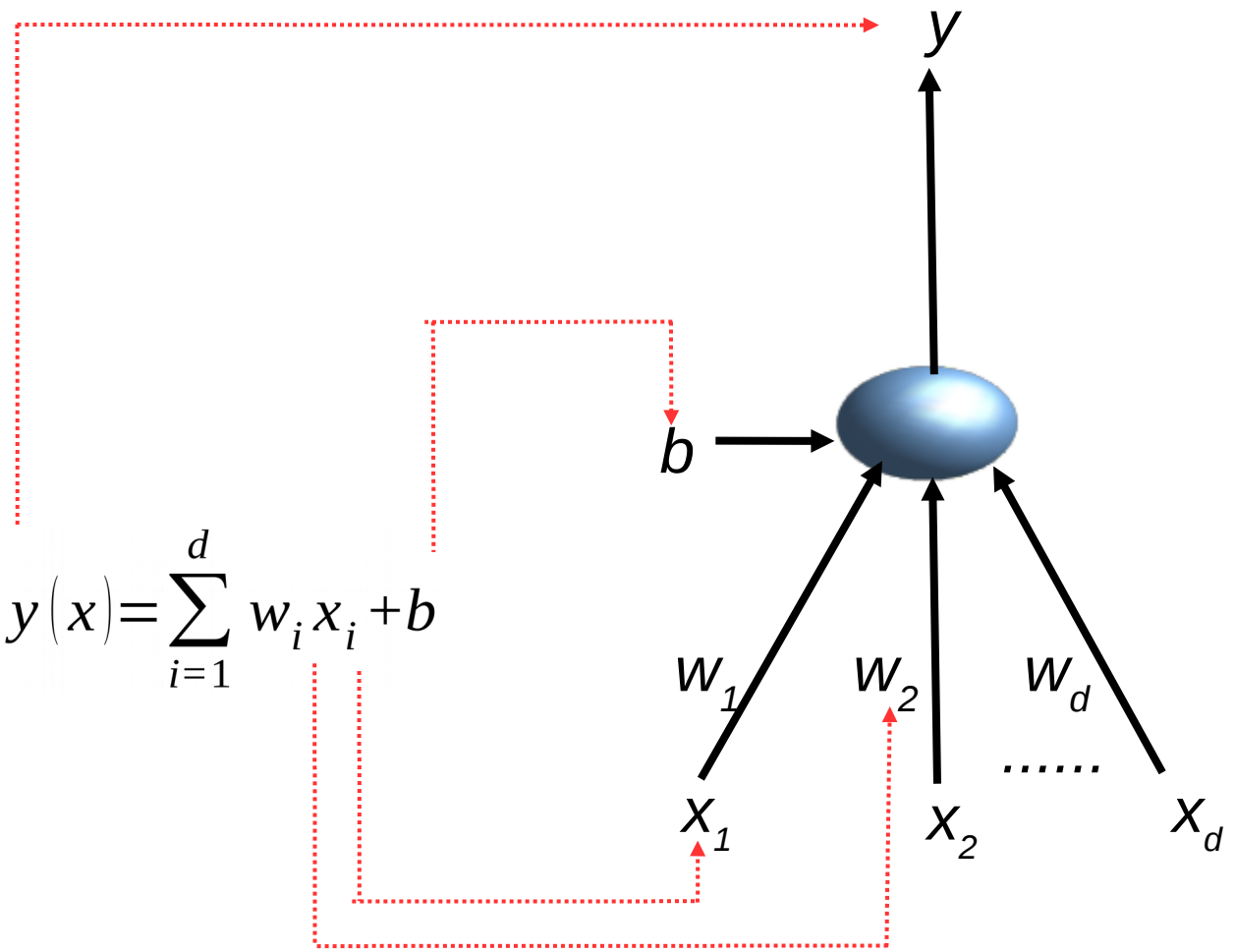
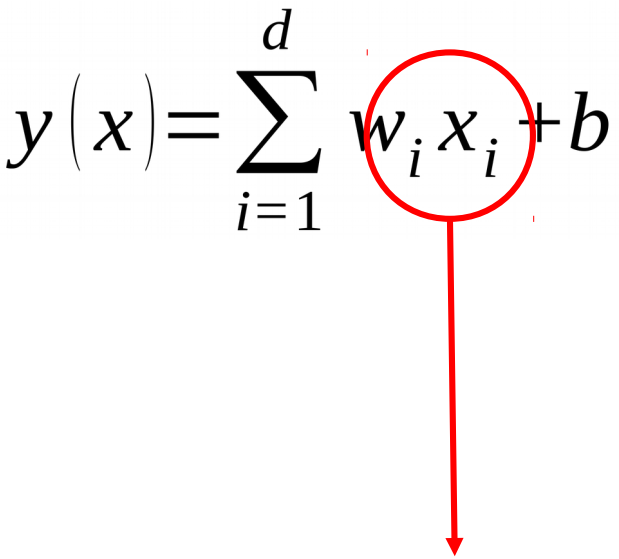
Neural Network



Neural Network



Dot product of two very large matrices

```
layerInput = self.weights[i].dot(input.T[i])
```

Convolutional Neural Network

$$y(x) = \sum_{i=1}^{d} w_i x_i + b$$

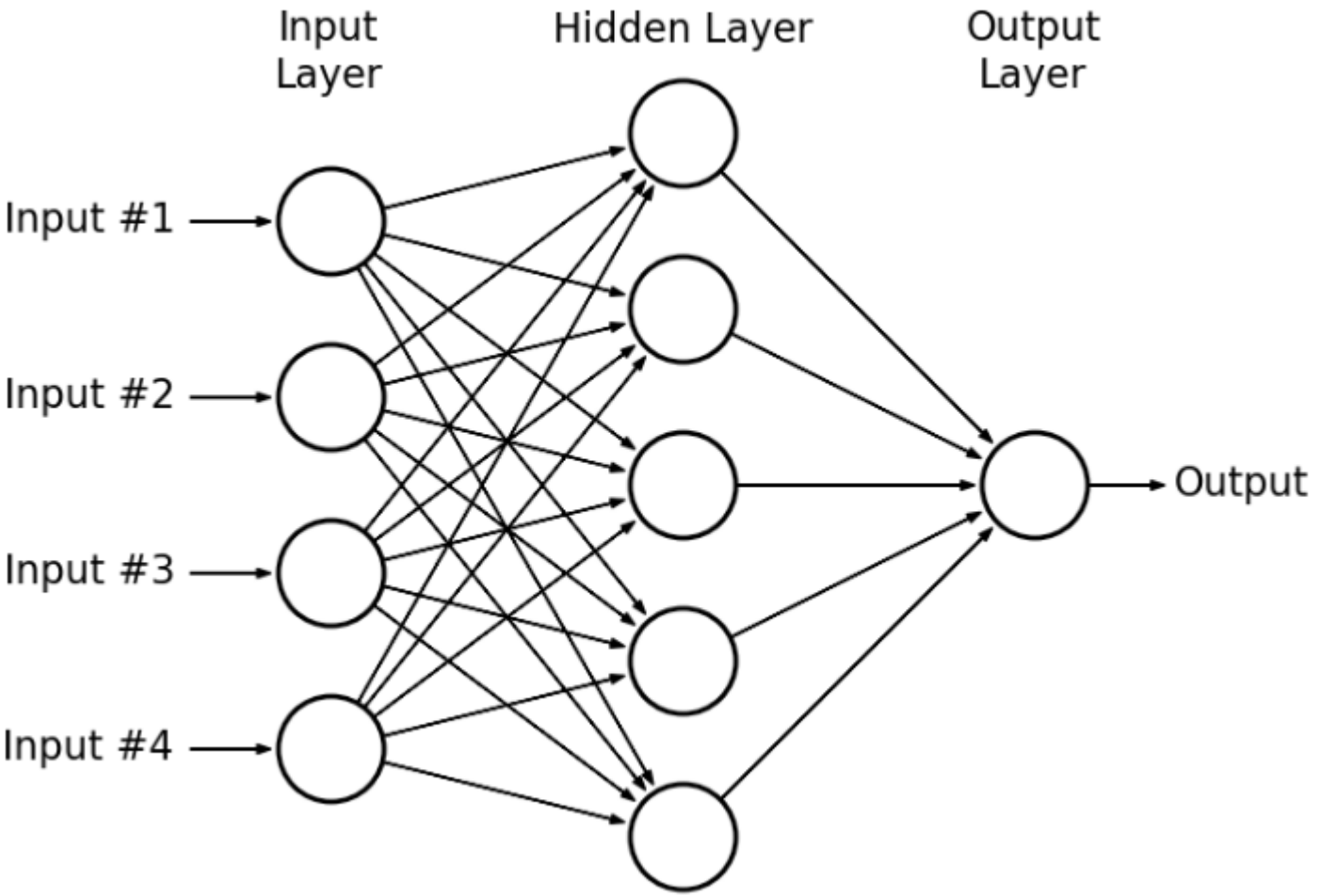
Convolution of two very large matrices

```
y = scipy.signal.convolve(input[i,:],w[i,:])
```

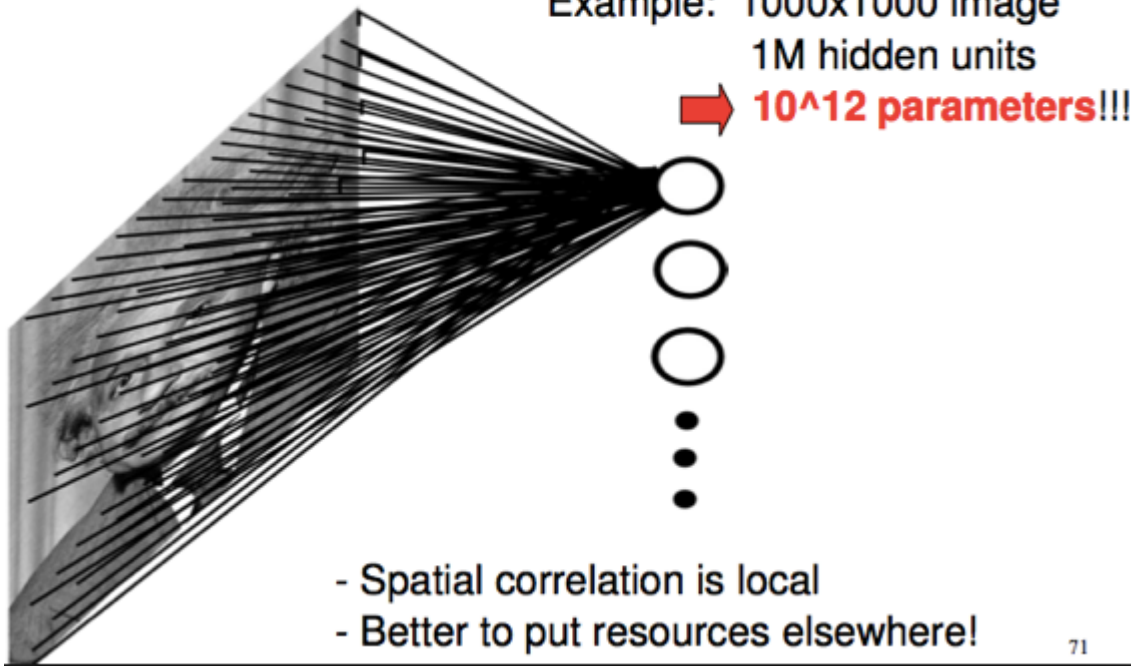
OR

```
y = scipy.signal.convolve2d(input[:,:,i],w[:,:,i,j])
```

Classical Neural Network



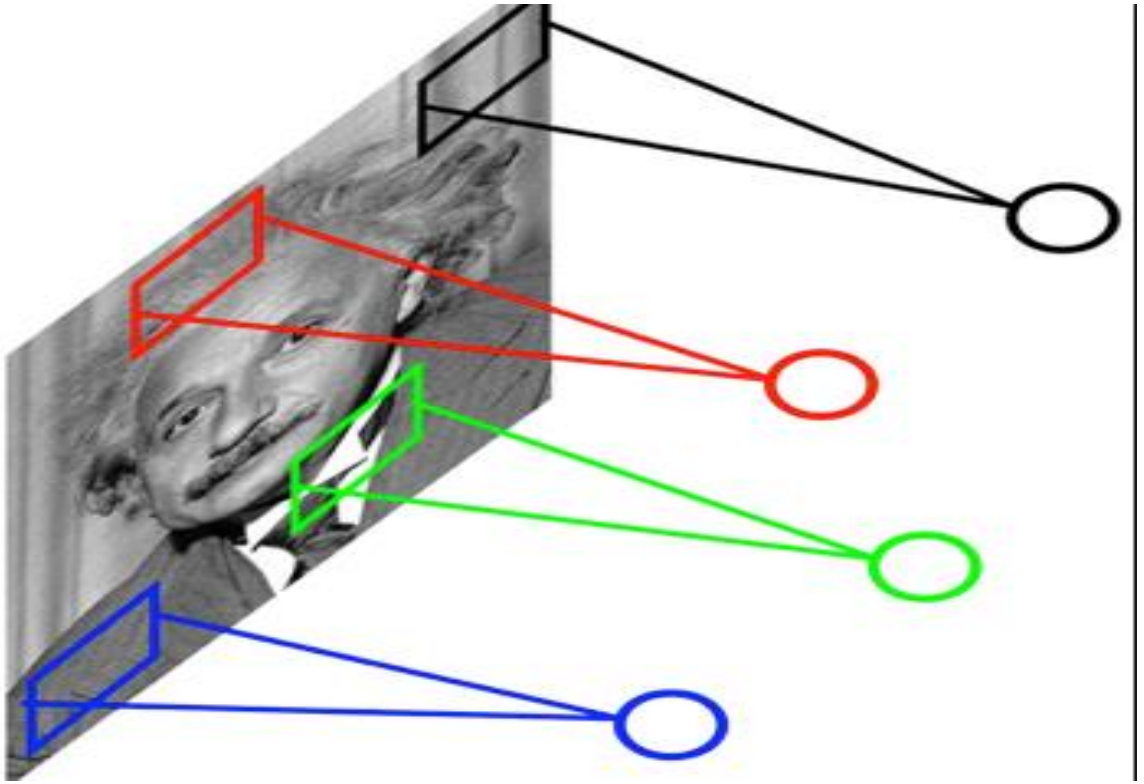
FULLY CONNECTED NEURAL NET



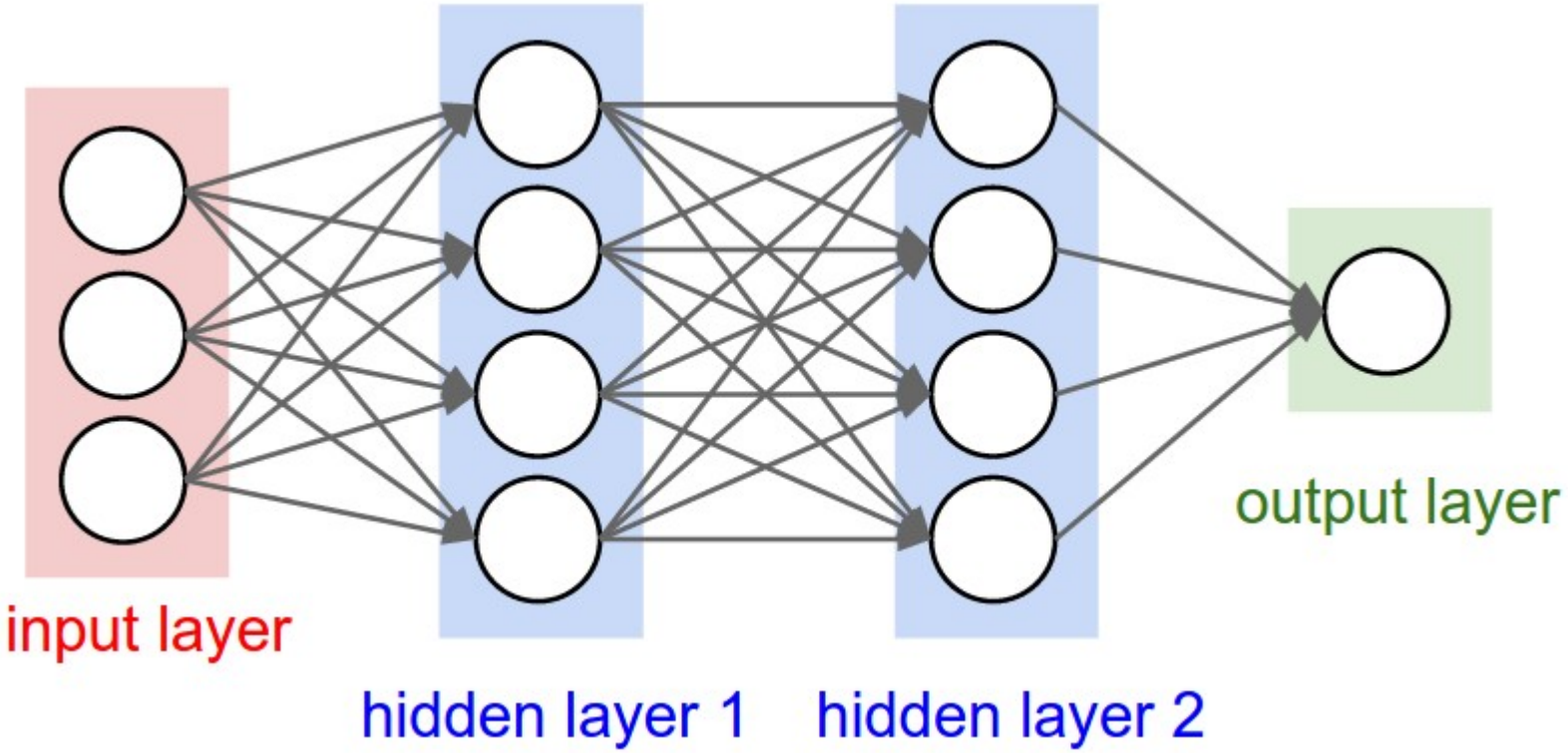
Example: 1000x1000 image
1M hidden units
10^12 parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!

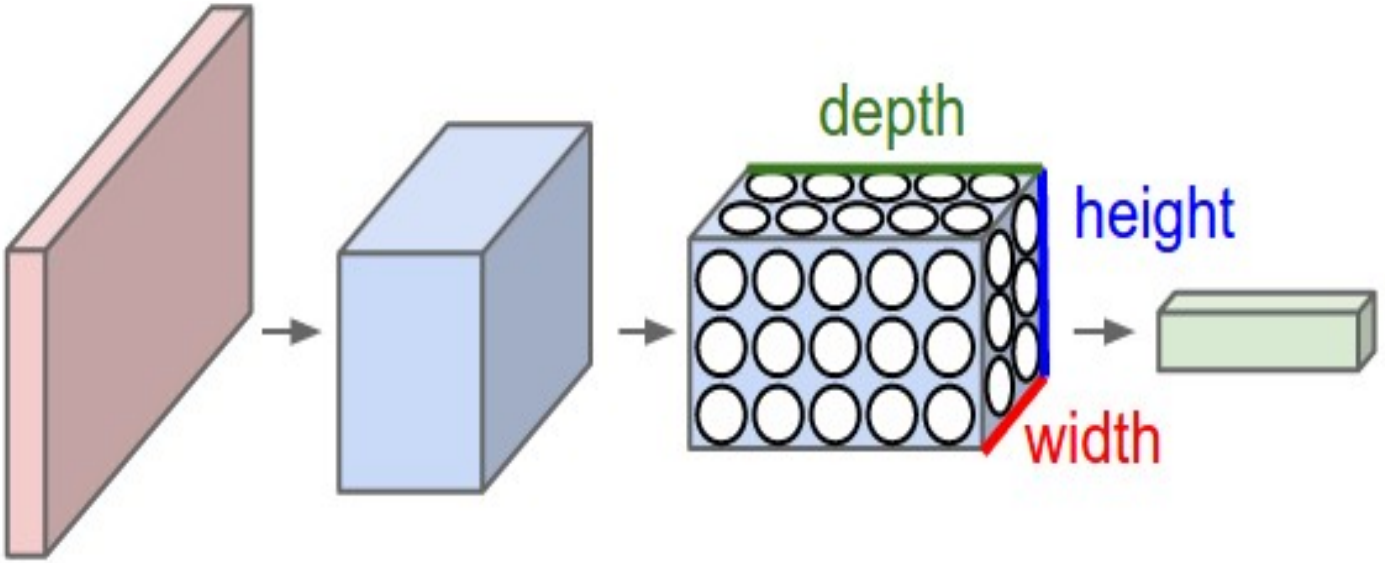
LOCALLY CONNECTED NEURAL NETWORK



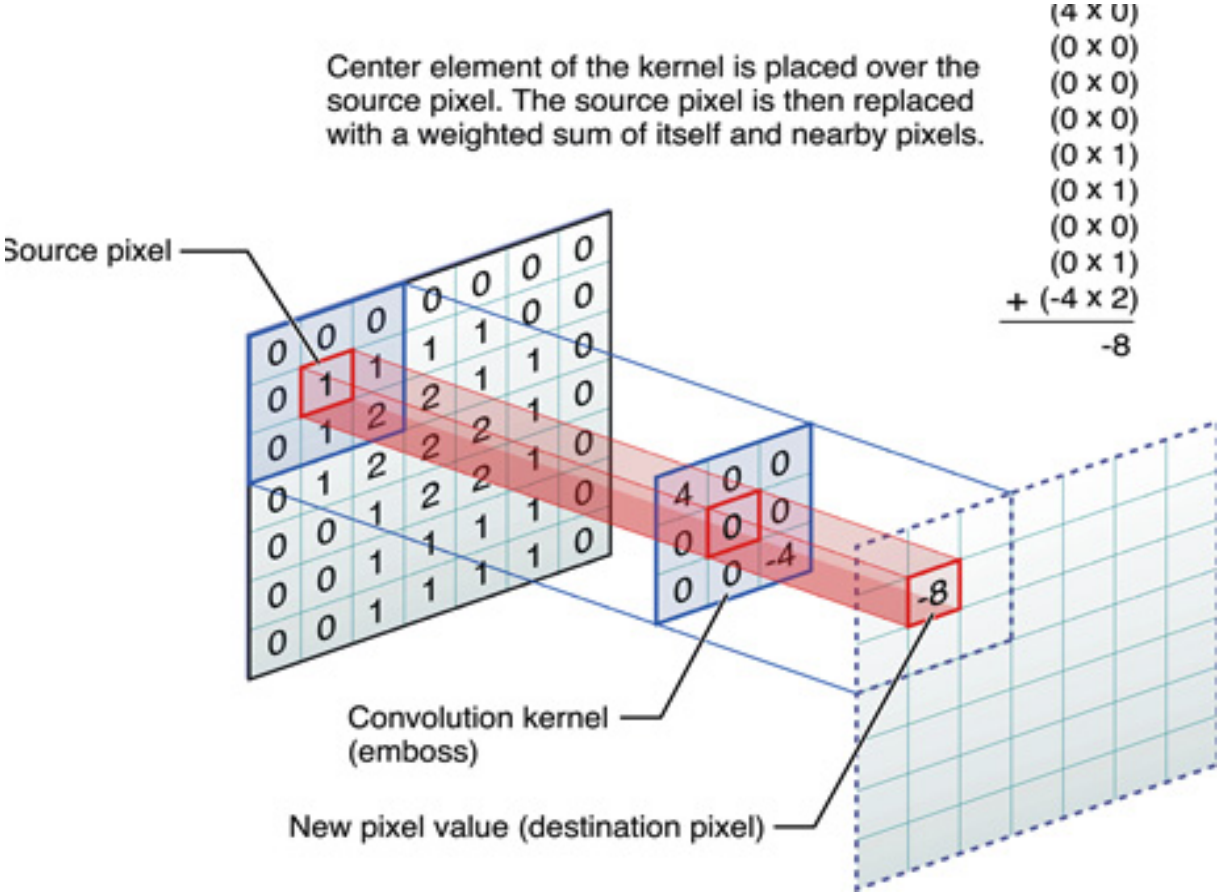
Neural Network



Convolutional Neural Network

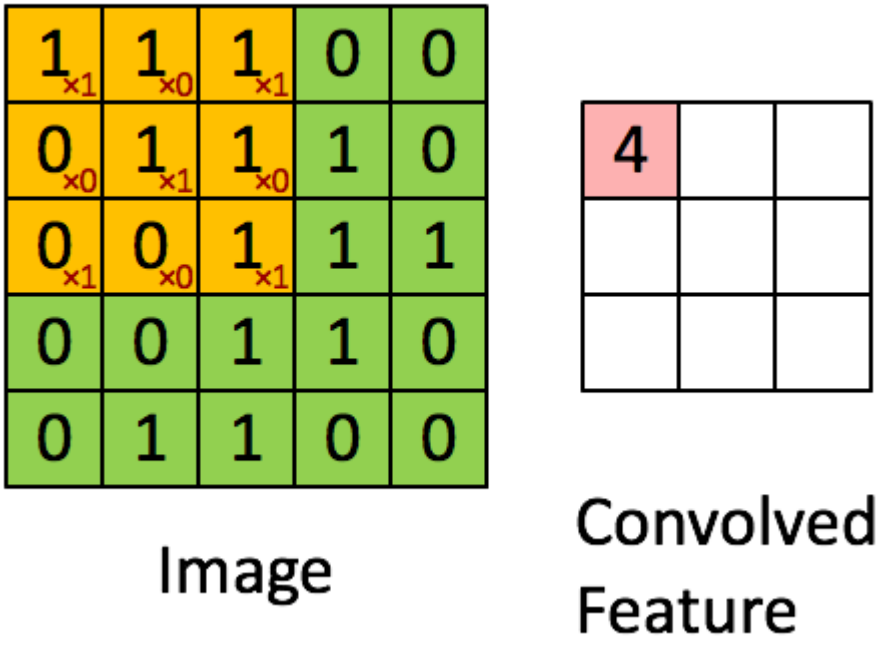


Convolutional Neural Network

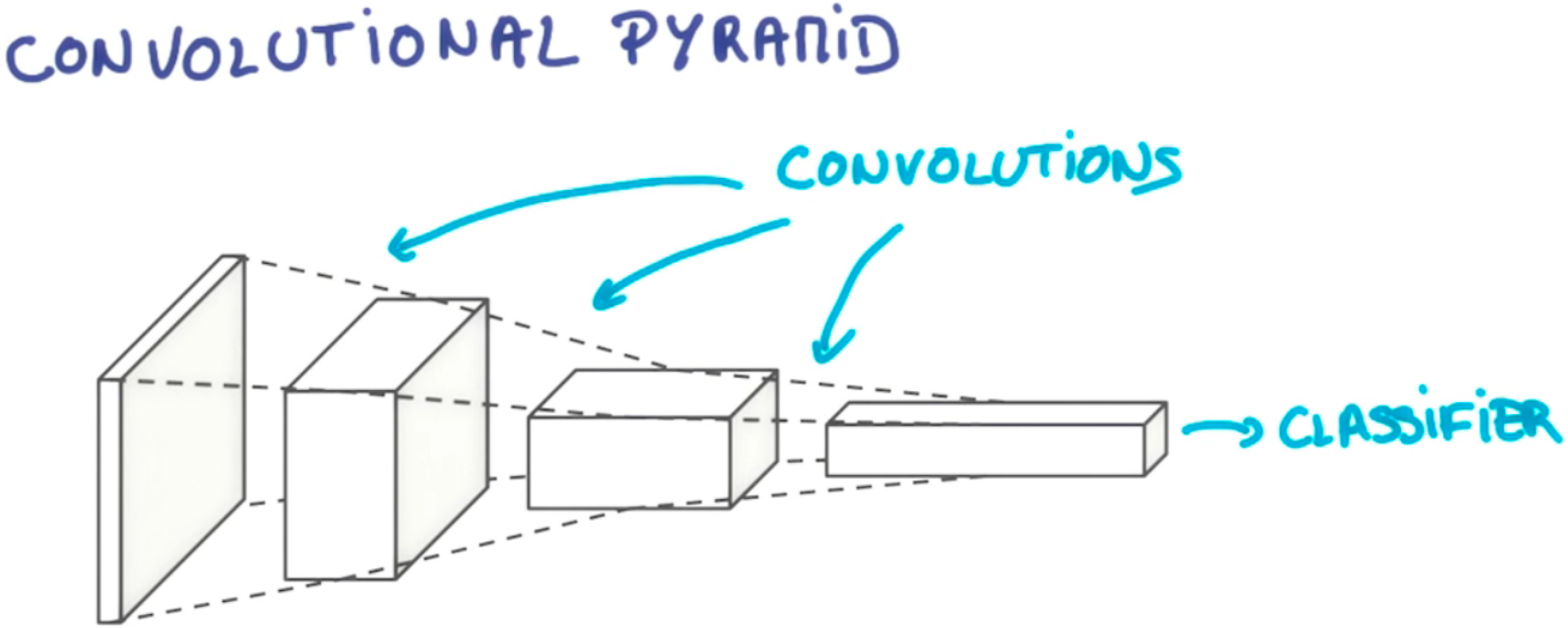


Canny Kernel

Convolutional Neural Network



Convolutional Neural Network



Forward propogation

Convolutional Neural Network

CNNs have three main components :

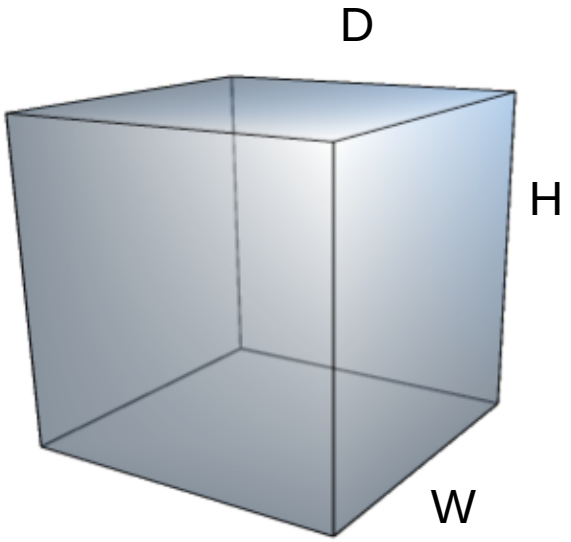
1) Convolutional Layers

2) Pooling layer

3) Fully Connected layer

Spatial Size of Output Volume

W x H x D



$$SZ = \frac{(W - F + 2P)}{S} + 1$$

Spatial Size of Output Volume

W x H x D

$$SZ = \frac{(W - F + 2P)}{S} + 1$$

W = size of input

F = size of receptive field

P = padding size

S = stride

Spatial Size of Output Volume

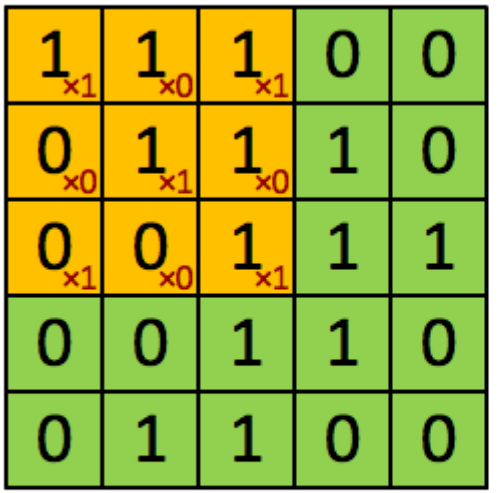
$$SZ = \frac{(W - F + 2P)}{S} + 1$$

W = size of input

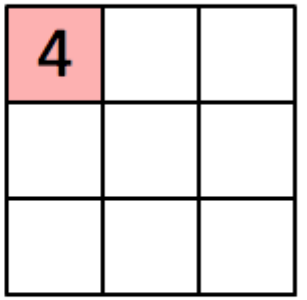
F = size of receptive field

P = padding size

S = stride



Image



Convolved Feature

W = 5 (5x5)

F = 3 (3x3)

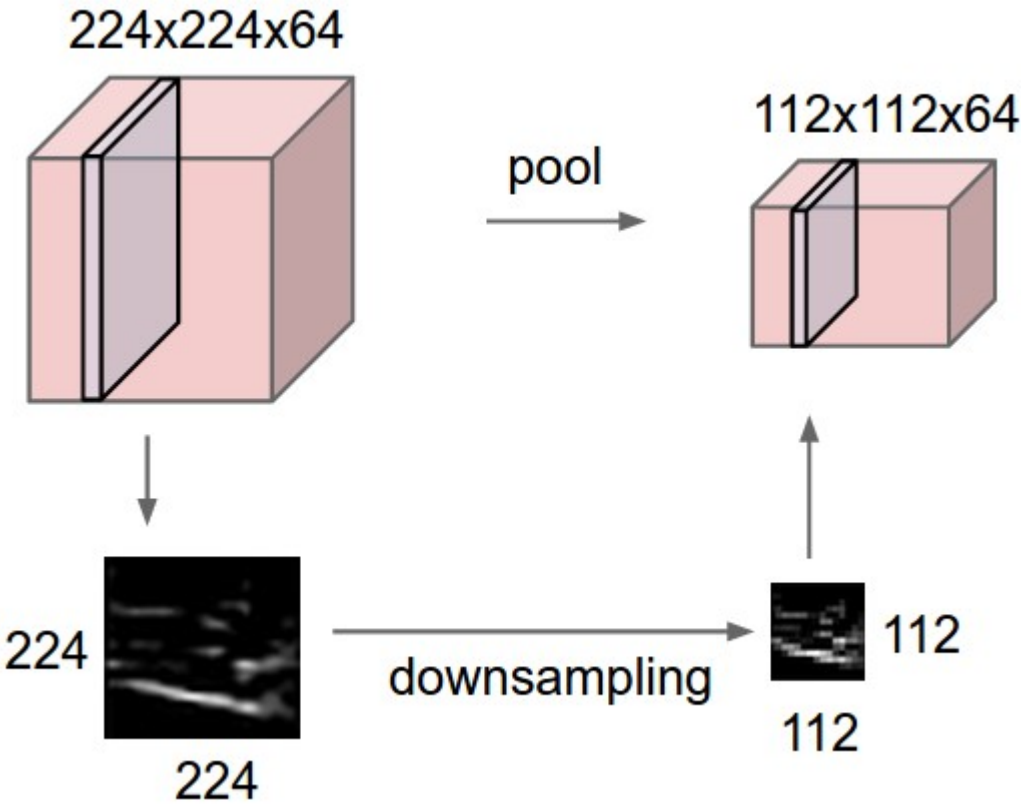
P = 0

S = 1

SZ = 3

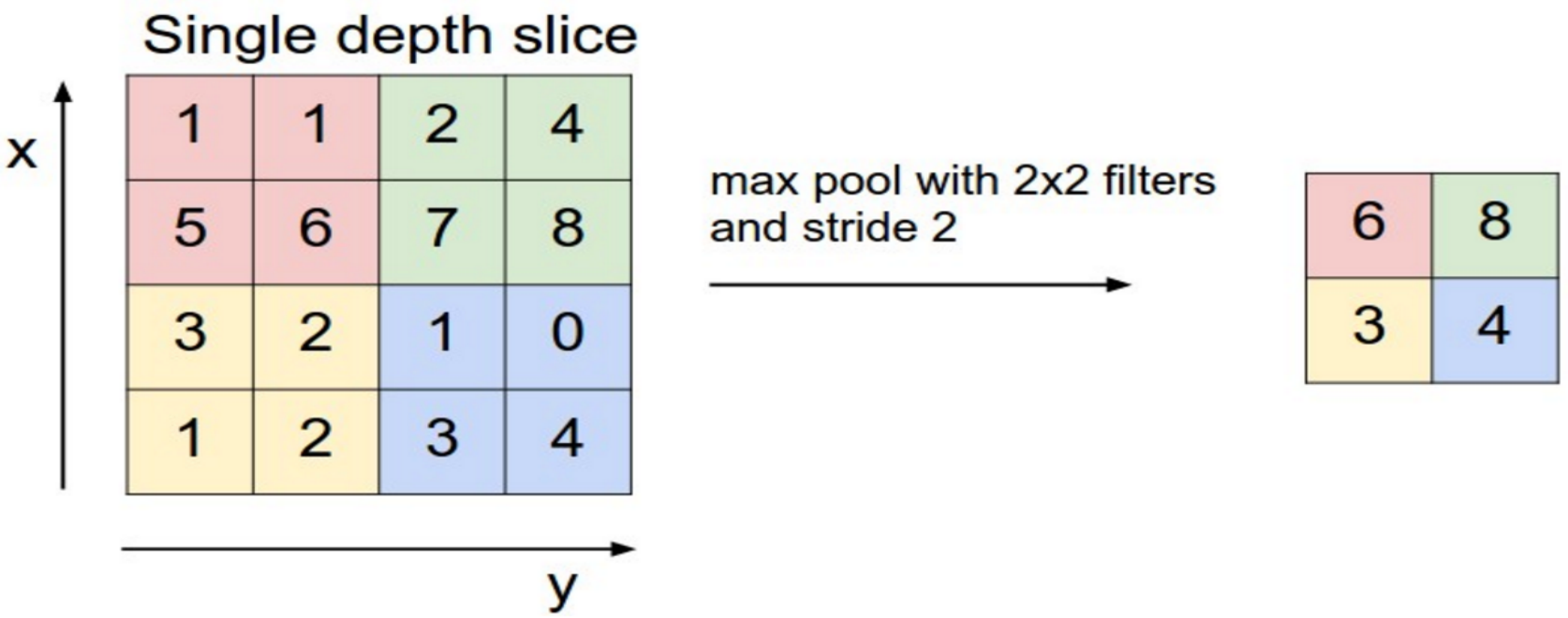
Pooling

Reduces *spatial size* of input by subsampling



Pooling

Reduces *spatial size* of input by subsampling



Fully Connected Layer

Convert spatial connections with volume to a single layer so it can go into a regular NN

Example : Ouput at one layer is 7x7x512

We need to input them into a NN of 1024 inputs

-- Use window of size 7x7 with stride = 1 and pad = 0

$$SZ = \frac{(W - F + 2P)}{S} + 1$$

SZ = 1x 1x 1024