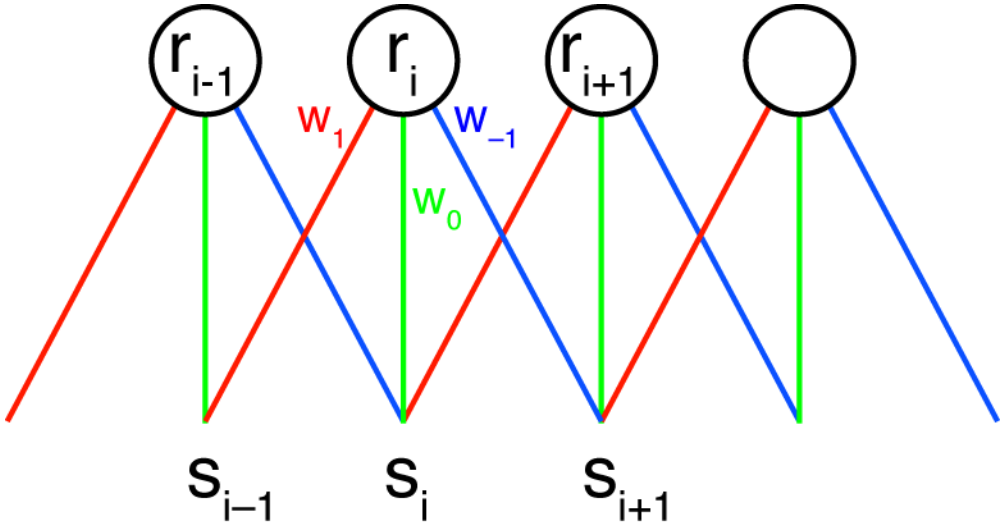
Convolutional networks

Sebastian Seung

Convolutional network

- Neural network with spatial organization
  - every neuron has a location
  - usually on a grid
- Translation invariance
  - synaptic strength depends on locations only through spatial separation.
- Locality
  - synapses only between nearby neurons.

Locality and translation invariance in 1d



$$r_i = f\left(w_1 s_{i-1} + w_0 s_i + w_{-1} s_{i+1}\right)$$

Regular vs. convolutional network

- "stimulus" s and "response" r in 1d

$$r_i = f\left(\sum_j W_{ij} s_j\right) \qquad r_i = f\left(\sum_j w_{i-j} s_j\right)$$

- constrained weight matrix $W_{ij} = w_{i-j}$
- w has local support

Toeplitz matrix

- For each diagonal, the elements are identical.

$$\begin{pmatrix} \ddots & \ddots & \ddots & & \\ \ddots & w_0 & w_{-1} & w_{-2} & \\ \ddots & w_1 & w_0 & w_{-1} & \ddots \\ & w_2 & w_1 & w_0 & \ddots \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

Convolution + nonlinearity

- linear filtering operation

$$\left(\mathbf{w} * \mathbf{s}\right)_i = \sum_j w_{i-j} s_j = \sum_j w_j s_{i-j}$$

- nonlinear transformation

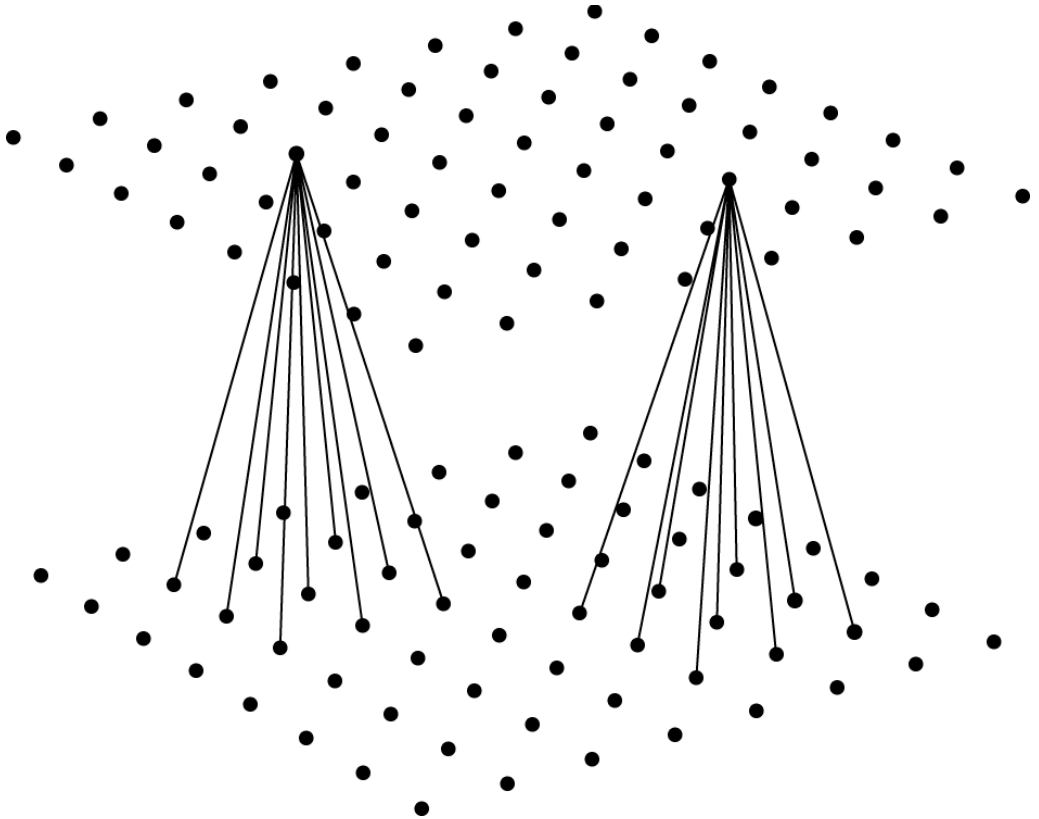
$$r_i = f\left(\left(w * s\right)_i\right)$$

Translation invariance: definition

- Operand
  - does not change when the translation operator is applied
- Operator
  - commutes with the translation operator.

Any translation invariant linear operator is a convolution.

Convolutional network (2d)



$$r_{ij} = f\left(\left(\mathbf{w} * \mathbf{s}\right)_{ij}\right)$$

$$\left(\mathbf{w} * \mathbf{s}\right)_{ij} = \sum_{kl} w_{i-k,j-l} s_{kl} = \sum_{kl} w_{kl} s_{i-k,j-l}$$

Convolution in d dimensions

- Sum over multi-indices
  - points on hypercubic lattice

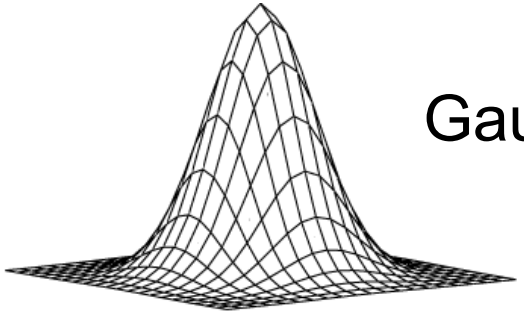
$$\mathbf{i} = (i_1, i_2, \ldots, i_d) \quad \mathbf{j} = (j_1, j_2, \ldots, j_d)$$

$$(\mathbf{w} * \mathbf{s})_{\mathbf{i}} = \sum_{\mathbf{j}} w_{\mathbf{i}-\mathbf{j}} s_{\mathbf{j}}$$

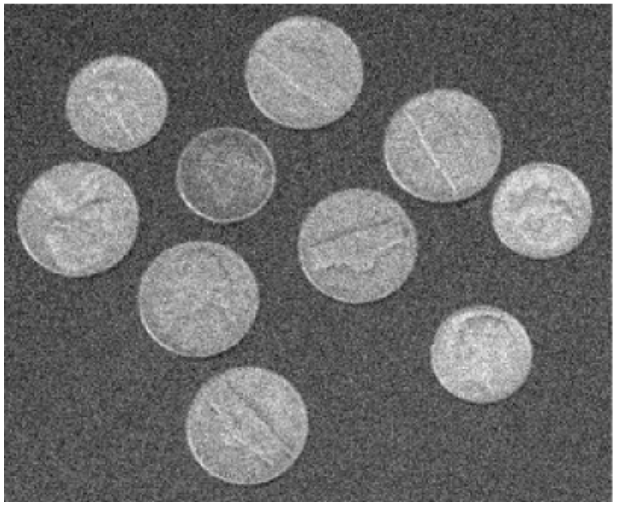
Convolution + nonlinearity

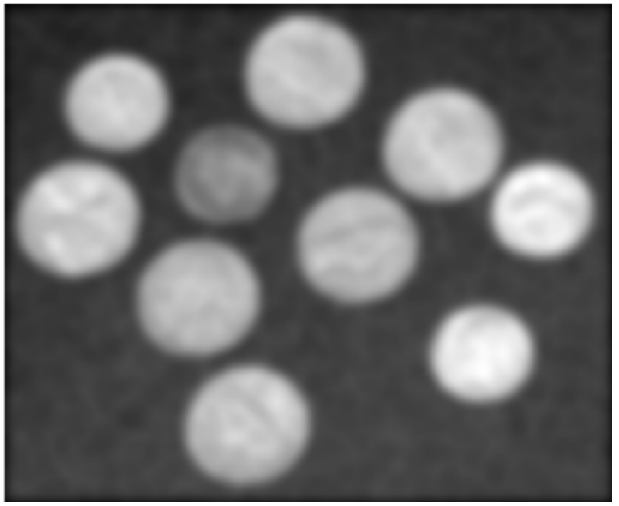
- Can represent basic operations in image processing.
- Can be used to model the first stages of the visual system.

Noise suppression

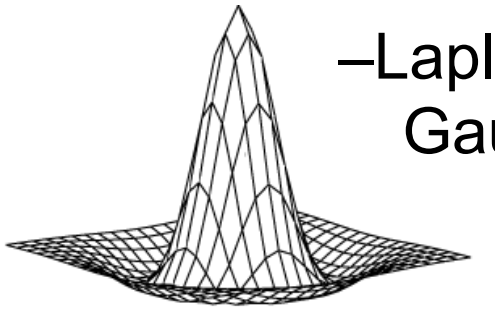


Gaussian filtering





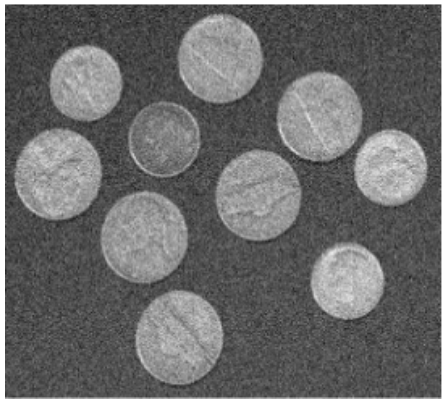
Edge detection

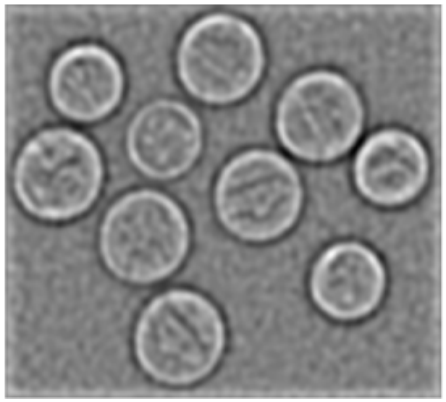


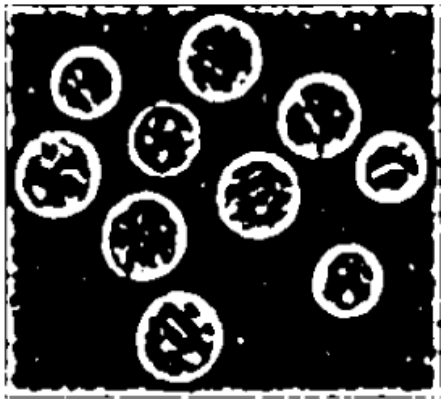
–Laplacian of Gaussian

filtering

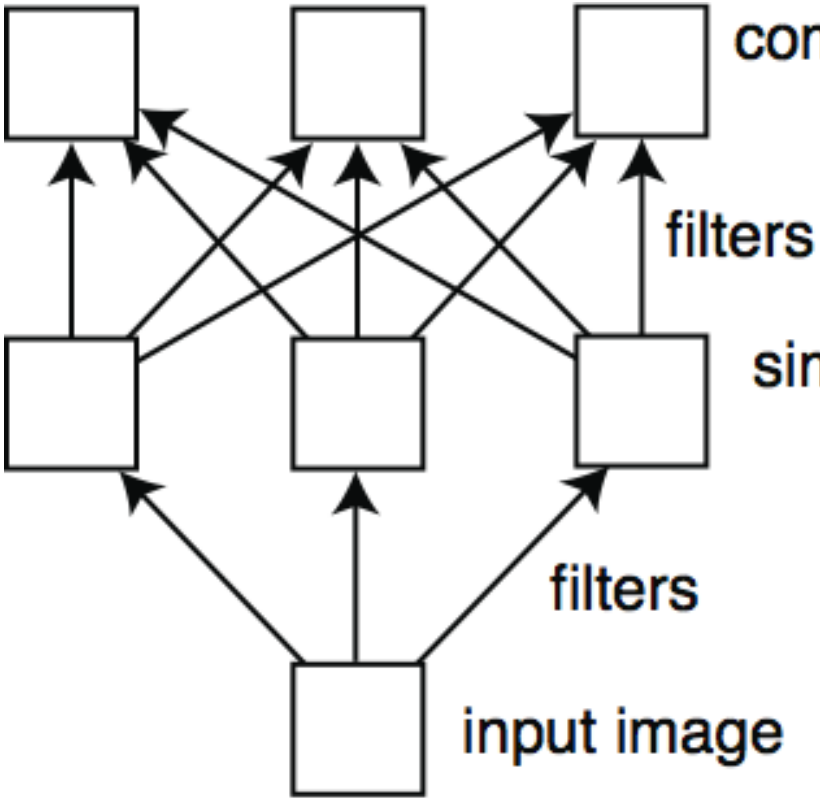
thresholding







Convolutional network



$$\mathbf{I}^{\alpha} = \mathbf{f}\left(\sum_{\beta} \mathbf{w}^{\alpha\beta} * \mathbf{I}^{\beta} + b_{\alpha} \right)$$

Feature map

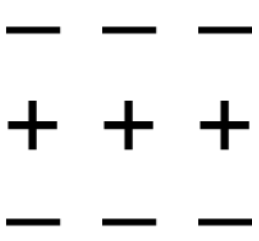
- array of detectors of the same feature
- multiple feature maps in single layer
  - each neuron has a location and a feature index.

$$I_i^{\alpha} = f\left(\sum_{\beta j} w_{i-j}^{\alpha\beta} I_j^{\beta} + b_{\alpha}\right)$$

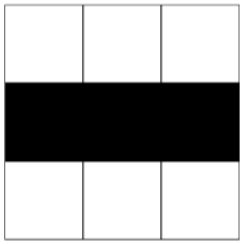
Conjunction neurons as feature detectors

- An image is composed of features.
- Logical conjunction can detect features.

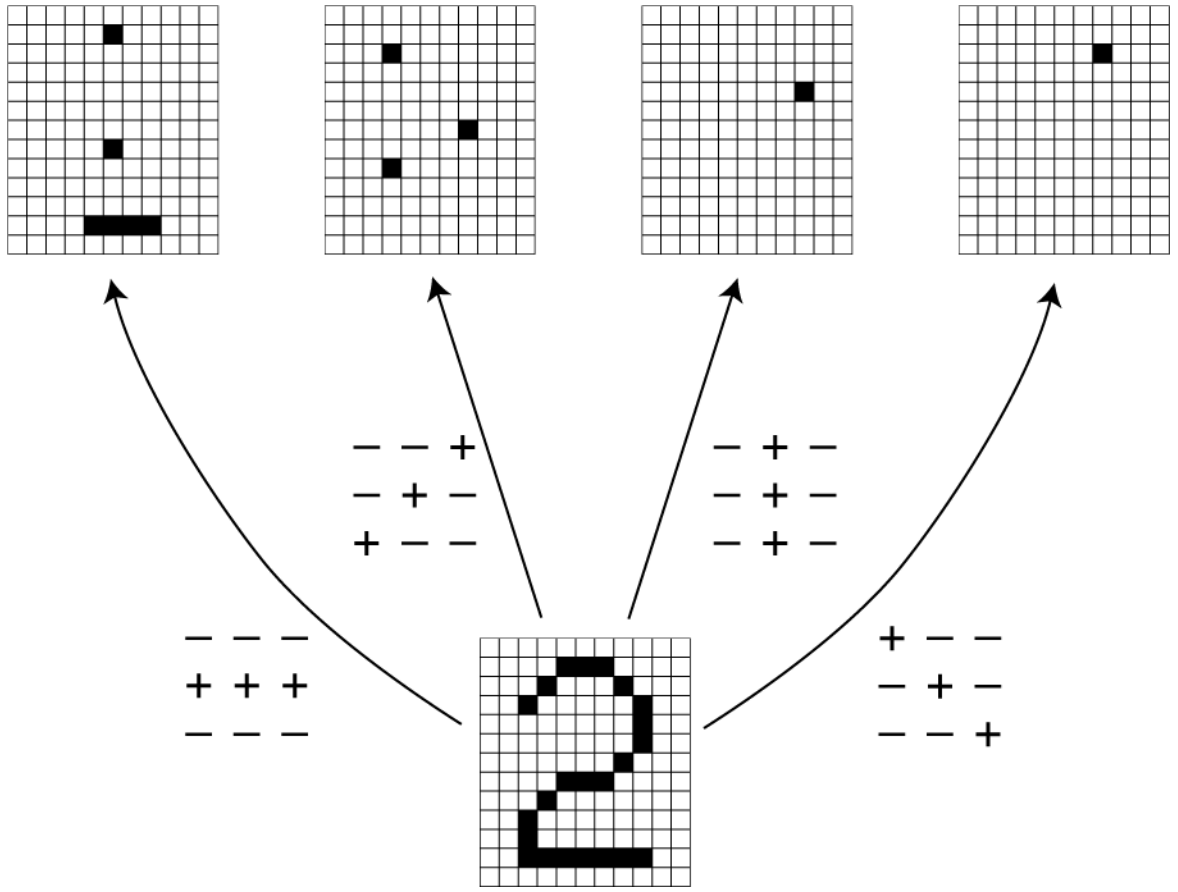
synaptic weights



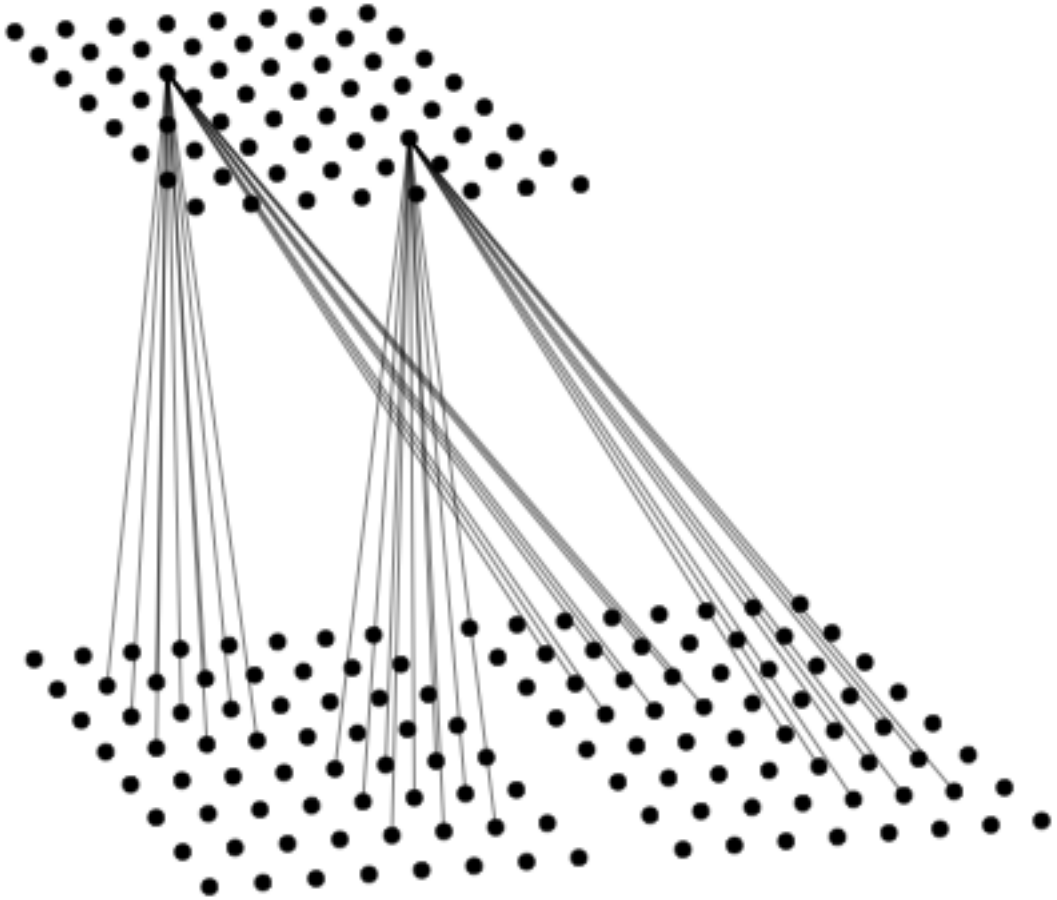
trigger feature

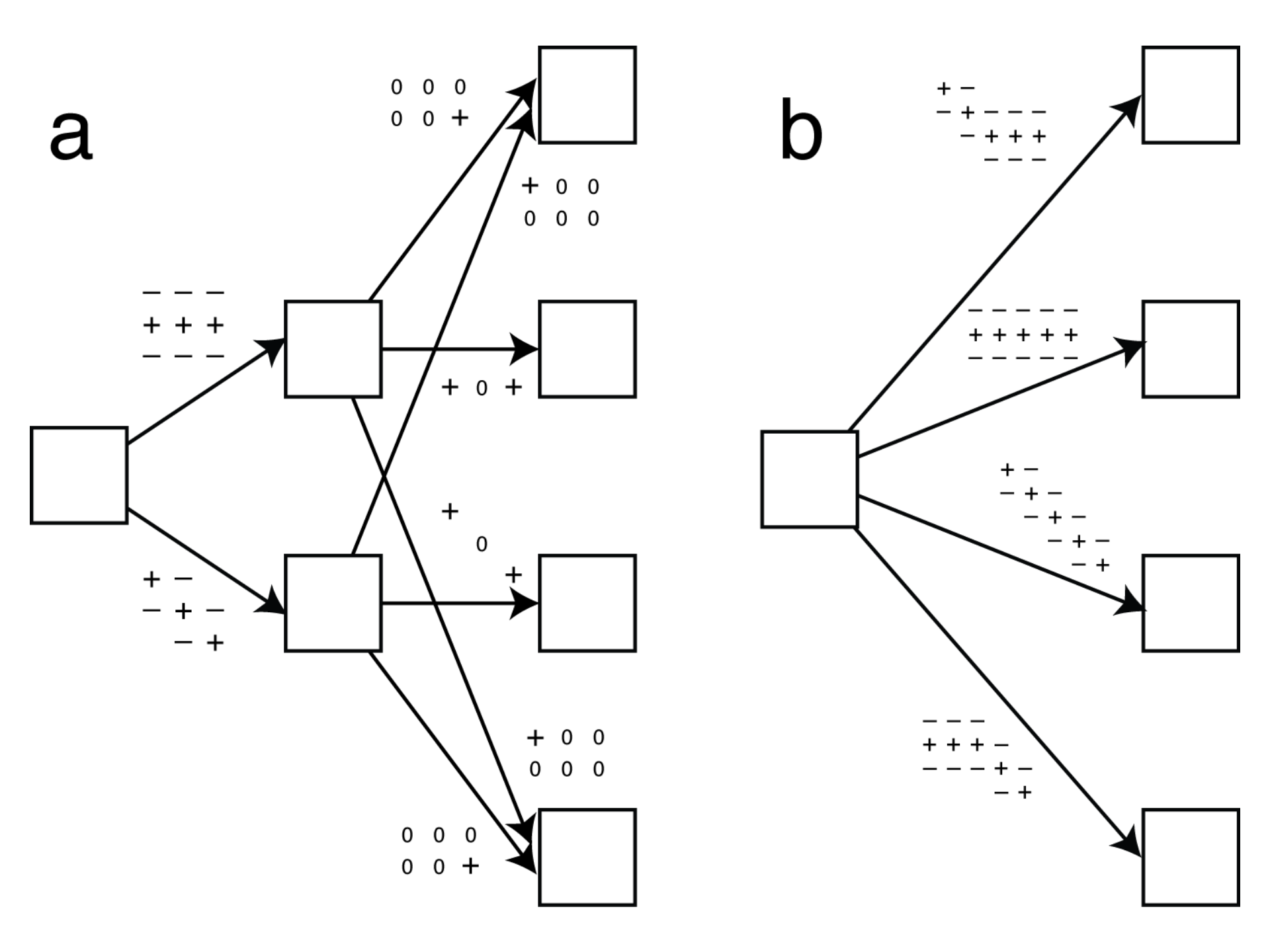


Divergent convolution creates multiple feature maps



Convergent convolution builds detectors for complex features




a
b

Depth-size tradeoff revisited

- Any circuit with two layers of AND gates can be collapsed to a single layer.
- When is the collapsed circuit larger?

The deeper network is more efficient when:

A. Disjunctive normal form is compact.
B. Conjunctive normal form is compact.
C. The fan-out of the hidden layer is large.
D. The fan-in of the hidden layer is large.

Generalize layered structure to directed acyclic graph

- Nodes are feature maps
- Edges are convolutions

Forward pass

$$\begin{aligned} \mathbf{u}^{\alpha} &= \sum_{\beta} \mathbf{w}^{\alpha\beta} * \mathbf{x}^{\beta} + b^{\alpha}\mathbf{1} \\ \mathbf{x}^{\alpha} &= \mathbf{f}(\mathbf{u}^{\alpha}) \end{aligned}$$

Backward pass

$$\hat{x}_j^\beta = \sum_{i\alpha} \delta_i^\alpha w_{i-j}^{\alpha\beta} \quad \hat{\mathbf{x}}^\beta = \sum_\alpha \boldsymbol{\delta}^\alpha * \bar{w}^{\alpha\beta}$$

$$\delta_j^\beta = f'(u_j^\beta)\hat{x}_j^\beta$$

Backward pass

- Convolution is multiplication by a matrix of the form $W_{ij} = w_{i-j}$
- The matrix transpose is

$$W_{ji} = w_{j-i}$$

- Therefore, use spatial inversion of filters
  - i.e., flip the filter about each axis

Weight update

- The weights of neurons in the same feature map are tied together.
- How to perform gradient learning when parameters are shared?

Gradient “sharing”

$$x(t) = t \qquad y(t) = t$$

$$\begin{aligned} \frac{d}{dt} f(x(t), y(t)) &= \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \\ &= \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \end{aligned}$$

Gradient learning with parameter sharing

- Find the derivatives with respect to each of the individual parameters, treating them as independent. Then sum all of these to find the derivative with respect to the shared parameter.

For Whoever Shares, to Him More Gradient Will Be Given

corruption of Mark 4:25

Weight update for a convolution is cross-correlation

$$\Delta W_{ij}^{\alpha\beta} = \eta \delta_i^\alpha x_j^\beta$$

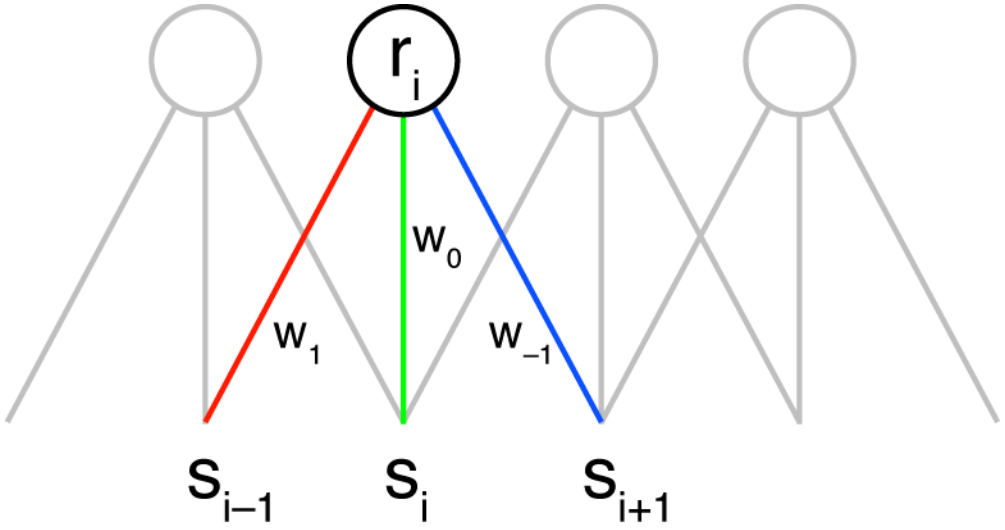
$$\Delta w_k^{\alpha\beta} = \eta \sum_{\substack{i,j \\ i-j=k}} \delta_i^\alpha x_j^\beta$$

$$= \eta \sum_j \delta_{j+k}^\alpha x_j^\beta$$

Receptive field

$$r_i = f\left(w_1 s_{i-1} + w_0 s_i + w_{-1} s_{i+1}\right)$$

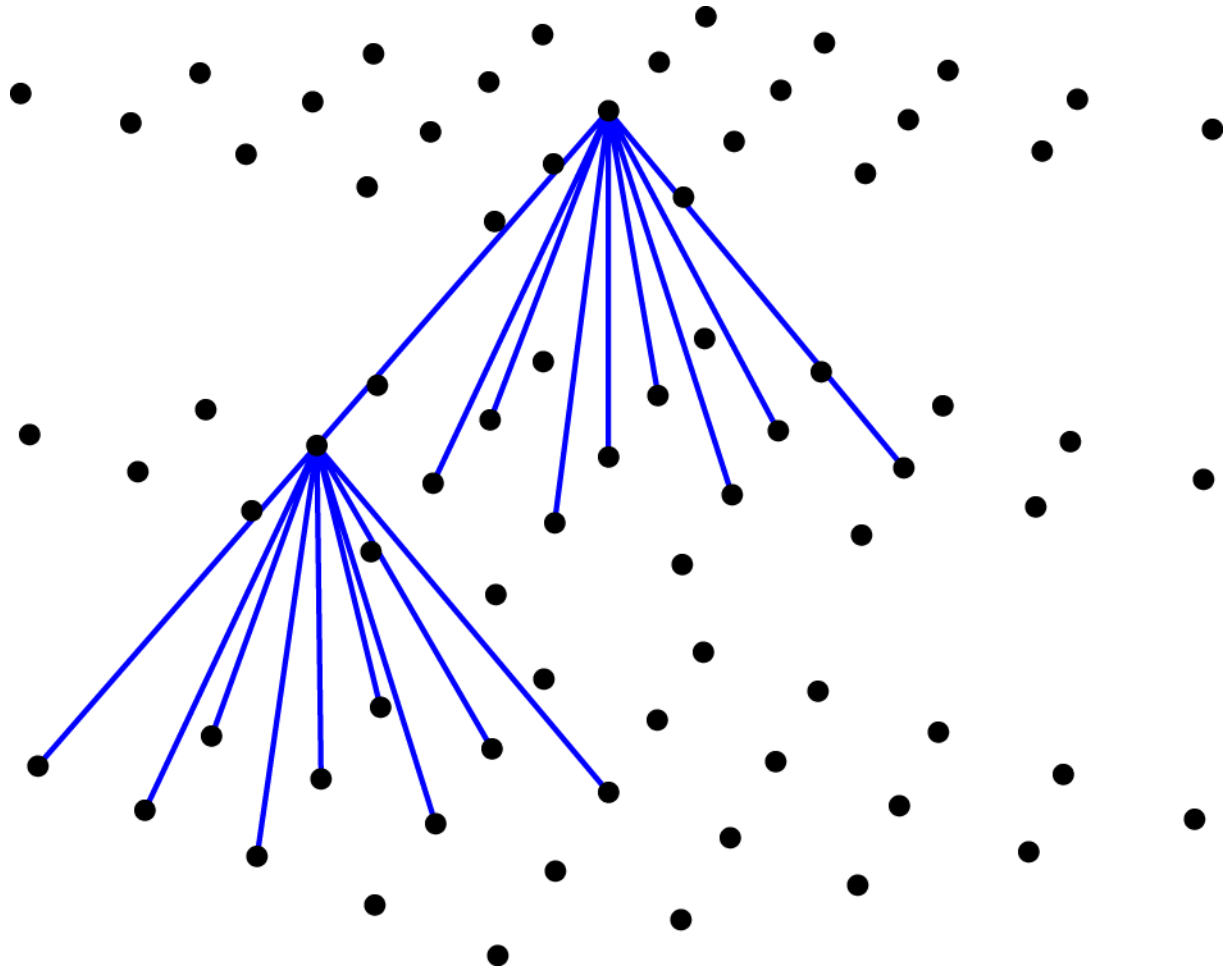
$$\begin{pmatrix} w_1 & w_0 & w_{-1} \end{pmatrix}$$



Receptive field (structural definition)

- The locations of the neurons in the input layer that are connected to a neuron
- Paths of one or more synapses.

Receptive field size increases



After L convolutions with a $k \times k$ filter, how large is the receptive field?

A. $(Lk – L + 1) \times (Lk – L + 1)$
B. $(Lk – 1) \times (Lk – 1)$
C. $(2k – 1) \times (2k – 1)$
D. $(L + k) \times (L + k)$
E. $(Lk – k + 2) \times (Lk – k + 2)$

Projective field

$$r_i = f\left(w_1 s_{i-1} + w_0 s_i + w_{-1} s_{i+1}\right)$$

$$\begin{pmatrix} w_{-1} & w_0 & w_1 \end{pmatrix}$$