



# Jaringan Berbasis Kompetisi

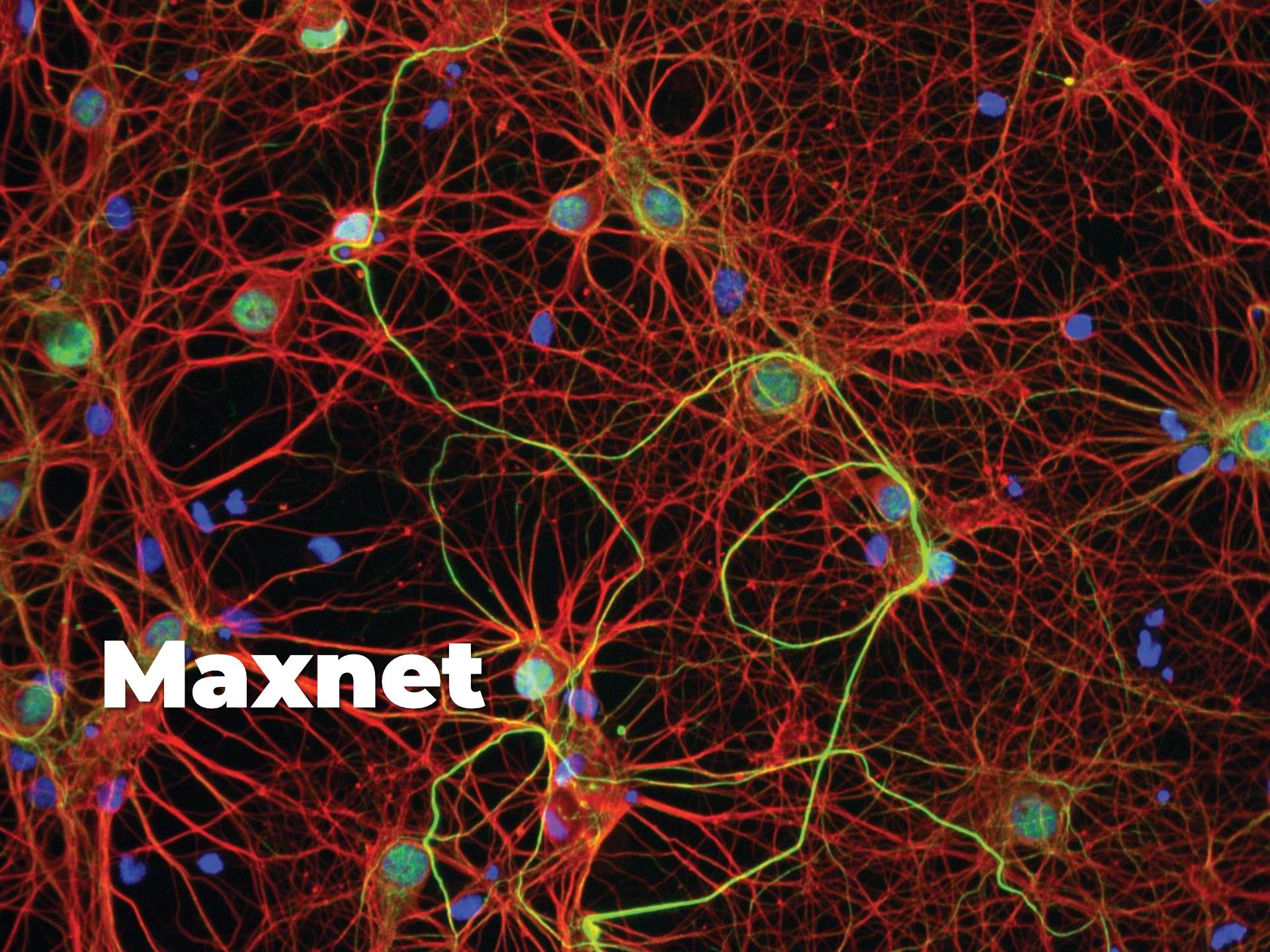
Randy Cahya Wihandika, S.ST., M.Kom.

# Topik Hari Ini

- JST berbasis kompetisi
- Maxnet
- Mexican hat
- Hamming net

# JST Berbasis Kompetisi

- JST dapat memberikan hasil yang ambigu
- Pada kasus klasifikasi, JST dapat memberikan lebih dari satu kelas *output*
- JST berbasis kompetisi memilih satu *output*
- Neuron saling “berkompetisi”
- Mekanismenya sering kali disebut *winner takes all*



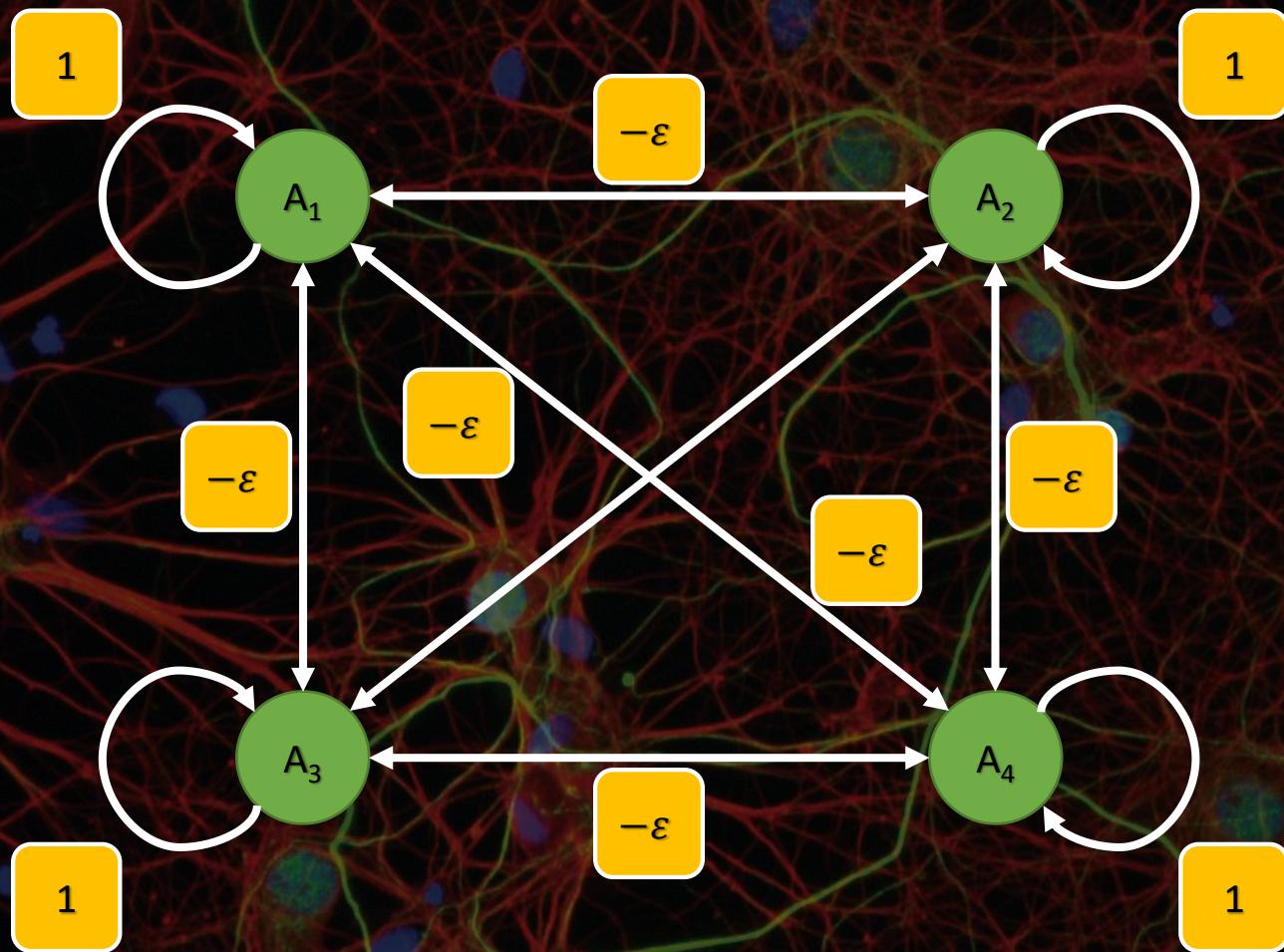
**Maxnet**

# Maxnet

- Lippmann (1987)
- Semua neuron saling terhubung
- Tidak ada proses pelatihan; nilai bobot tetap
- Fungsi aktivasi:

$$f(x) = \begin{cases} x & \text{jika } x \geq 0 \\ 0 & \text{lainnya} \end{cases}$$

# Arsitektur Maxnet



# Algoritme Maxnet

- I. Set nilai  $\varepsilon: 0 < \varepsilon < \frac{1}{m}$ ,  $m$  adalah banyaknya neuron

Set nilai bobot:  $w_{ij} = \begin{cases} 1 & \text{jika } i = j \\ -\varepsilon & \text{jika } i \neq j \end{cases}$

2. Selama kondisi berhenti belum tercapai, lakukan langkah 3–5

# Algoritme Maxnet

3. Hitung nilai aktivasi setiap neuron:

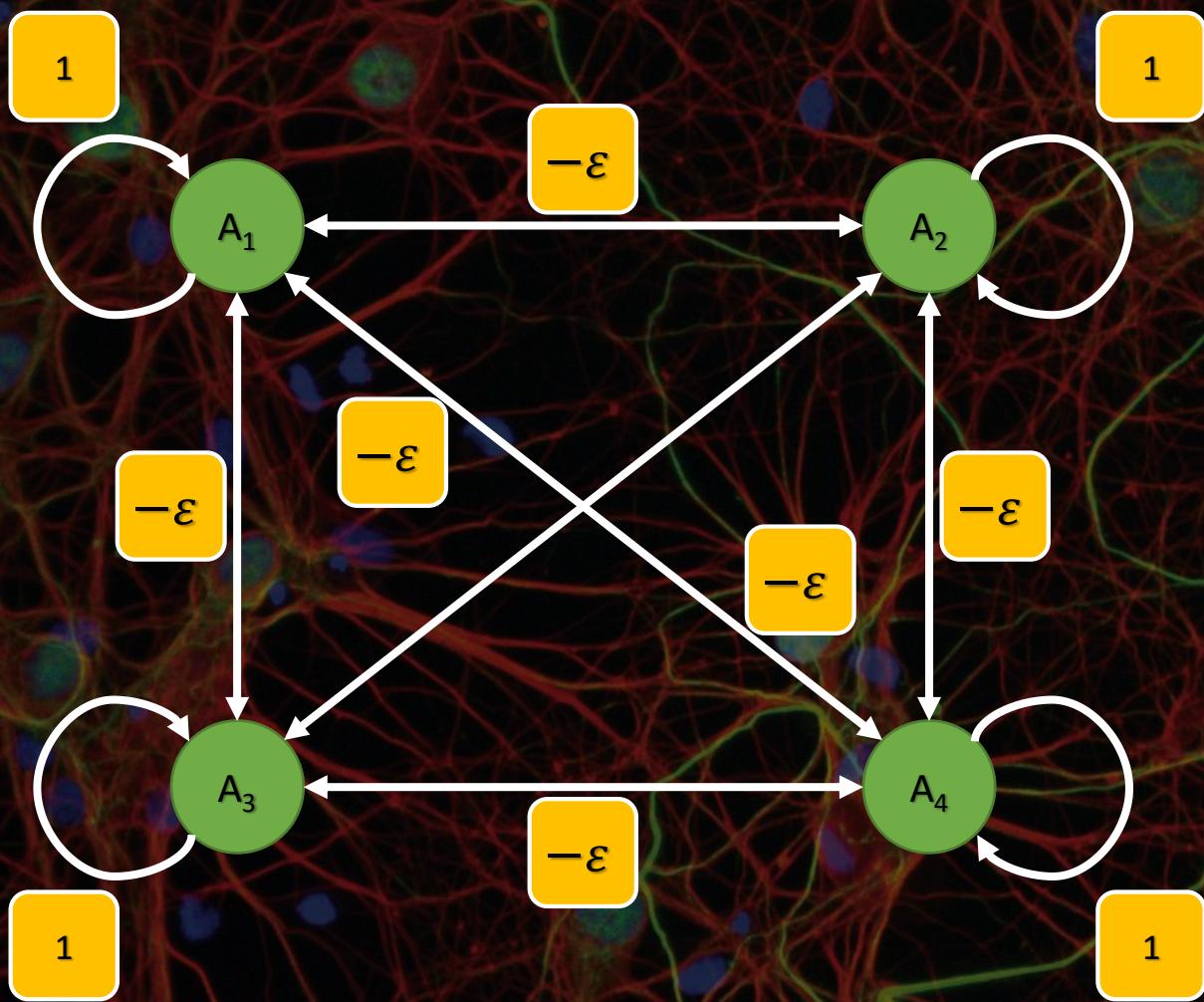
$$a'_j = f \left[ a_j - \varepsilon \sum_{k \neq j} a_k \right]$$

4. Ubah nilai aktivasi lama dengan yang baru:

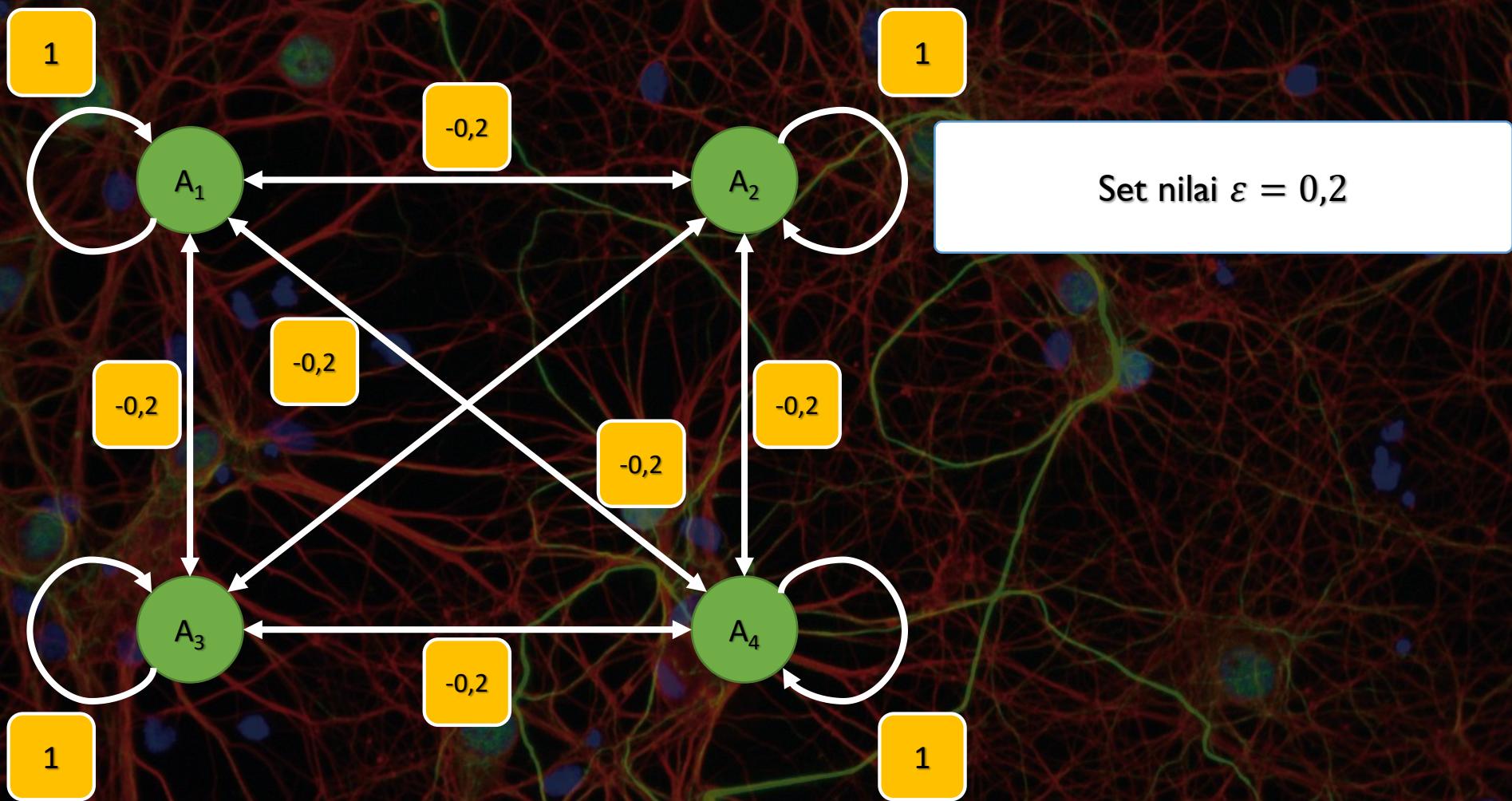
$$a_j = a'_j$$

5. Lanjutkan proses hingga hanya ada satu neuron dengan nilai aktivasi bukan nol

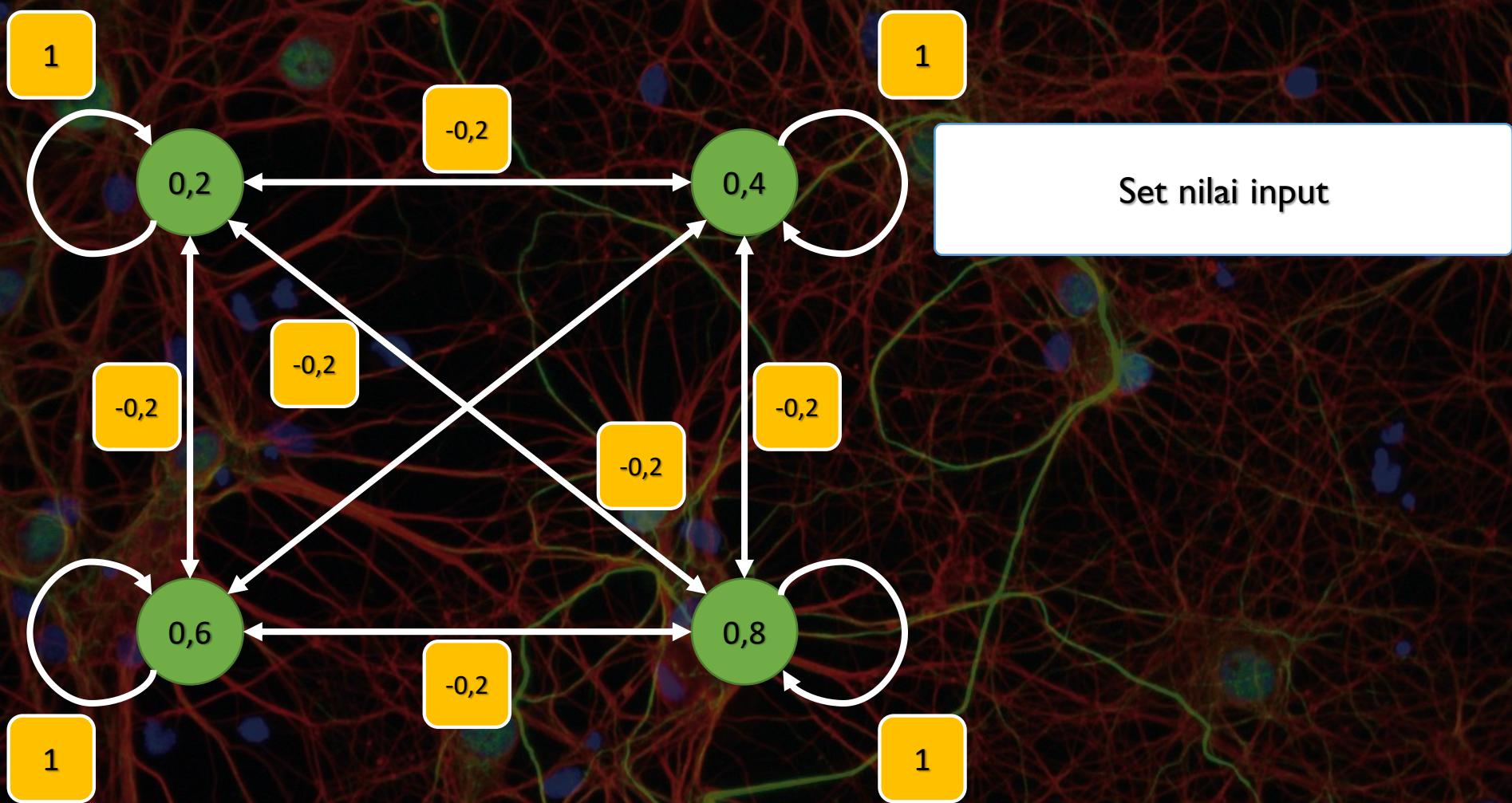
# Contoh



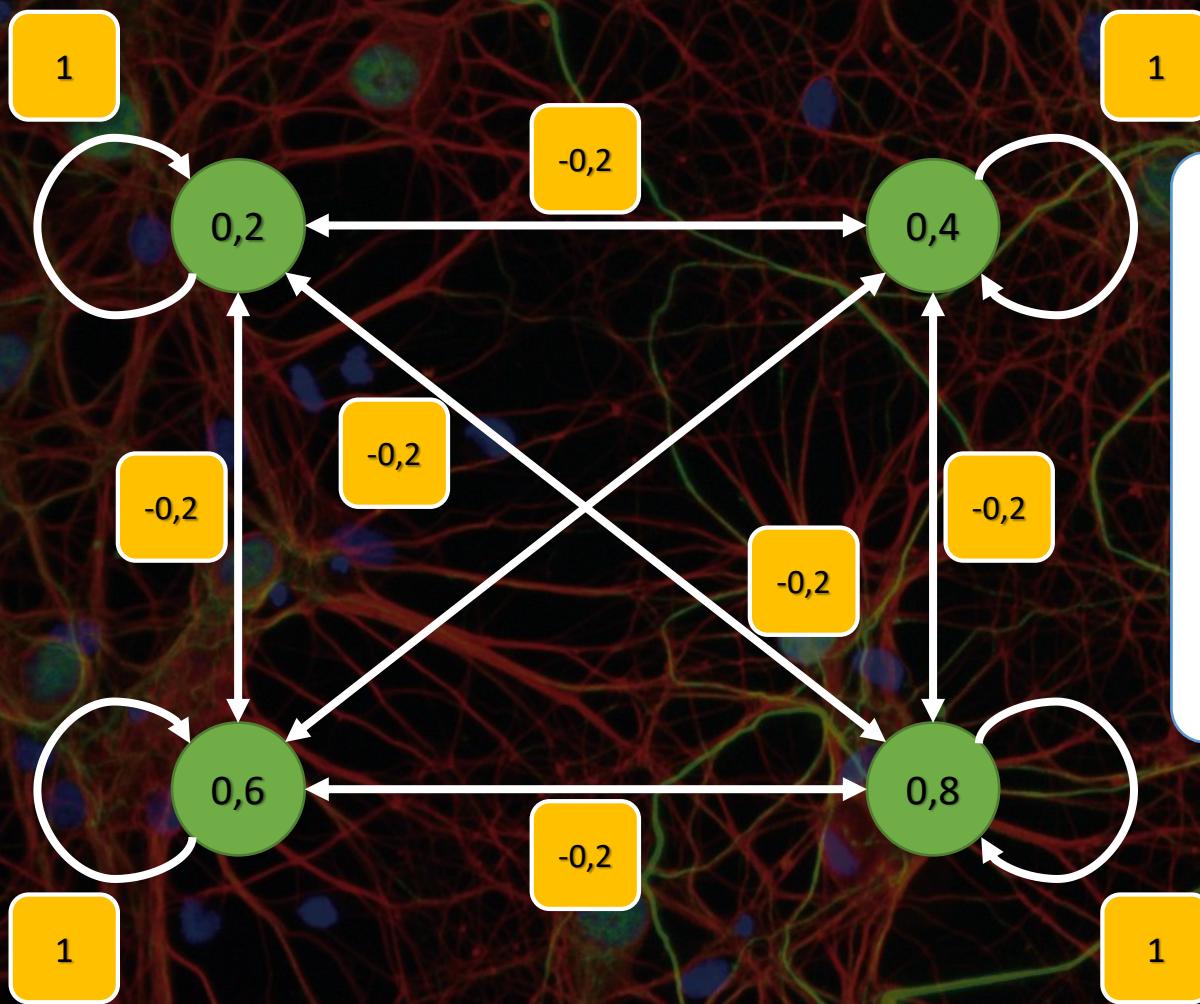
# Contoh



# Contoh

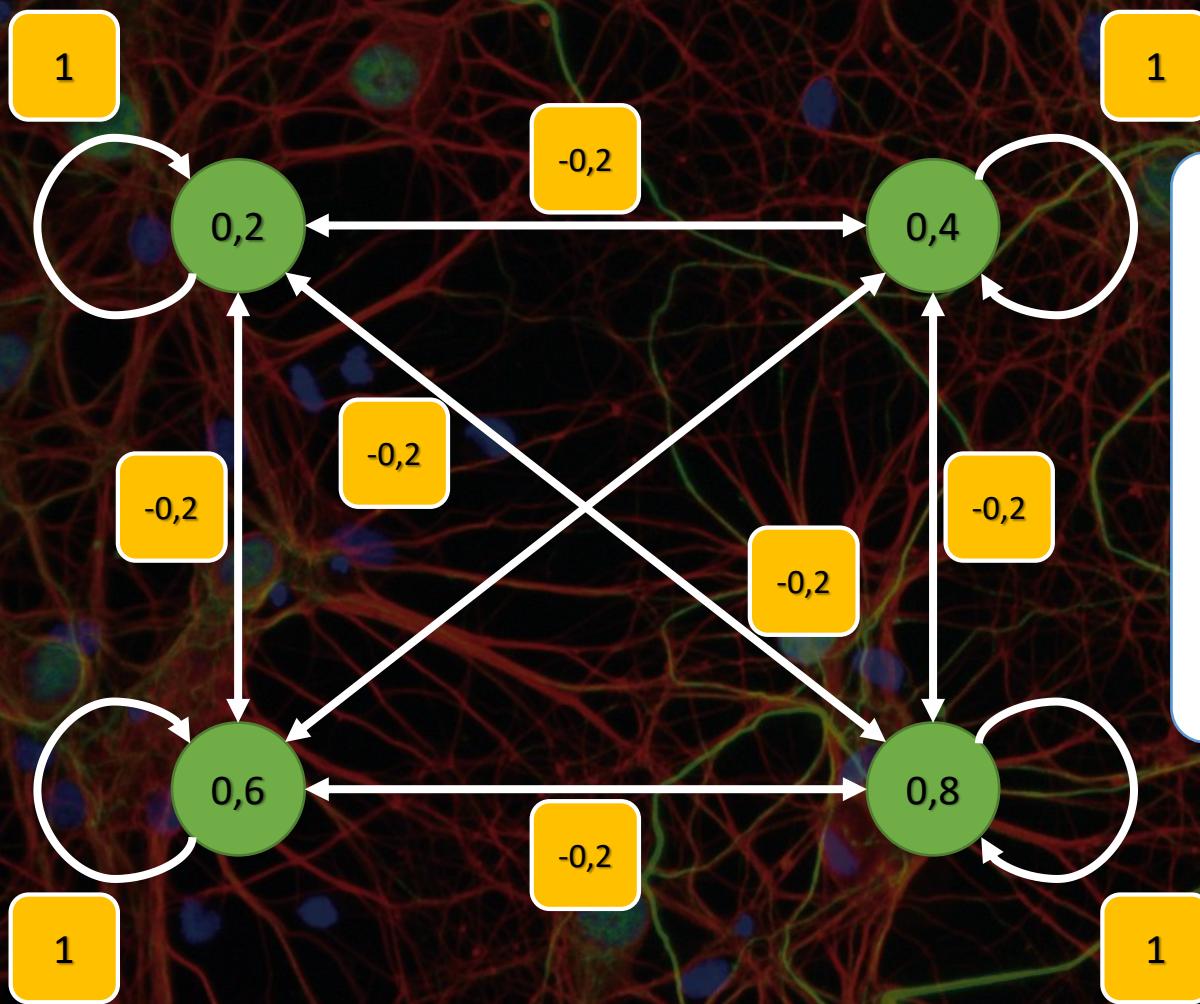


# Contoh



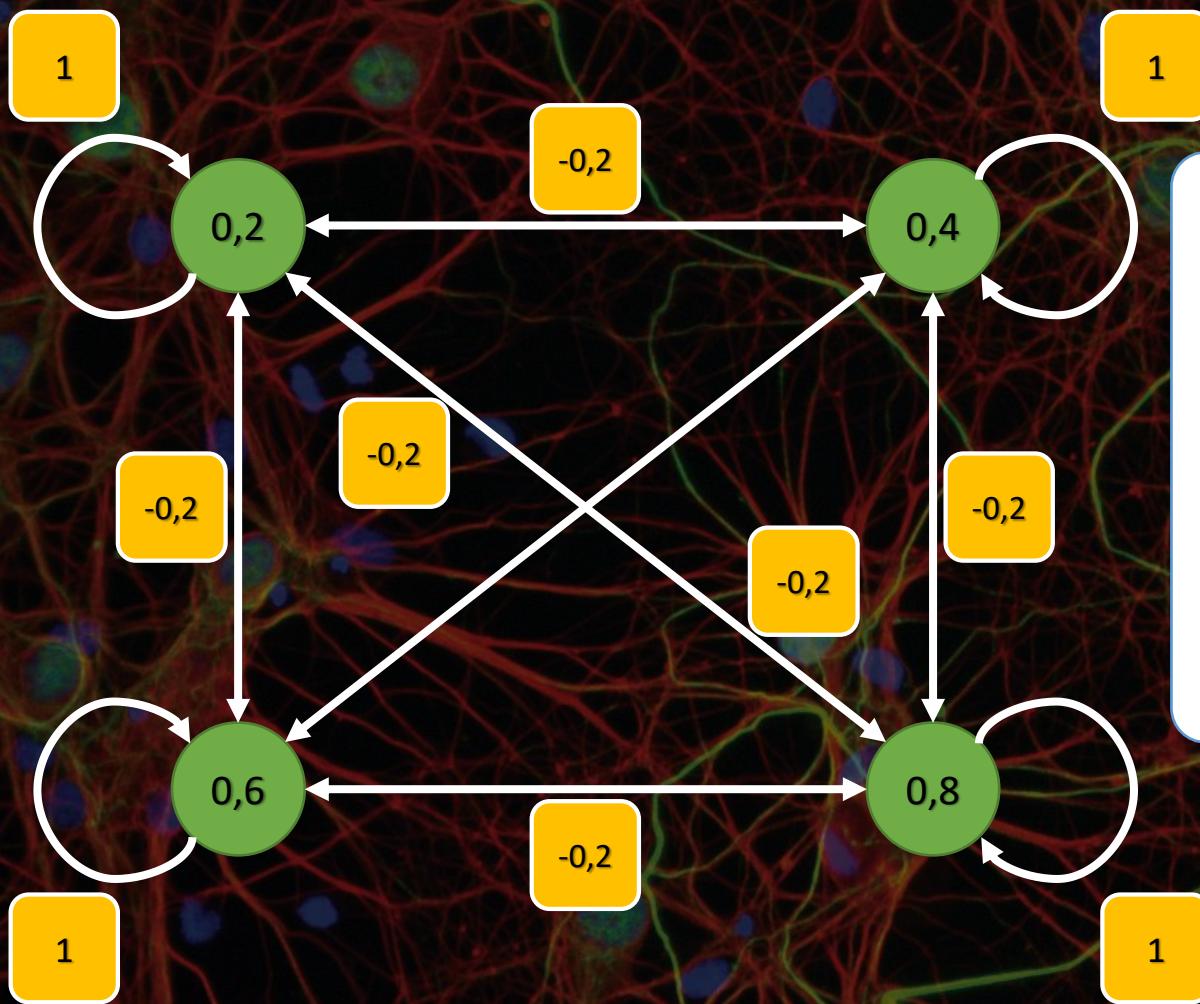
$$\begin{aligned}a'_j &= f \left[ a_j - \varepsilon \sum_{k \neq j} a_k \right] \\a'_1 &= f[a_1 - \varepsilon(a_2 + a_3 + a_4)] \\&= f[0,2 - 0,2(0,4 + 0,6 + 0,8)] \\&= f(0,2 - 0,36) \\&= f(-0,16) \\&= 0\end{aligned}$$

# Contoh



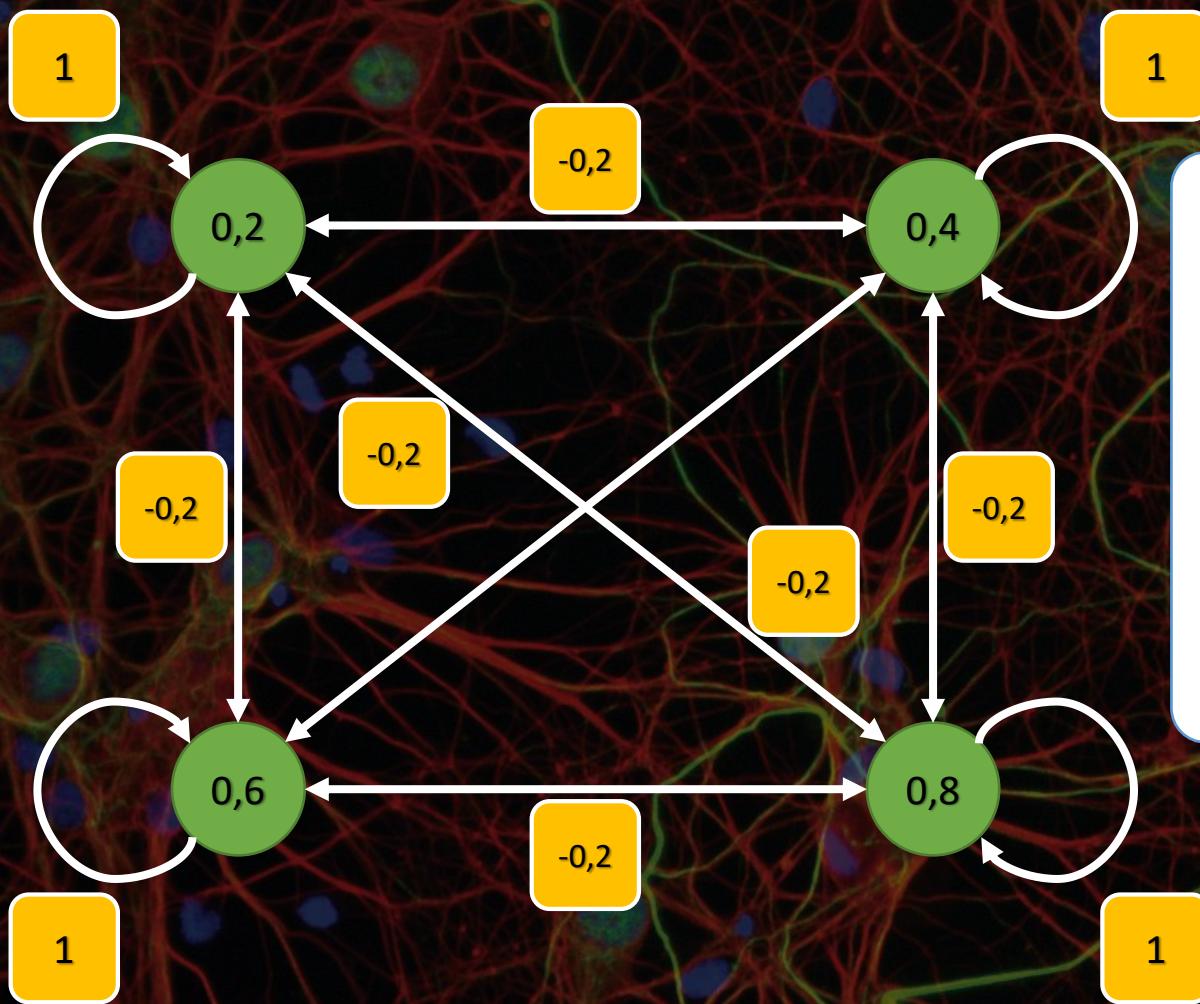
$$\begin{aligned}a'_j &= f \left[ a_j - \varepsilon \sum_{k \neq j} a_k \right] \\a'_2 &= f[a_2 - \varepsilon(a_1 + a_3 + a_4)] \\&= f[0,4 - 0,2(0,2 + 0,6 + 0,8)] \\&= f(0,4 - 0,32) \\&= f(0,08) \\&= 0,08\end{aligned}$$

# Contoh



$$\begin{aligned}a'_j &= f \left[ a_j - \varepsilon \sum_{k \neq j} a_k \right] \\a'_3 &= f[a_3 - \varepsilon(a_1 + a_2 + a_4)] \\&= f[0,6 - 0,2(0,2 + 0,4 + 0,8)] \\&= f(0,6 - 0,28) \\&= f(0,32) \\&= 0,32\end{aligned}$$

# Contoh



$$\begin{aligned}a'_j &= f \left[ a_j - \varepsilon \sum_{k \neq j} a_k \right] \\a'_4 &= f[a_4 - \varepsilon(a_1 + a_2 + a_3)] \\&= f[0,8 - 0,2(0,2 + 0,4 + 0,6)] \\&= f(0,8 - 0,24) \\&= f(0,56) \\&= 0,56\end{aligned}$$

# Contoh

Iterasi	A1	A2	A3	A4
1	0,0	0,08	0,32	0,56
2	0,0	0,0	0,192	0,48
3	0,0	0,0	0,096	0,442
4	0,0	0,0	0,008	0,422
5	0,0	0,0	0,0	0,421

# Implementasi Maxnet

```
import numpy as np

def act(x):
    return x if x >= 0 else 0

def maxnet(a, e=None):
    if e is None:
        e = np.random.uniform(0, 1 / len(a))

    while np.count_nonzero(a) > 1:
        a_new = np.zeros(len(a))

        for i in range(len(a)):
            s = sum([a[j] for j in range(len(a)) if j != i])
            a_new[i] = act(a[i] - e * s)

        a = a_new

    return np.argmax(a)
```

# Implementasi Maxnet

```
a = [.2, .4, .6, .8]
e = .2
m = maxnet(a, e)

print(a[m])
```

A fluorescence microscopy image showing a dense network of neurons. The neurons are stained with red, likely for actin filaments, and blue, likely for nuclei or specific markers. Several neurons are highlighted with green outlines, forming a pattern that resembles a Mexican Hat. These outlined neurons have bright green nuclei and green-stained processes. The background is black.

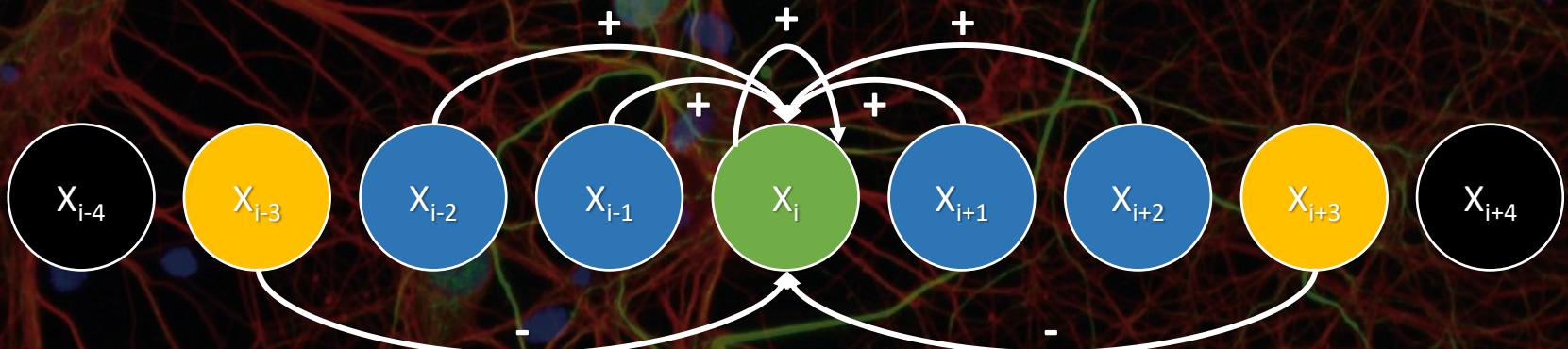
**Mexican Hat**

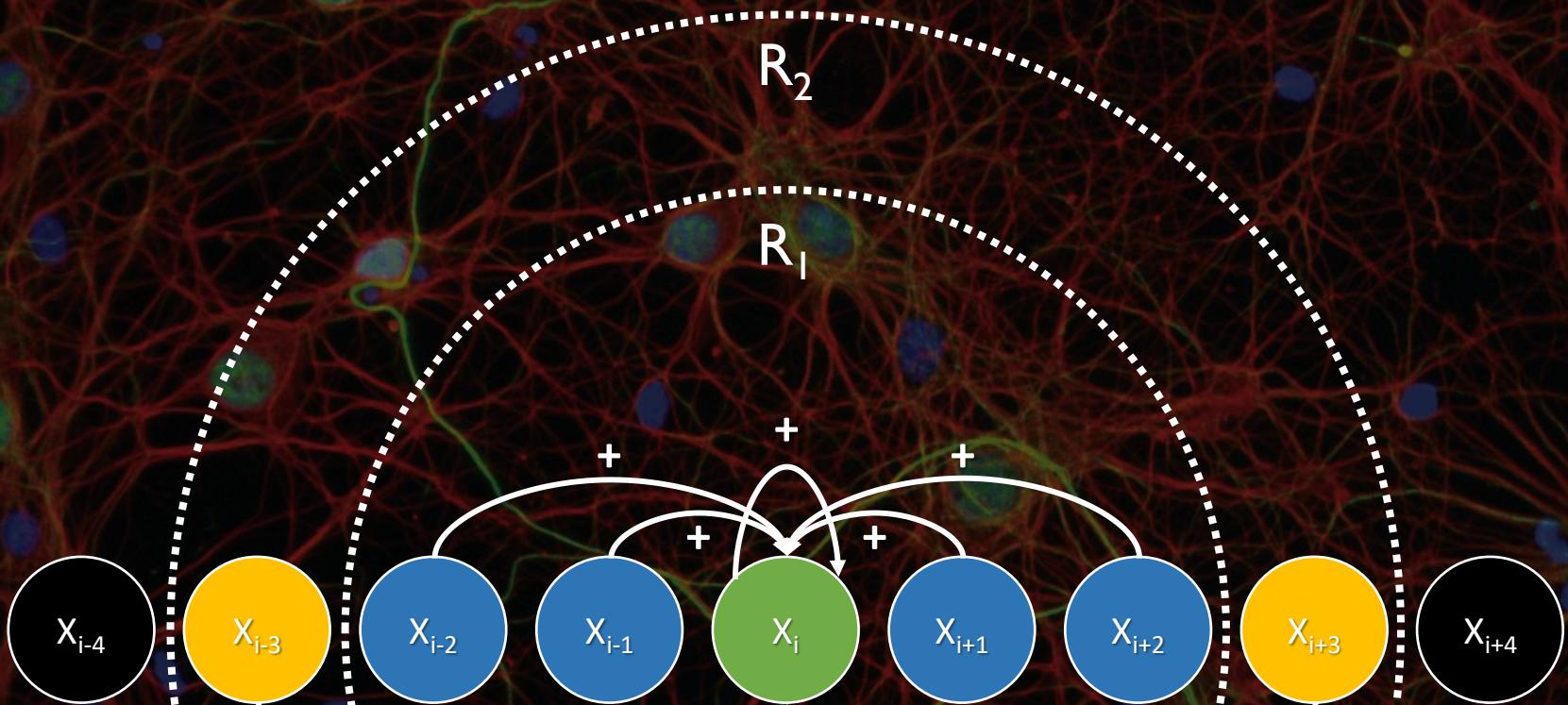
# Mexican Hat

- Bentuk jaringan kompetisi yang lebih umum dibandingkan Maxnet
- Kohonen (1987)

# Mexican Hat

- Neuron yang **dekat** (*cooperative neighbors*) dengan  $X_i$  memiliki bobot **positif**
- Neuron yang **jauh** (*competitive neighbors*) dengan  $X_i$  memiliki bobot **negatif**
- Neuron yang sangat jauh tidak terhubung dengan  $X_i$  (bobot nol)





# Algoritme Mexican Hat

1. Inisialisasi  $t_{max}$  (iterasi maks.),  $R_1$ ,  $R_2$   
Inisialisasi bobot  $C_1$  untuk neuron dalam radius  $R_1$  dan  $C_2$  untuk neuron dalam radius  $R_2$
2. Masukkan input:  
$$x = s$$
3. Selama  $t < t_{max}$ , lakukan langkah 4–8

# Algoritme Mexican Hat

4. Hitung input setiap neuron:

$$x'_i = C_1 \sum_{k=-R_1}^{R_1} x_{i+k} + C_2 \sum_{k=-R_2}^{-R_1-1} x_{i+k} + C_2 \sum_{k=R_1+1}^{R_2} x_{i+k}$$

5. Aplikasikan fungsi aktivasi:

$$x_i = \min(x_{max}, \max(0, x_i))$$

6. Simpan nilai aktivasi pada variabel

# Algoritme Mexican Hat

7. Tambahkan indeks iterasi:

$$t = t + 1$$

8. Jika  $t = t_{max}$ , hentikan proses

# Contoh

- Parameter:

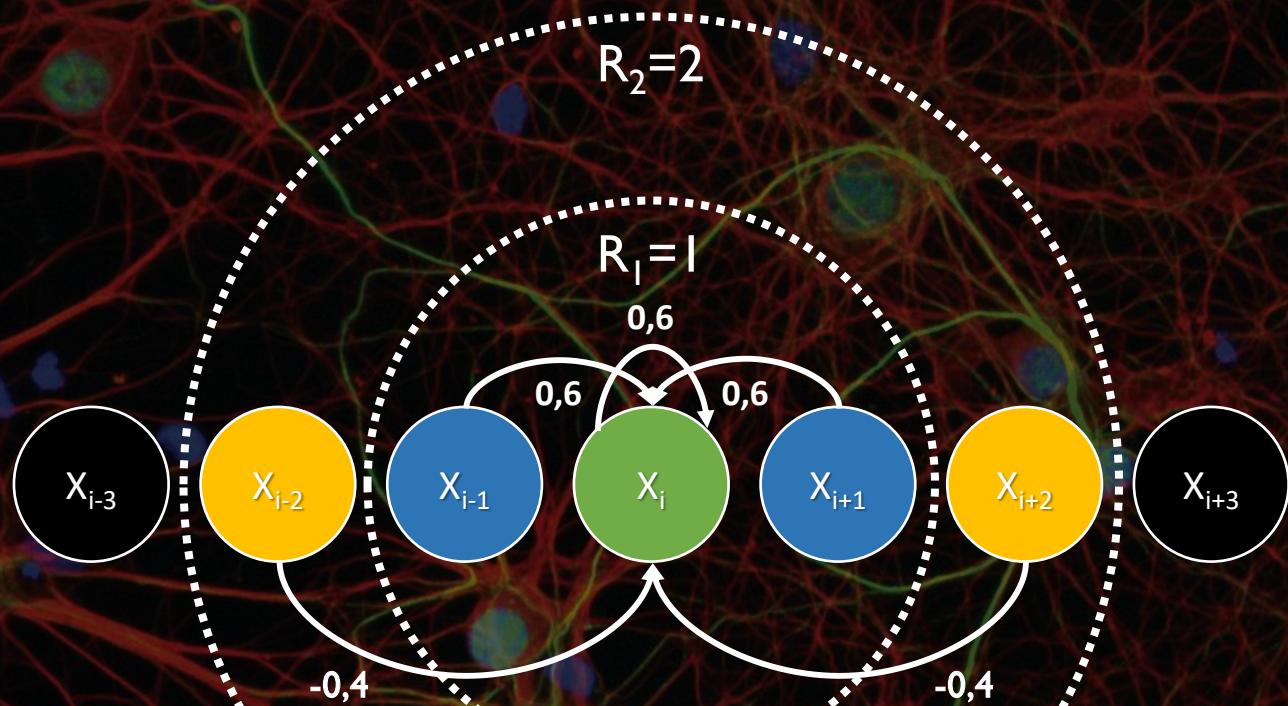
$$R_1 = 1$$

$$R_2 = 2$$

$$C_1 = 0,6$$

$$C_2 = -0,4$$

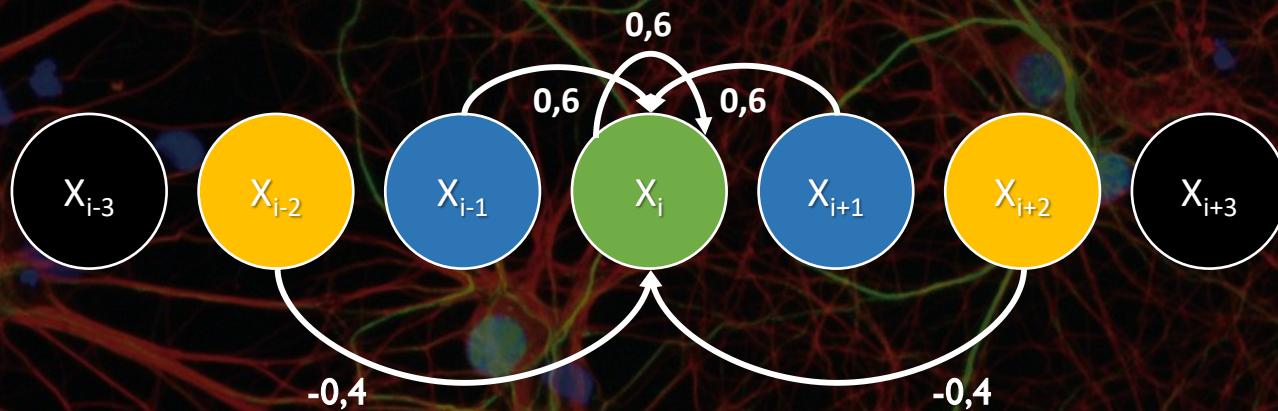
# Contoh



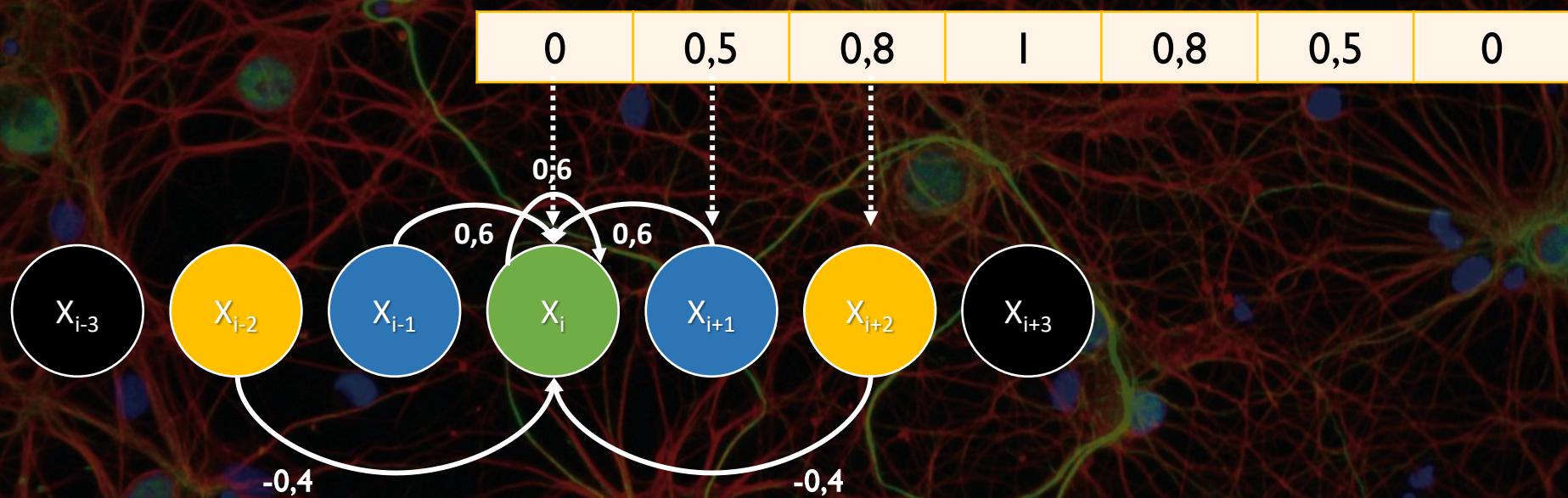
# Contoh

- Input: 

0	0,5	0,8	1	0,8	0,5	0
---	-----	-----	---	-----	-----	---

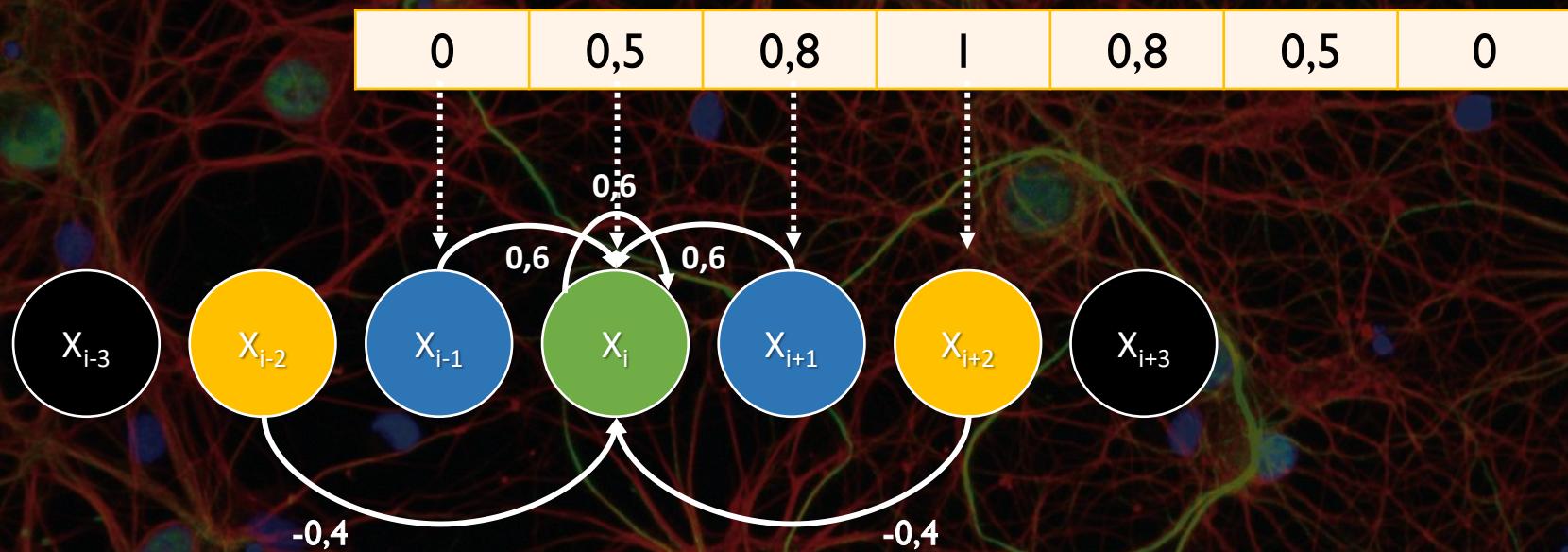


# Contoh



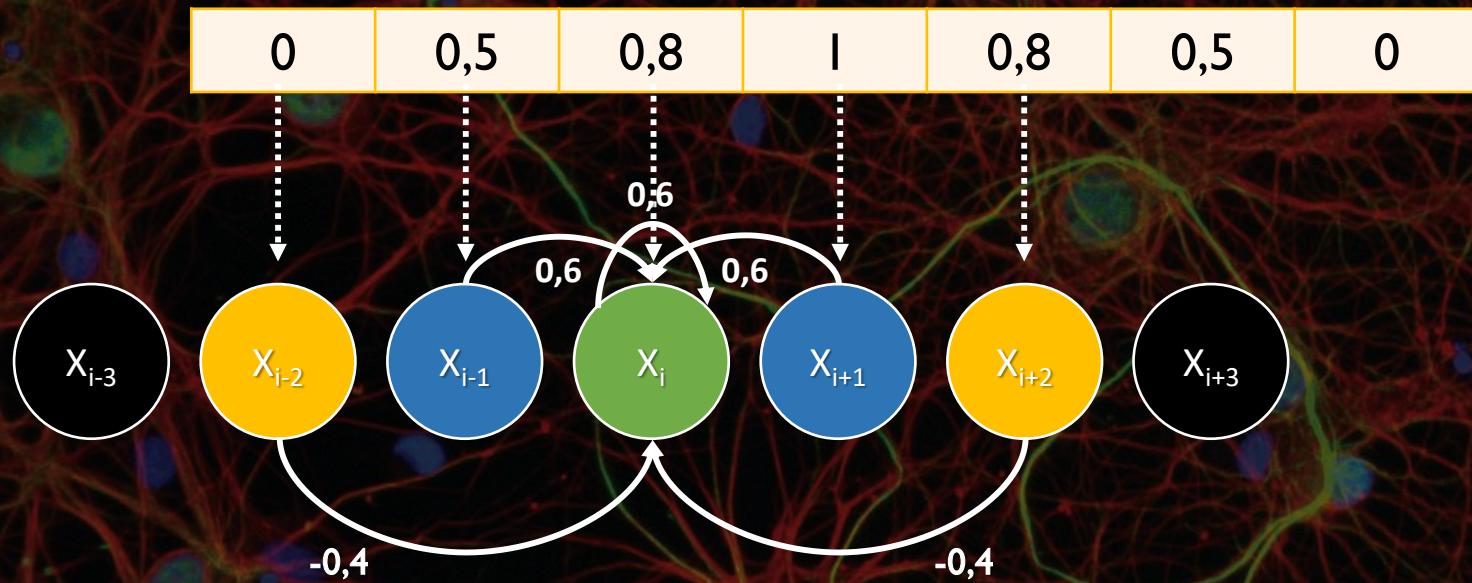
$$x_1 = 0,6(0) + 0,6(0,5) - 0,4(0,8) = -0,2$$

# Contoh



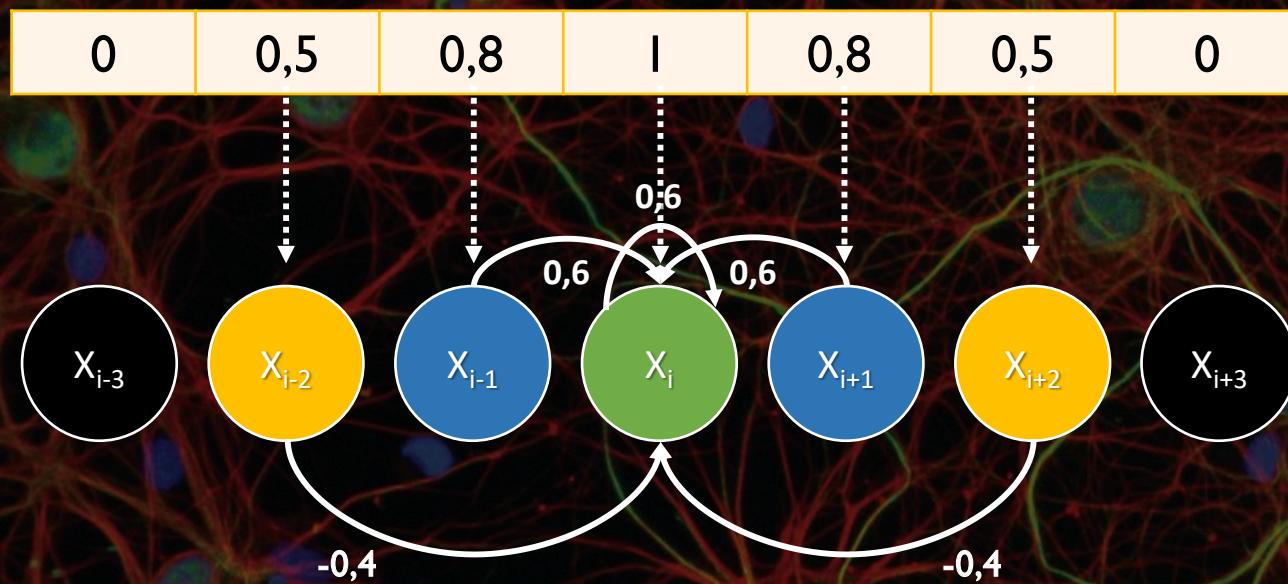
$$x_2 = 0,6(0) + 0,6(0,5) + 0,6(0,8) - 0,4(1) = 0,38$$

# Contoh



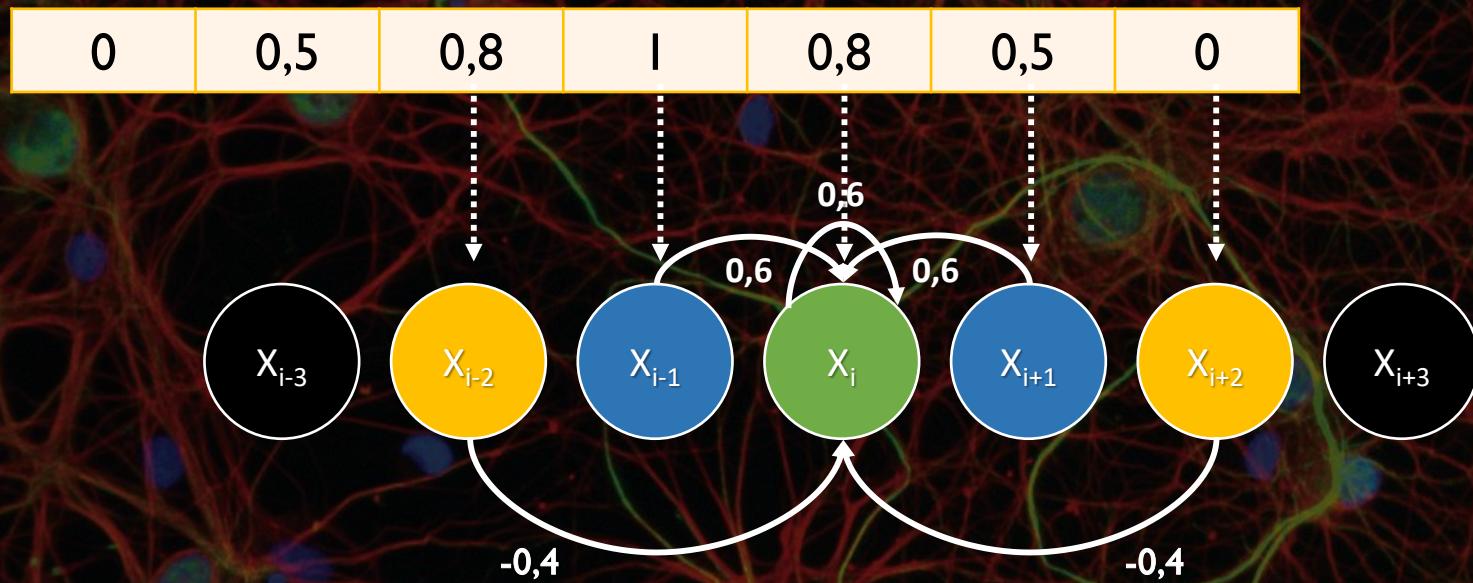
$$x_3 = -0,4(0) + 0,6(0,5) + 0,6(0,8) + 0,6(1) - 0,4(0,8)$$
$$= 1,06$$

# Contoh



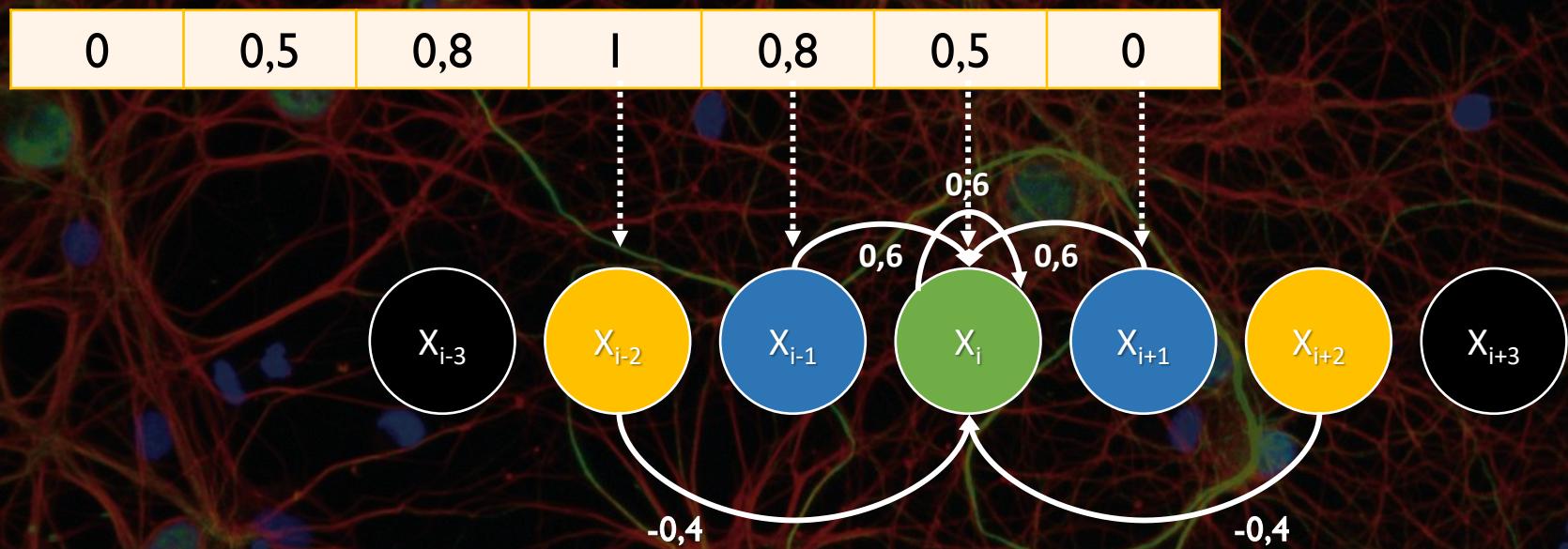
$$\begin{aligned}x_4 &= -0,4(0,5) + 0,6(0,8) + 0,6(1) + 0,6(0,8) - 0,4(0,5) \\&= 1,16\end{aligned}$$

# Contoh



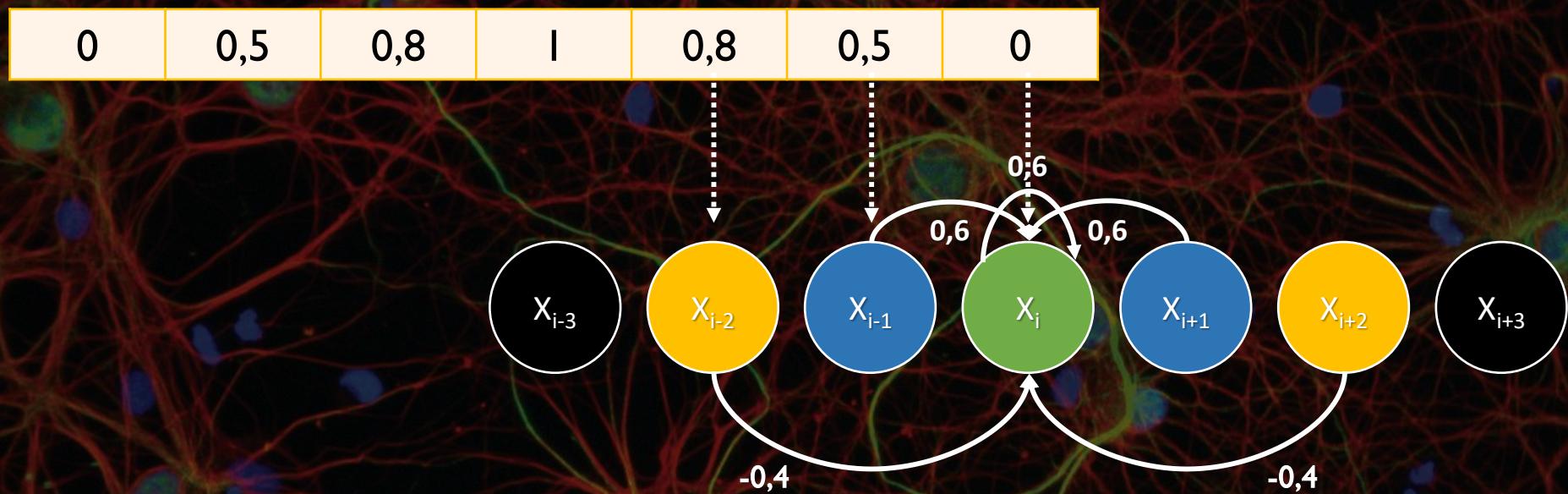
$$\begin{aligned}x_5 &= -0,4(0,8) + 0,6(1) + 0,6(0,8) + 0,6(0,5) - 0,4(0) \\&= 1,06\end{aligned}$$

# Contoh



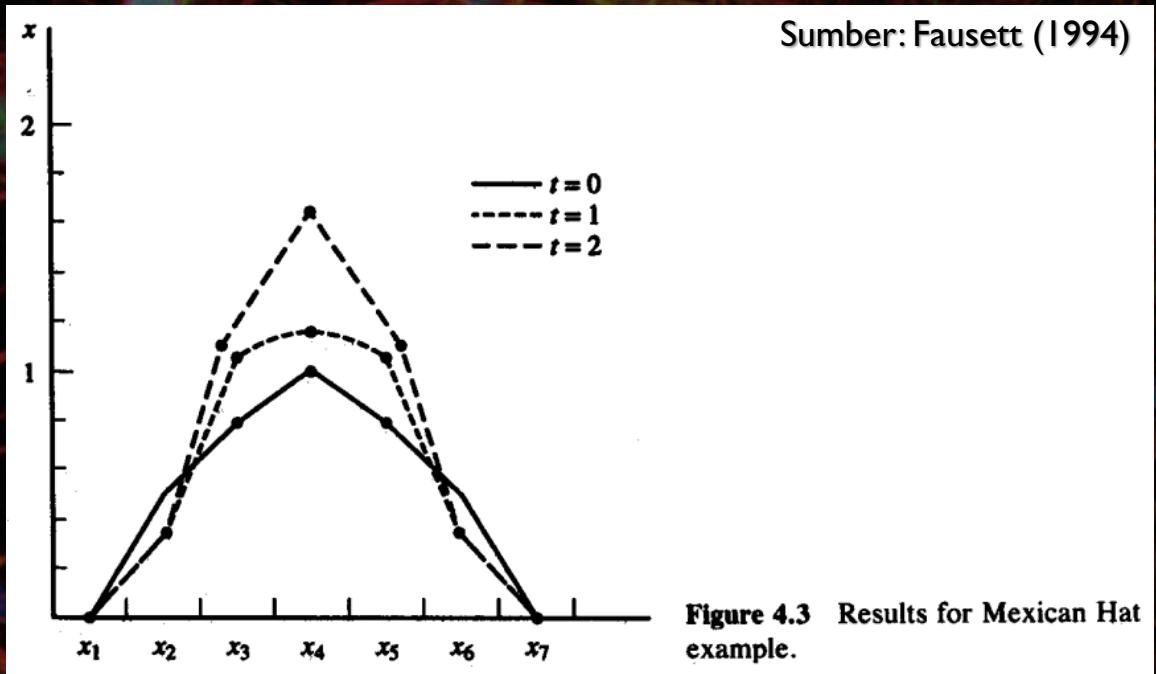
$$x_6 = -0,4(1) + 0,6(0,8) + 0,6(0,5) + 0,6(0) = 0,38$$

# Contoh



$$x_7 = -0,4(0,8) + 0,6(0,5) + 0,6(0) = -0,2$$

# Contoh



Iterasi	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$
0	0	0,5	0,8	1	0,8	0,5	0
1	0	0,38	1,06	1,16	1,06	0,38	0
2	0	0,39	1,14	1,66	1,14	0,39	0

# Implementasi Mexican Hat

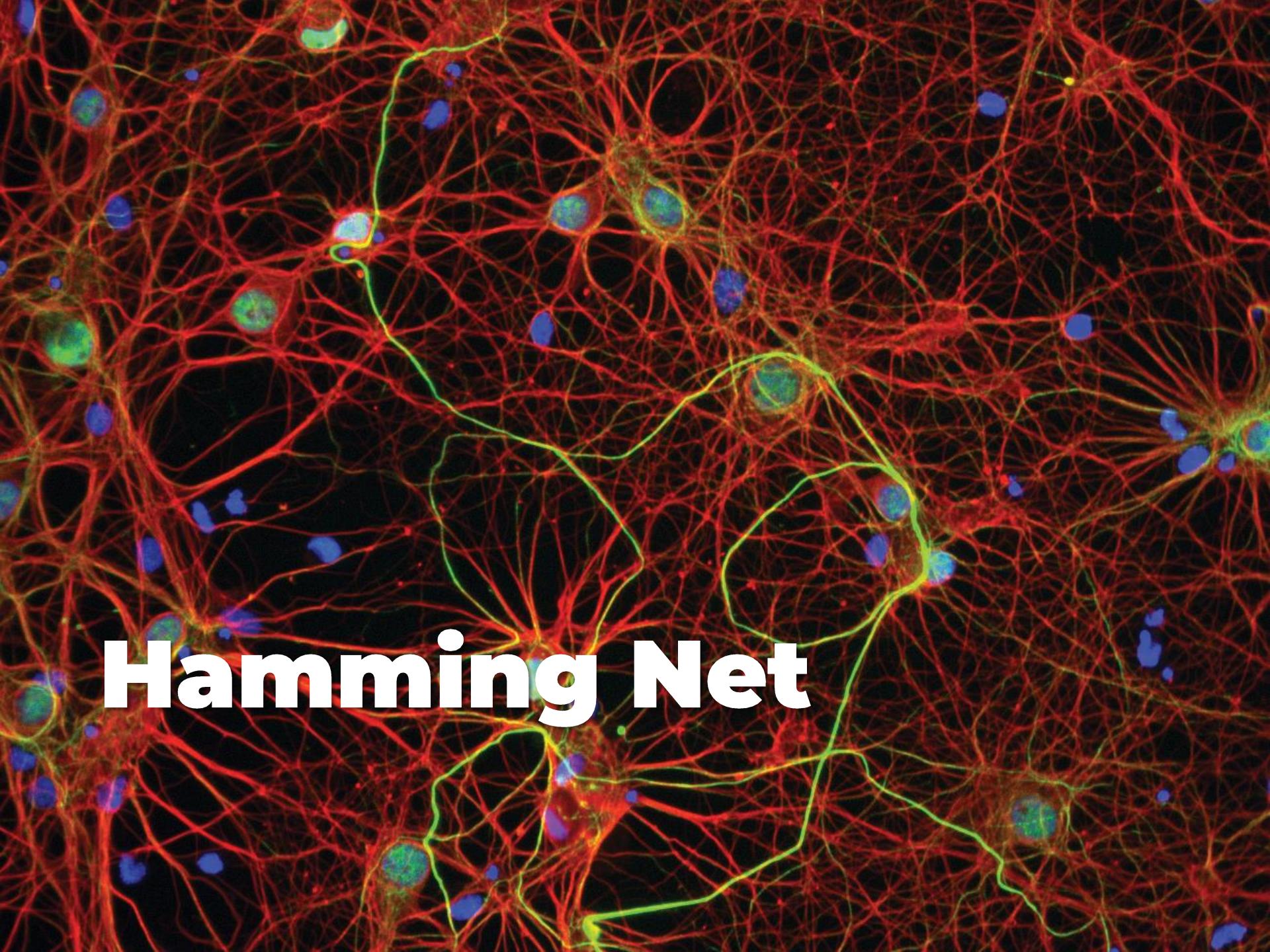
```
import numpy as np

def act(x):
    return [0 if i < 0 else i if 0 <= i <= 2 else 2 for i
in x]

def mexhat(x, r2, c1, c2, t_max):
    k = [c1] * (r2 * 2 + 1)
    k[0] = k[-1] = c2

    for t in range(t_max):
        print(x)
        x = act(np.convolve(x, k, 'same'))

x = [0, .5, .8, 1, .8, .5, 0]
mexhat(x, 2, .6, -.4, 3)
```



# Hamming Net

# Hamming Net

- Lippmann (1987)
- *Maximum likelihood classifier*
- Menentukan *exemplar vector* yang paling mirip dengan sebuah *input vector*
- Menggunakan Maxnet sebagai *subnet*

# Hamming Net

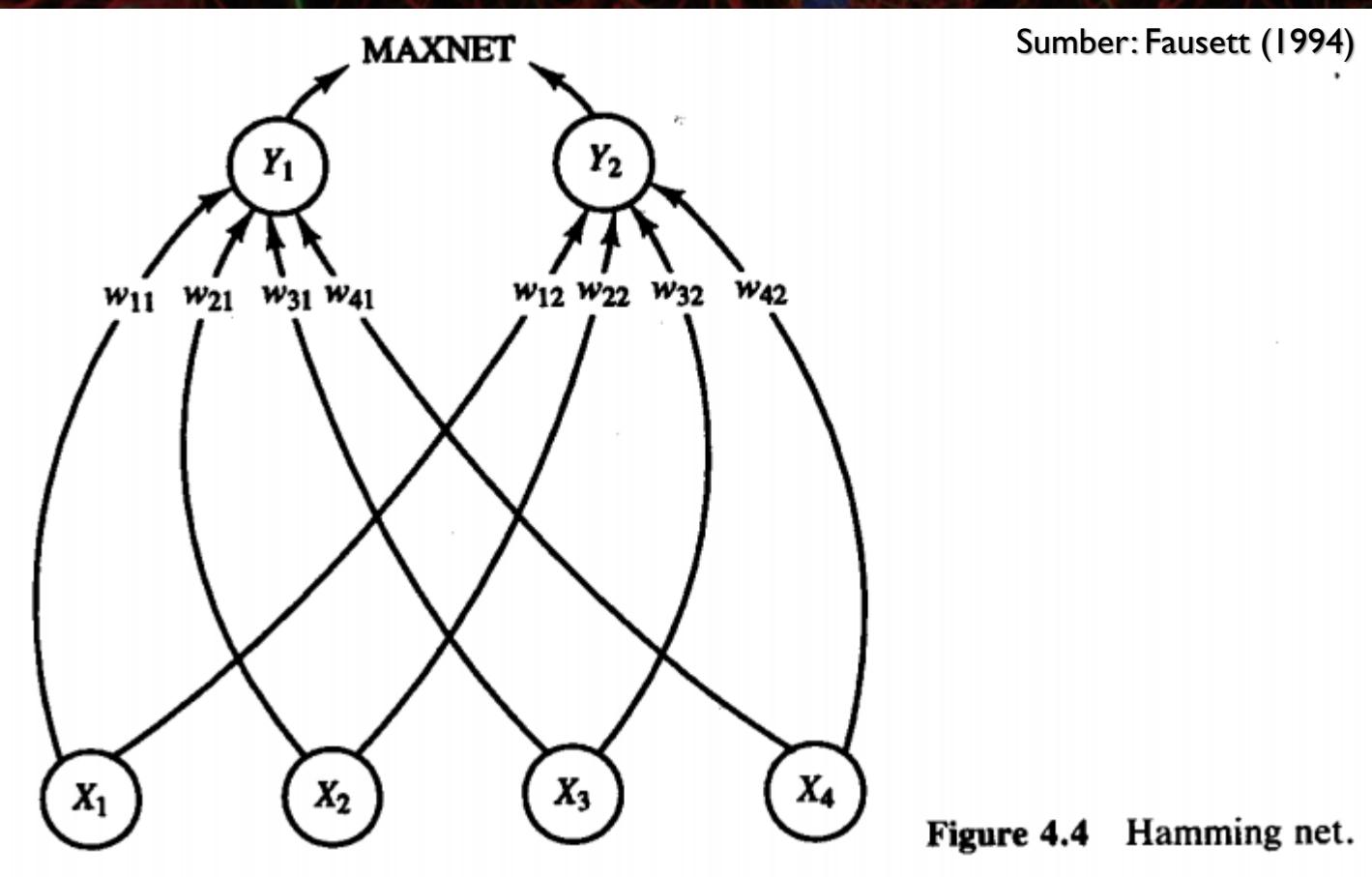


Figure 4.4 Hamming net.

# Hamming Net

- $x_1, x_2, \dots, x_n$  adalah *input vector* dengan jumlah elemen sebanyak  $n$
- $e_1, e_2, \dots, e_m$  adalah *exemplar vector* sebanyak  $m$

# Algoritme Hamming Net

- I. Inisialisasi bobot dan bias:

$$w_{ij} = \frac{e_i(j)}{n}$$

$$b_j = \frac{n}{2}$$

2. Untuk setiap *input vector*  $x$ , lakukan langkah 3–5

# Algoritme Hamming Net

3. Hitung  $y_{in}$ :

$$y_{in_j} = b_j + \sum_i x_i w_{ij}$$

4. Inisialisasi nilai aktivasi untuk Maxnet:

$$a_j = y_{in_j}$$

5. Operasikan Maxnet

# Contoh Hamming Net

- Exemplar vector:

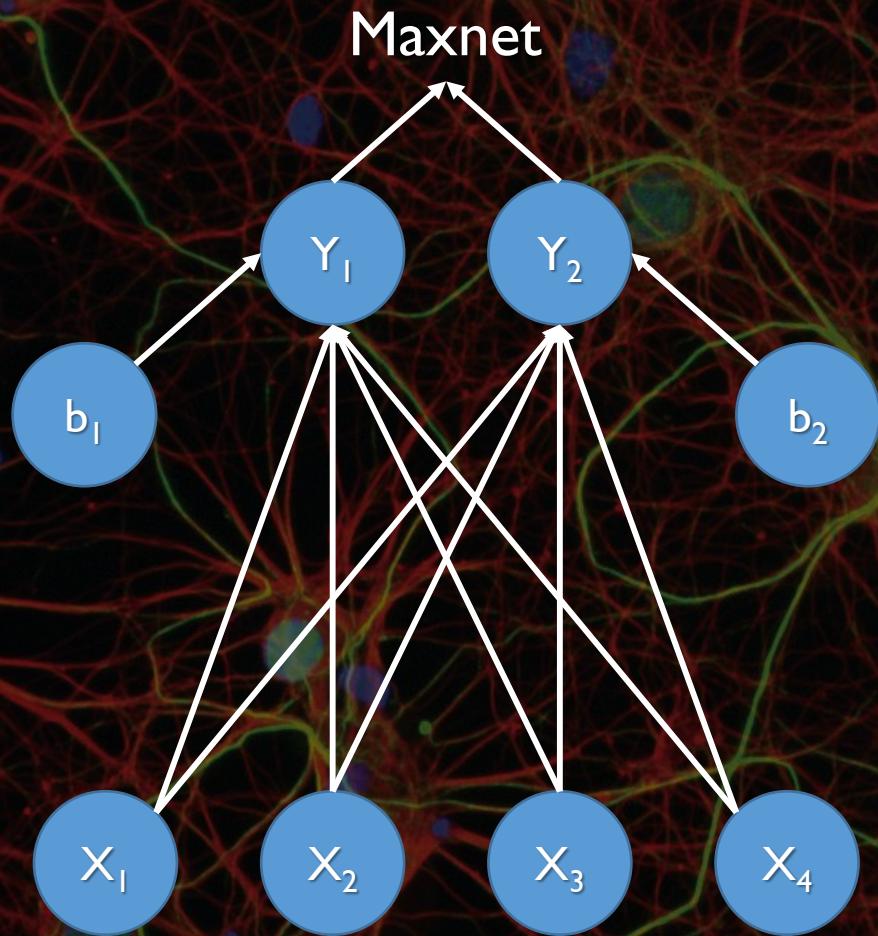
$$e(1) = (1, -1, -1, -1)$$

$$e(2) = (-1, -1, -1, 1)$$

- Input vector:

$$x = (1, 1, -1, -1)$$

# Contoh Hamming Net



# Contoh Hamming Net

I. Inisialisasi bobot:

$$e(1) = (1, -1, -1, -1)$$

$$e(2) = (-1, -1, -1, 1)$$

$$w_{ij} = \frac{e_i(j)}{2}$$

$$W = \begin{bmatrix} .5 & -.5 \\ -.5 & -.5 \\ -.5 & -.5 \\ -.5 & .5 \end{bmatrix}$$

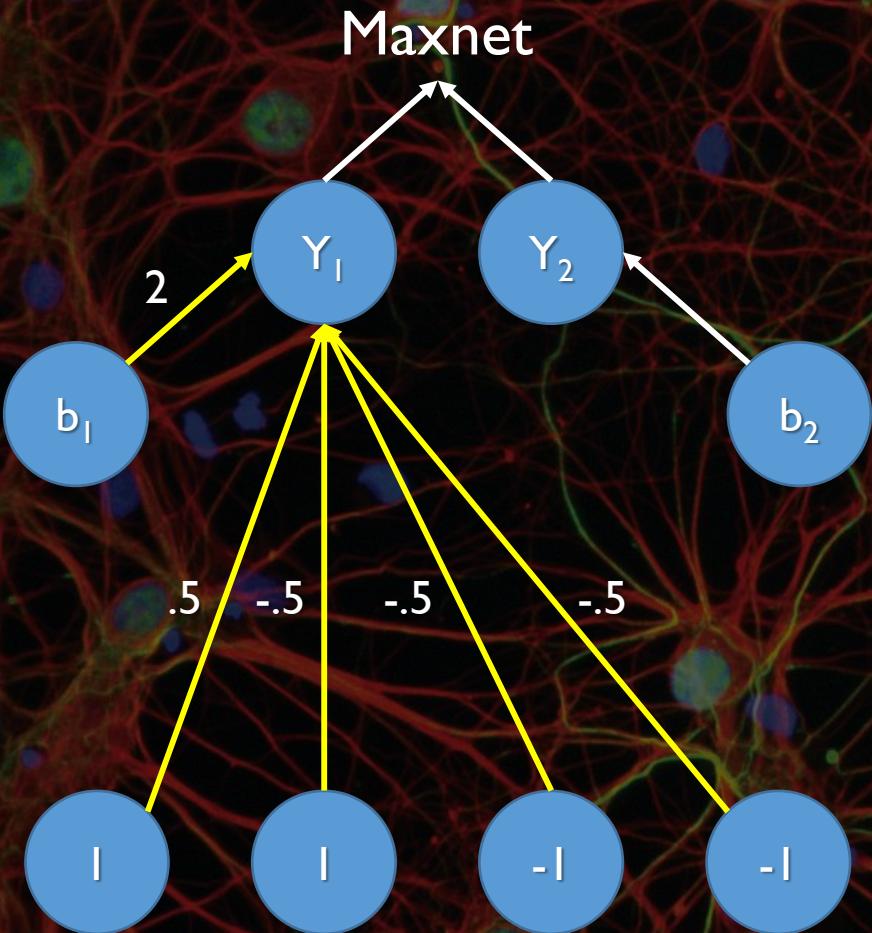
# Contoh Hamming Net

I. Inisialisasi bias:

$$b_j = \frac{n}{2}$$

$$b_1 = b_2 = \frac{4}{2} = 2$$

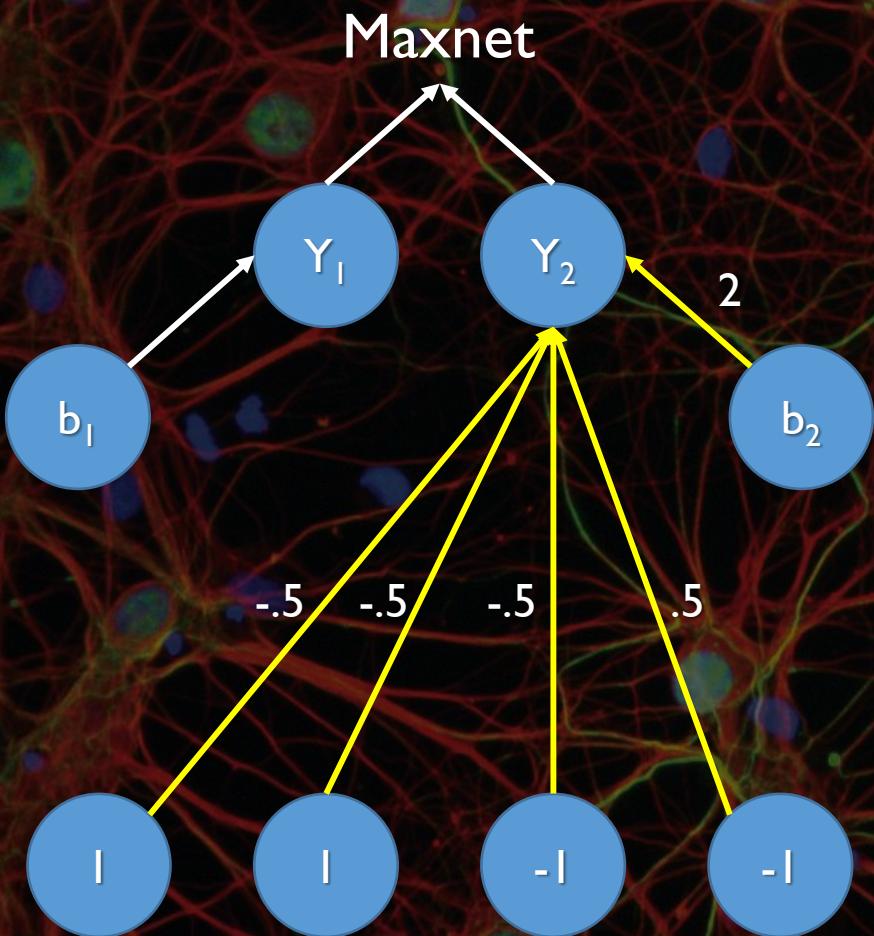
# Contoh Hamming Net



3. Hitung  $y_{in}$ :

$$y_{in_1} = b_1 + \sum_i x_i w_{i1} \\ = 2 + 1 = 3$$

# Contoh Hamming Net



3. Hitung  $y_{in}$ :

$$y_{in_2} = b_1 + \sum_i x_i w_{i2}$$
$$= 2 - 1 = 1$$

# Contoh Hamming Net

$e(1)$		-	-	-
$x$			-	-
Kemiripan	3			
$e(2)$	-	-	-	
$x$			-	-
Kemiripan	1			

# Implementasi Hamming Net

```
import numpy as np

def hamming(ex, x):
    w = np.array(ex) / 2
    b = len(w[0]) / 2
    y = [b + sum(x * w[i]) for i in range(len(w))]
    m = maxnet(y)

    return ex[m]

ex = [[1, -1, -1, -1],
      [-1, -1, -1, 1]]
x = [1, 1, -1, -1]
h = hamming(ex, x)

print(h)
```

A fluorescence microscopy image showing a dense network of neurons. The neurons are stained with a red fluorescent antibody against a protein like MAP2 or Tuj1, which labels the cell bodies and long, branching processes. Some neurons are highlighted with a bright green outline. Nuclei are stained with DAPI, appearing as small blue spots. The overall pattern is a complex web of red lines with green and blue puncta.

**Alhamdulillah.**