Neural networks

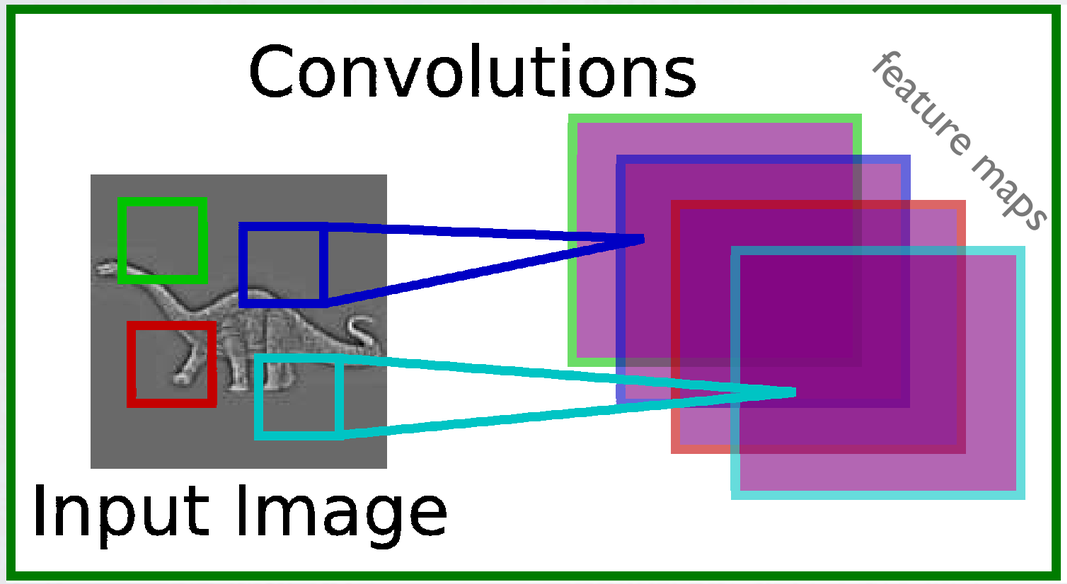
Computer vision - discrete convolution

COMPUTER VISION

Topics: parameter sharing

Jarret et al. 2009

- Each feature map forms a 2D grid of features
  - can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped



- x_i is the i[th] channel of input
- k_{ij} is the convolution kernel
- g_j is a learned scaling factor
- y_j is the hidden layer

(could have added a bias)

$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

COMPUTER VISION

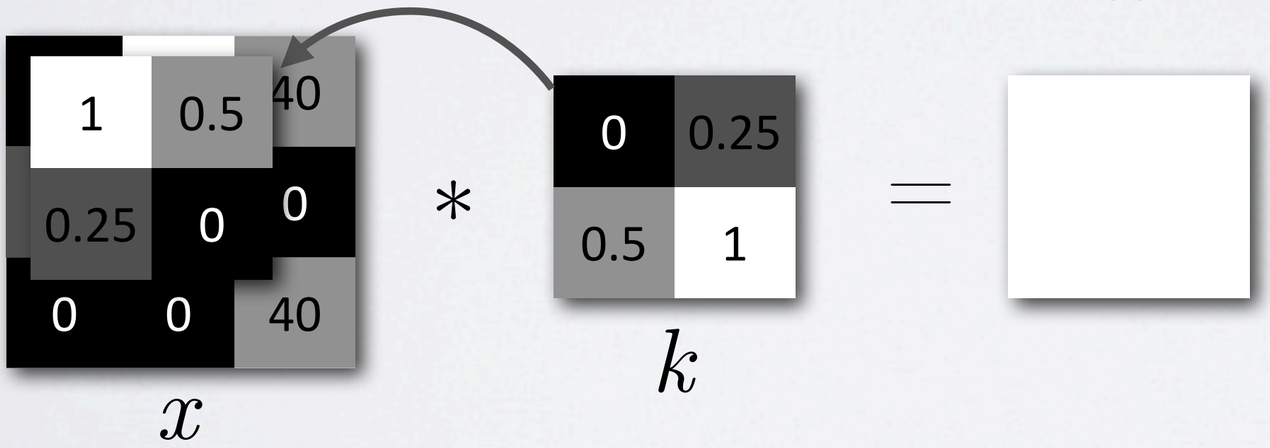
Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:

$\tilde{k} = k$ with rows and columns flipped



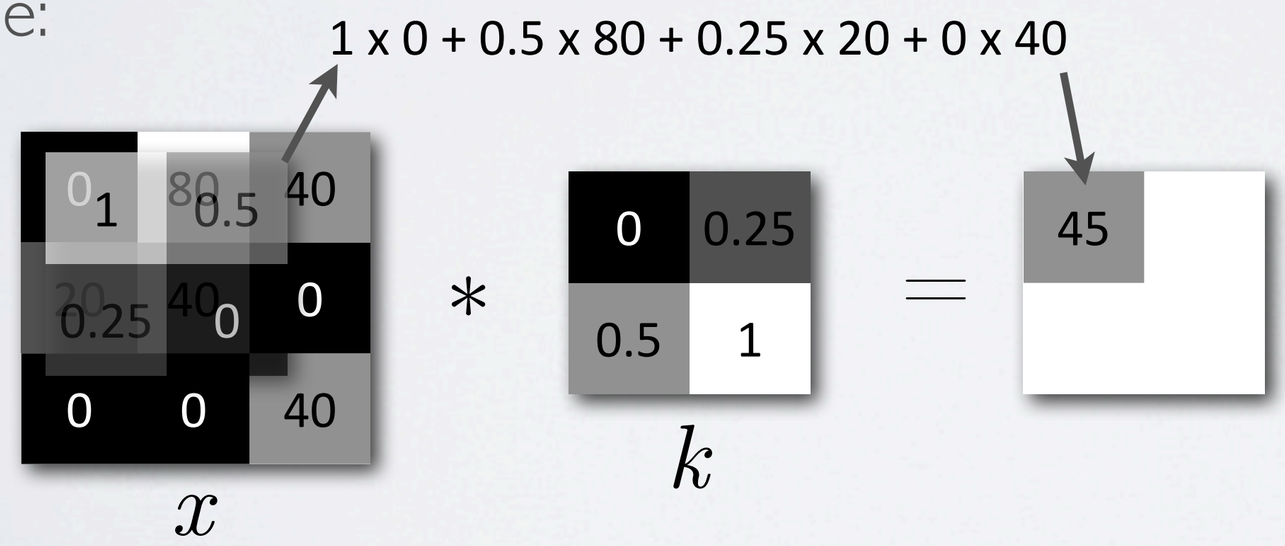
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



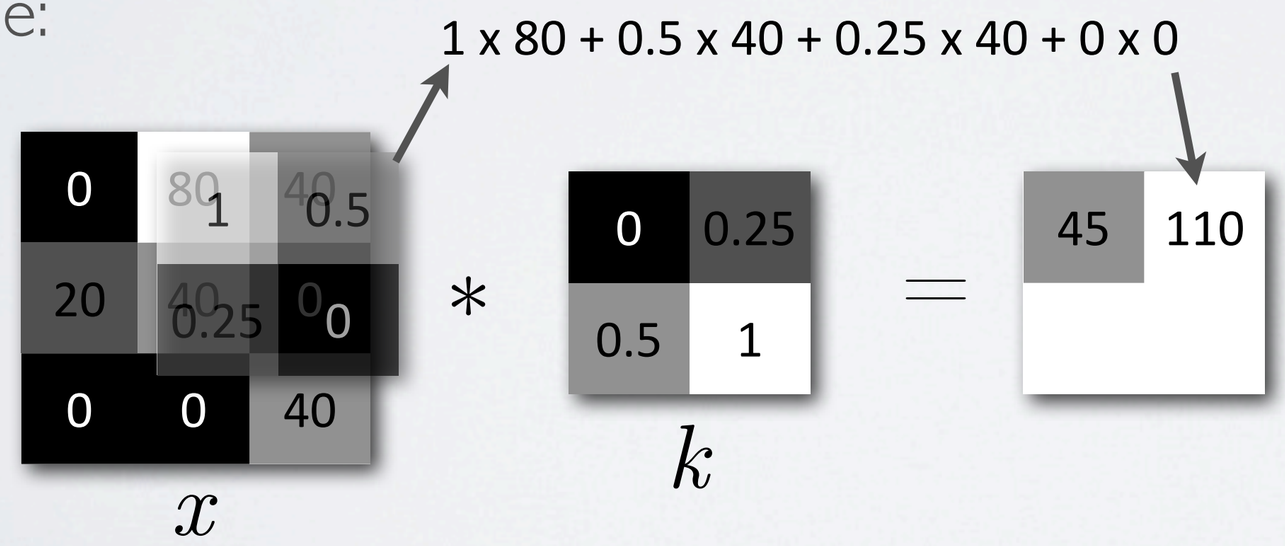
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



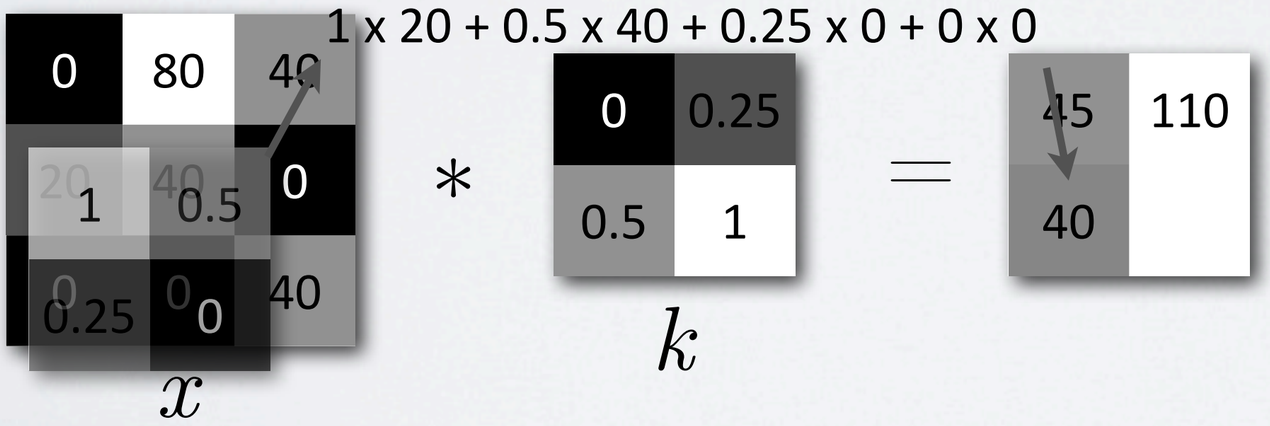
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



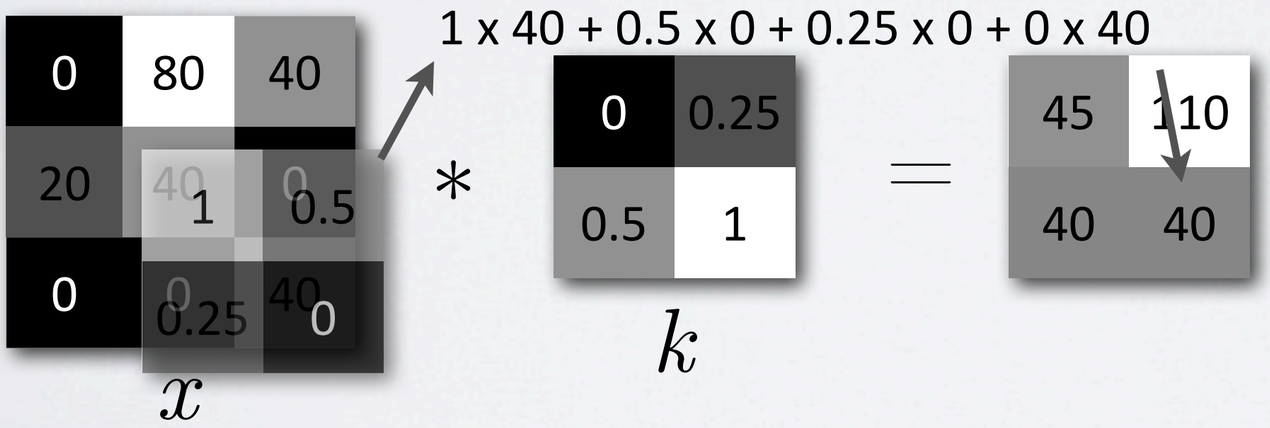
COMPUTER VISION

Topics: discrete convolution

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \; k_{r-p,r-q}$$

- Example:



COMPUTER VISION

Topics: discrete convolution

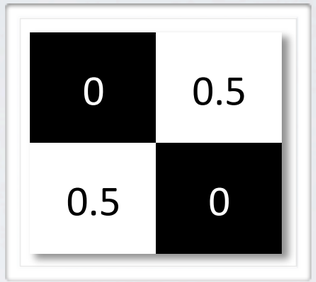
- Pre-activations from channel x_i into feature map y_j can be computed by:
  - getting the convolution kernel where $k_{ij} = \widetilde{W}_{ij}$ from the connection matrix W_{ij}
  - applying the convolution $x_i * k_{ij}$

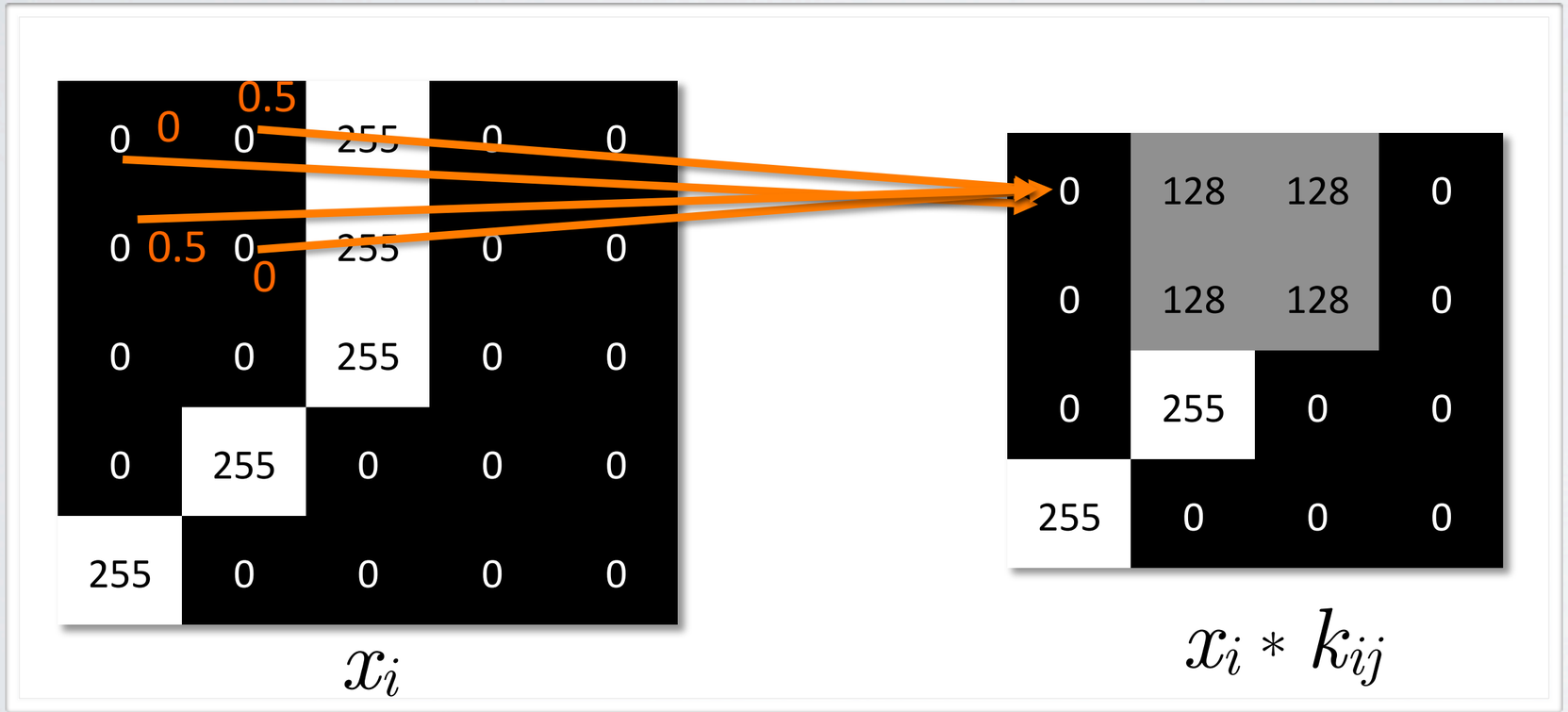
- This is equivalent to computing the discrete correlation of x_i with W_{ij}

COMPUTER VISION

Topics: discrete convolution

- Simple illustration: $x_i * k_{ij}$ where $W_{ij} = \tilde{W}_{ij}$

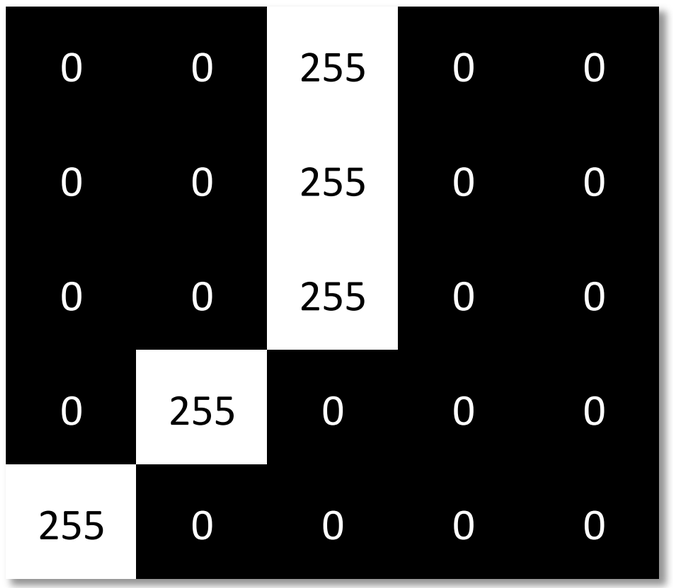




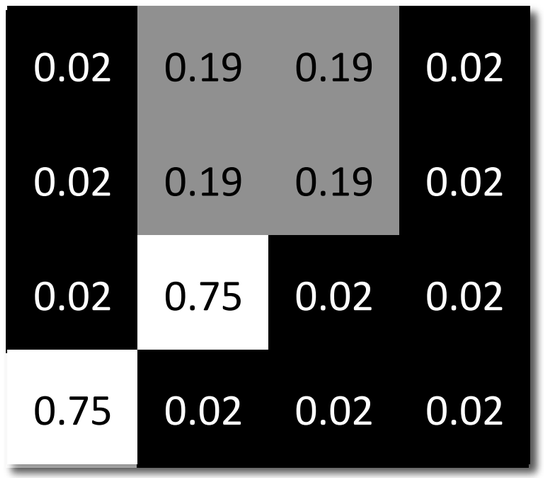
COMPUTER VISION

Topics: discrete convolution

- With a non-linearity, we get a detector of a feature at any position in the image



x_i



$$\mathrm{sigm}(0.02\ x_i * k_{ij} \text{-}4)$$

COMPUTER VISION

Topics: discrete convolution

- Can use "zero padding" to allow going over the borders ($\underline{*}$)