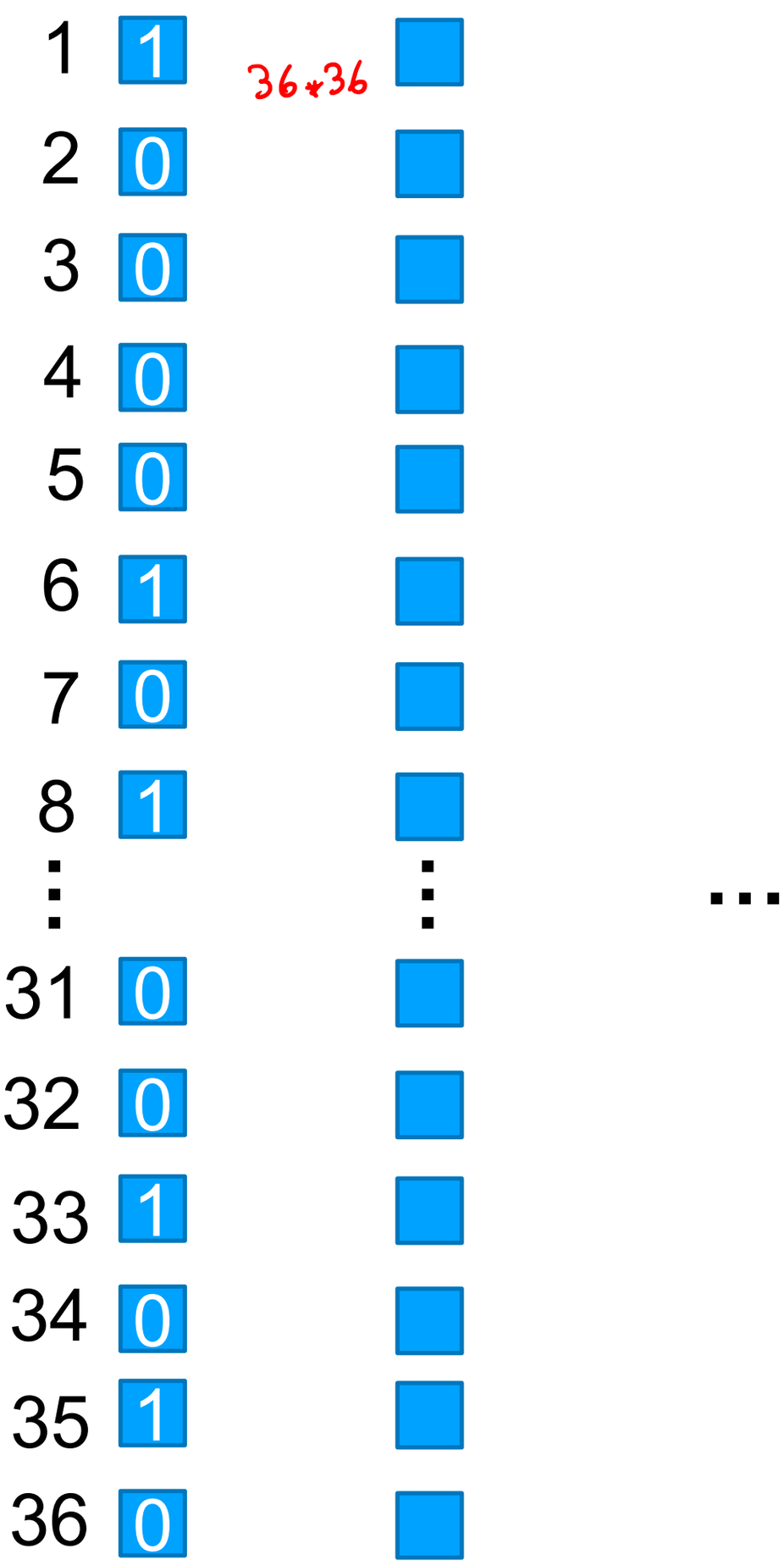
Fully-connected

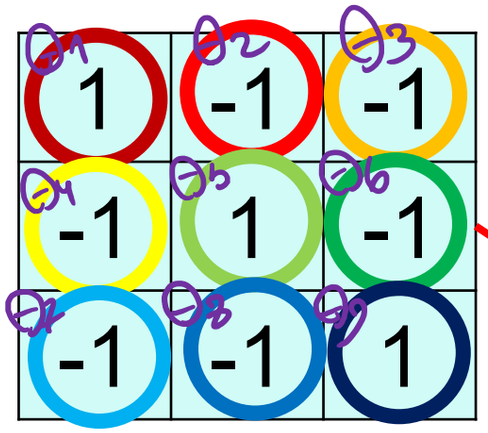


Conventional
Fully Connected
layers
(FC layers)

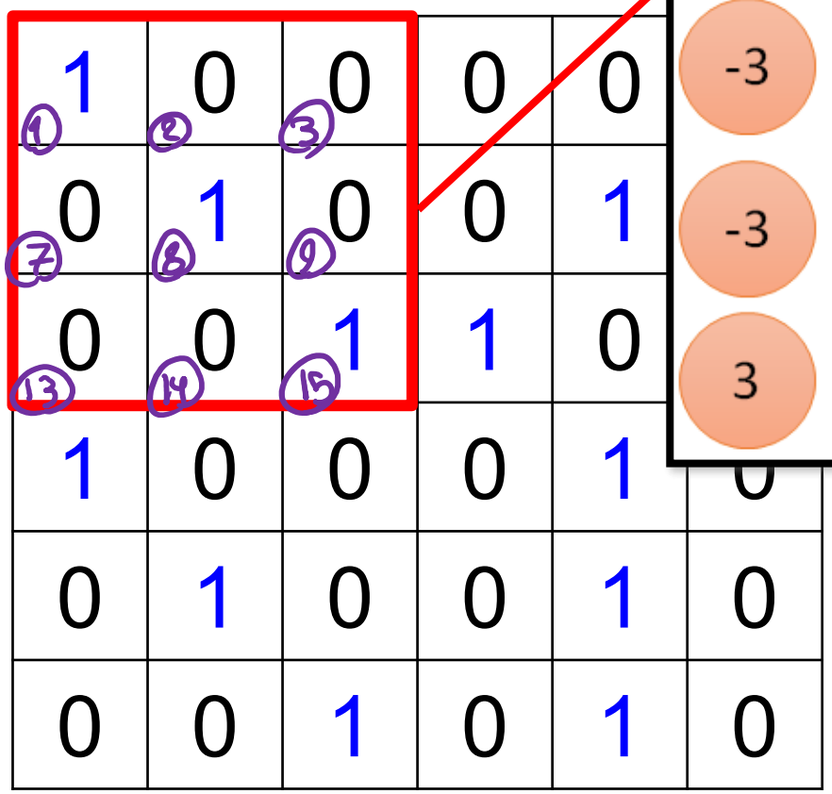
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

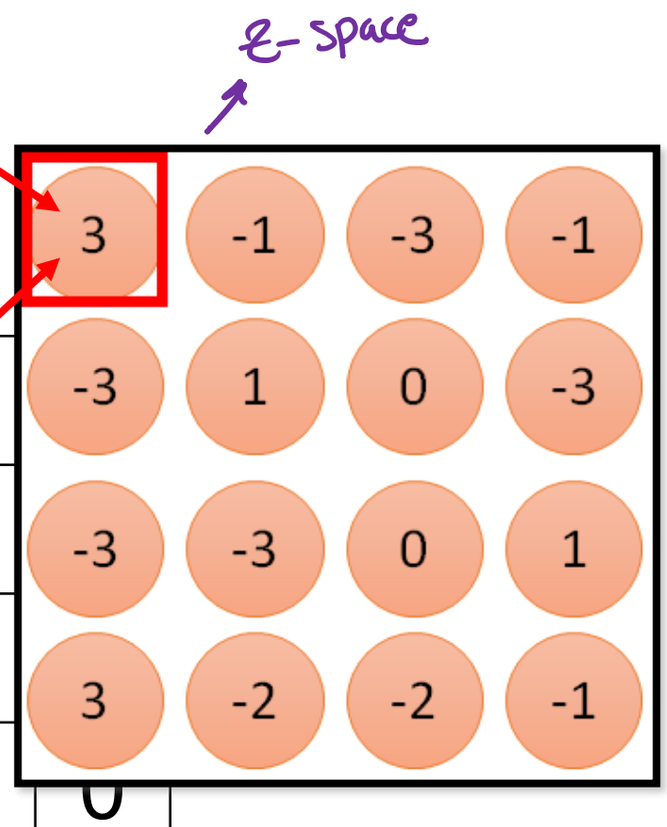




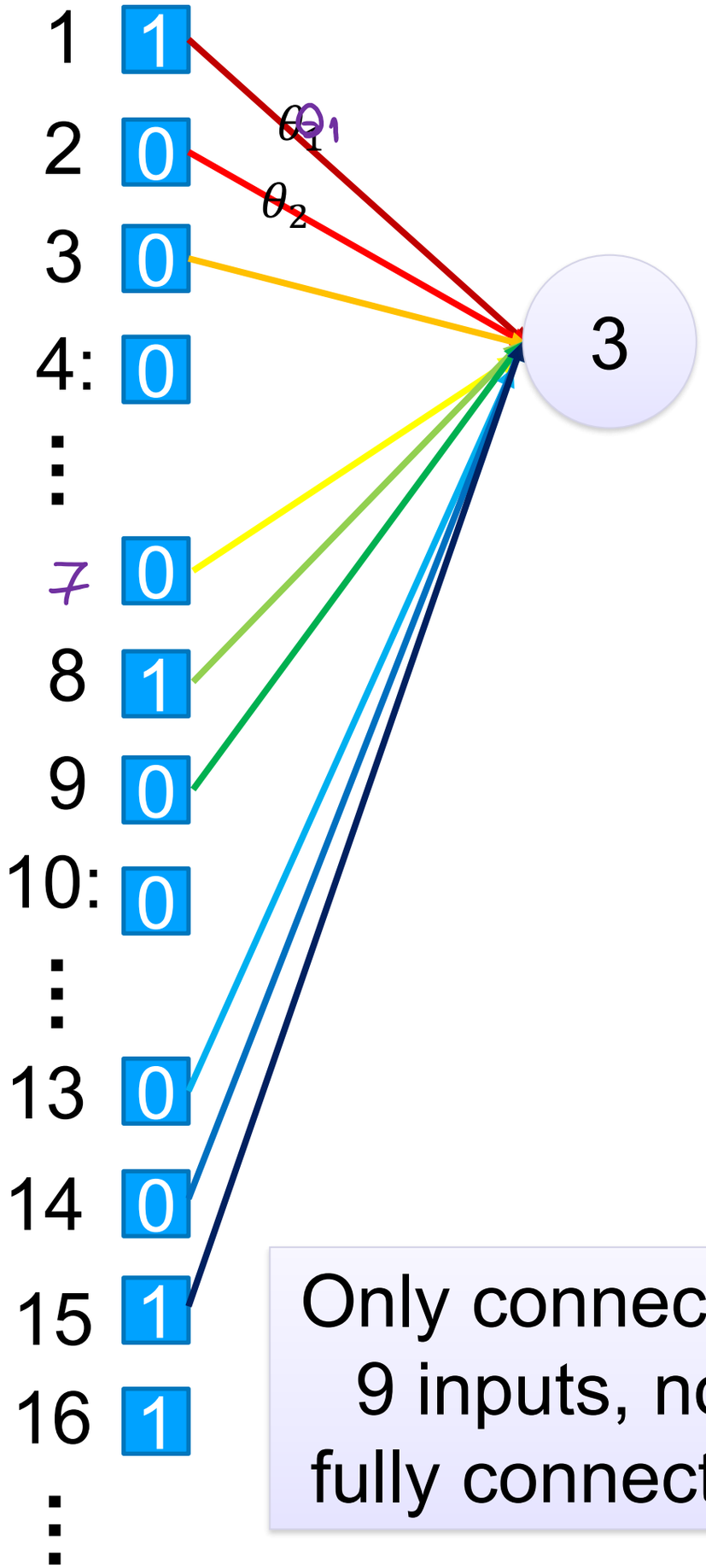
Filter 1



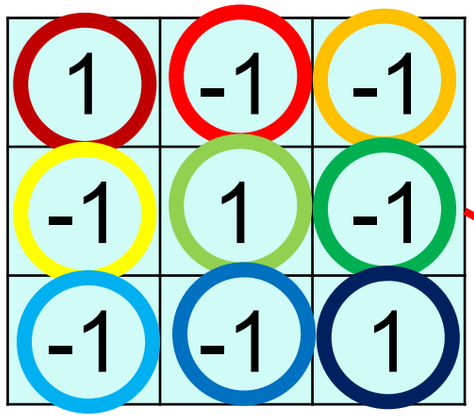
6 x 6 image



fewer parameters!

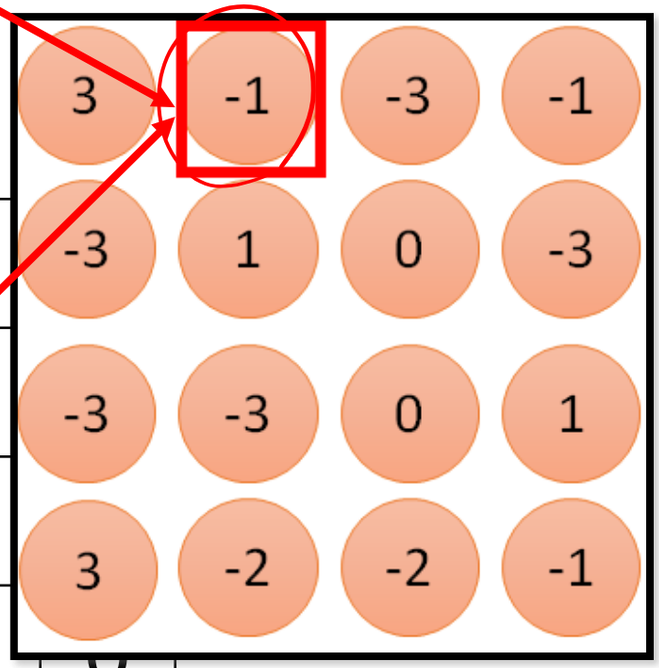


Only connect to 9 inputs, not fully connected



Filter 1

LCF

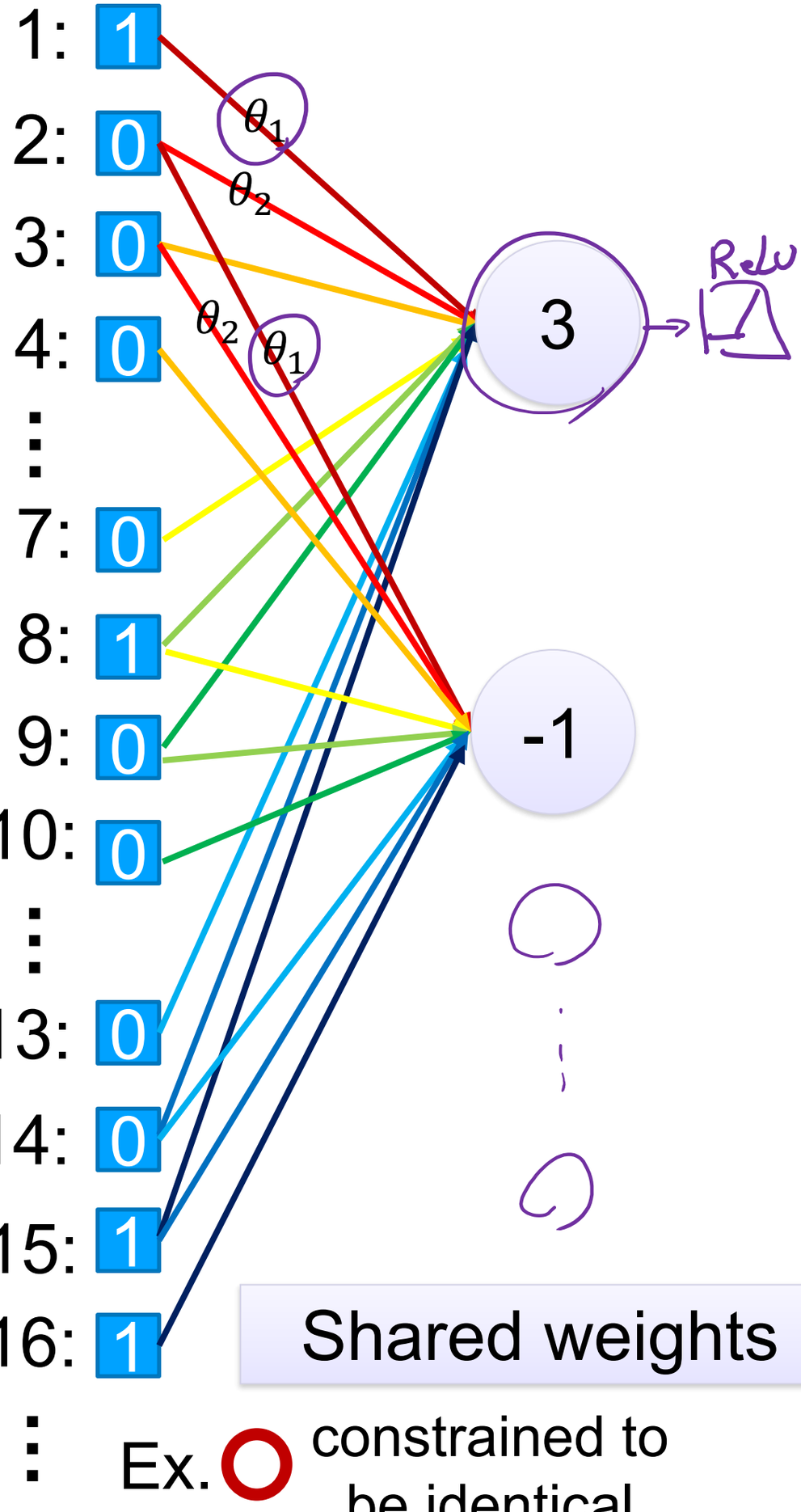


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Fewer parameters

Even fewer parameters



Shared weights

Ex. O constrained to be identical

The whole CNN

Convolution

Max Pooling

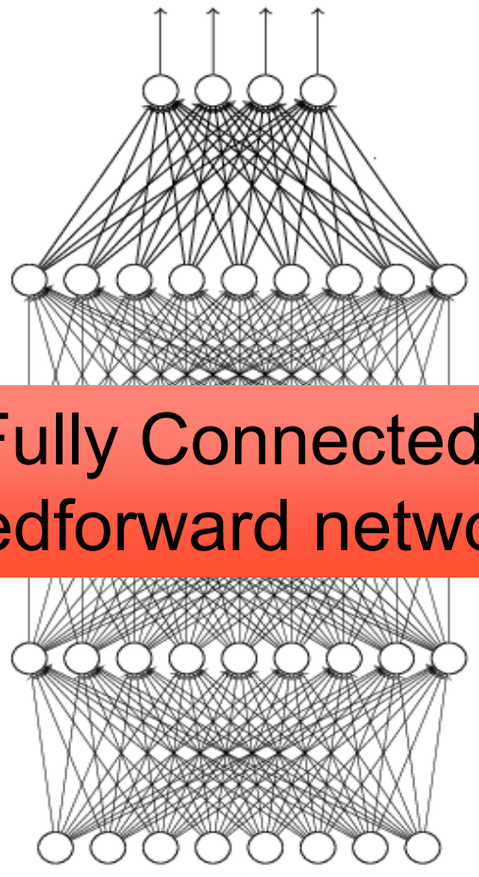
Convolution

Max Pooling

Can repeat many times

Flattened

Fully Connected Feedforward network



cat dog ……

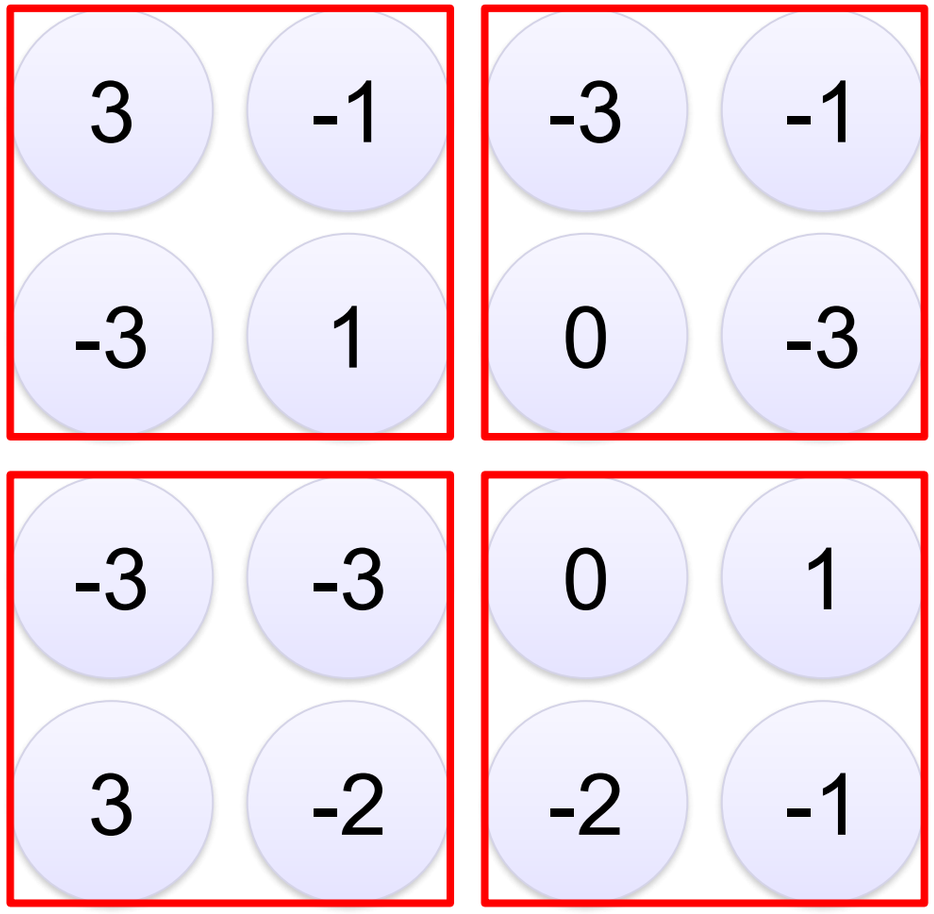
Max Pooling

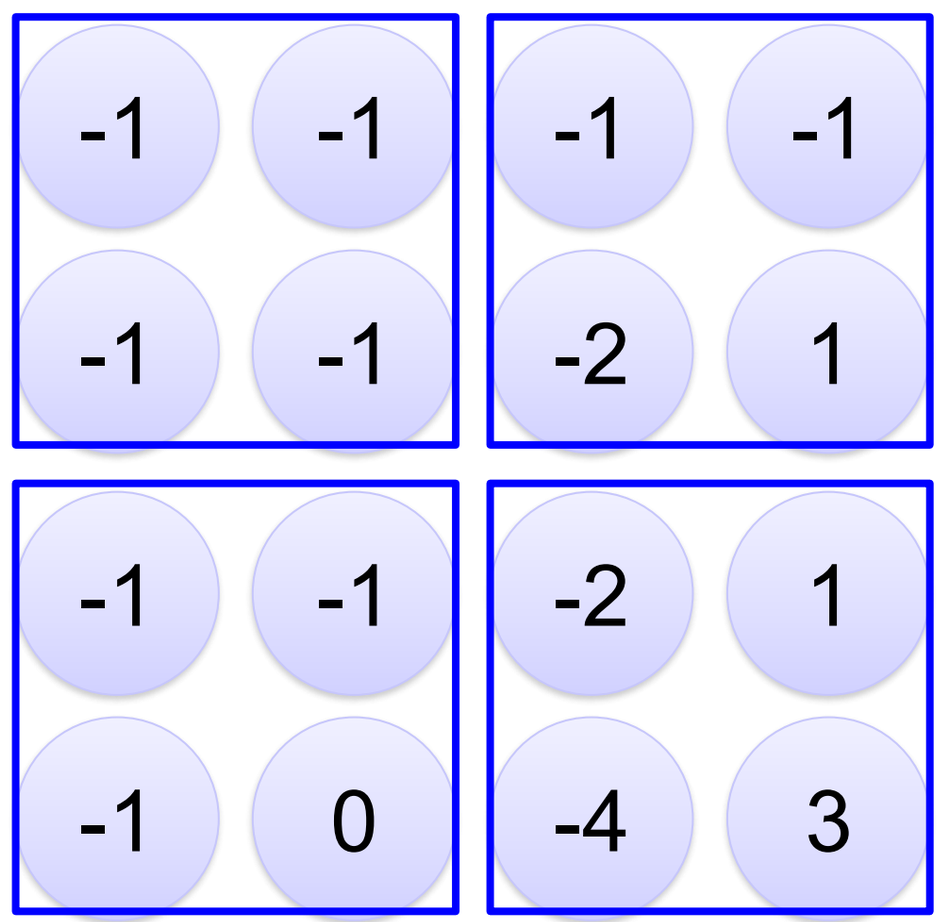
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2





Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image

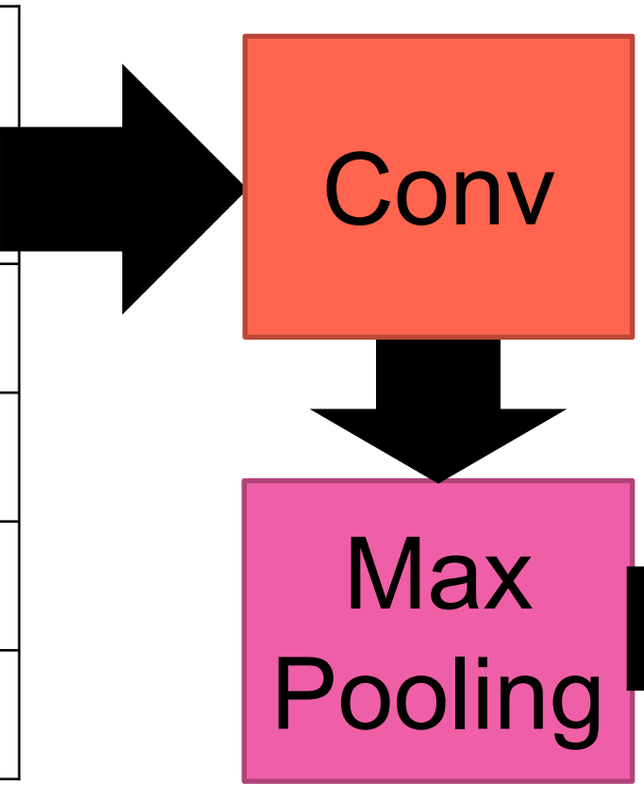
A CNN compresses a fully connected network in three ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

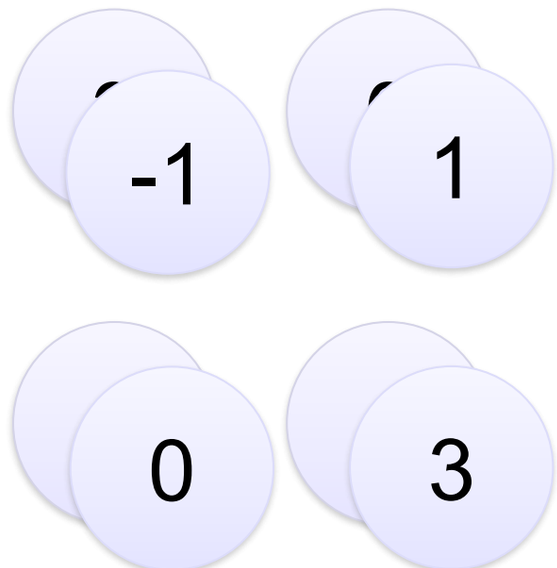
Max Pooling

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

6 x 6 image



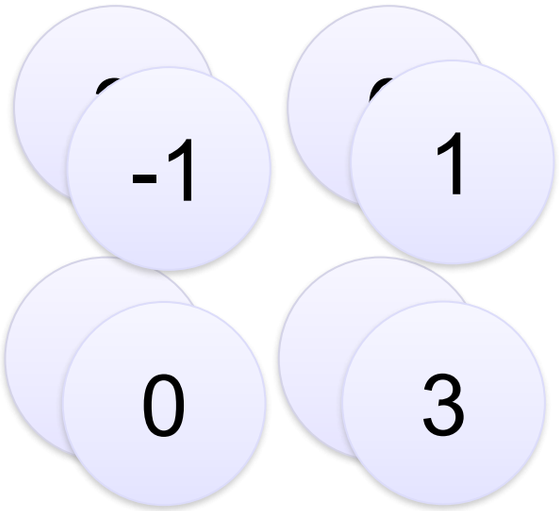
New image but smaller



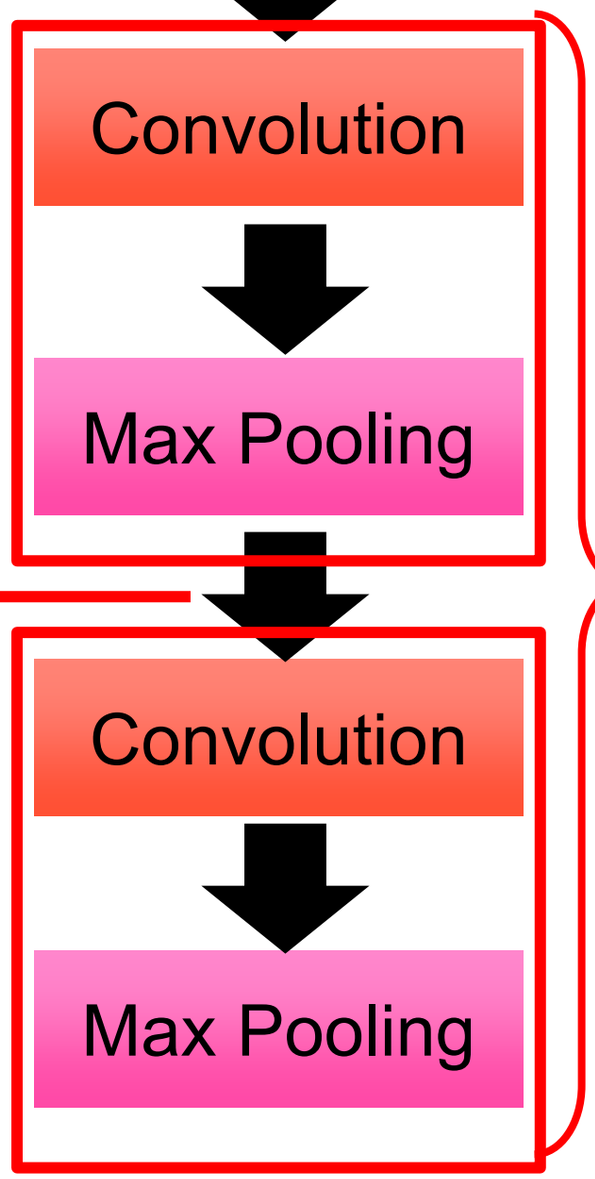
2 x 2 image

Each filter is a channel

The whole CNN



A new image



Smaller than the original image

The number of channels is the number of filters

The whole CNN

Convolution

Max Pooling

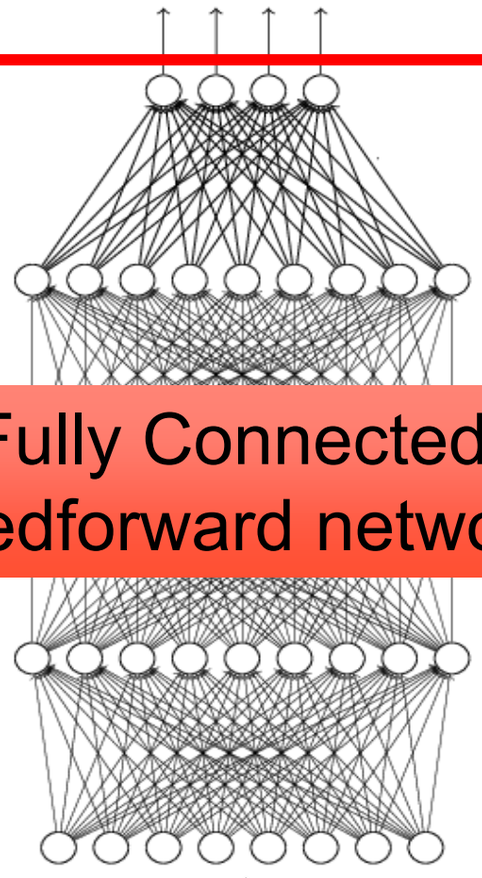
A new image

Convolution

Max Pooling

A new image

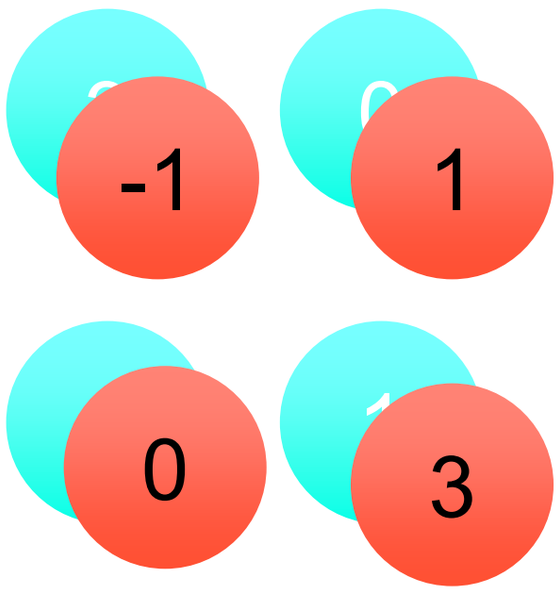
cat dog ……



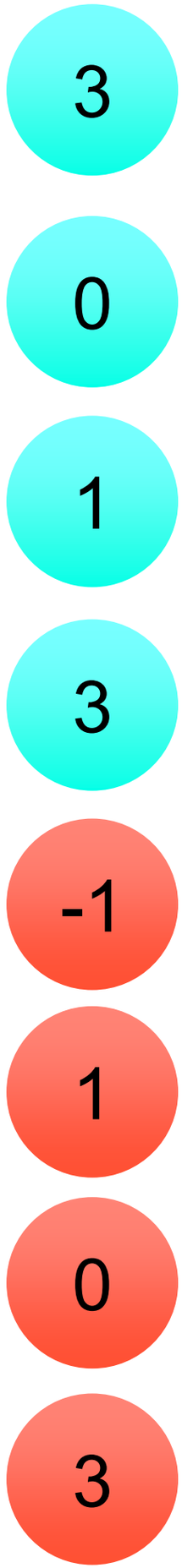
Fully Connected Feedforward network

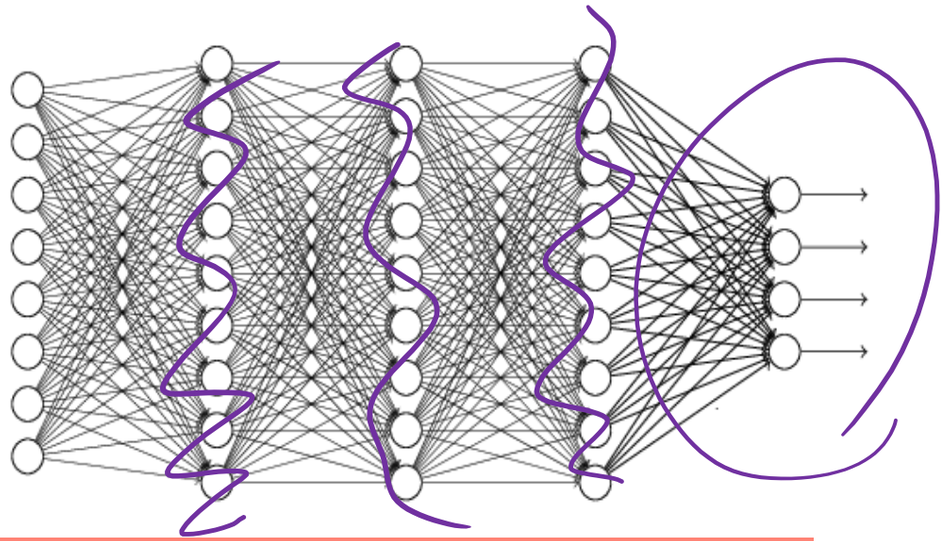
Flattened

Flattening



Flattened



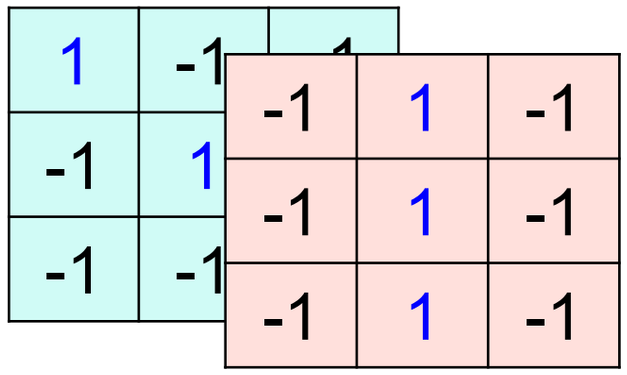


Fully Connected
Feedforward network

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



…

…

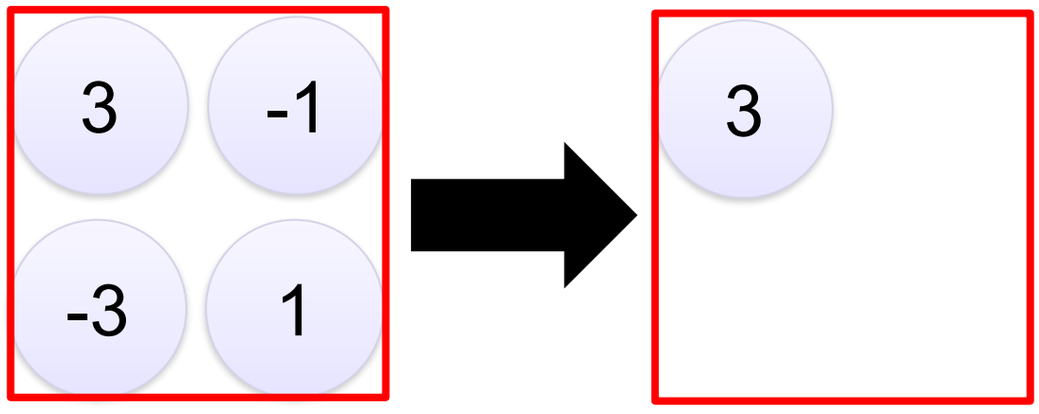
There are 25 3x3 filters.

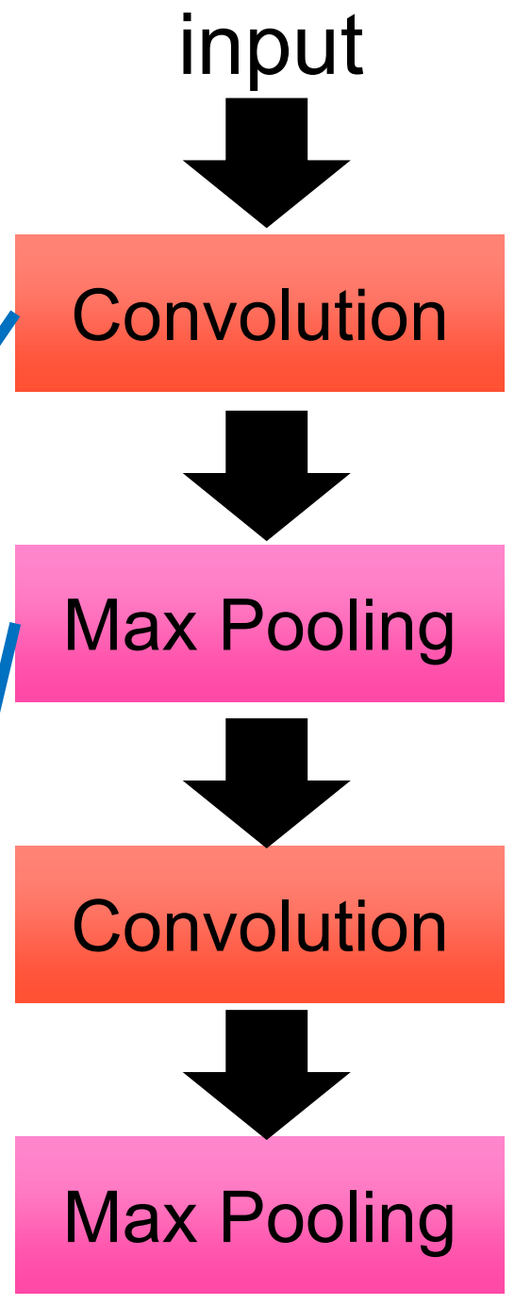
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

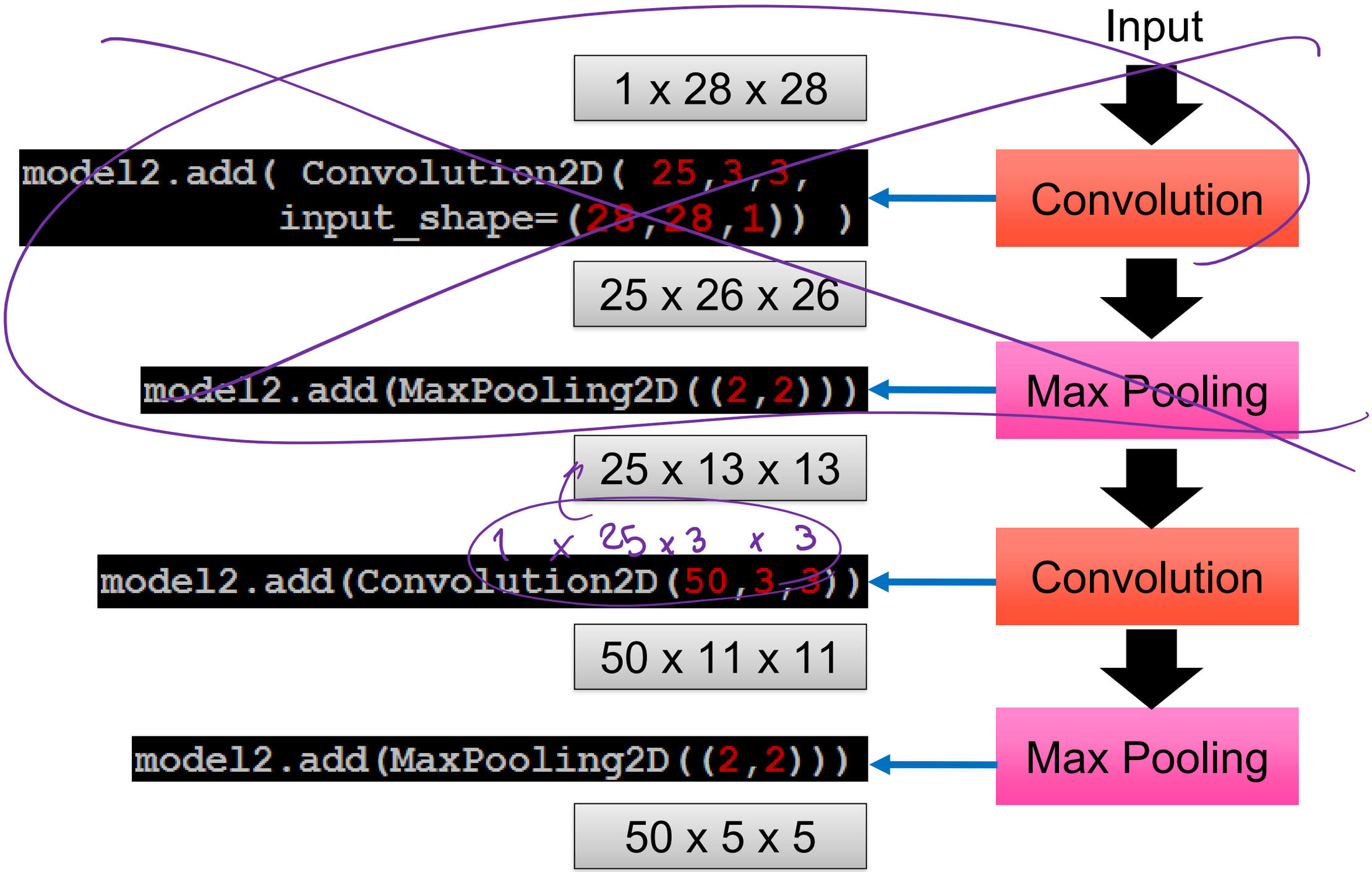
```
model2.add(MaxPooling2D((2,2)))
```





CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

Convolution + ReLU

25 x 26 x 26

Max Pooling

25 x 13 x 13

Convolution + ReLU

50 x 11 x 11

Max Pooling

50 x 5 x 5

Flattened

```
model2.add(Flatten())
```

1250

Fully connected feedforward network

```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

Output

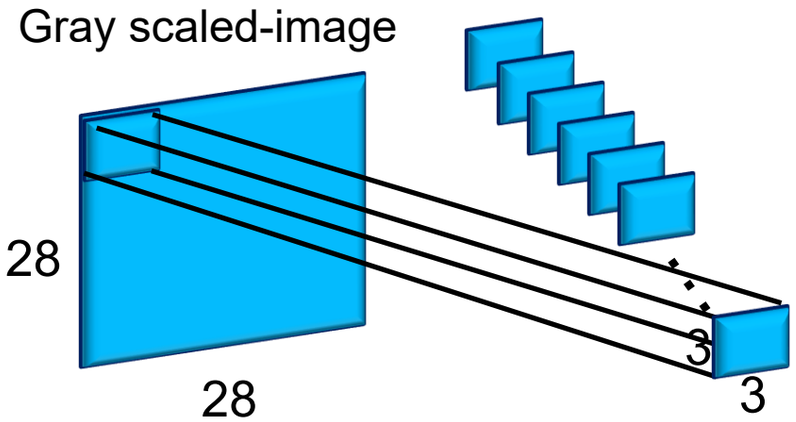
Number of Parameters

25X3X3+25 parameters

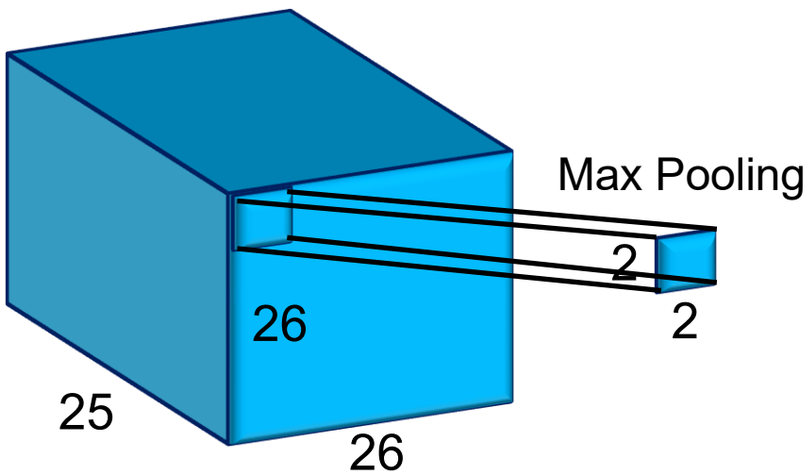
25 filters - Conv1

25: 3X3

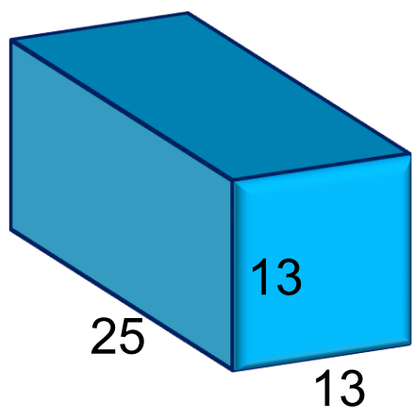
Gray scaled-image

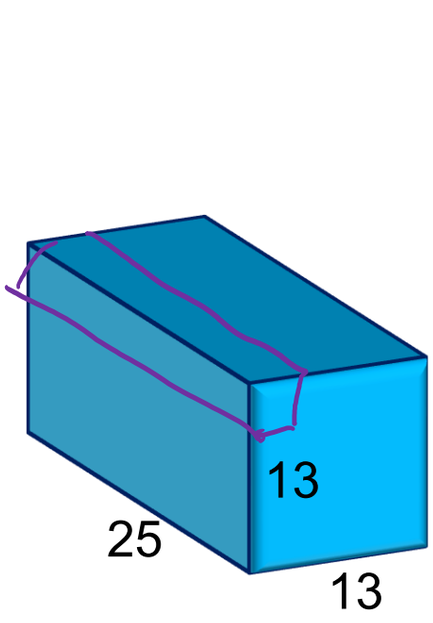


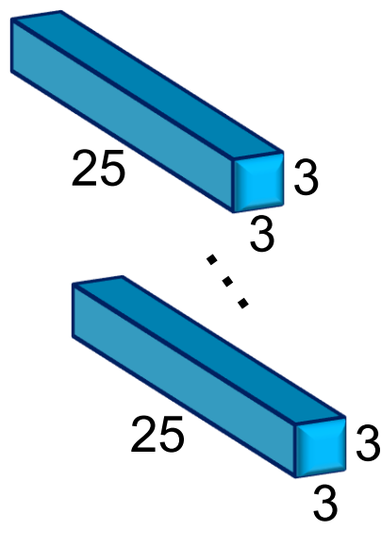
Convoluted result for 25 filters



Result after Max Pooling

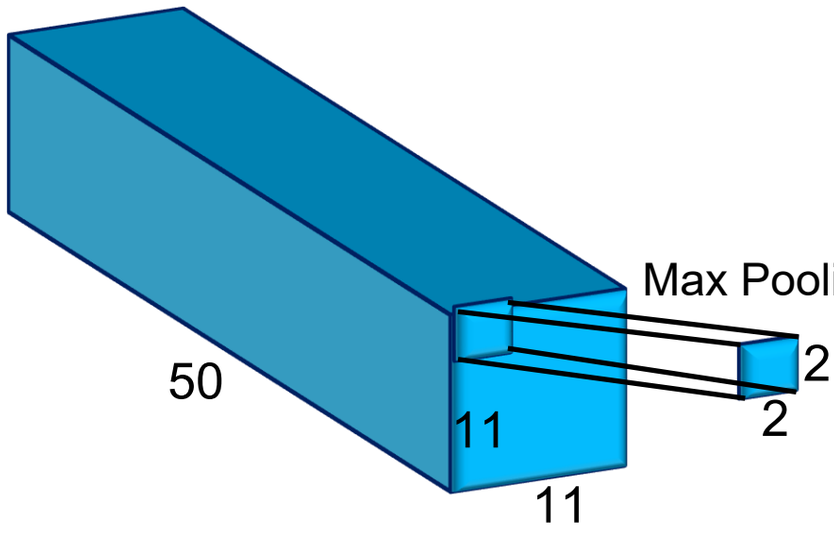






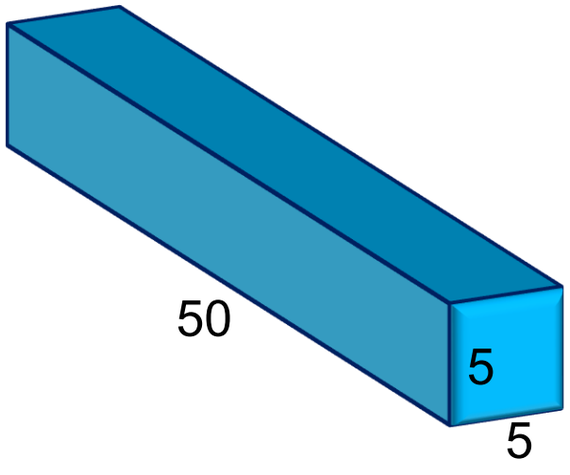
50: 3X3X25

50 filters – Conv2

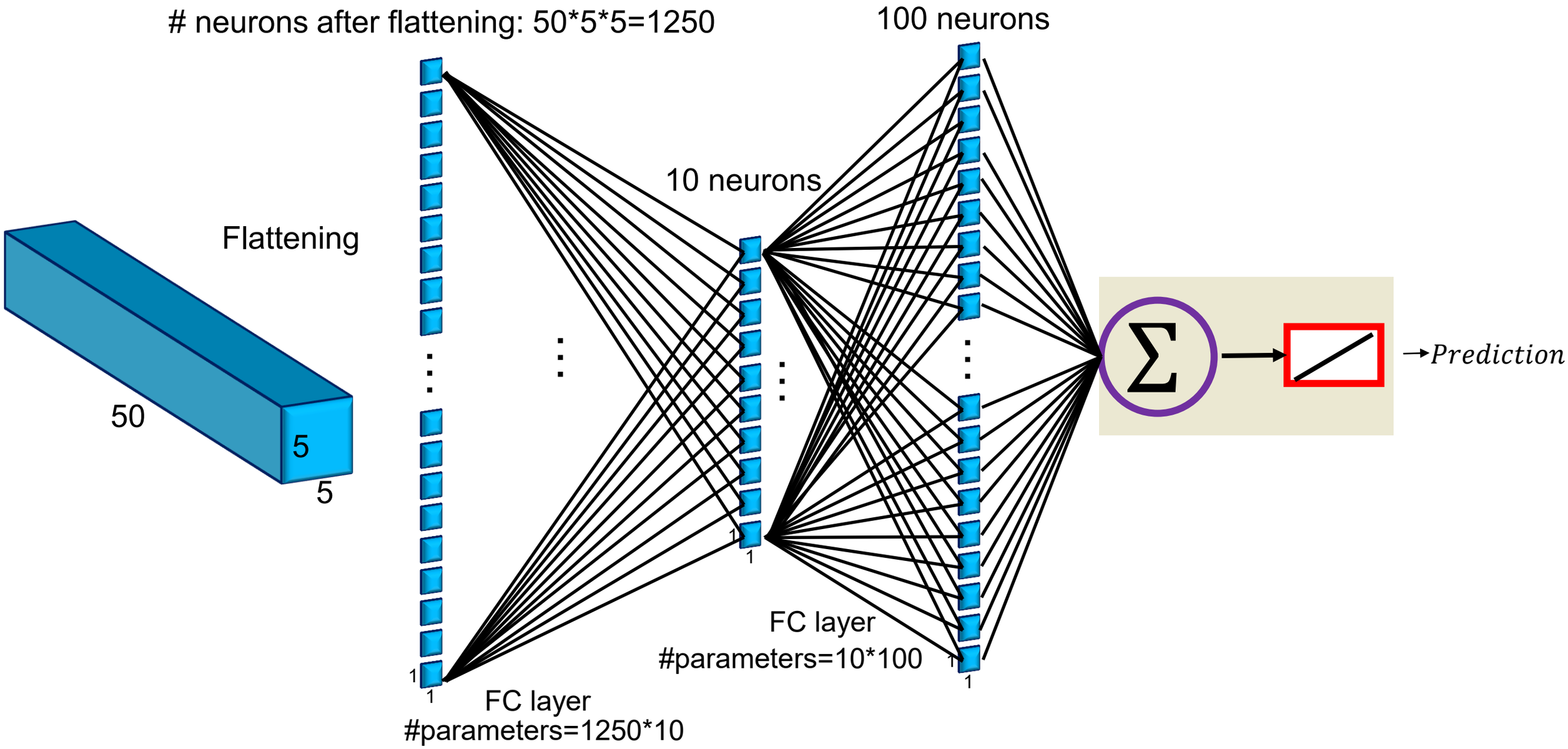


Convoluted result for 50 filters

Result after Max Pooling



50X3X3X25+50 parameters



10 CNN Architecture