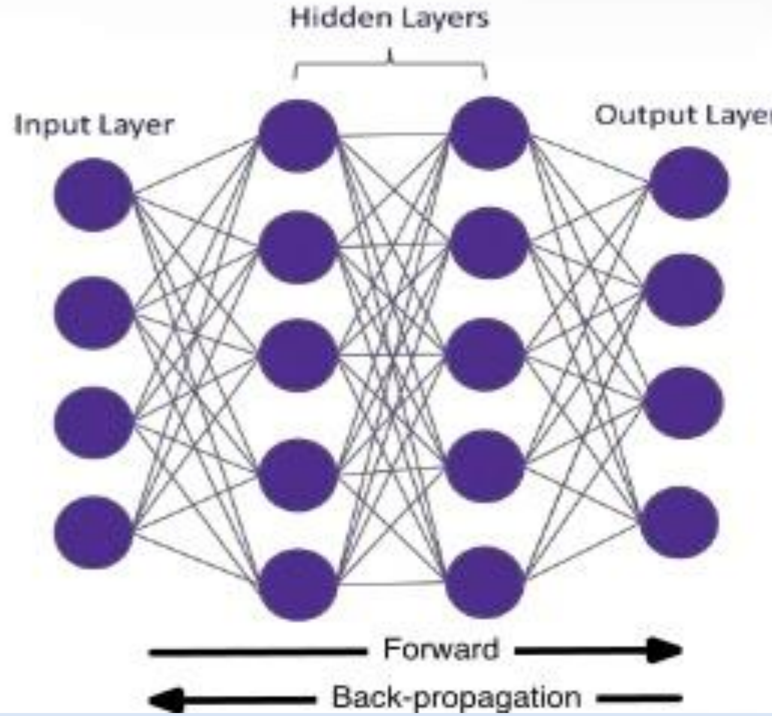
Deep Learning



Lecture 13: Convolution Neural Network

Presented by : Dr. Hanaa Bayomi
h.mobarz@fci-cu.edu.eg

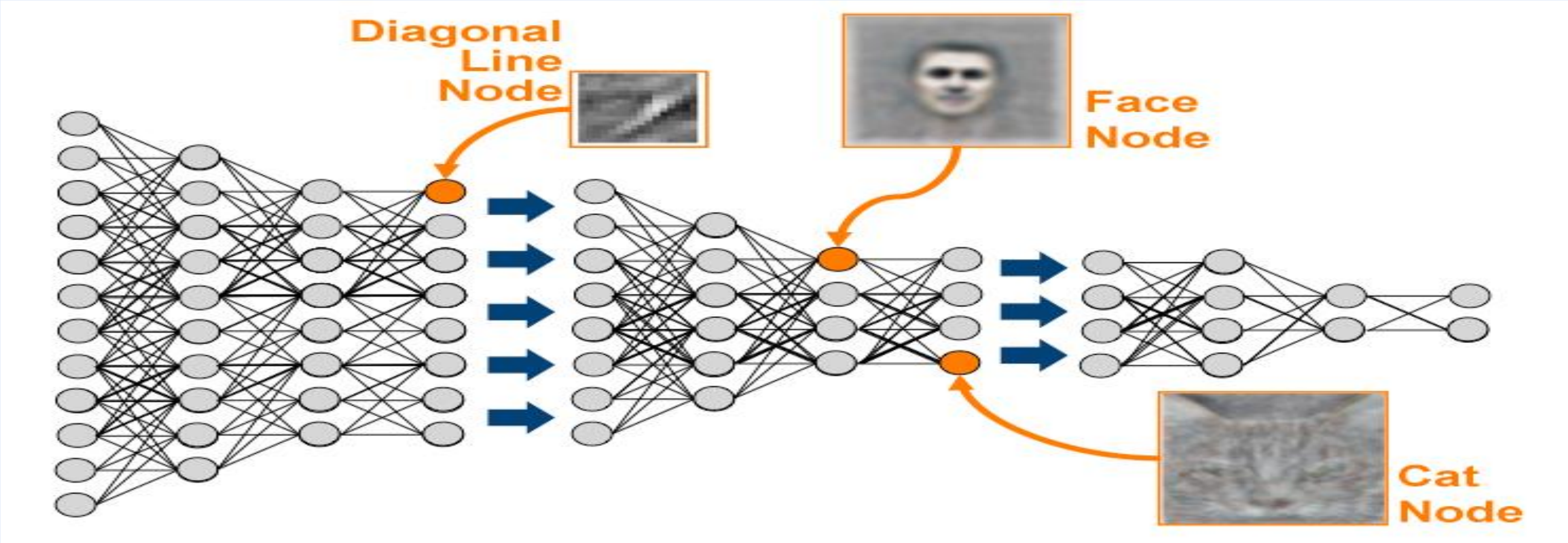
Deep Learning definition

- *Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.*
- Learning deep (many layered) neural networks
- The more layers in a Neural Network, the more abstract features can be represented

Deep Learning definition

E.g. Classify a cat:

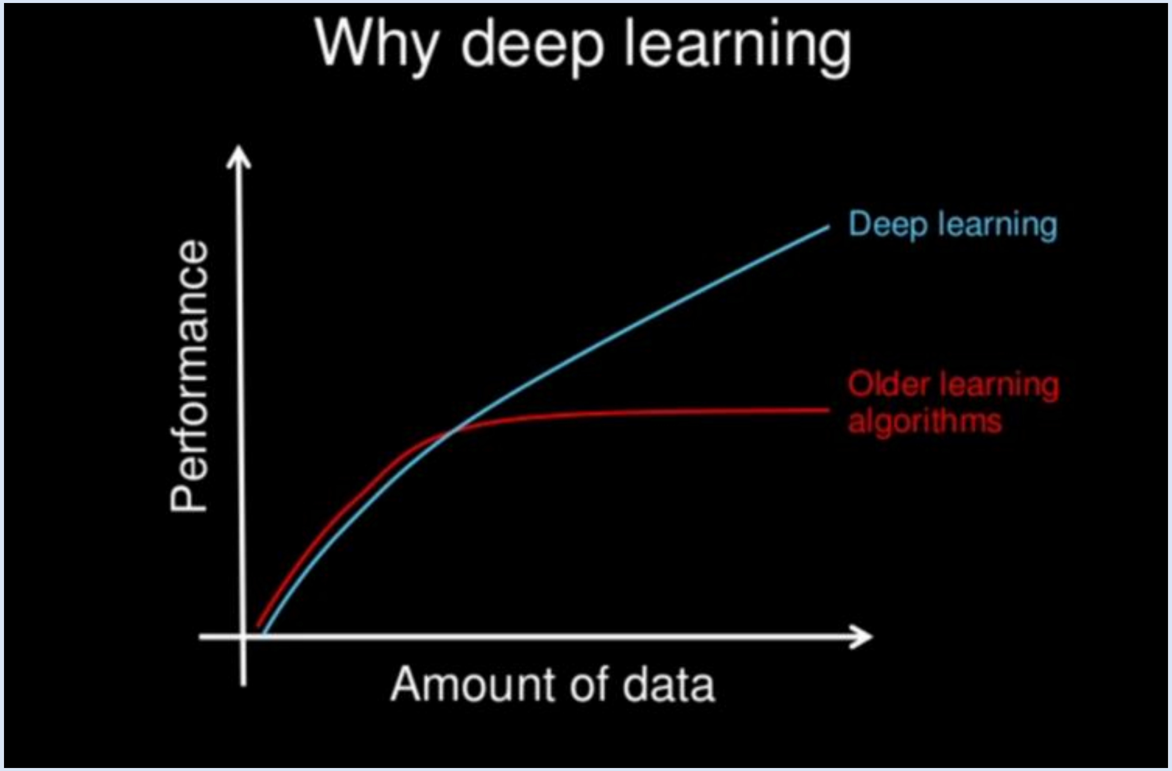
– Bottom Layers: Edge detectors, curves, corners straight lines
– Middle Layers: Fur patterns, eyes, ears
– Higher Layers: Body, head, legs
– Top Layer: Cat or Dog



Machine Learning VS Deep Learning

1- Data Dependency

- Deep learning need large amount of data to understand it perfectly



Machine Learning VS Deep Learning

2- Hardware Dependency

- Deep learning algorithms heavily depend on **high-end machines** This is because the requirements of deep learning algorithm include GPUs which are an integral part of its working.
- Machine Learning which can work on **low-end machines**.

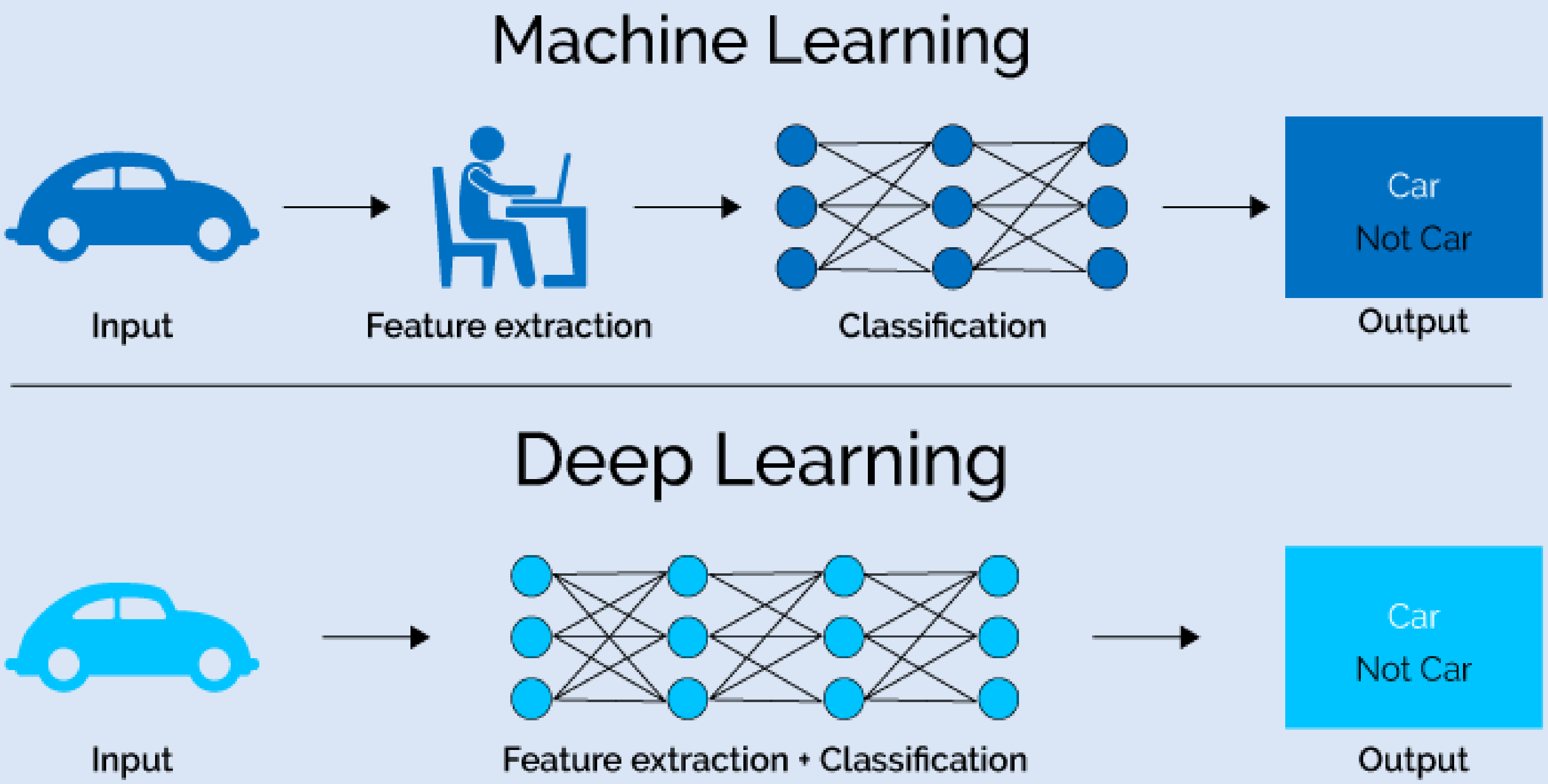
3- Execution time

- deep learning algorithm takes a long time to train. This is because there are so many parameters in a deep learning algorithm that training them takes longer than usual.

Machine Learning VS Deep Learning

4- Feature engineering

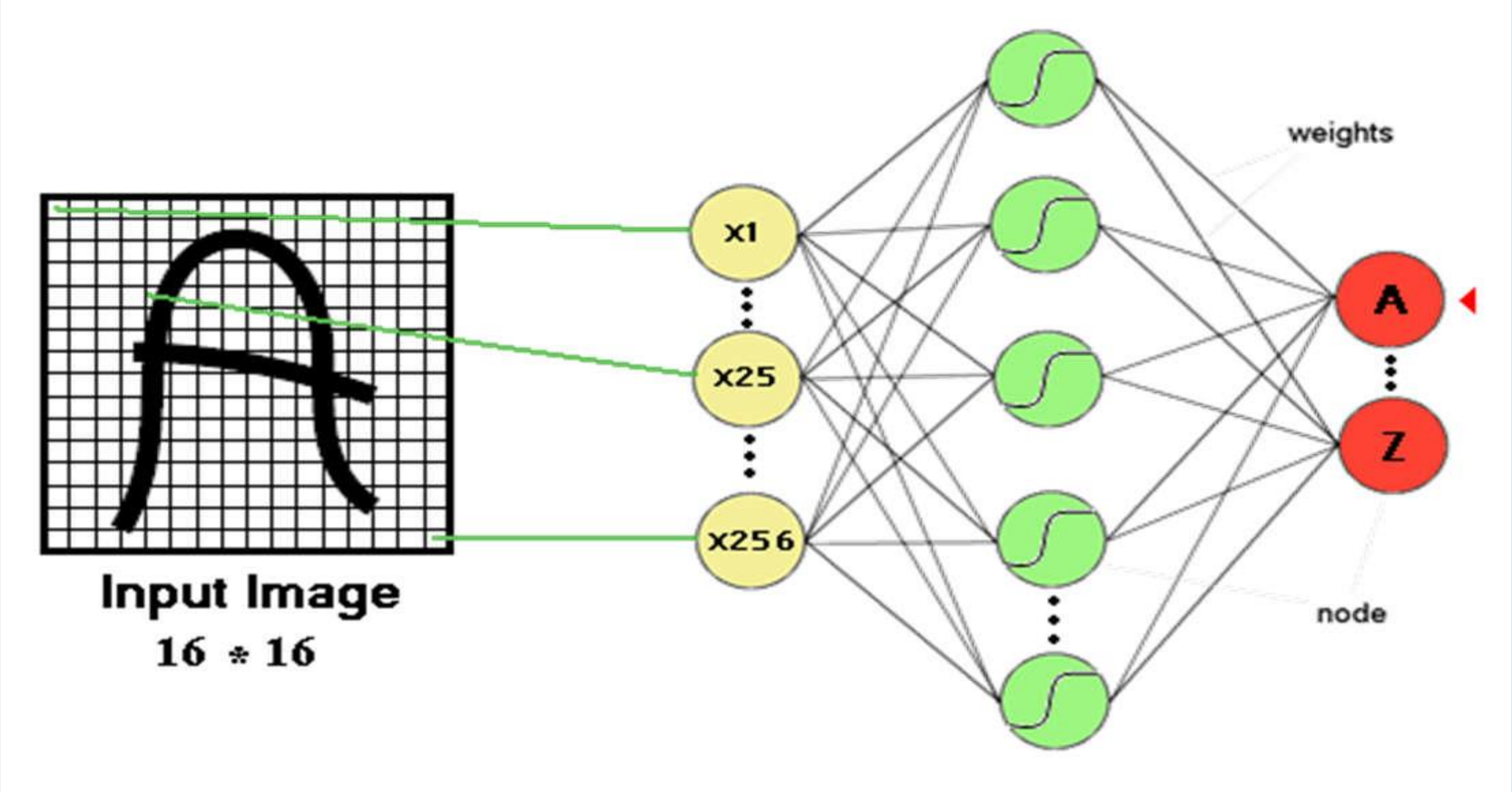
- Deep learning algorithms try to learn high-level features from data.



Convolutional Neural Network CNN

Multi-layer perceptron and image processing

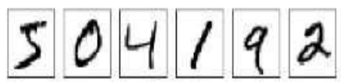
- One or more hidden layers
- Sigmoid activations functions



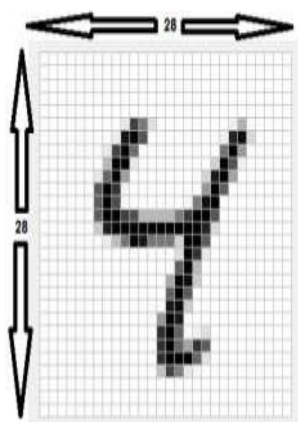
MNIST digits classification

MNIST Digits Classification

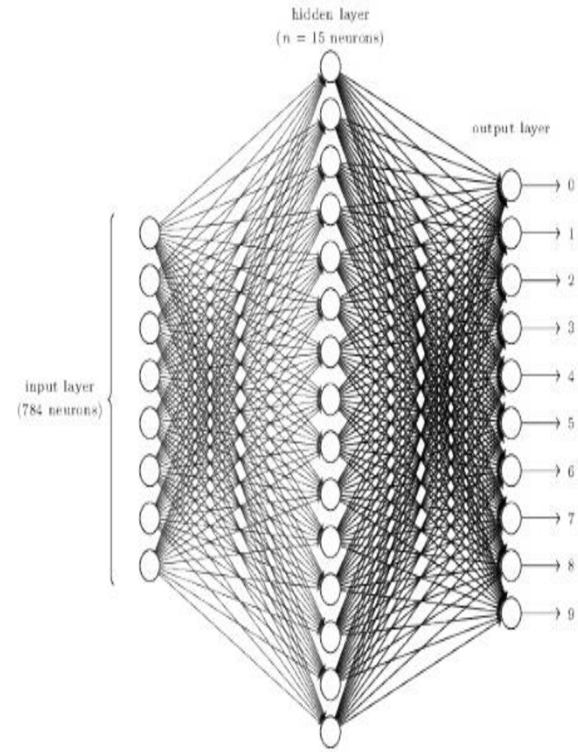
Segmented digits



MNIST digit format (28 x 28 = 784 pixels)



Source: Neural Networks and Deep Learning. Michael Nielsen.

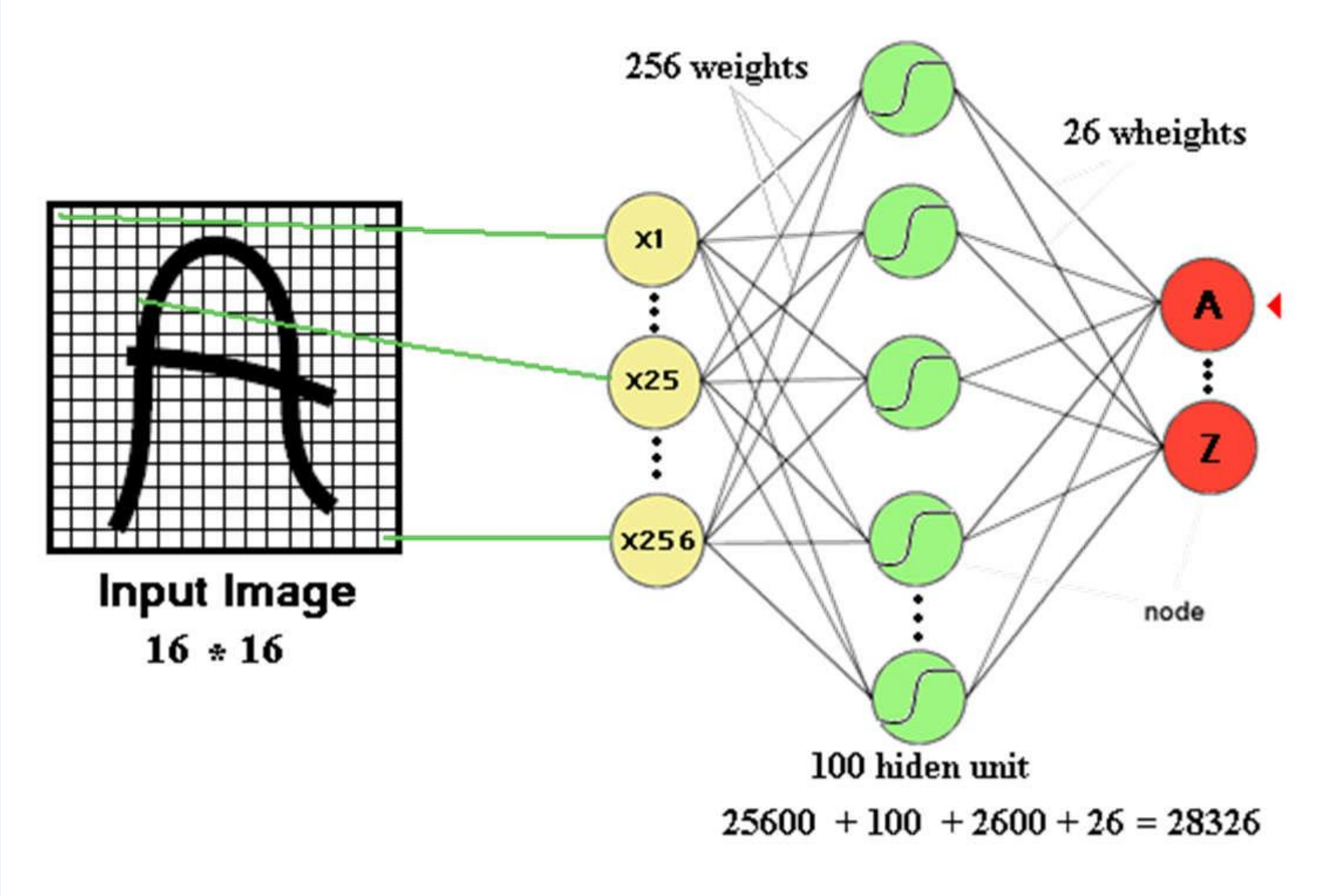


Source: Neural Networks and Deep Learning. Michael Nielsen.

- 2.225 of 10.000 test images (22.25 % accuracy)
- An SVM classifier can get 9.435 of 10.000 (% 94.35)

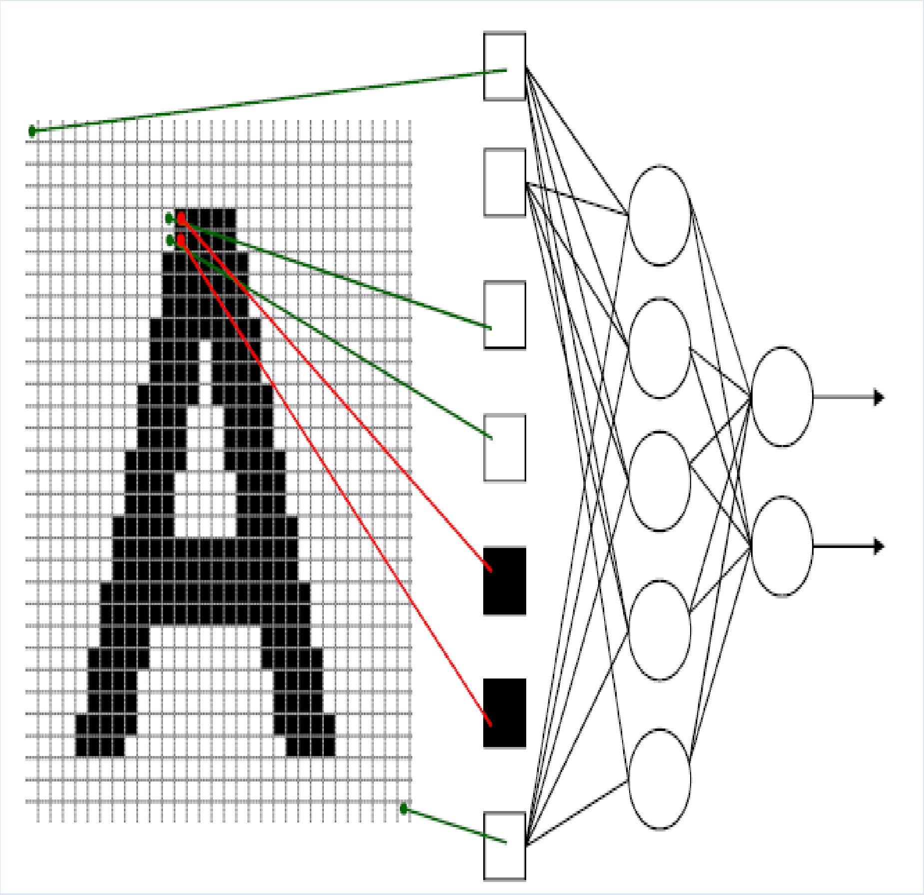
Drawbacks of previous neural networks

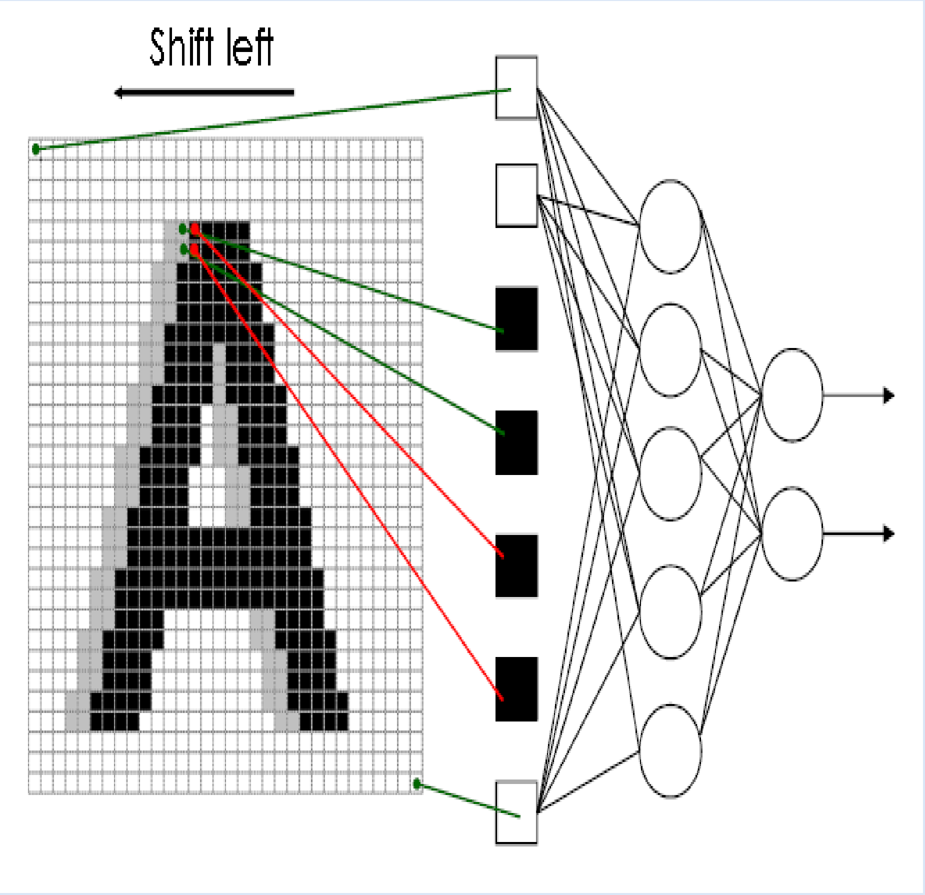
- The number of **trainable parameters** becomes extremely large



Drawbacks of previous neural networks

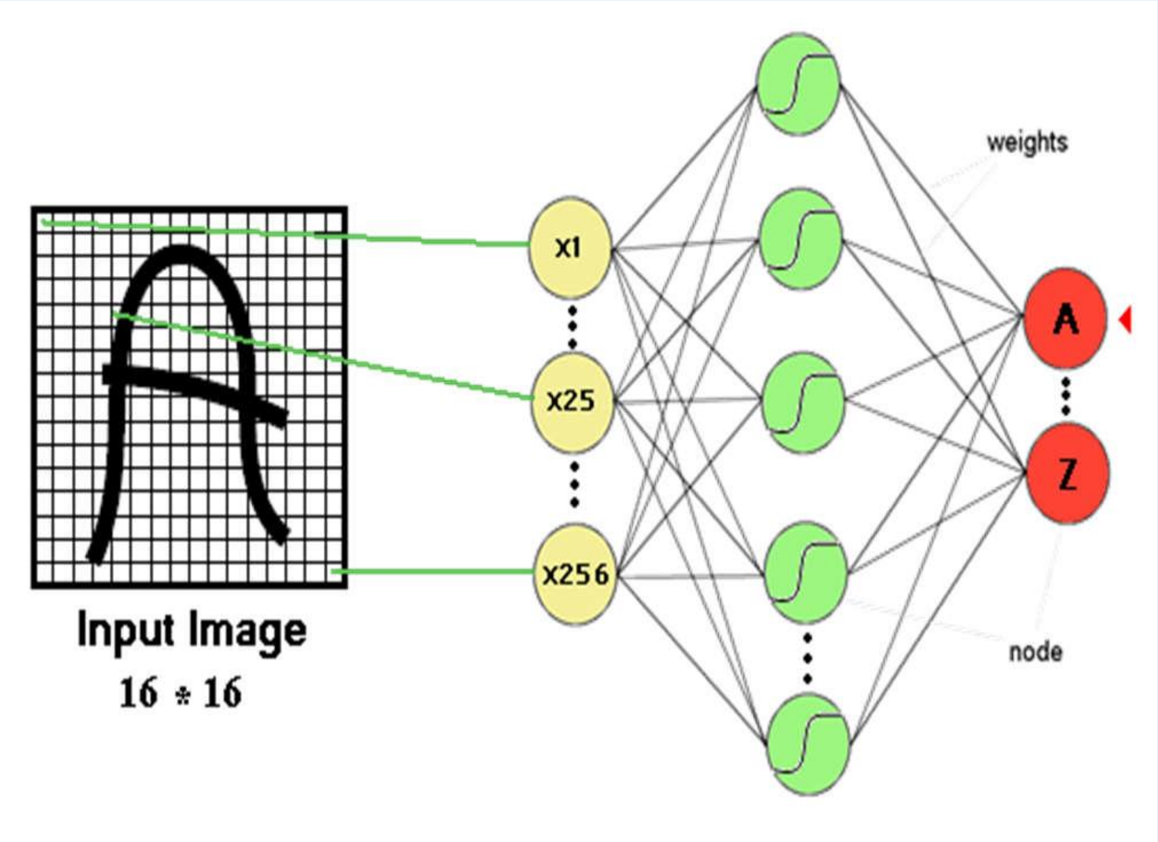
- Little or no invariance to **shifting, scaling, and other forms of distortion**





Drawbacks of previous neural networks

- The topology of the input data is completely ignored
- work with raw data.



Motivation

- **We can design neural networks that are specifically adapted for such problems**

1- Must deal with very high-dimensional inputs
   - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels

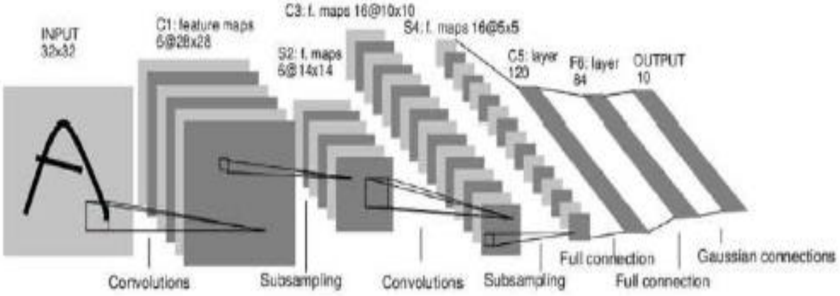
2- Can exploit the 2D topology of pixels (or 3D for video data)

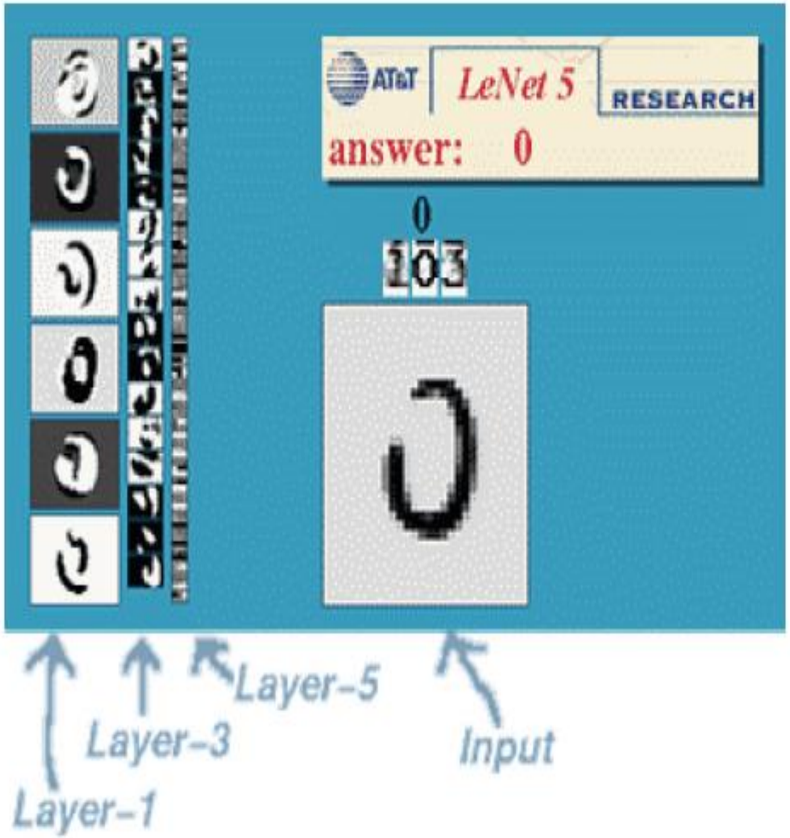
3- Can build in invariance to certain variations we can expect
   - Translations,Scalling, illumination, etc.

History

Yann LeCun

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.





Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998

Convolutional neural Network

- CNN is a feed-forward network that can extract topological properties from an image.

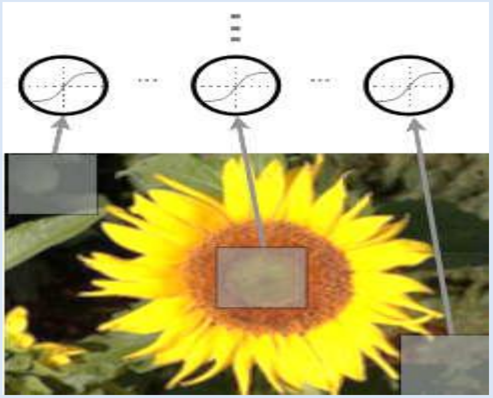
- Like almost every other neural networks they are trained with a version of the back-propagation algorithm.

- Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.

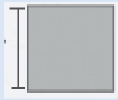
- They can recognize patterns with extreme variability (such as handwritten characters).

Convolutional neural Network

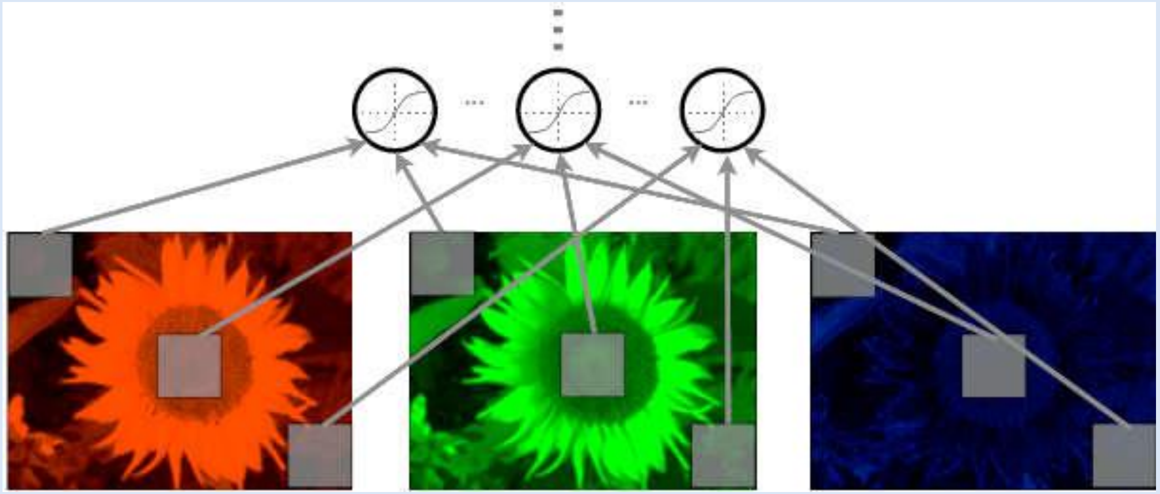
Convolutional networks leverage these ideas



1. local connectivity

 = *receptive field*

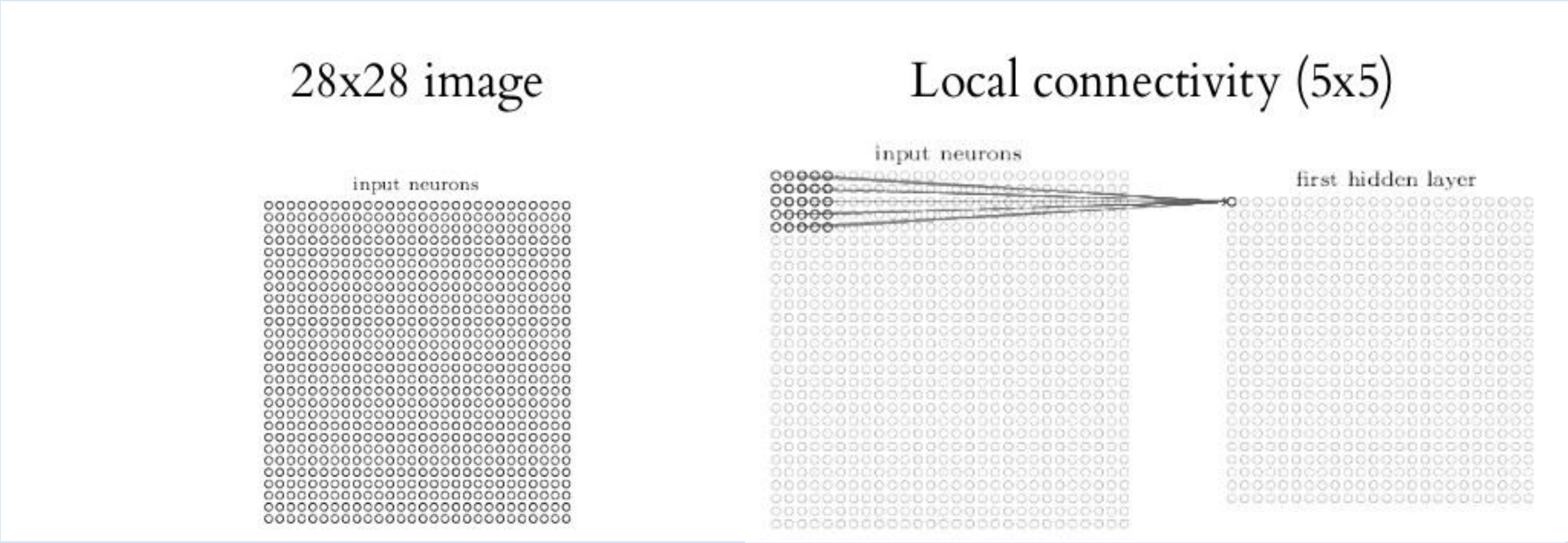
- Each hidden unit is connected only to a sub region (patch) of the input image
- It is connected to all channels
  - 1 if greyscale image
  - 3 (R, G, B) for color image



Convolutional neural Network

Convolutional networks leverage these ideas

2. parameter sharing



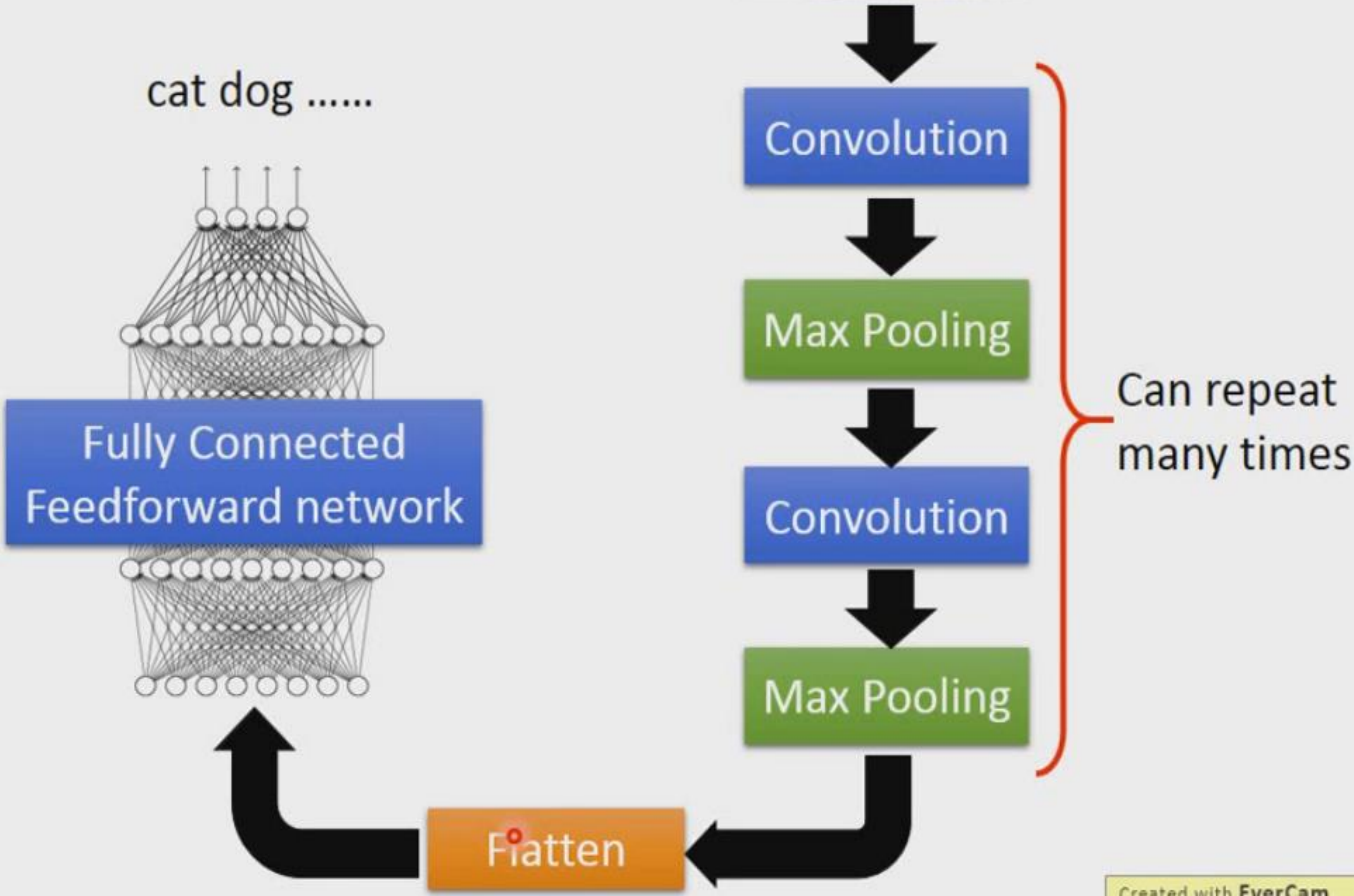
3. pooling / subsampling hidden units

Convolutional neural Network

➢ Convnets contain one or more of each of the following layers:

1. convolution layer
2. ReLU (rectified linear units) layer (element wise threshold)
3. pooling layer
4. fully connected layer
5. loss layer (during the training process)

The whole CNN



The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

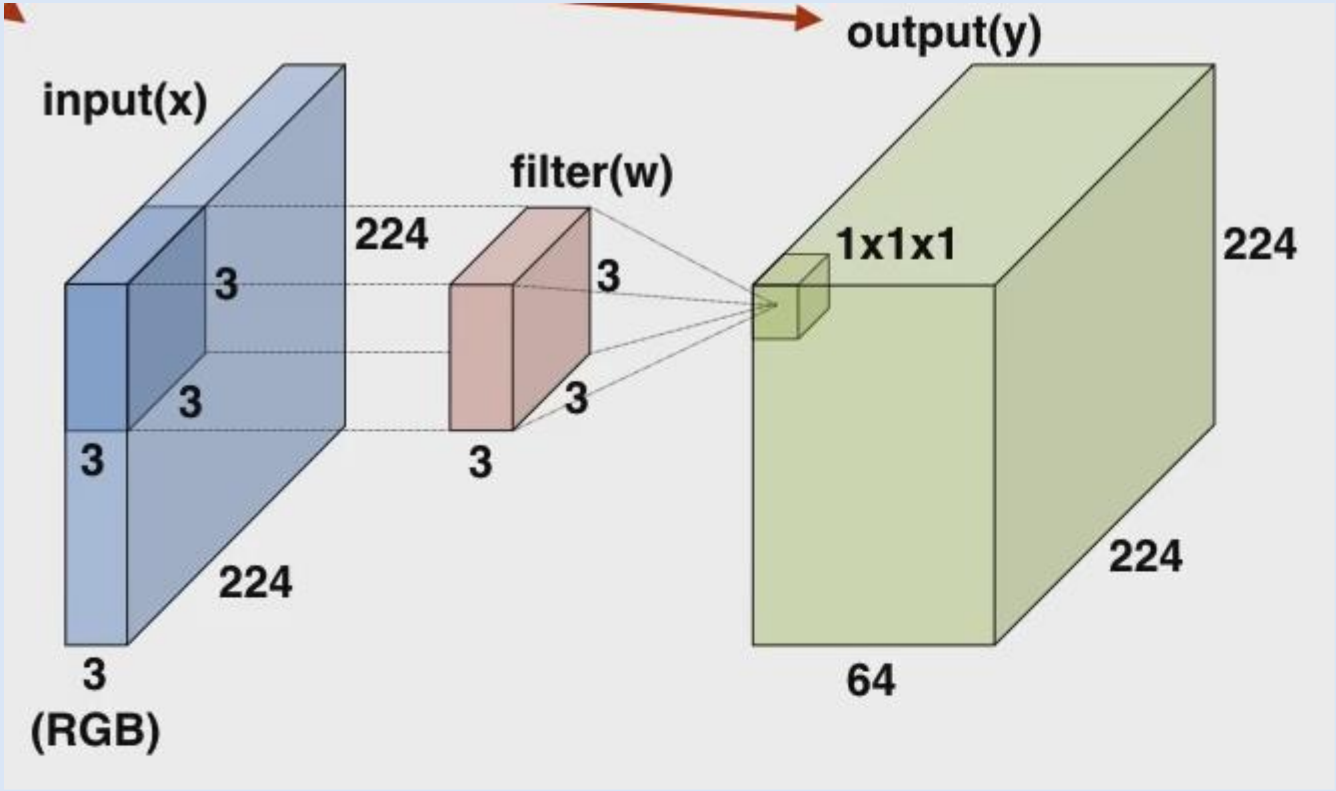
Max Pooling

Can repeat many times

Flatten

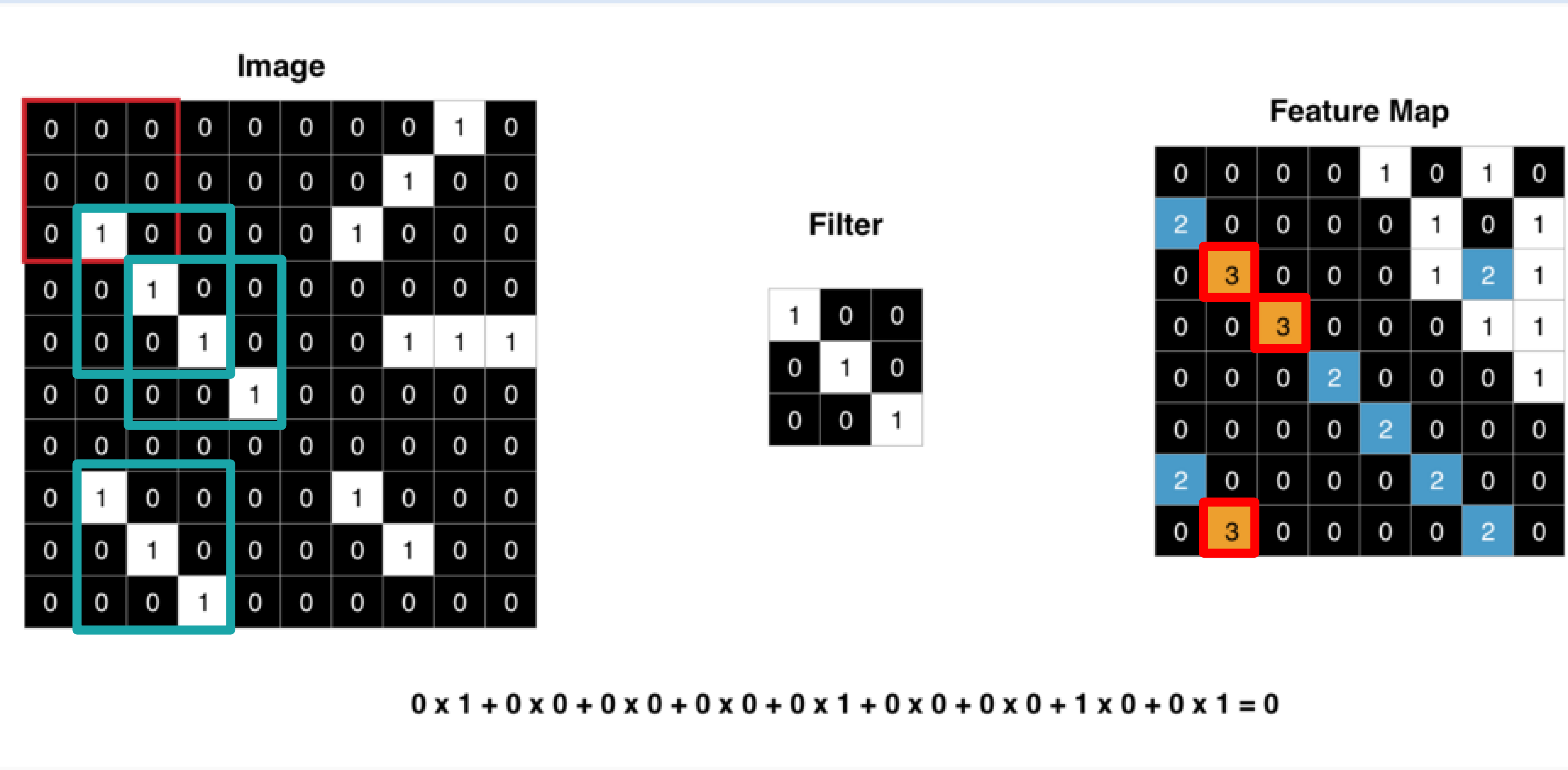
1- Convolution layer

a convnet processes an image using a matrix of weights called filters (or features) that detect specific attributes such as diagonal edges, vertical edges, etc. Moreover, as the image progresses through each layer, the filters are able to recognize more complex attributes.

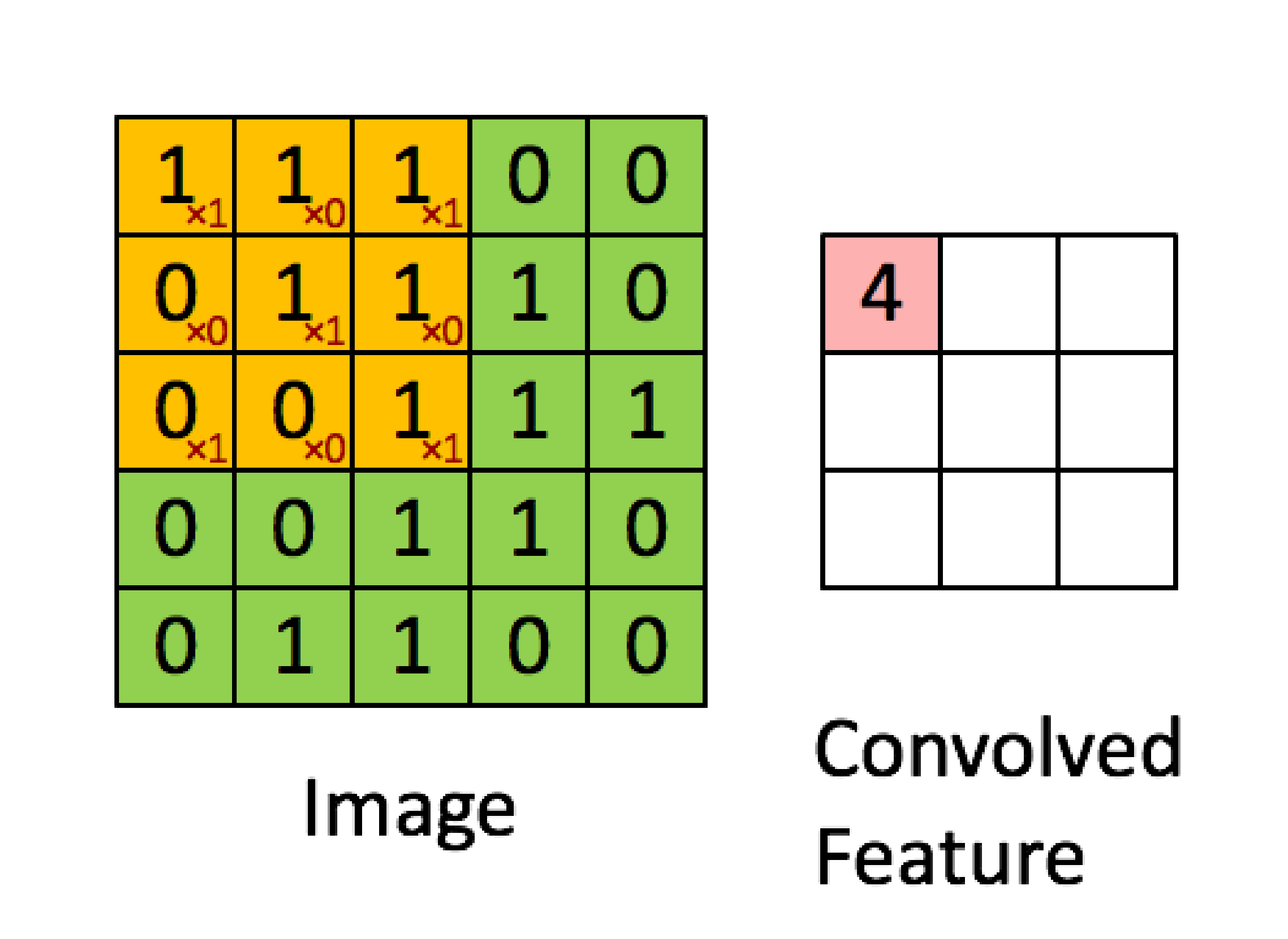


Convolution layer

The convolution layer is always the first step in a convnet. Let's say we have a 10 x 10 pixel image, here represented by a 10 x 10 x 1 matrix of numbers:



Convolution Example



Kernels as Feature Detectors

Can think of kernels as a "local feature detectors"

Vertical Line Detector

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

Horizontal Line Detector

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

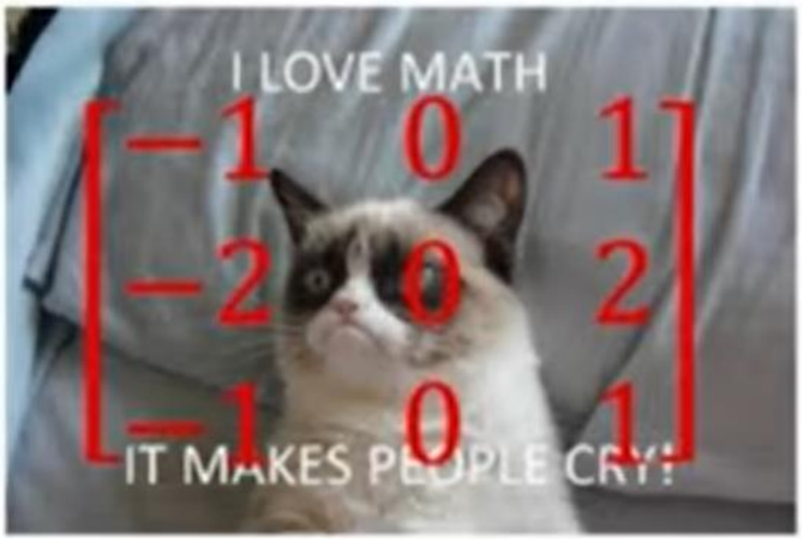
Corner Detector

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

- For example: Canny, Sobel, Gaussian blur, smoothing, low- level segmentation, morphological filters, Gabor filters,...

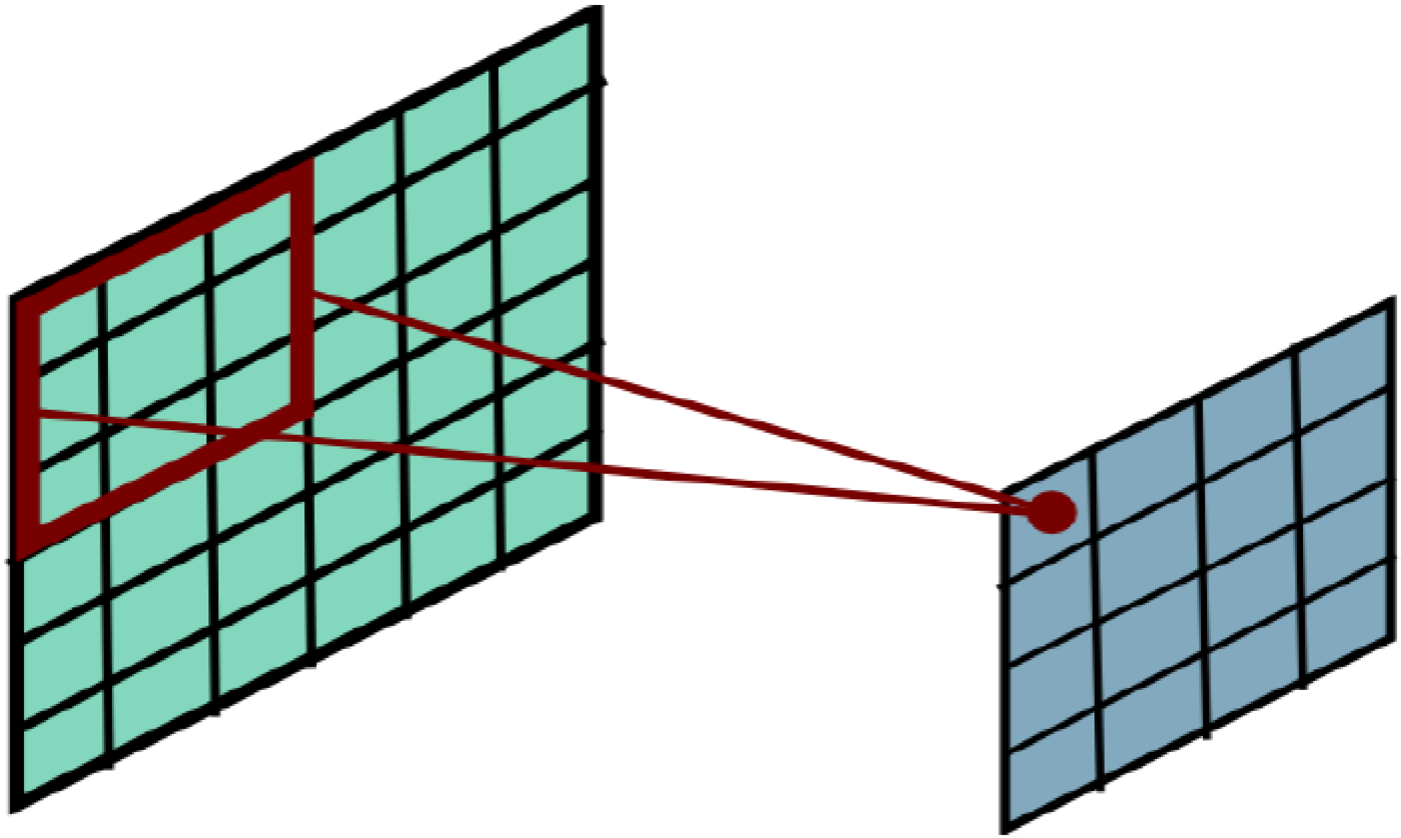
Example

e.g. Sobel 2-D filter

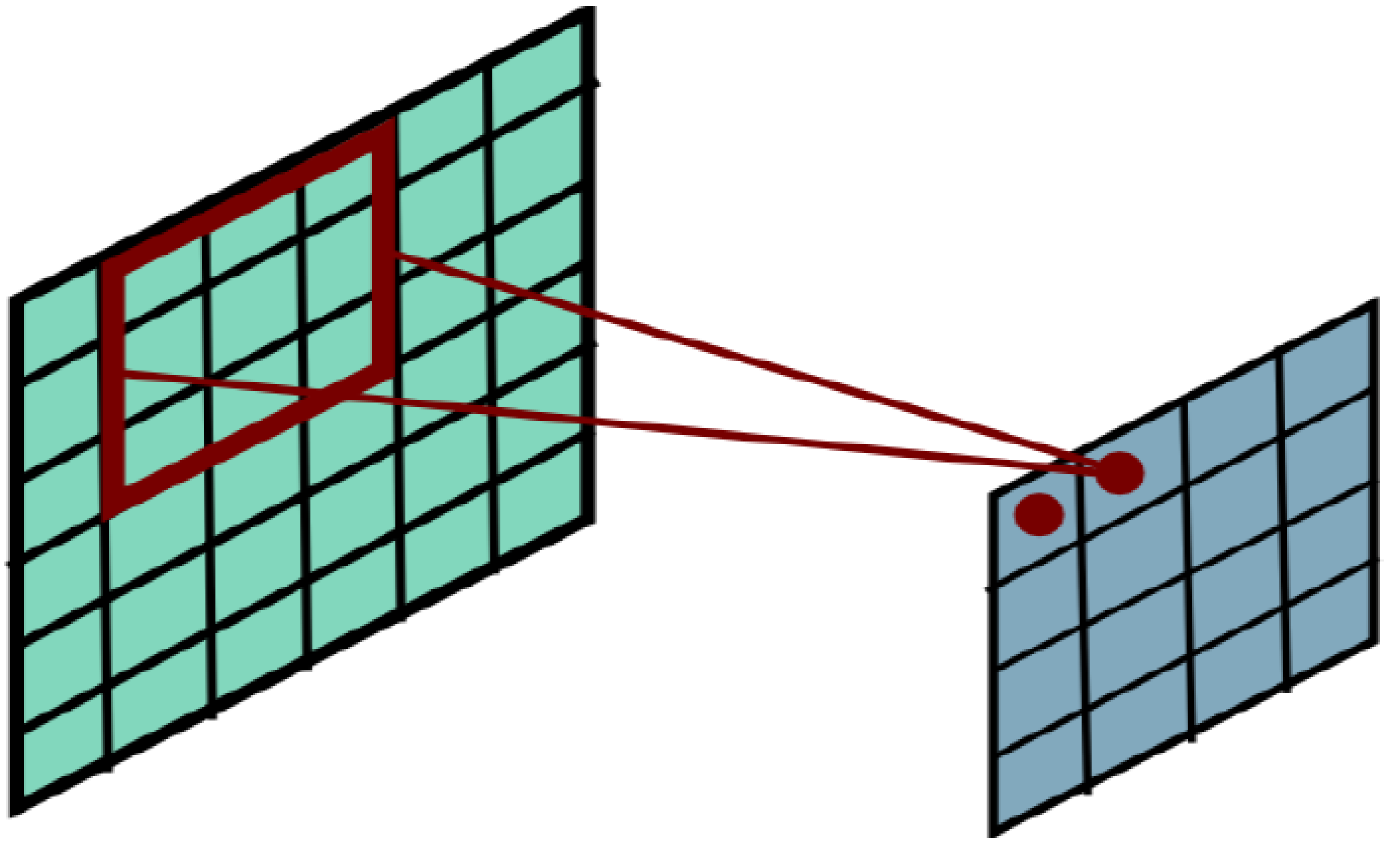


Detect contour

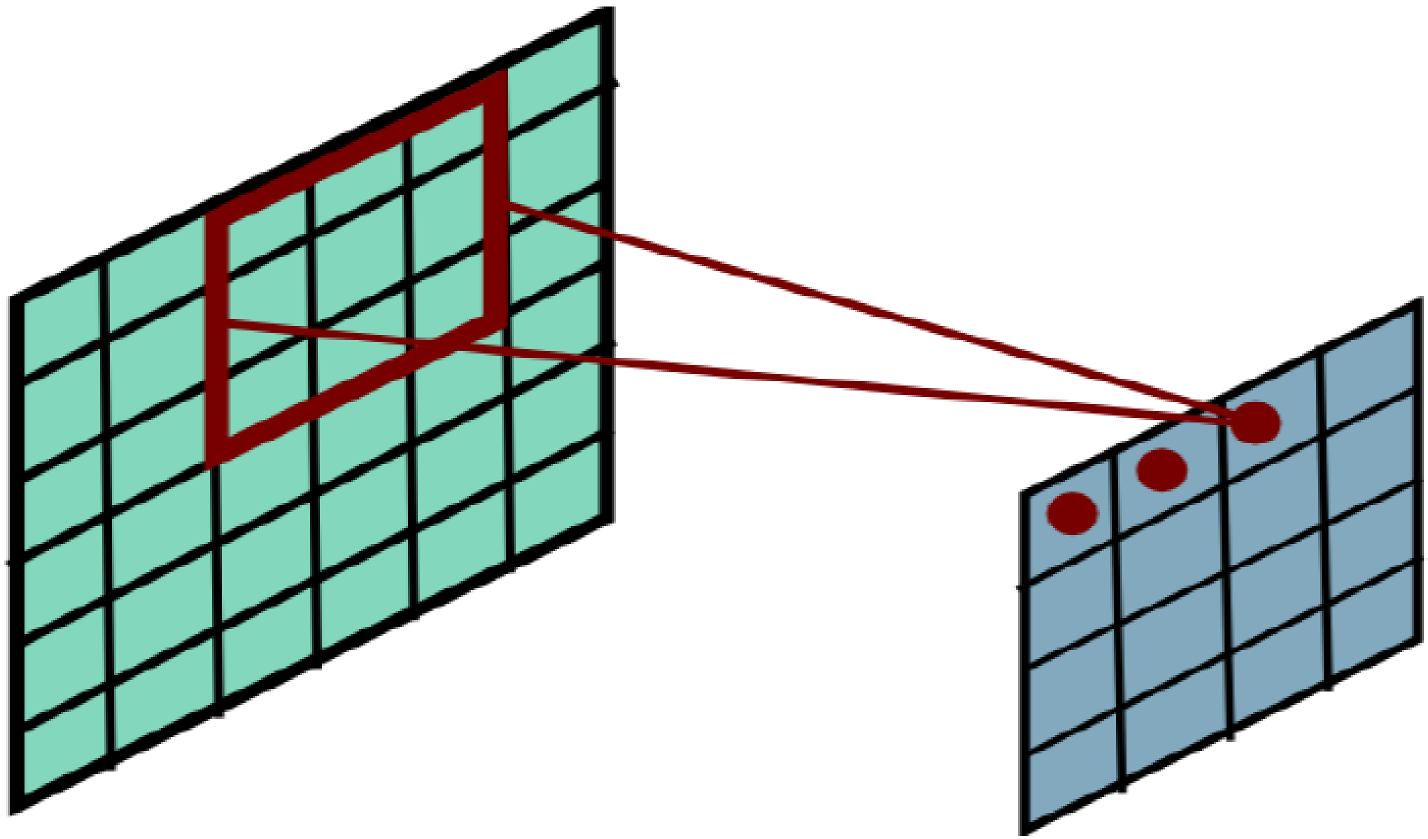
Convolutional Layer



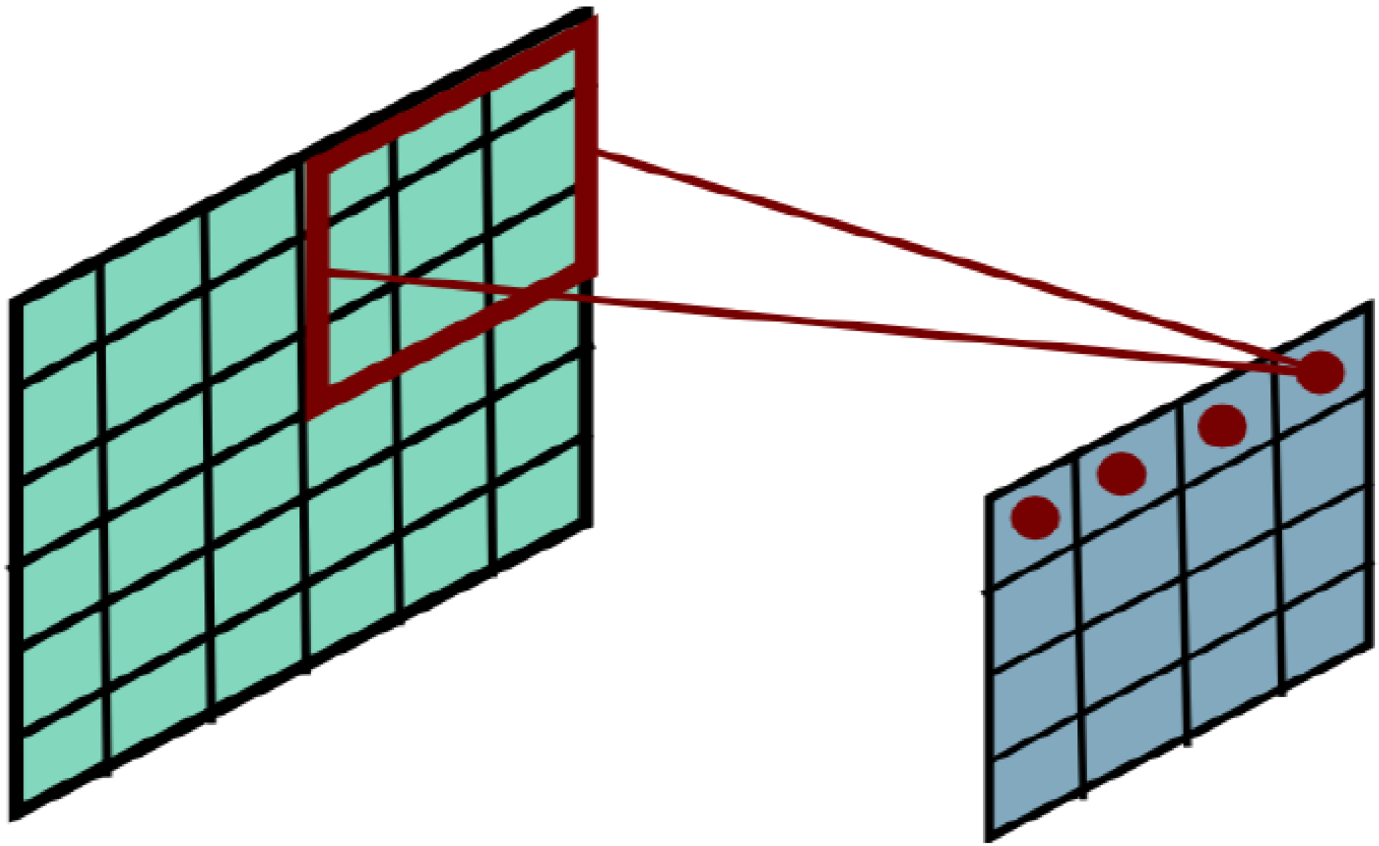
Convolutional Layer



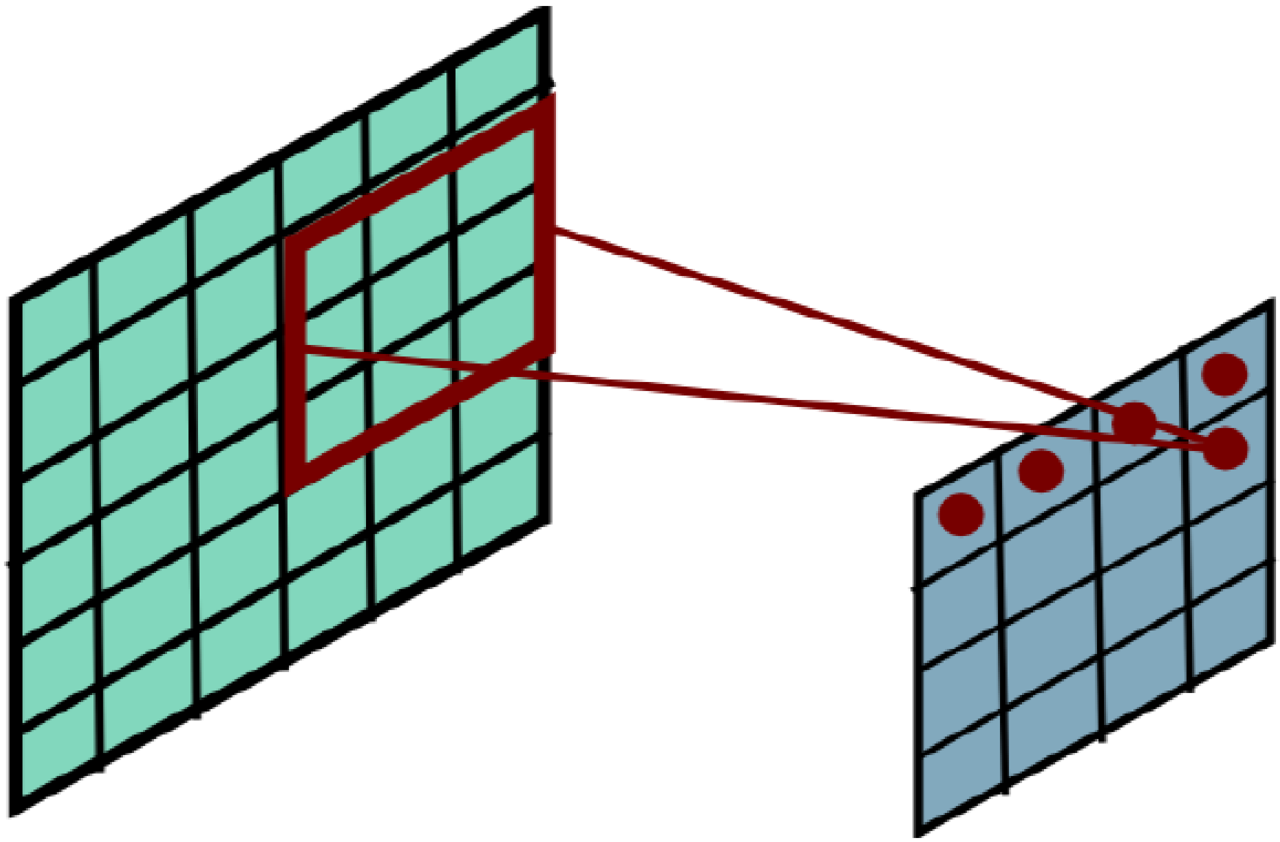
Convolutional Layer



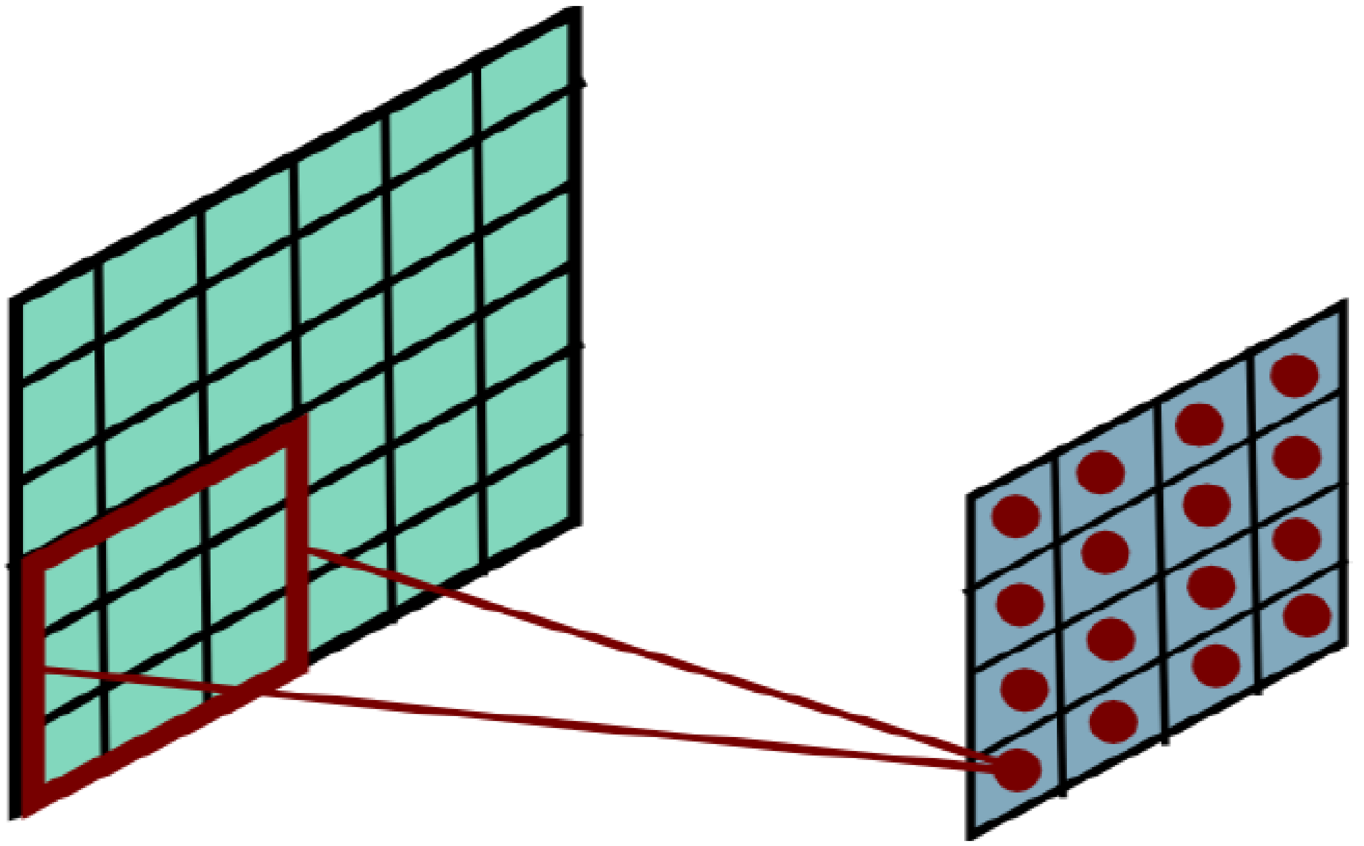
Convolutional Layer



Convolutional Layer

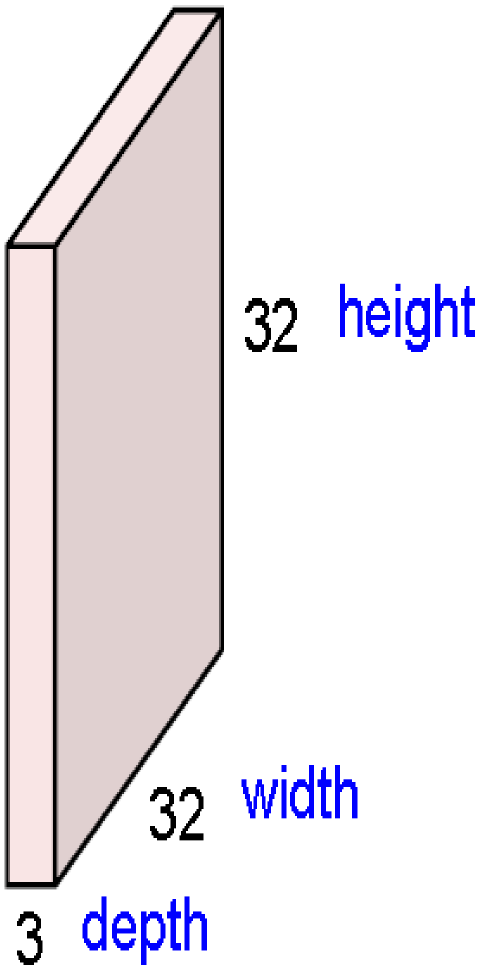


Convolutional Layer

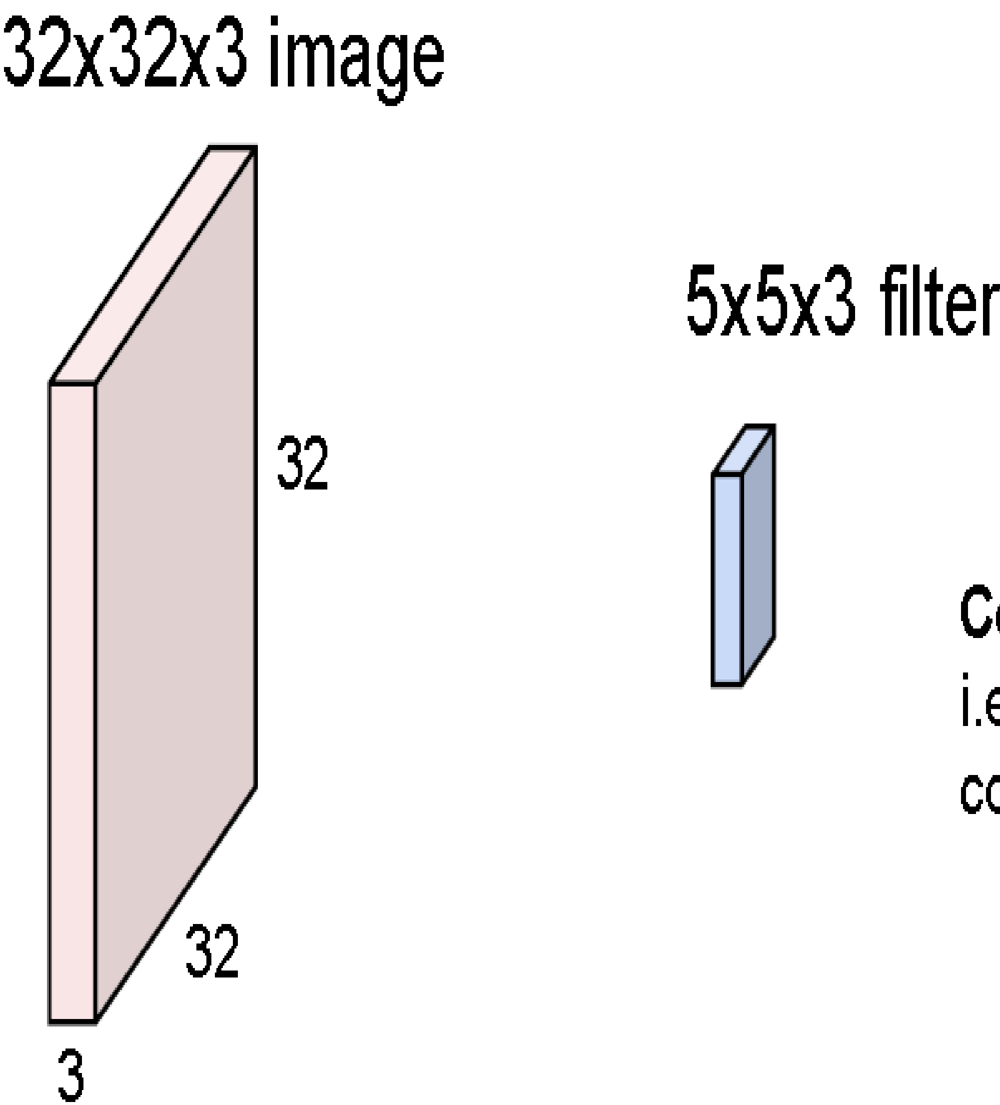


Convolution Layer

32x32x3 image



Convolution Layer

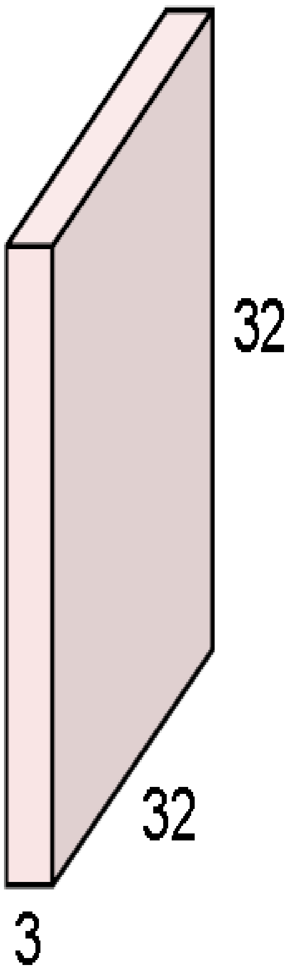


Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

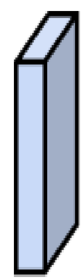
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

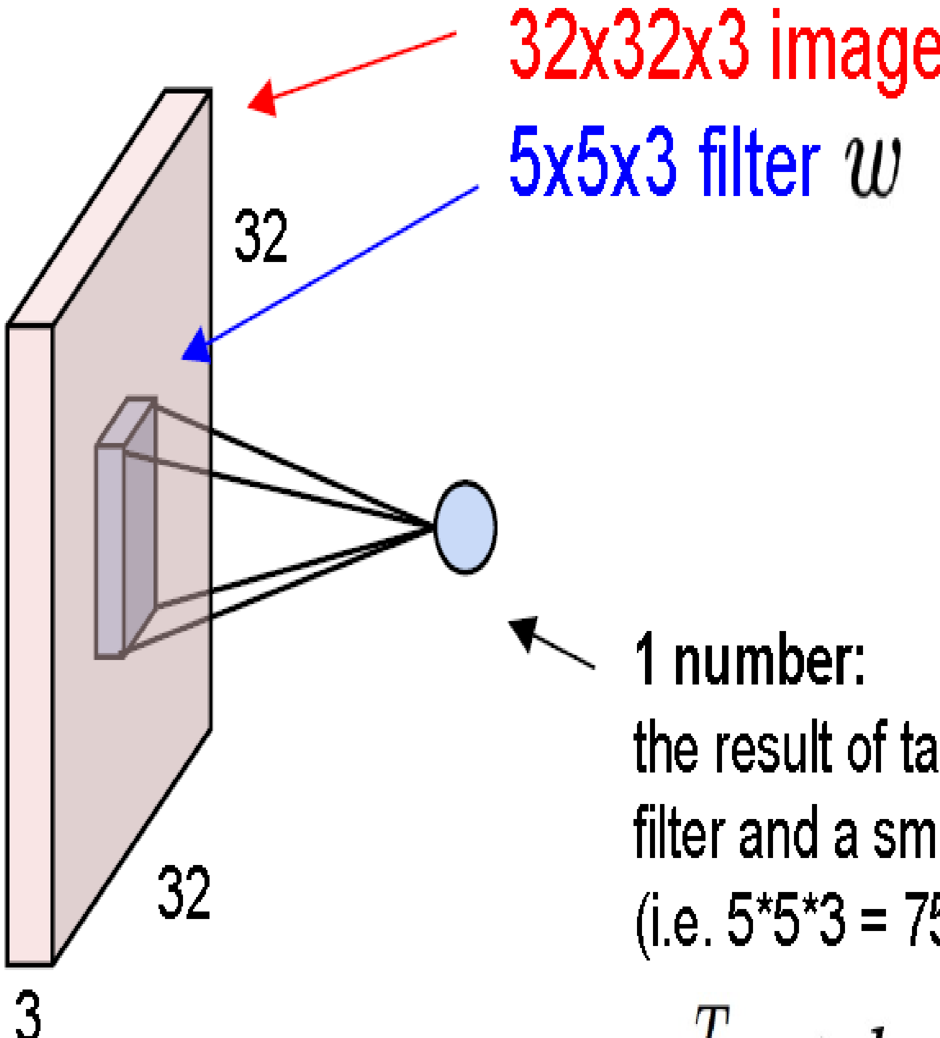


5x5x3 filter



Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

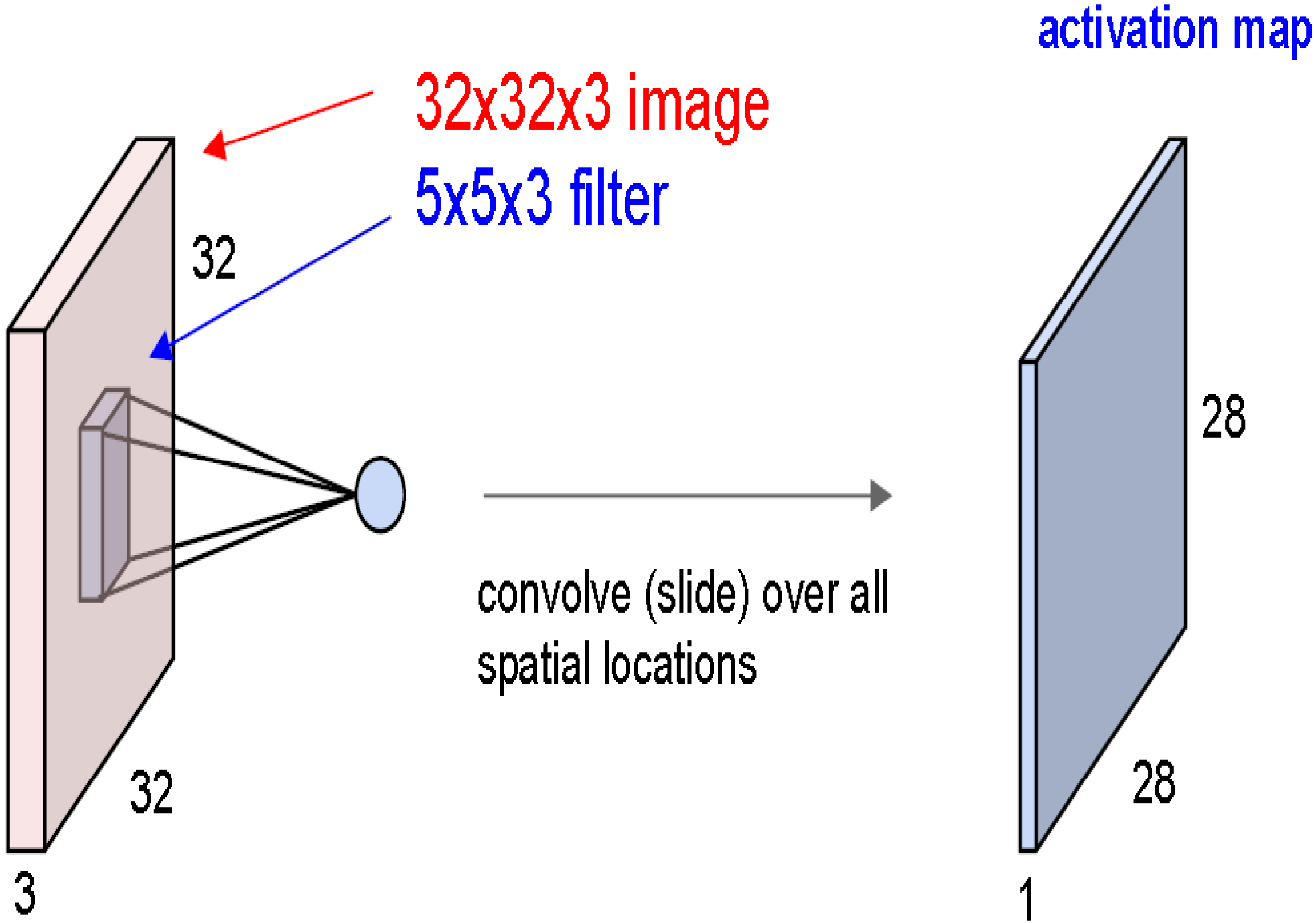
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

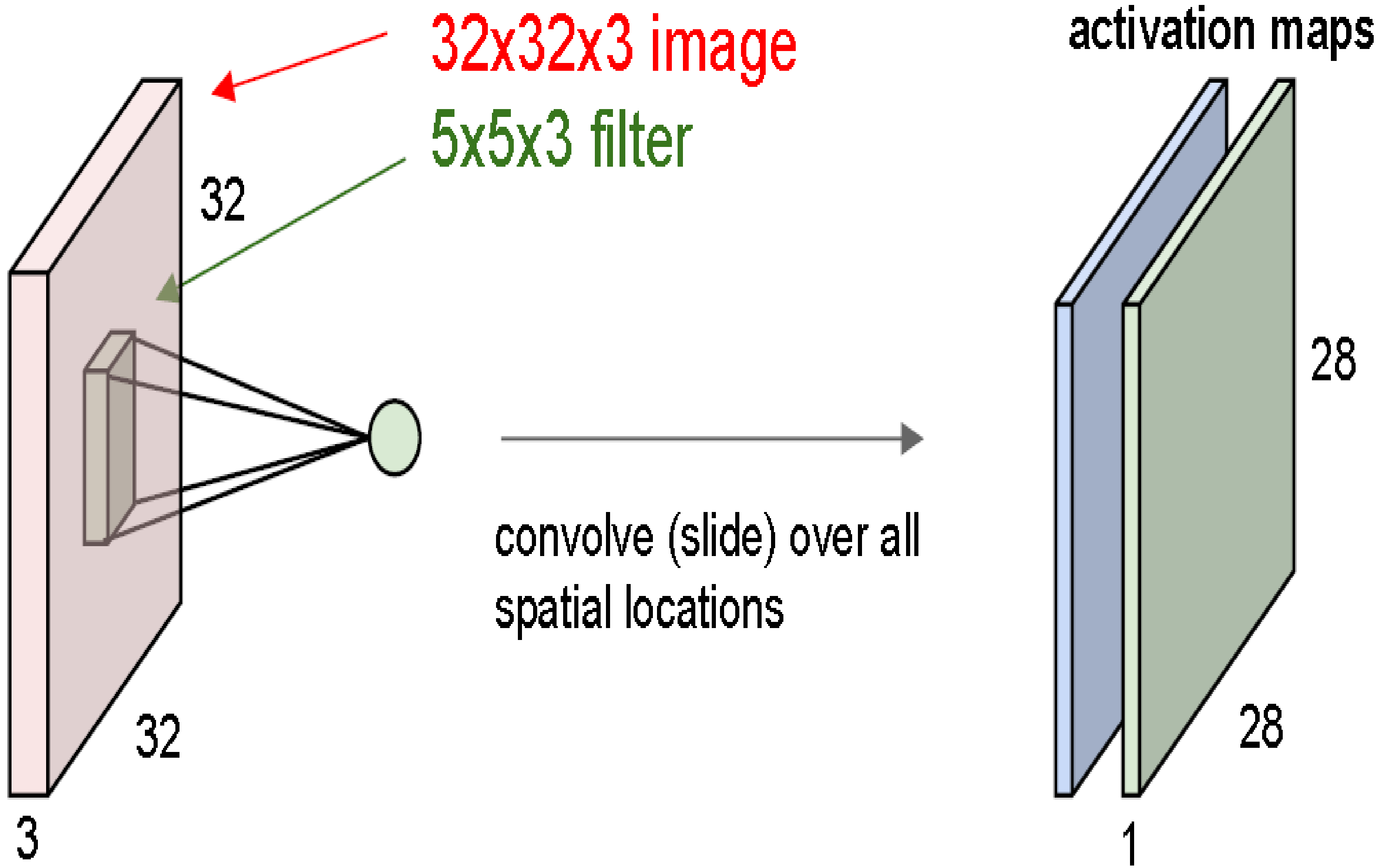
$$w^T x + b$$

Convolution Layer

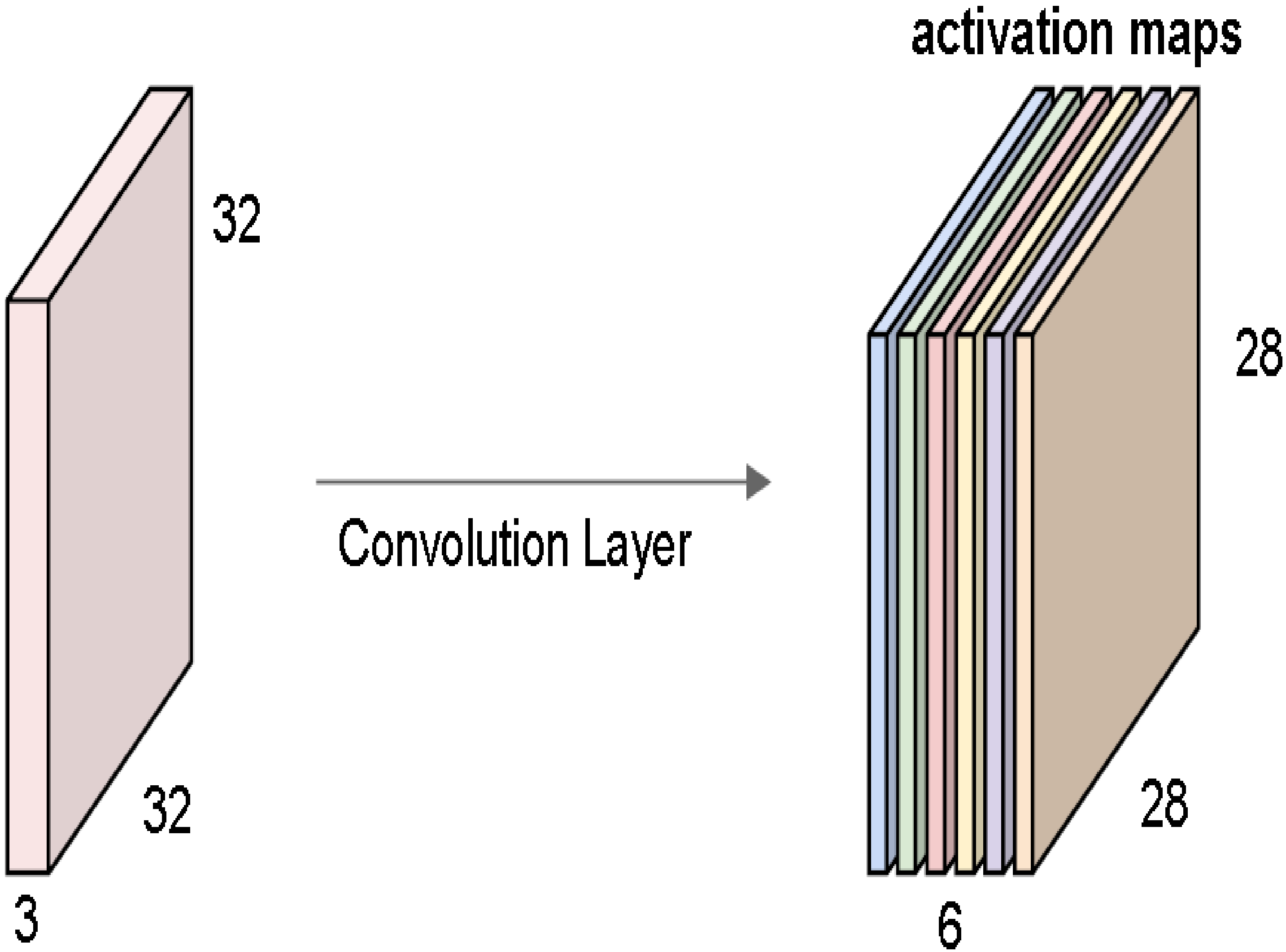


Convolution Layer

consider a second, green filter

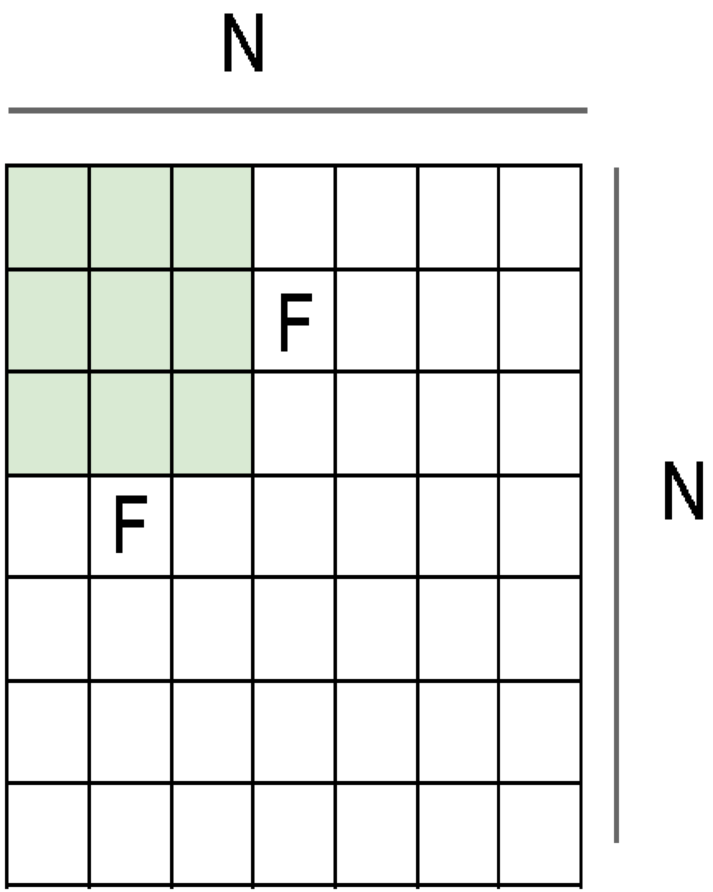


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

stride



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

2- ReLU Layer $f(x) = max(0,y)$

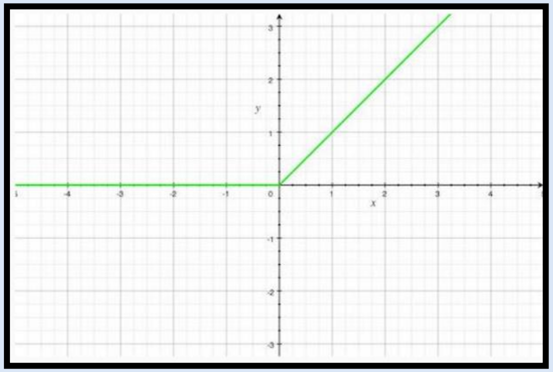
- The ReLU (short for rectified linear units) layer commonly follows the convolution layer.

- The addition of the ReLU layer allows the neural network to account for non-linear relationships, i.e. the ReLU layer allows the convnet to account for situations in which the relationship between the pixel value inputs and the convnet output is not linear.

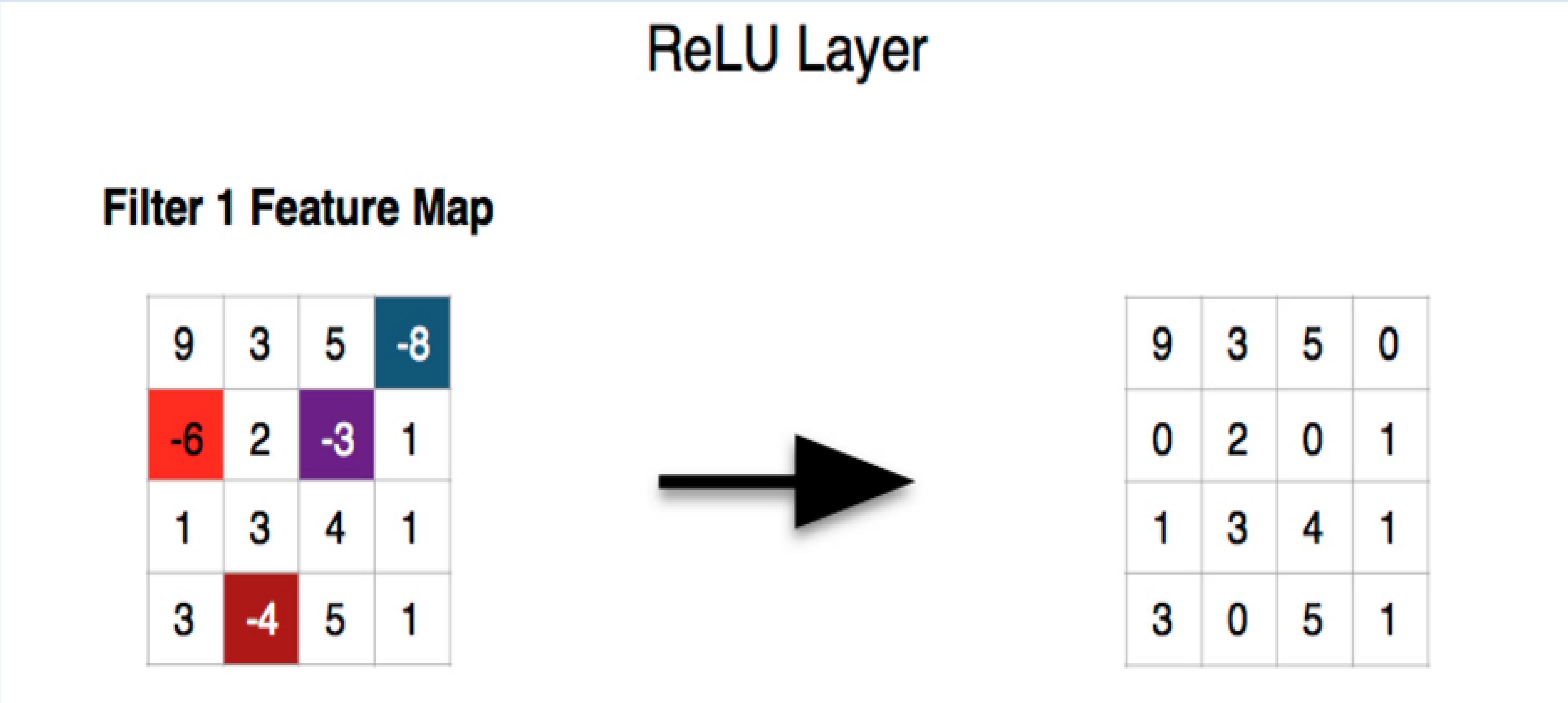
- the convolution operation is a linear one. $\mathbf{y = w_1x_1 + w_2x_2 + w_3x_3 + ...}$

- The ReLU function takes a value **y** and returns 0 If y is negative and **y** if **y** is positive.

Rectified linear (ReLU) : max(0,y)
- Simplifies backprop
- Makes learning faster
- Make feature sparse



2- ReLU Layer $f(x) = \max(0,x)$

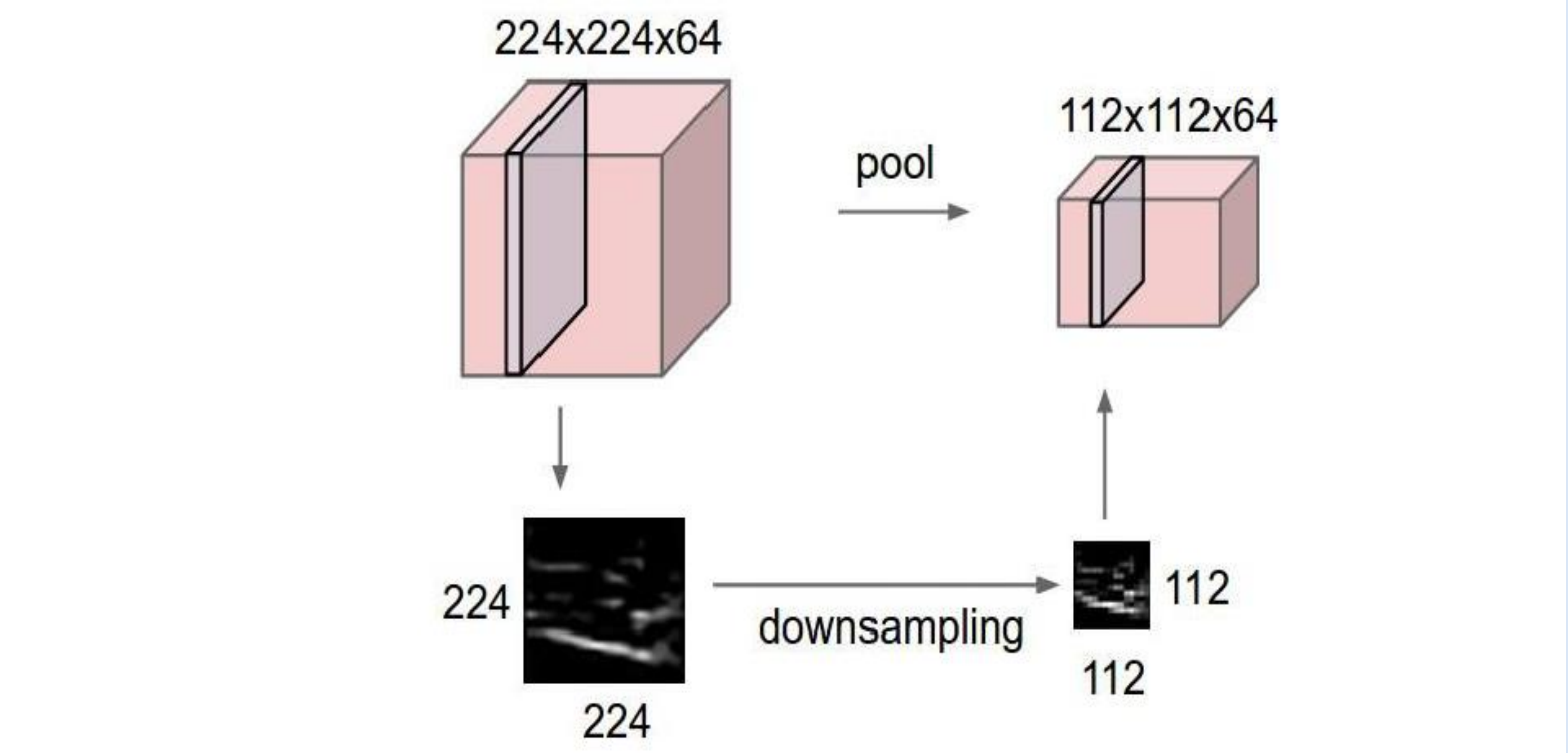


Other functions such as tanh or the sigmoid function can be used to add non-linearity to the network, but ReLU generally works better in practice.

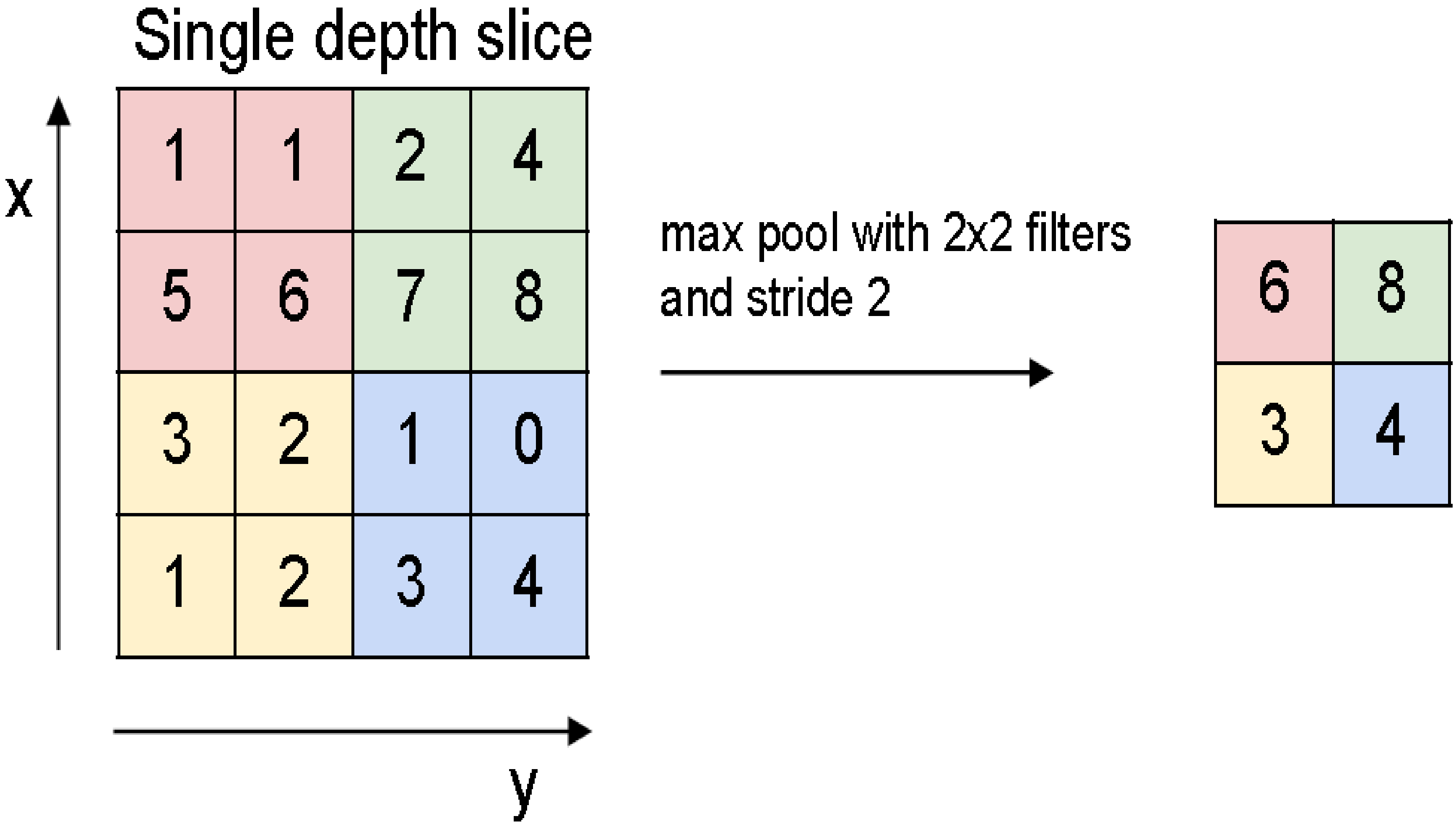
3- Pooling layer

• the pooling layer makes the convnet less sensitive to small changes in the location of a feature

• Pooling also reduces the size of the feature map, thus simplifying computation in later layers.



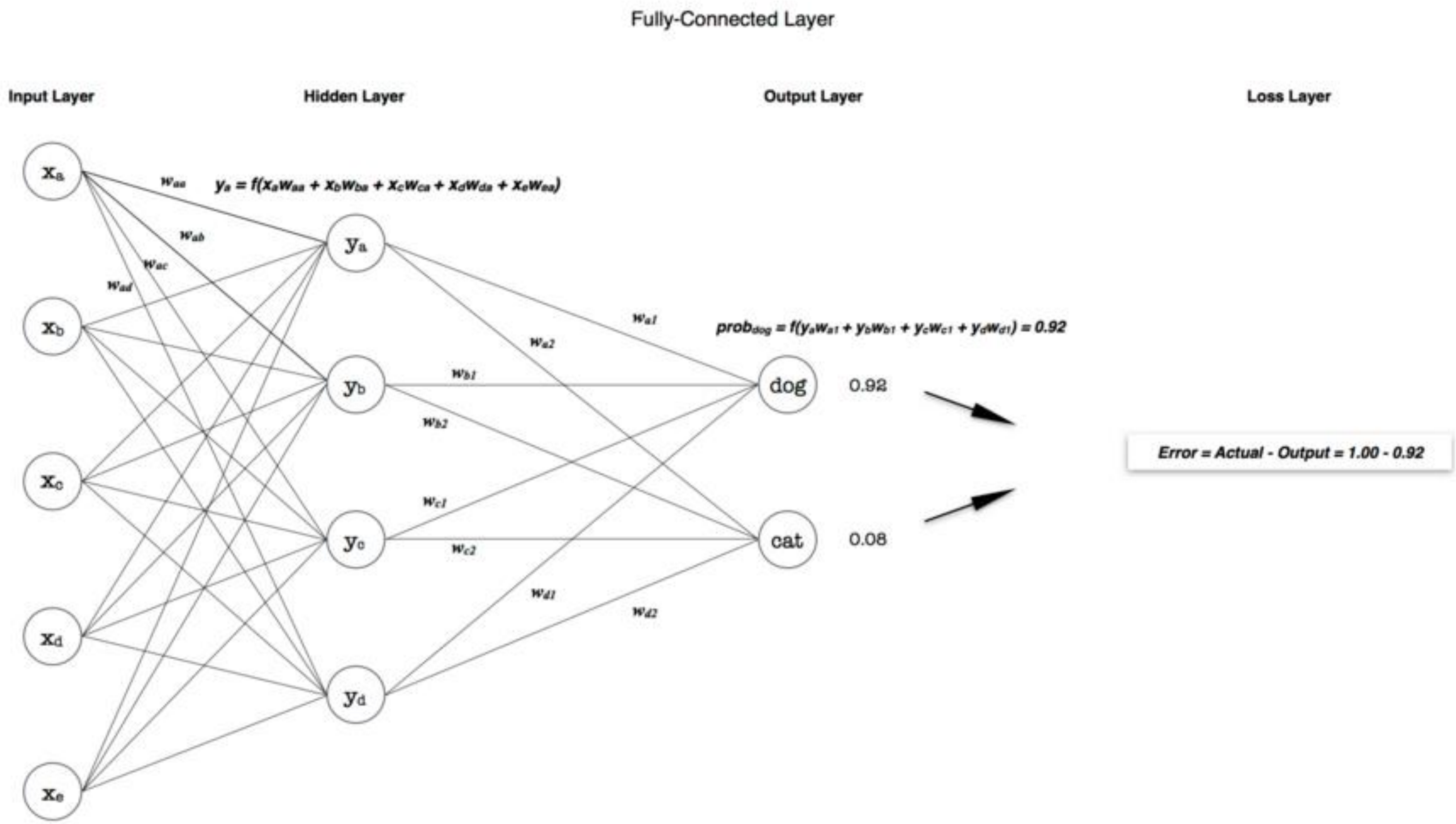
MAX POOLING



Single depth slice
x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
max pool with 2x2 filters
and stride 2
6 8
3 4

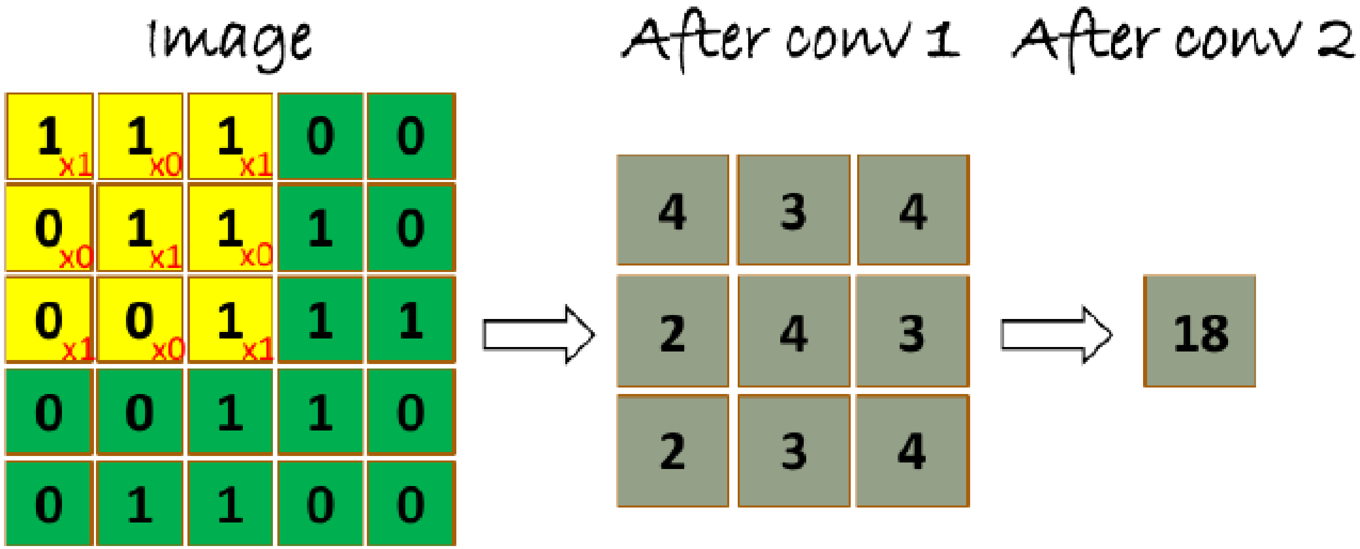
4- fully connected NN + loss layers

The fully-connected layer is where the final "decision" is made.



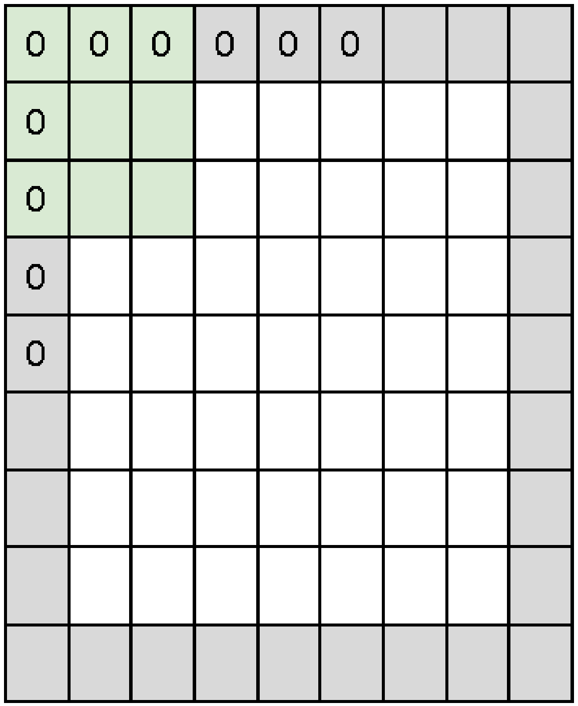
Problem, Image will vanish if applying more filters in cascade

- Our images get smaller and smaller
- Not too deep architectures
- Details are lost



Original 5x5
After apply one filter of size 3x3 , output will be3x3
After applying second filter on the output (cascaded filters)
Output will be 1x1

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

The formula for calculating the output size for any given conv layer is

$$O = \frac{(W - K + 2P)}{S} + 1$$

where O is the output height/length, W is the input height/length, K is the filter size, P is the padding, and S is the stride.

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Common settings:

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

CNN

what do they learn?