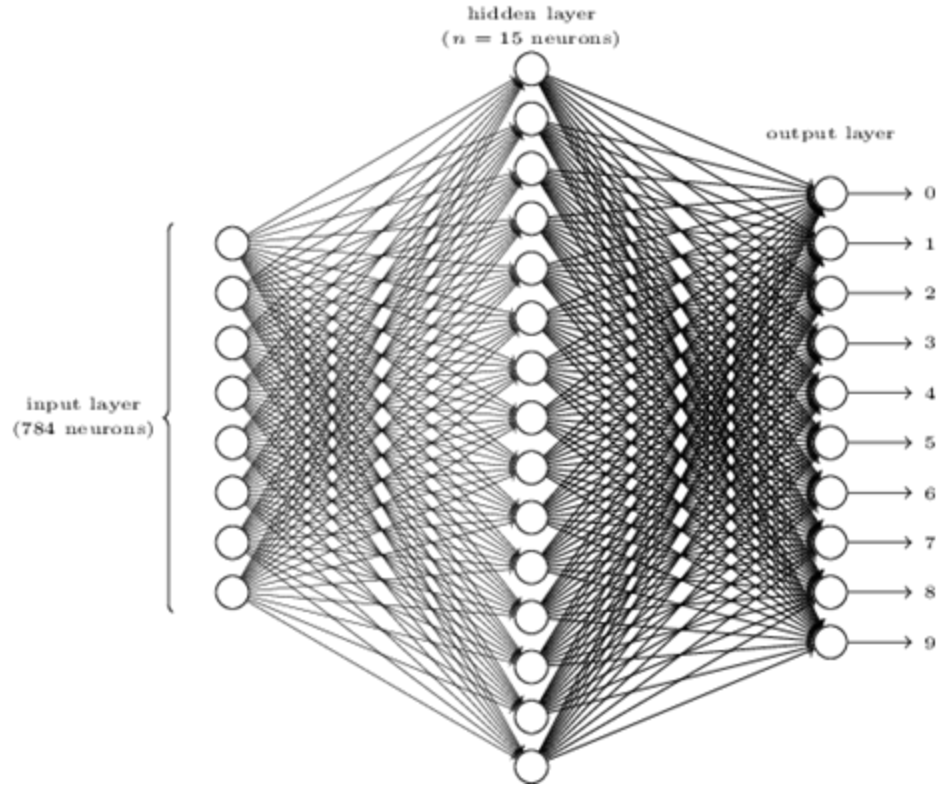


# *Convolutional Neural Networks*

**Nail Ibrahimli**

# Recognizing Digits with Neural Nets.

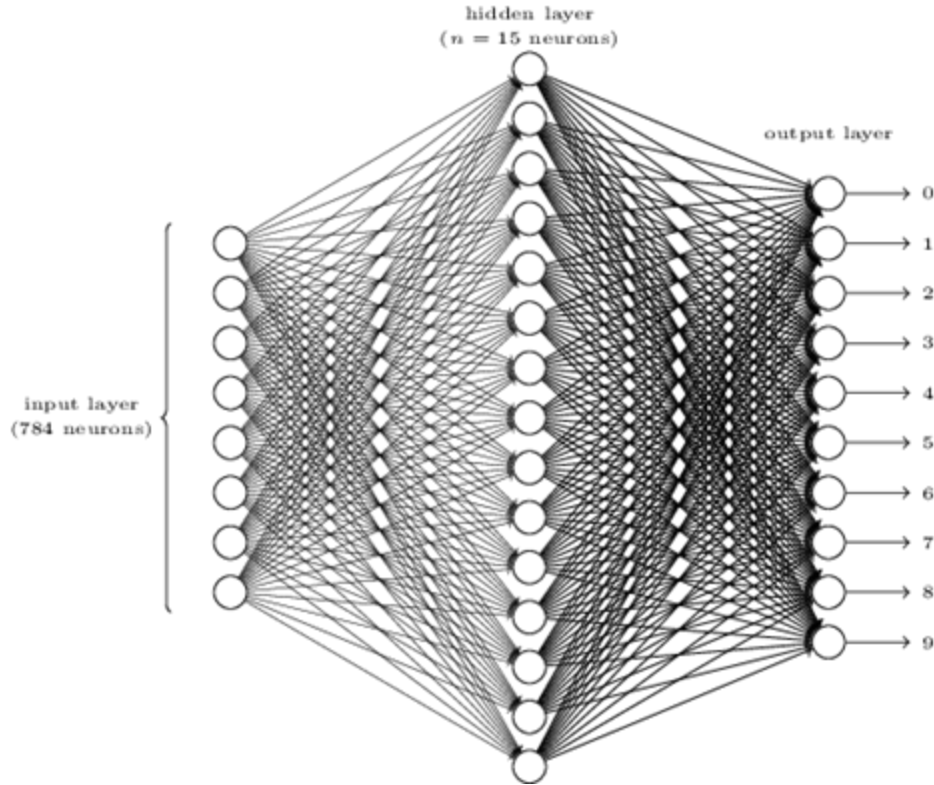


For  $x$  representing digit 6:

$$y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0)^T$$

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

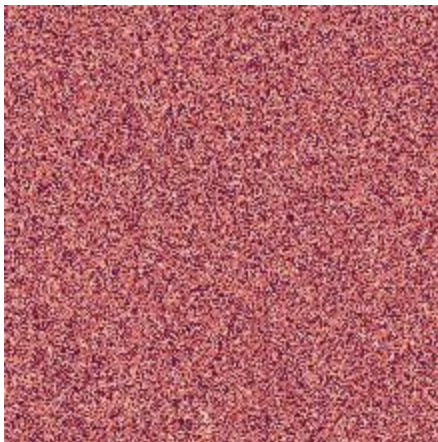
# Complexity of the World:



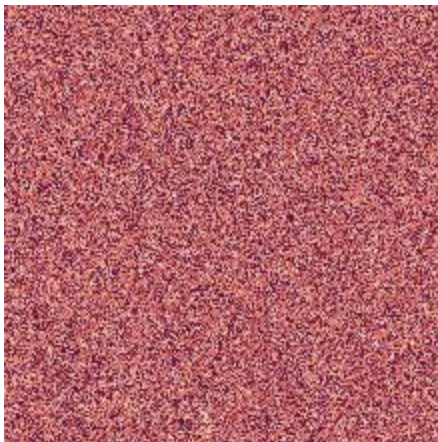
Images have much higher resolutions and input dimensions.

Number of the categories and classes are usually much more than 10.

# *Image signals - locality*



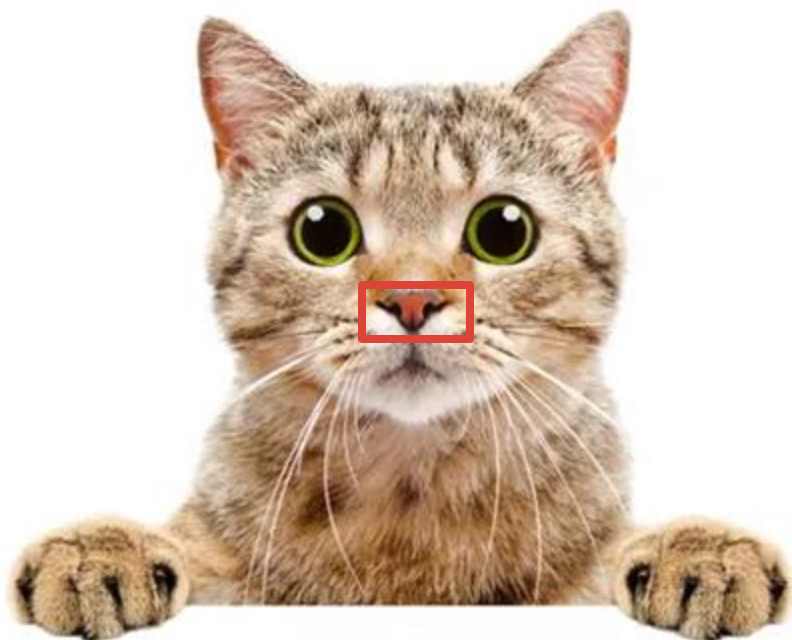
# *Image signals - locality*



# *Image signals - locality*



# *Image signals - locality*





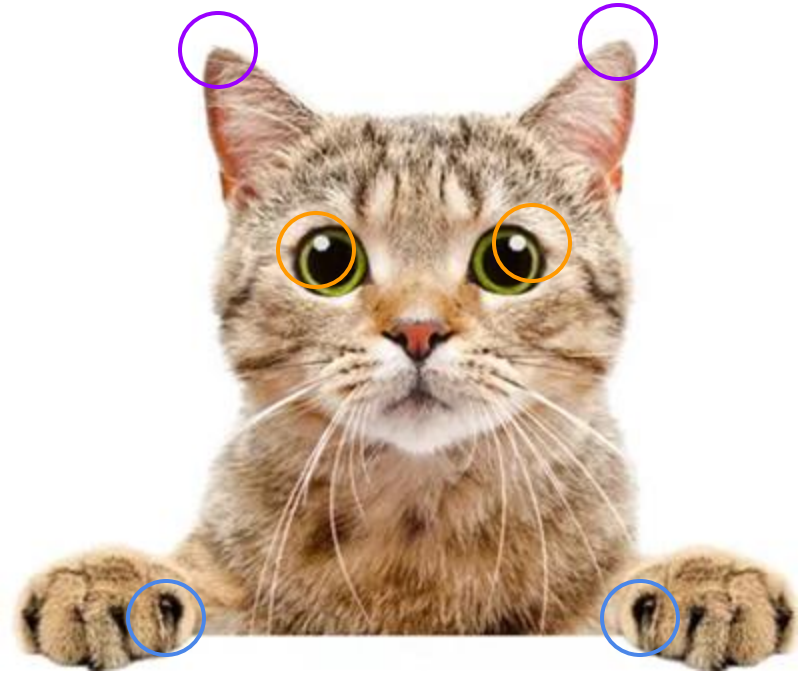
# *Image signals - stationarity*



Image credits: Andy Warhol



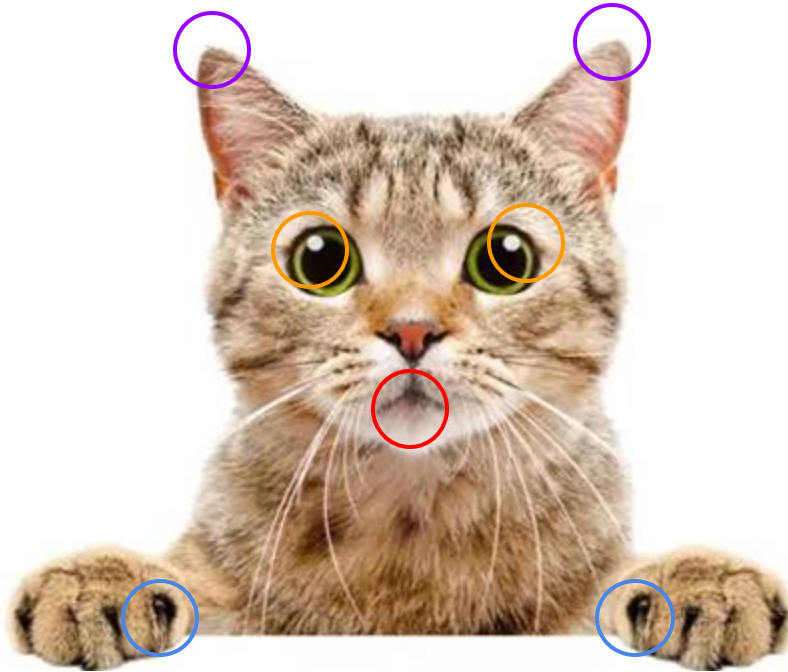
# *Image signals - stationarity*



# *Image signals - compositionality*



# *Image signals - compositionality*

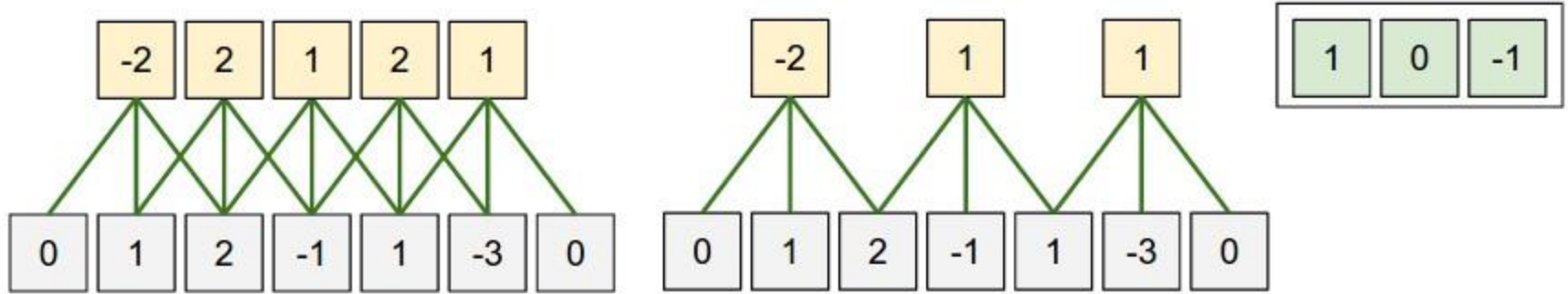


# *Image signals properties*

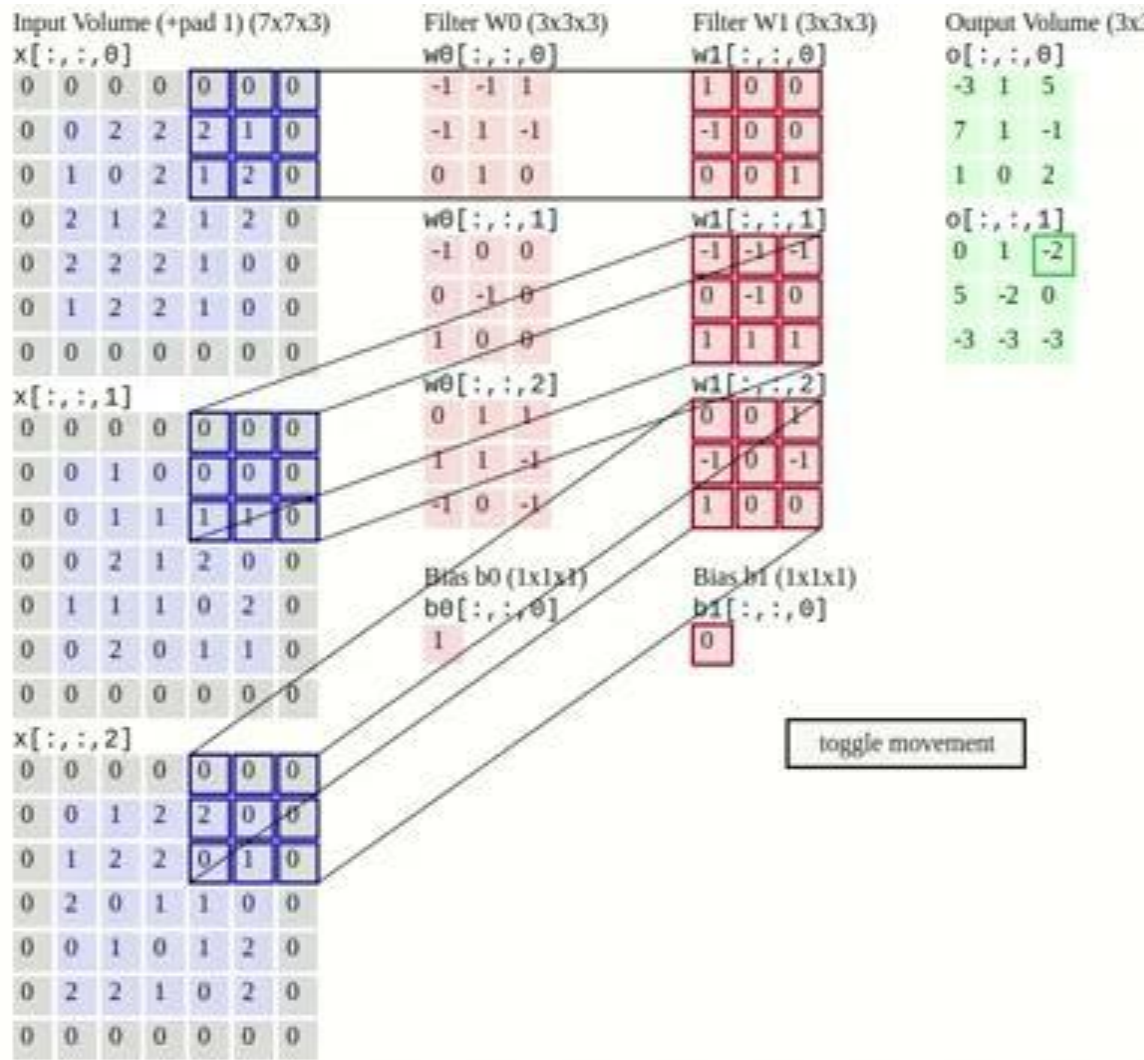
- Locality - neighboring pixels are correlated
- Stationarity - similar features can occur multiple times in different positions in the image plane
- Compositionality - natural images are composed of features



# One dimensional convolution



# Image convolution



# Kernels

Input image



Convolution  
Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map



Input image



Kernel

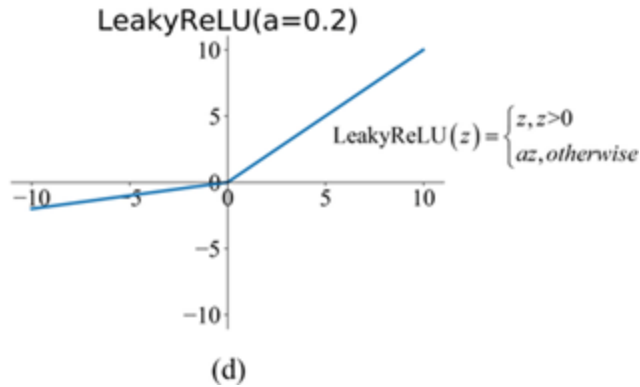
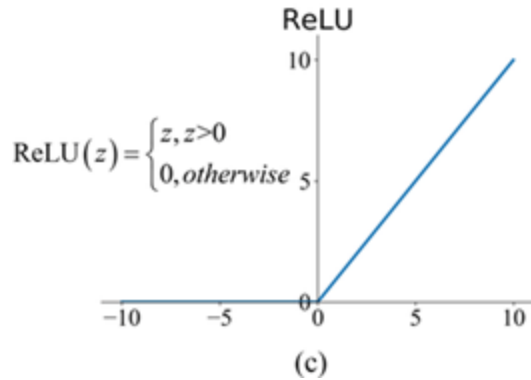
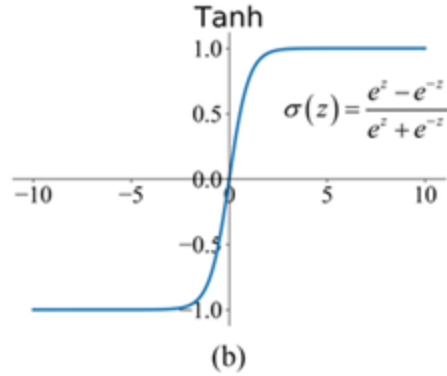
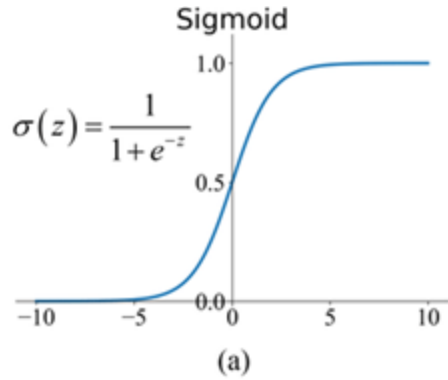
$$\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$

Feature map

















# Activations

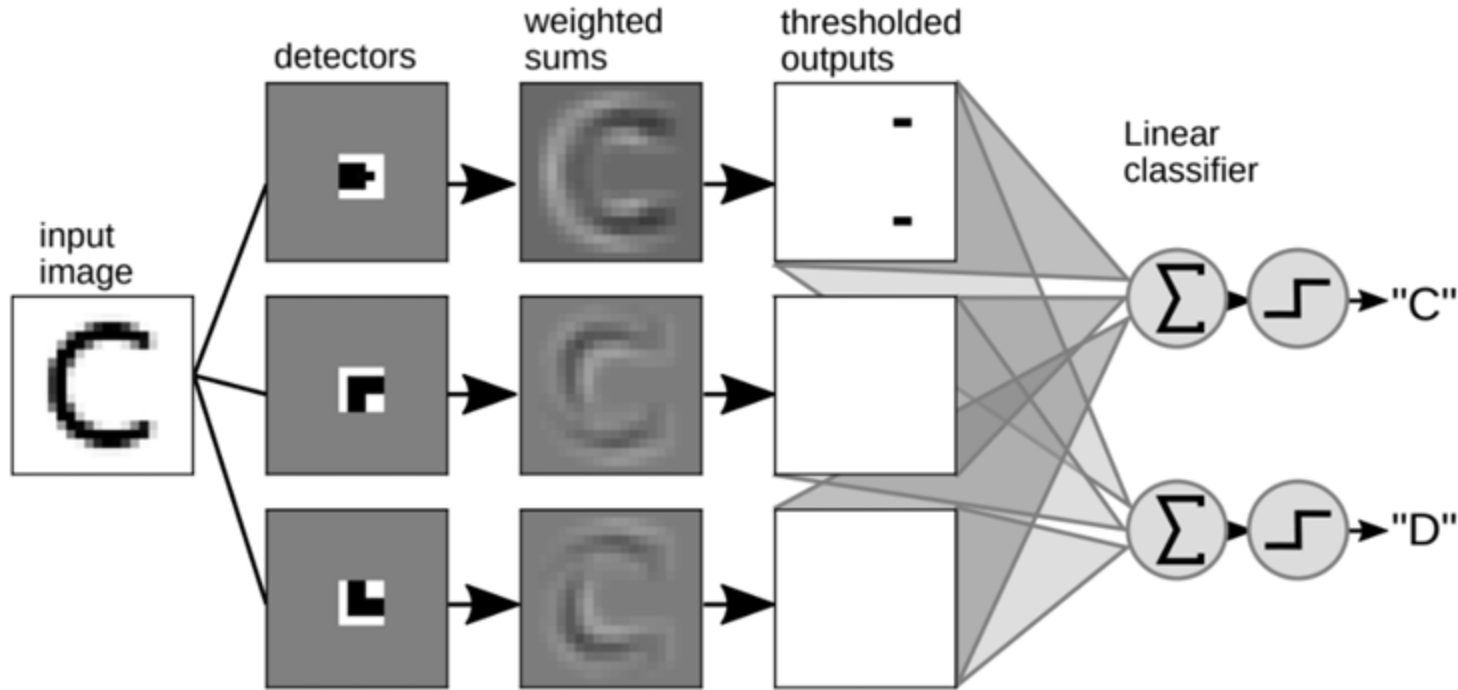


# Activations

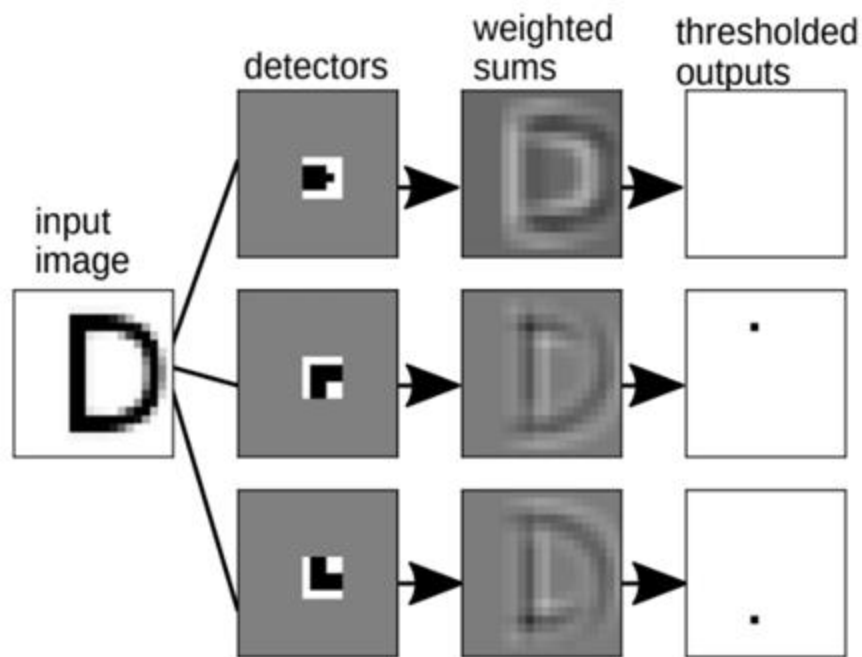
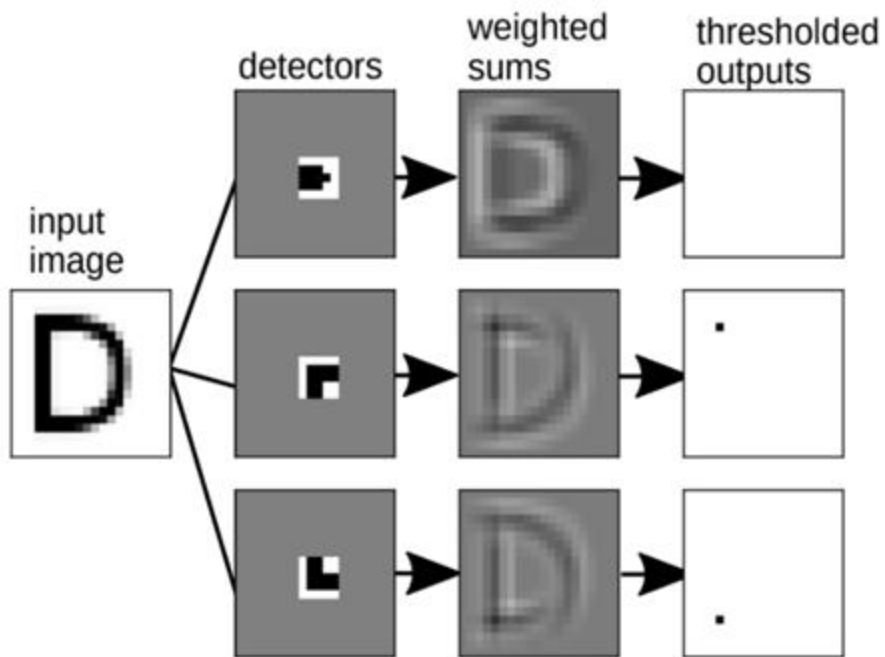
[PyTorch activation functions](#)

<p>Sigmoid</p>  $y = \frac{1}{1 + e^{-x}}$	<p>Tanh</p>  $y = \tanh(x)$	<p>Step Function</p>  $y = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	<p>Softplus</p>  $y = \ln(1 + e^x)$
<p>ReLU</p>  $y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	<p>Softsign</p>  $y = \frac{x}{1 +  x }$	<p>ELU</p>  $y = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$	<p>Log of Sigmoid</p>  $y = \ln\left(\frac{1}{1 + e^x}\right)$
<p>Swish</p>  $y = \frac{x}{1 + e^{-x}}$	<p>Sinc</p>  $y = \frac{\sin(x)}{x}$	<p>Leaky ReLU</p>  $y = \max(0.01x, x)$	<p>Mish</p>  $y = x(\tanh(\text{softplus}(x)))$

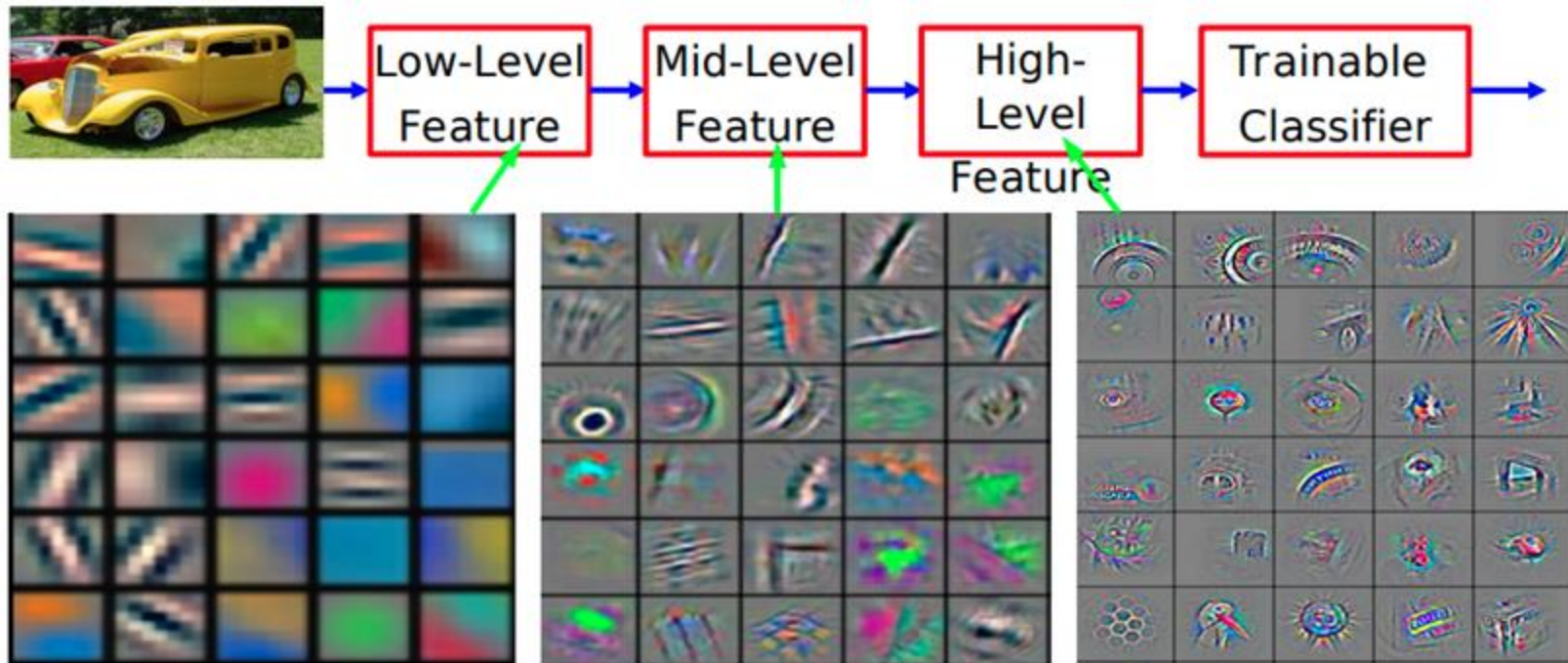
# Convolution motivation



# Convolution motivation



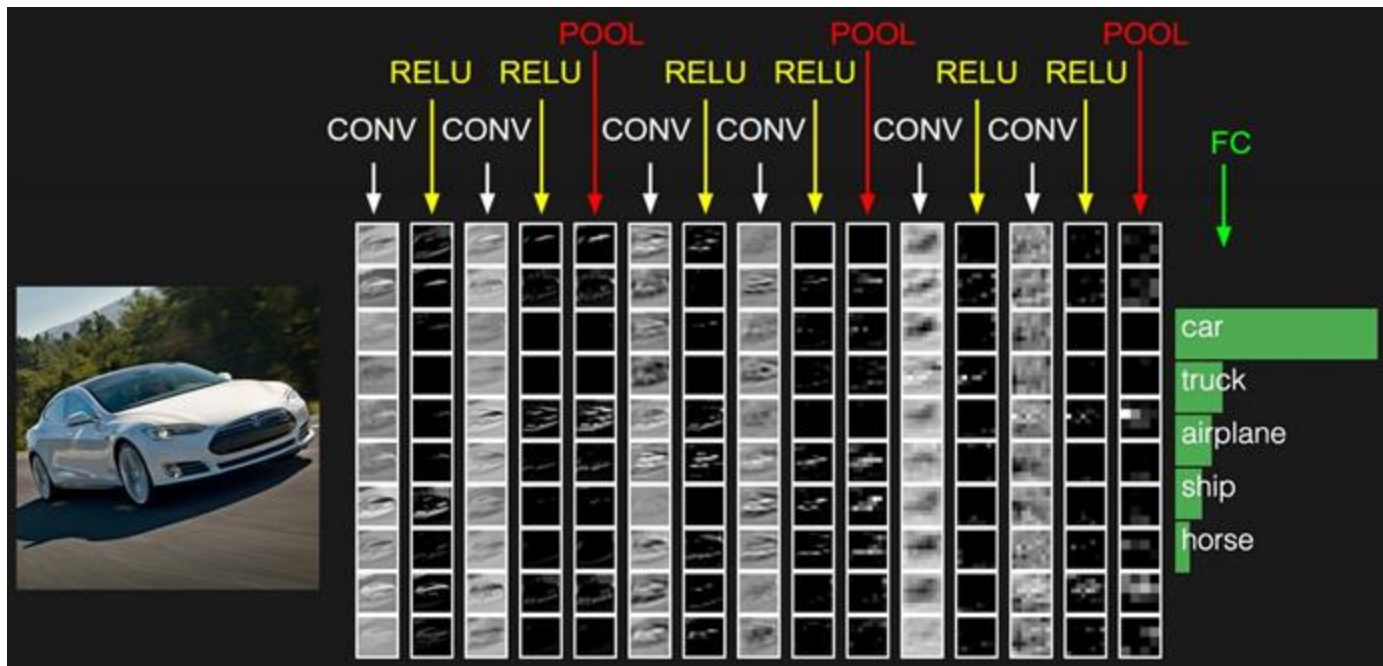
# Convolutional features



Slide credit: Yann Lecun

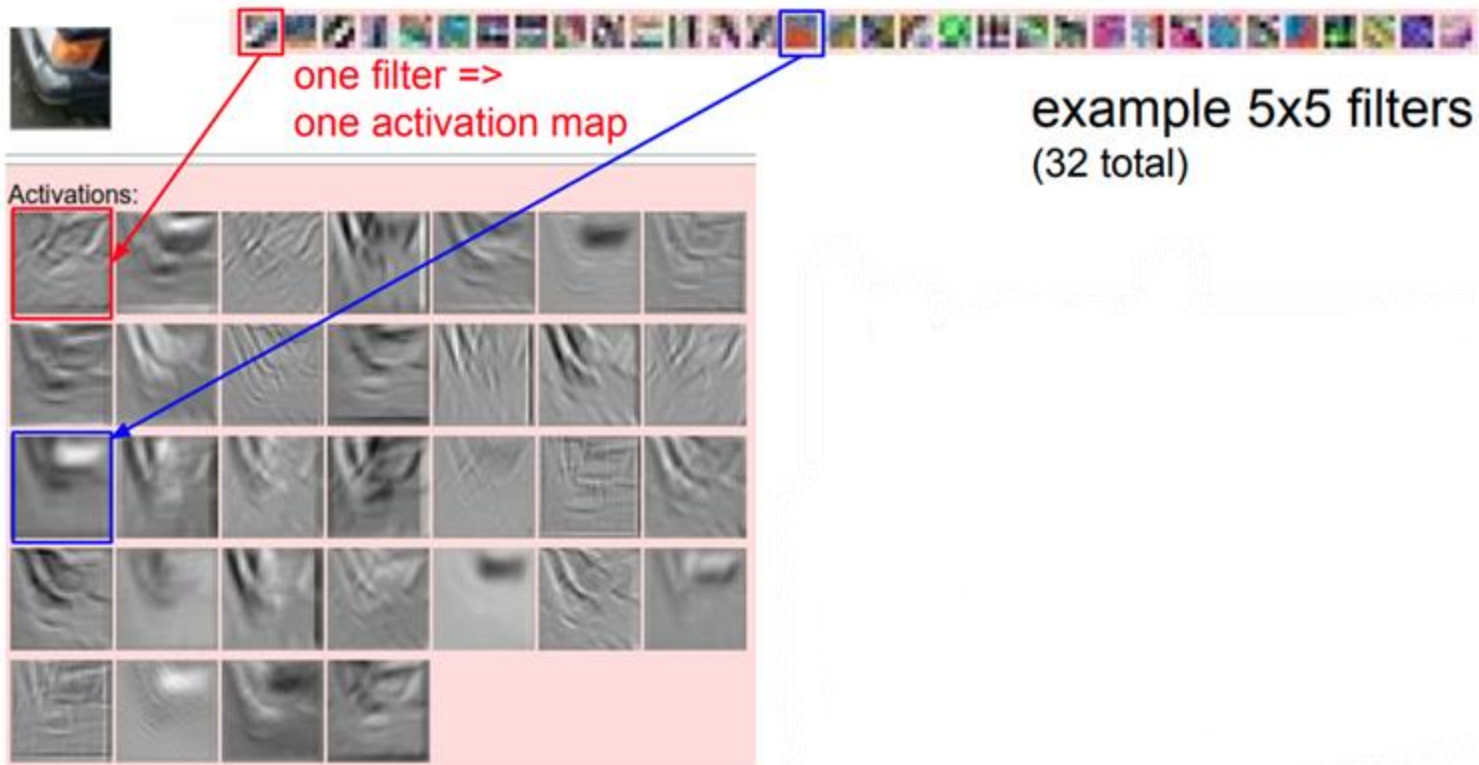
Image credit: Visualizing and Understanding Convolutional Networks (Zeiler & Fergus, 2013)

# Convolutional features



Slide credit: Andrej Karpathy

# Convolutional kernels





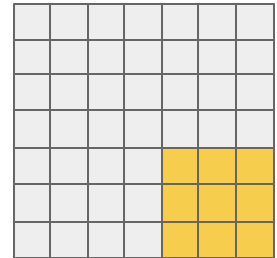
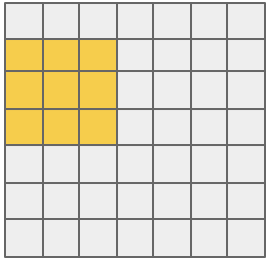
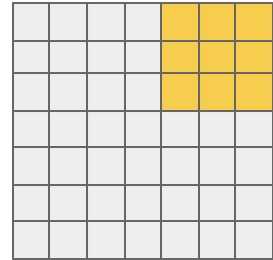
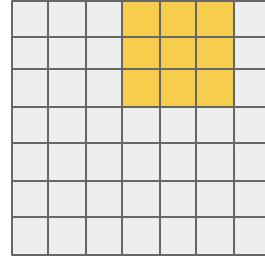
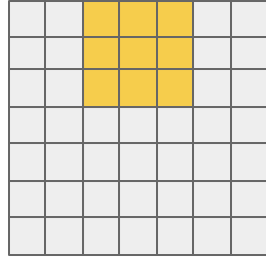
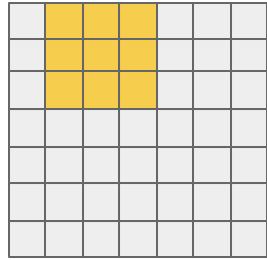
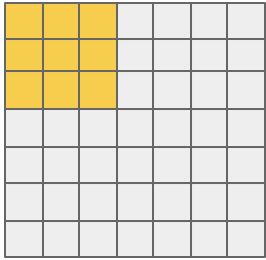
# *Convolutional low-level features*



Image credit: Stanford CS231n

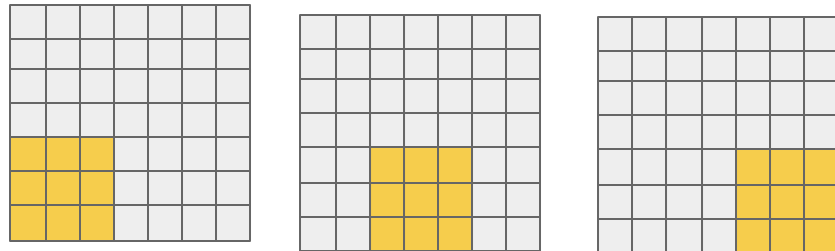
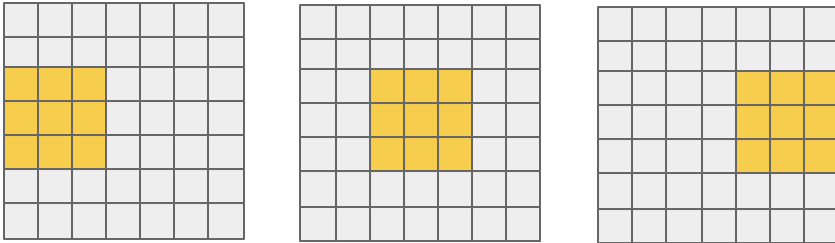
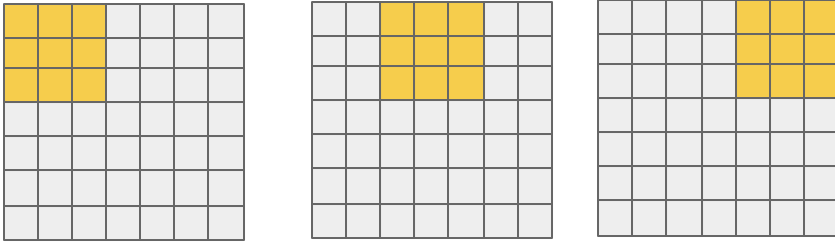
# Convolution operation

$N=7, F=3, S=1$



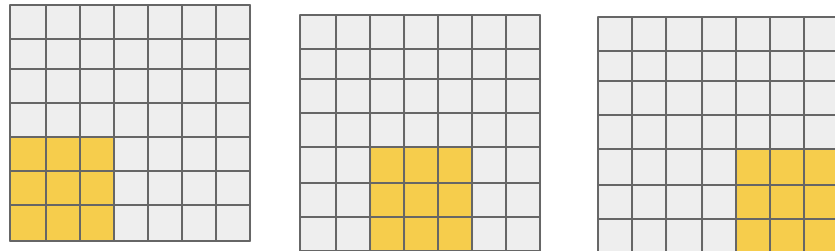
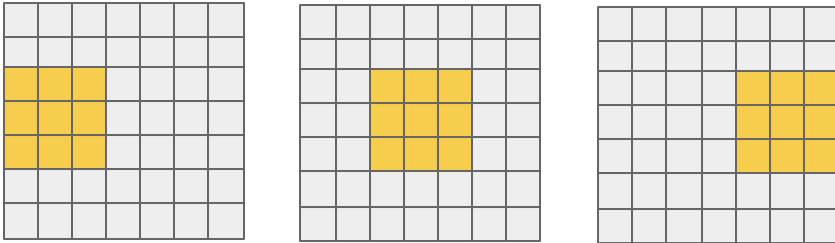
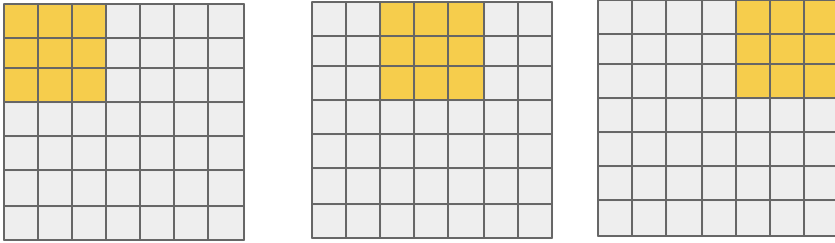
# Convolution operation

$N=7, F=3, S=2$



# Convolution operation

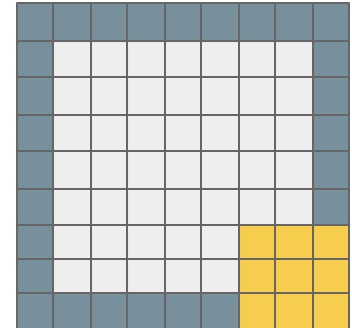
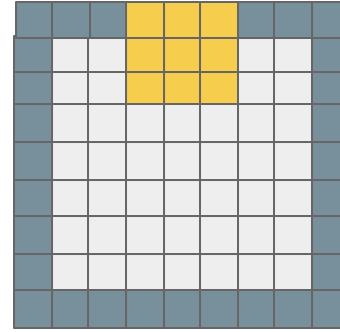
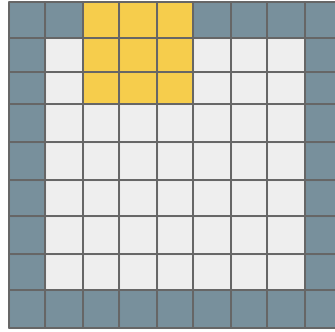
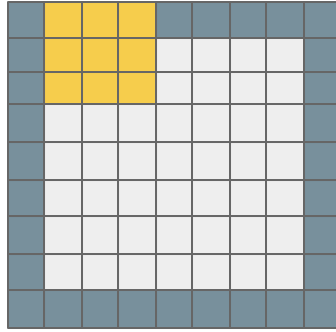
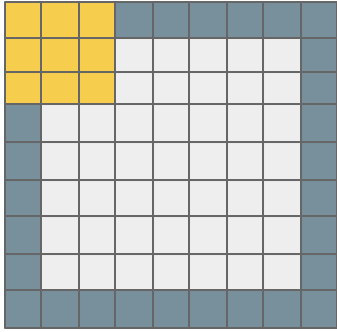
$N=7, F=3, S=2$



$$\text{Output} = (N-F)/S+1$$

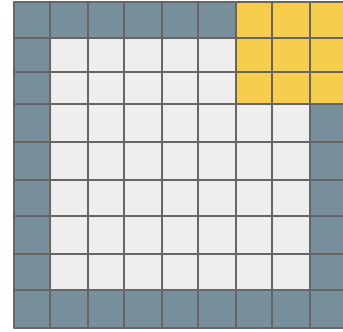
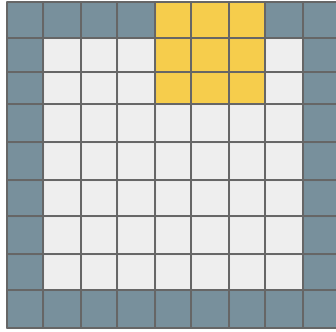
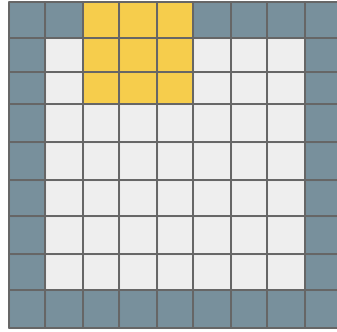
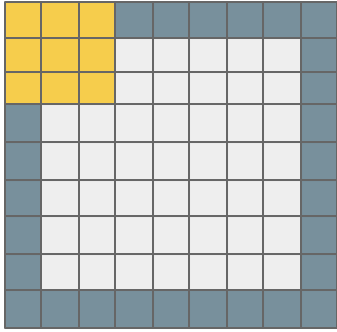
# Convolution operation

$N=7$ ,  $F=3$ ,  $S=1$ ,  $P=1$

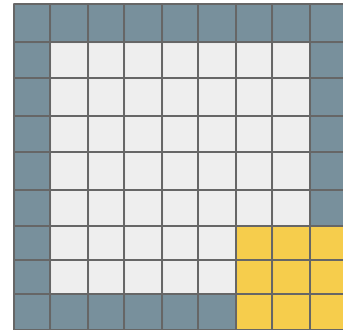


# Convolution operation

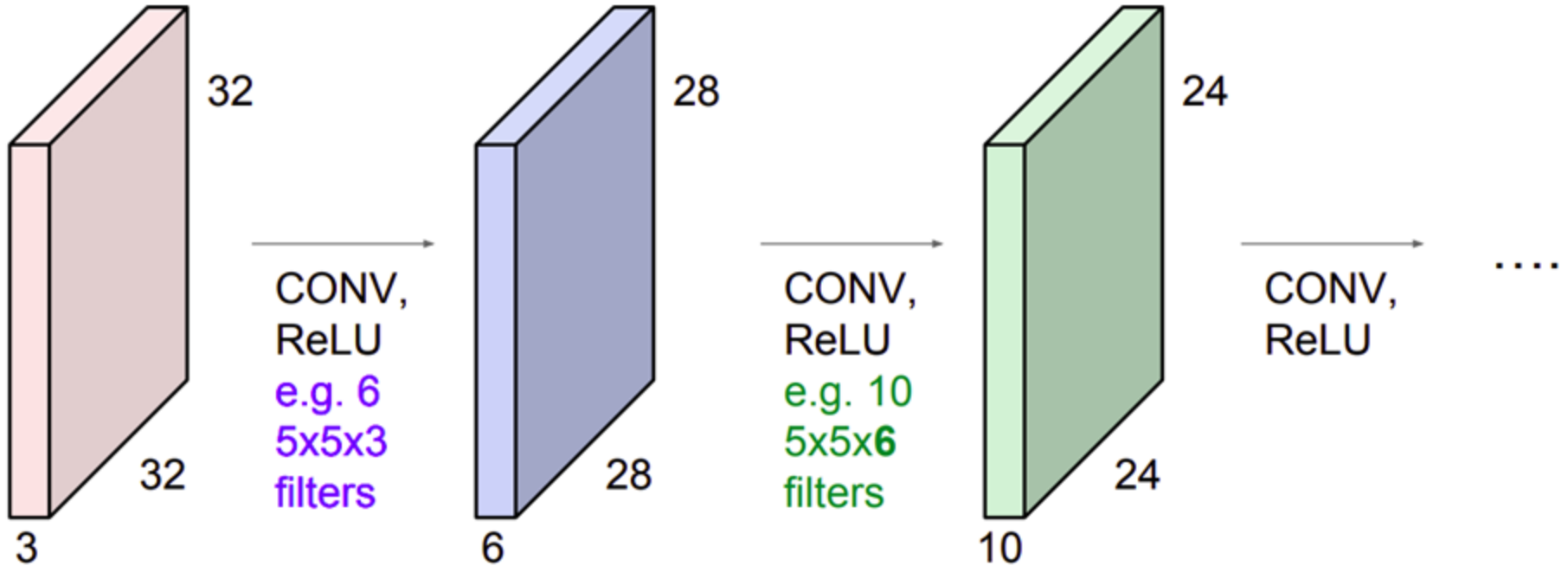
$N=7$   $F=3$ ,  $S=2$ ,  $P=1$



$$\text{Output} = (N-F+2P)/S+1$$

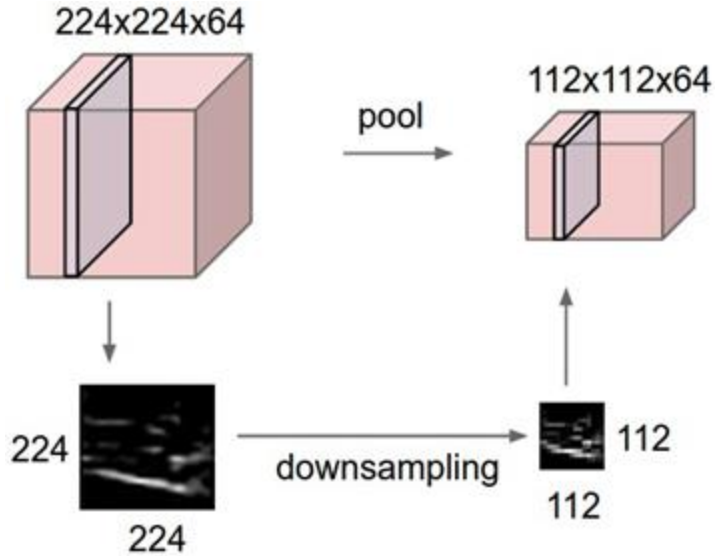


# Number of parameters



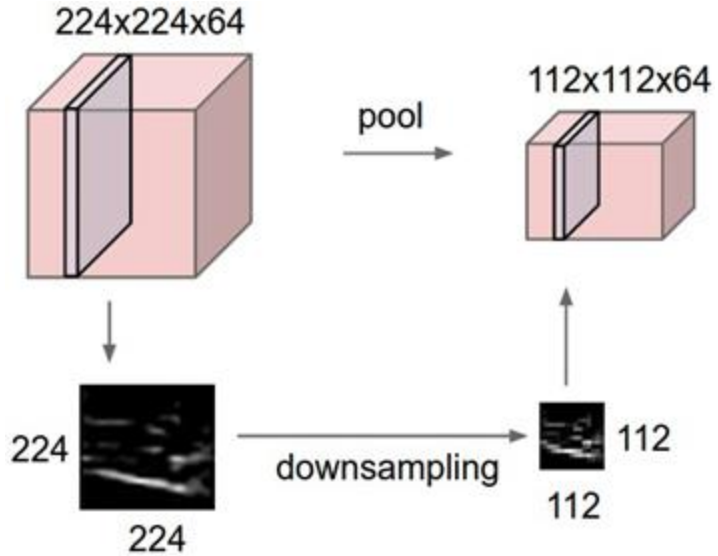


# Pooling layer



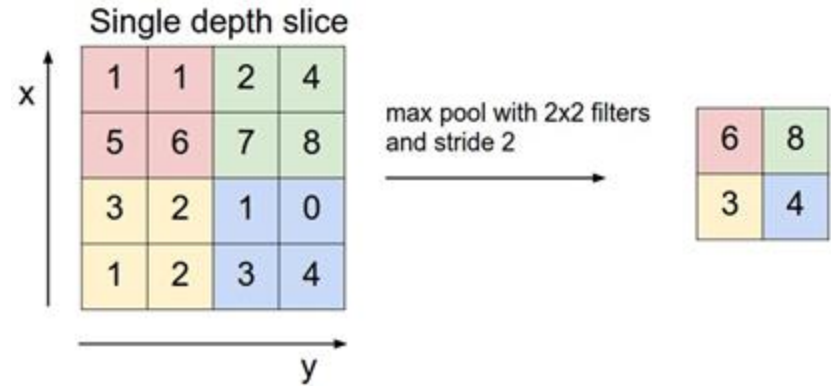
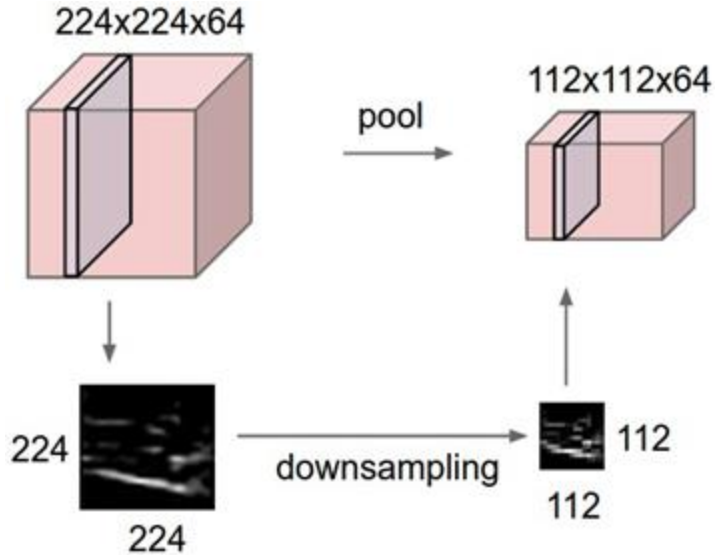
Advantages?

# Pooling layer

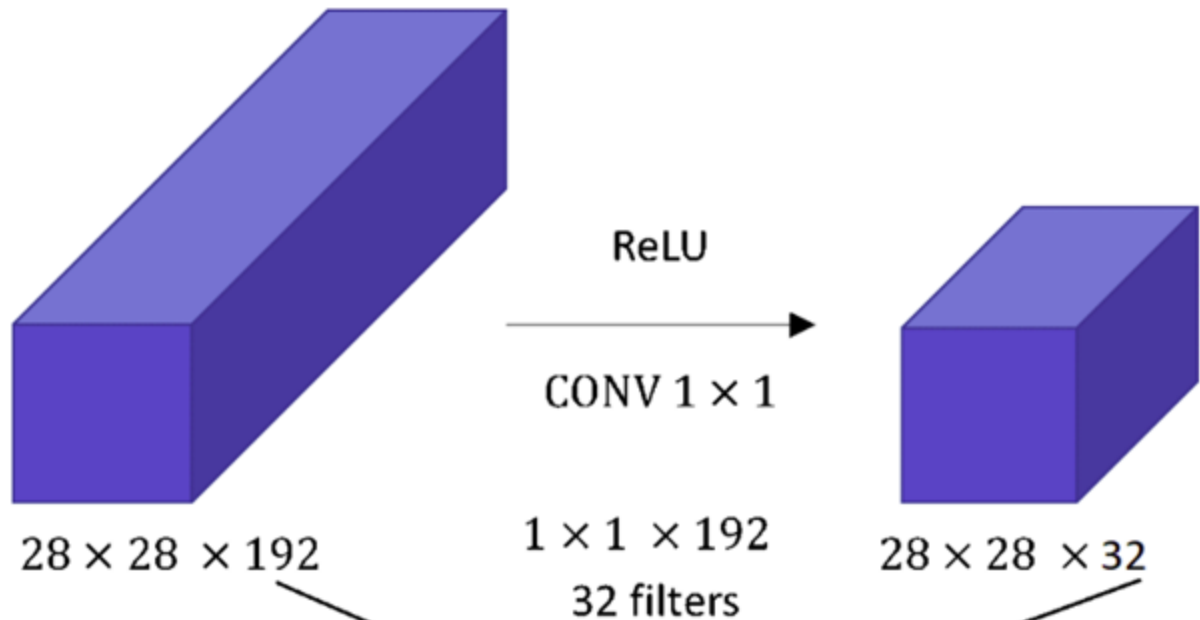


- reduce the spatial size of the representation
- control overfitting.

# Pooling layer (Maxpool)

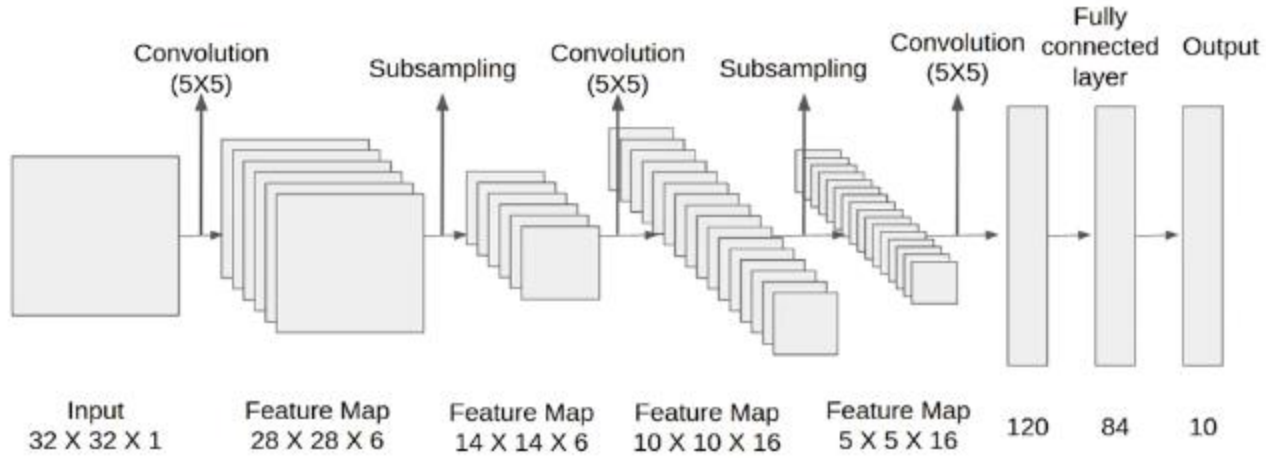


# 1x1 Convolutions

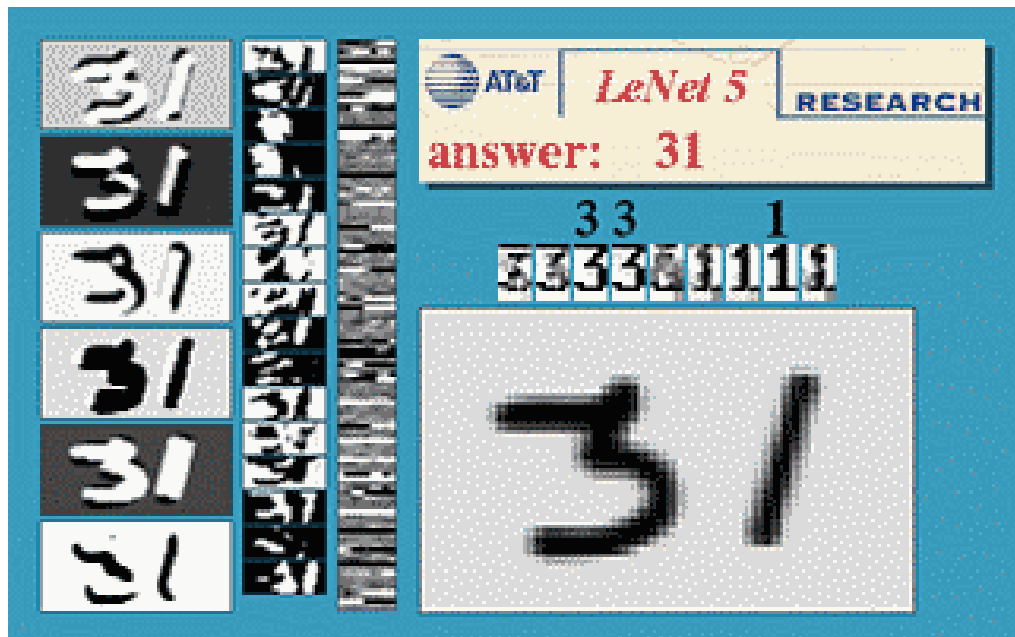


A number of filters goes from 192 to 32.

# LeNet5

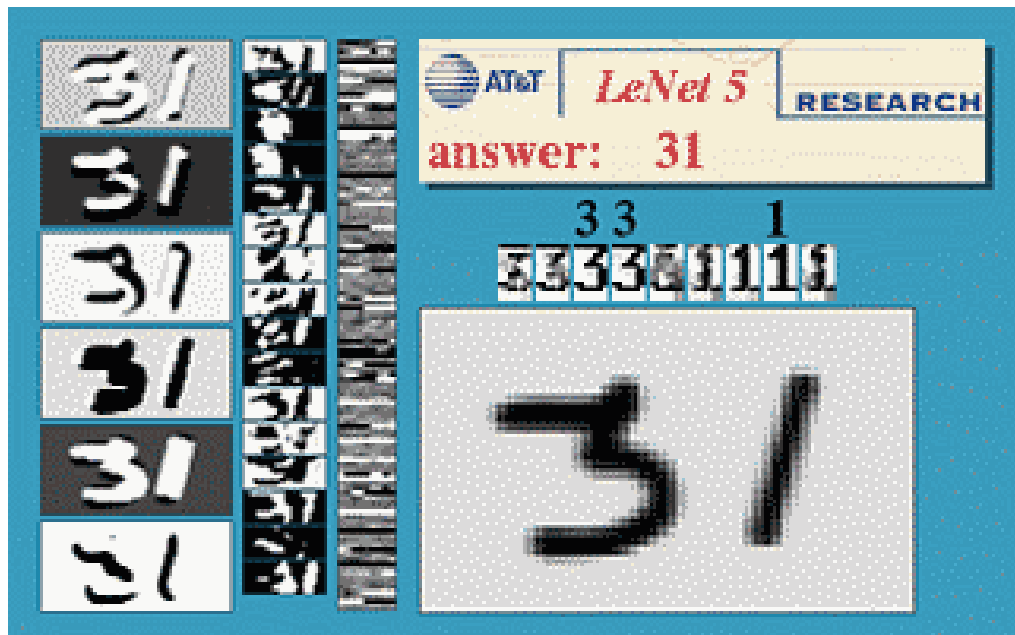


# LeNet5



Credit: Yann Lecun

# LeNet5



```
class LeNet5(nn.Module):  
    def __init__(self):  
        super().__init__()  
        self.conv1 = nn.Conv2d(1, 20, 5, 1)  
        self.conv2 = nn.Conv2d(20, 20, 5, 1)  
        self.fc1 = nn.Linear(4*4*20, 500)  
        self.fc2 = nn.Linear(500, 10)  
  
    def forward(self, x):  
        x = F.relu(self.conv1(x))  
        x = F.max_pool2d(x, 2, 2)  
        x = F.relu(self.conv2(x))  
        x = F.max_pool2d(x, 2, 2)  
        x = x.view(-1, 4*4*20)  
        x = F.relu(self.fc1)  
        x = self.fc2(x)  
        return F.logsoftmax(x, dim=1)
```

# AlexNet

