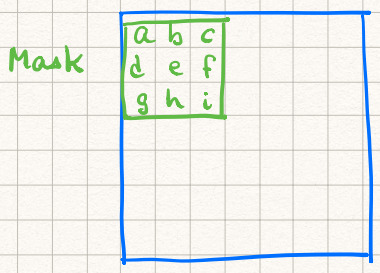
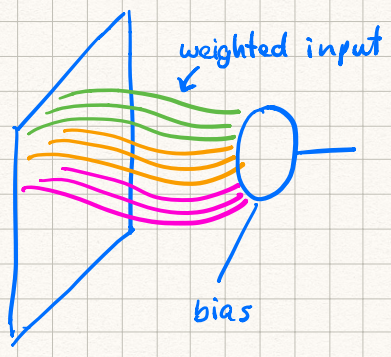
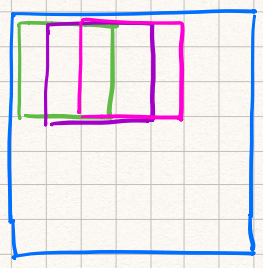
Convolutional neural networks

Convolutions

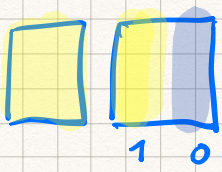






the mask advances to the right and down

Useful convolutions

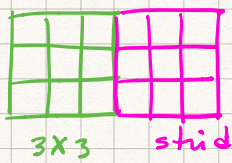


$\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$ edge detector, vertical

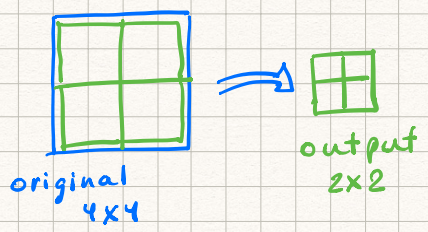
$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$ edge detector, horizontal

$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ averaging

A mask has a size $f \times f$ and advances by a stride S:



The stride can produce a subsampling :



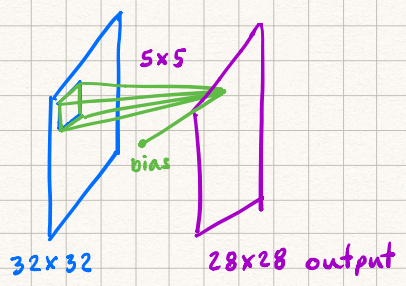
Computationally

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} a & b & a & b \\ c & d & c & d \\ a & b & a & b \\ c & d & c & d \end{bmatrix} \rightarrow \begin{bmatrix} a & 0 & 0 & b \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c & 0 & 0 & d \end{bmatrix}$$

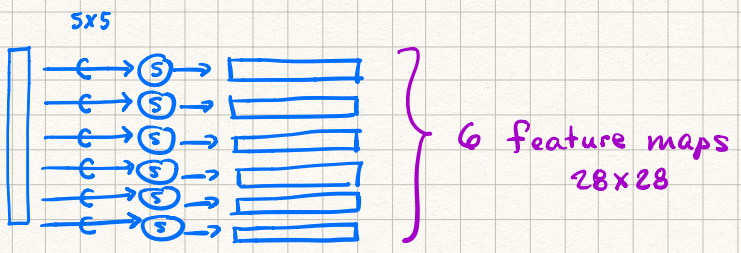
$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} a & 0 & 0 & b \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c & 0 & 0 & d \end{pmatrix} \rightarrow \begin{pmatrix} a & 0 & 0 & b \\ c & 0 & 0 & d \end{pmatrix}$$

$$\begin{pmatrix} a & 0 & 0 & b \\ c & 0 & 0 & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

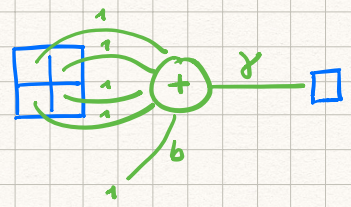
Le Net 5



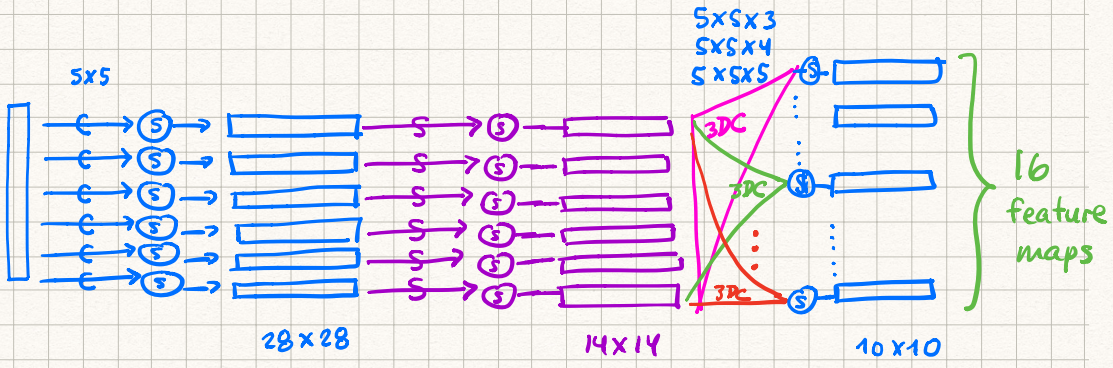
the convolution is based on shared weights

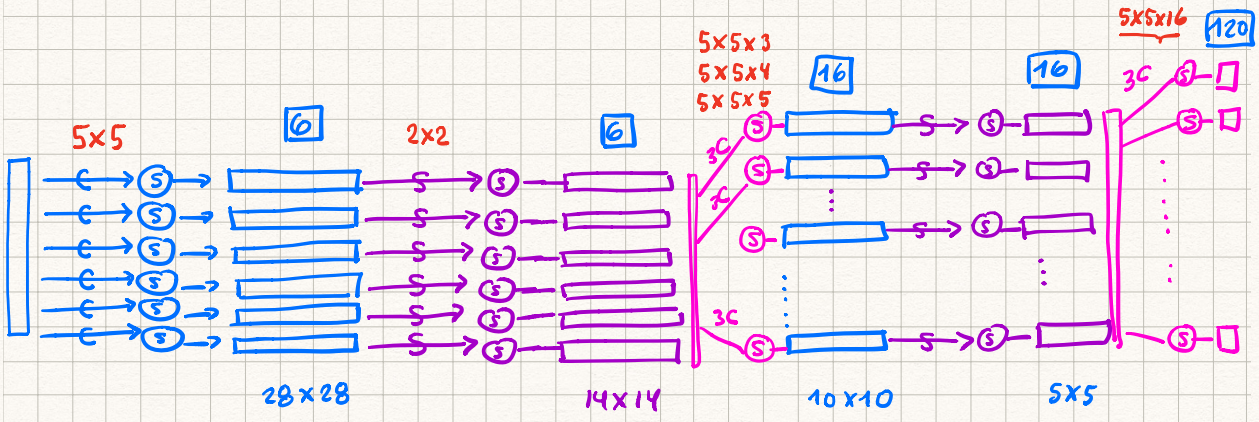


Subsampling convolution

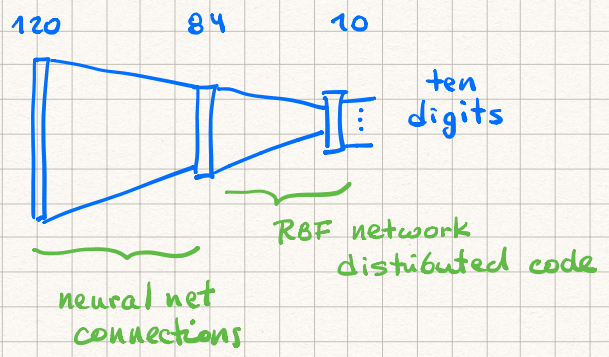


summarizes four pixels in one





Fully connected part



Details

LeNet 5 uses tanh as sigmoidal function

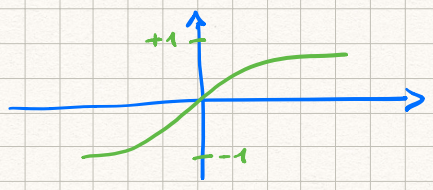
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$(e^x - e^{-x})\tanh x + (e^x + e^{-x})\tanh'(x) = e^x + e^{-x}$$

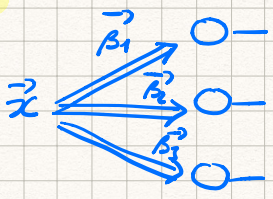
$$\frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} + \tanh'(x) = 1$$

$$\Rightarrow \quad \tanh'(x) = 1 - (\tanh x)^2$$

$\tanh x$ has range $+1, -1$



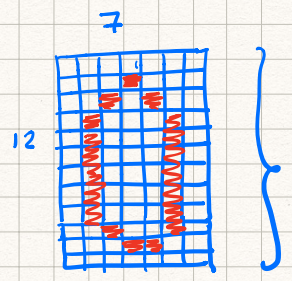
RBF-network



the unit whose β vector is closest to $\vec{x}$ "wins"

In LeNet the RBF network weights are not trainable.

Codes



The codes for each digit are vectors of length 84

The codes resemble the picture of the digit

Softmax function Softmin function

$$f_i(x_1,\dots,x_k) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$ ← normalization

Taking logs:

$$l_i(x_1,\dots,x_k) = x_i - \log\left(\sum e^{x_j}\right)$$

x_i: maximize x_i (correct class)

$\log\left(\sum e^{x_j}\right)$: keep the sum small

For minimum

Minimize $-l_i(x_1 \dots x_k)$

AlexNet