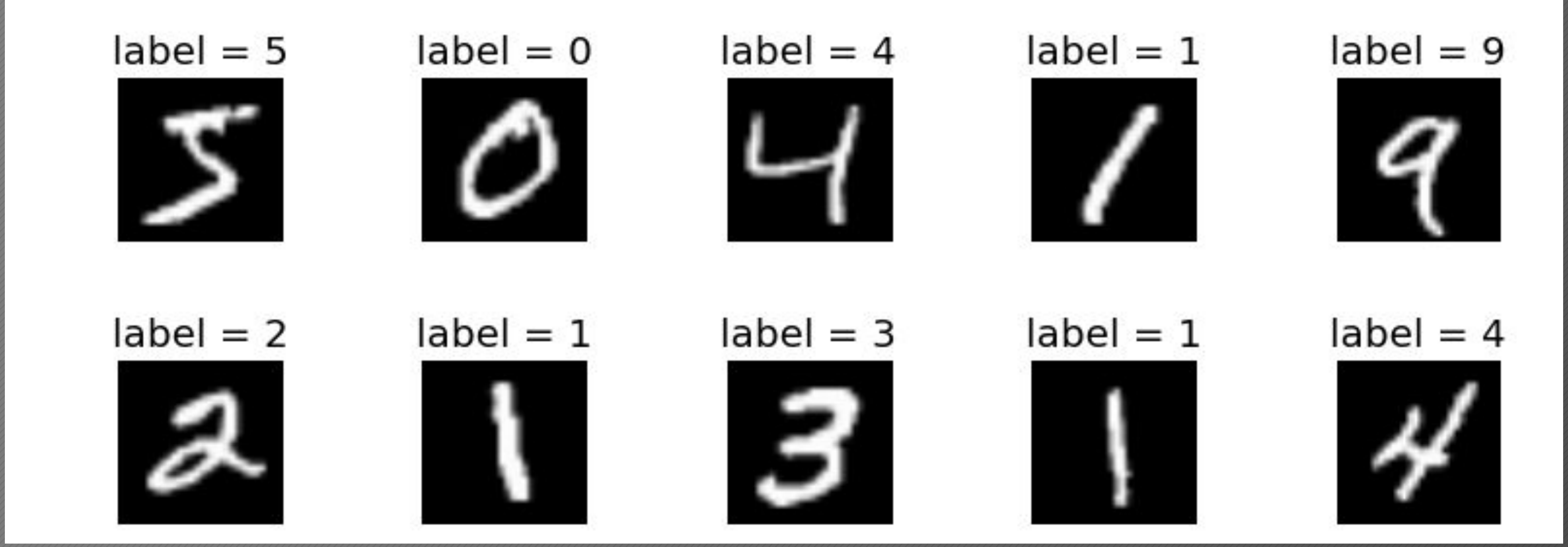
Rețele neurale

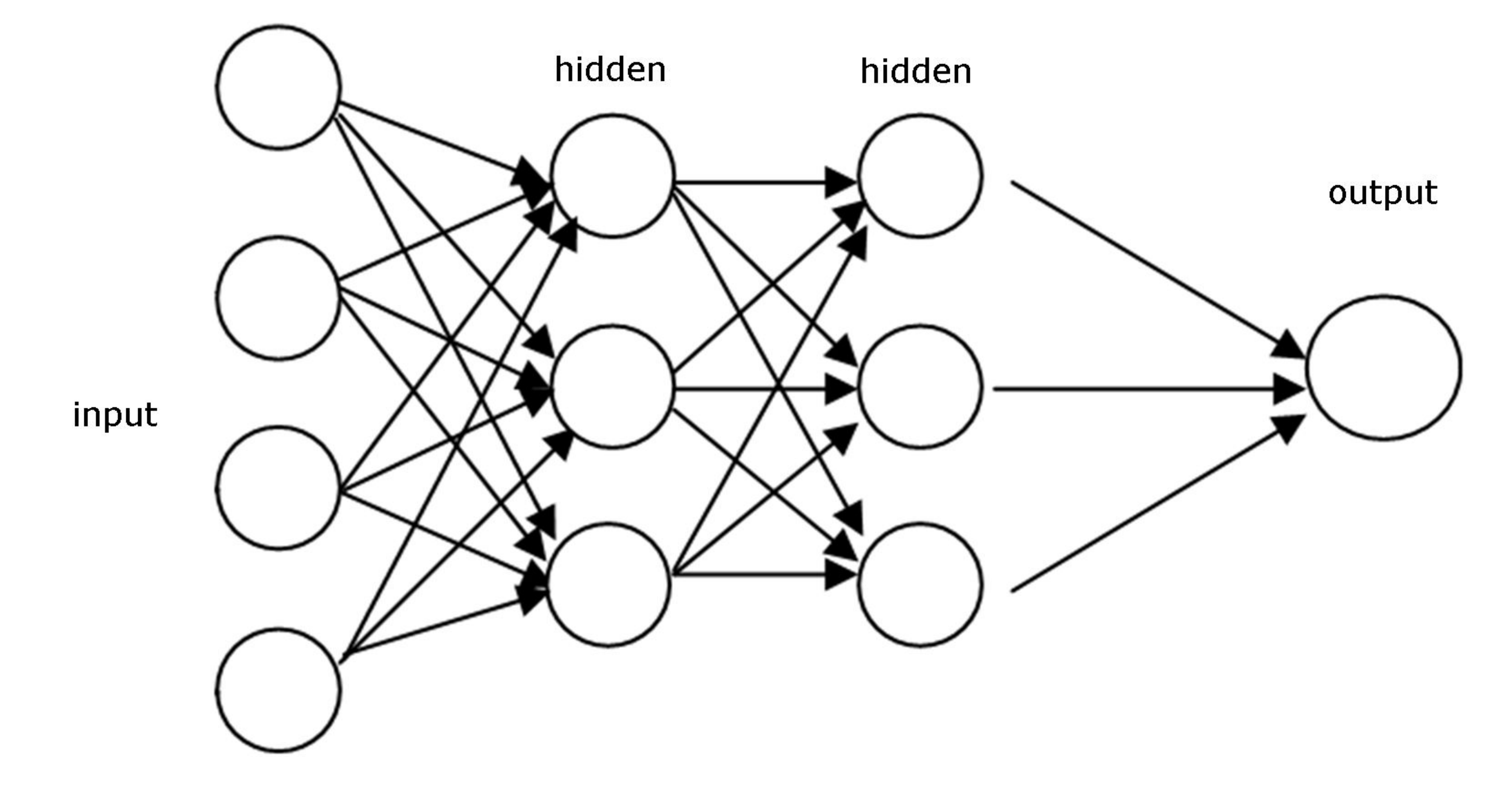
Învățare automată

Introducere

- Clasificare de imagini folosind o rețea neurală
- Rețea de tip feed forward
- Recunoașterea literelor în imagini (dataset MNIST)



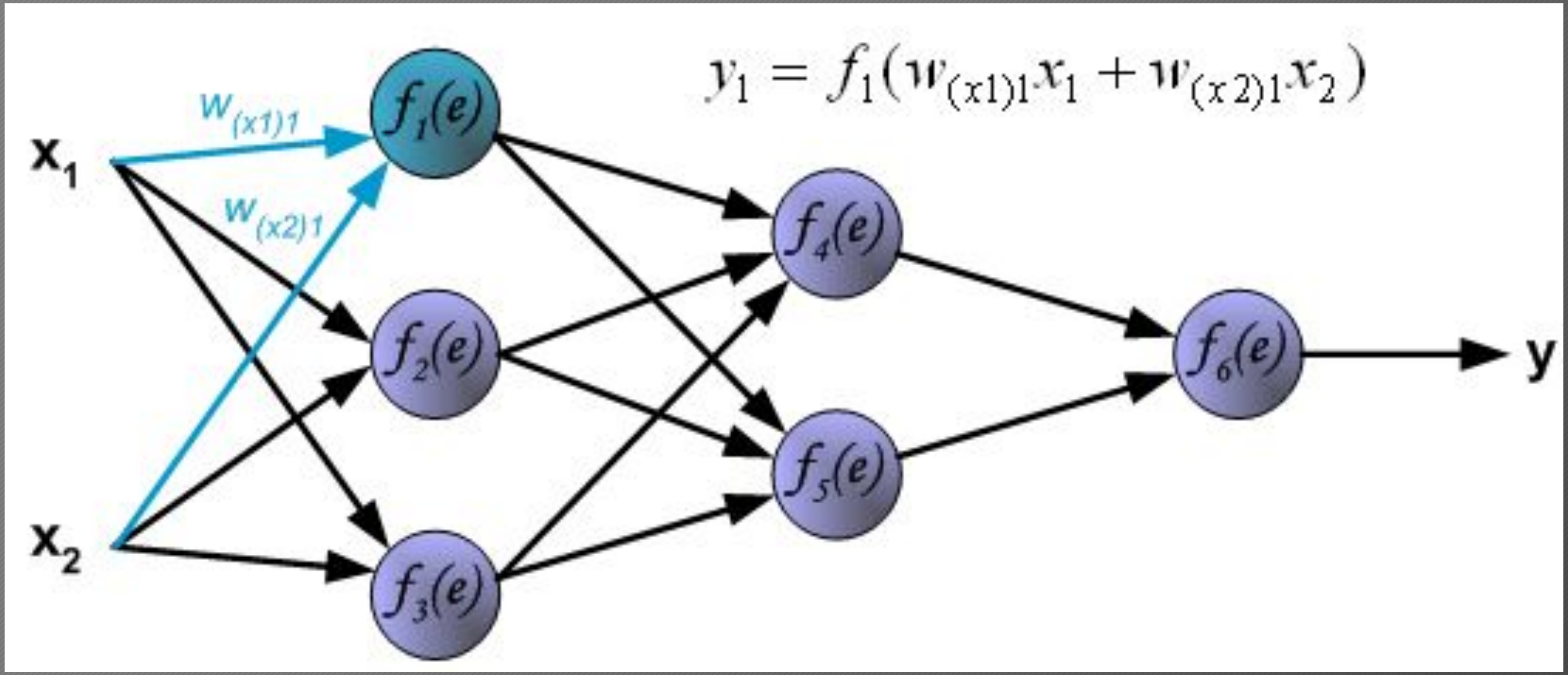
Feed forward



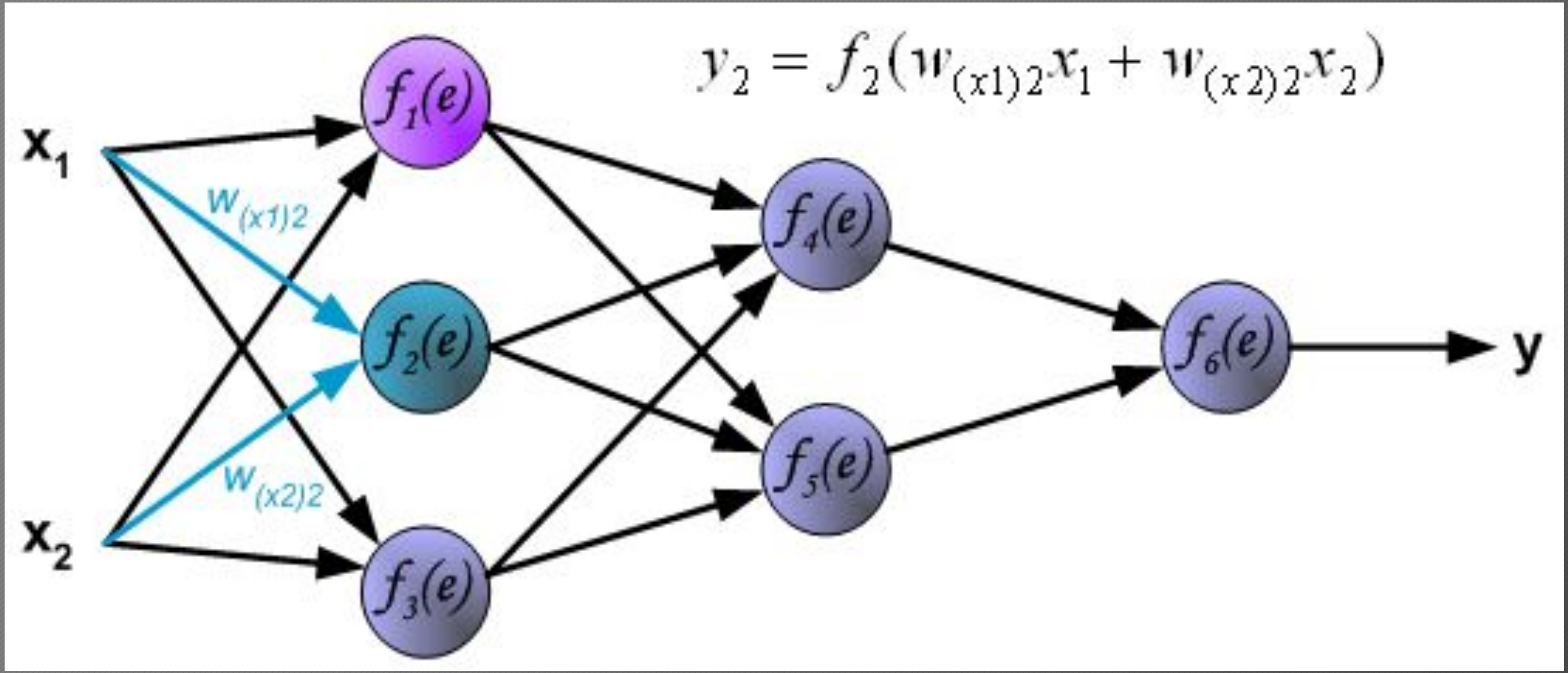
SGD

- **Stochastic gradient descent**:
  1. Forward
  2. Compute errors
  3. Backpropagation
  4. Update parameters

1. Forward

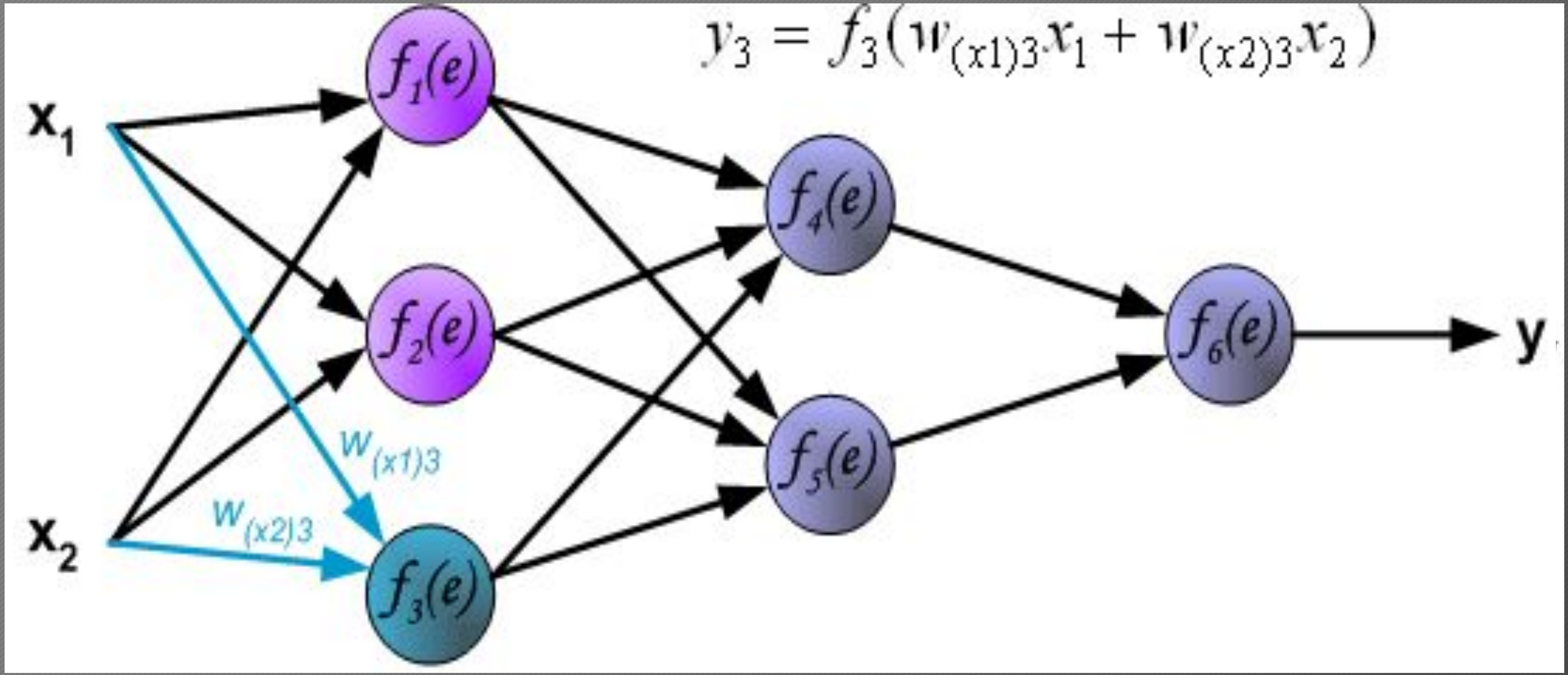


1. Forward



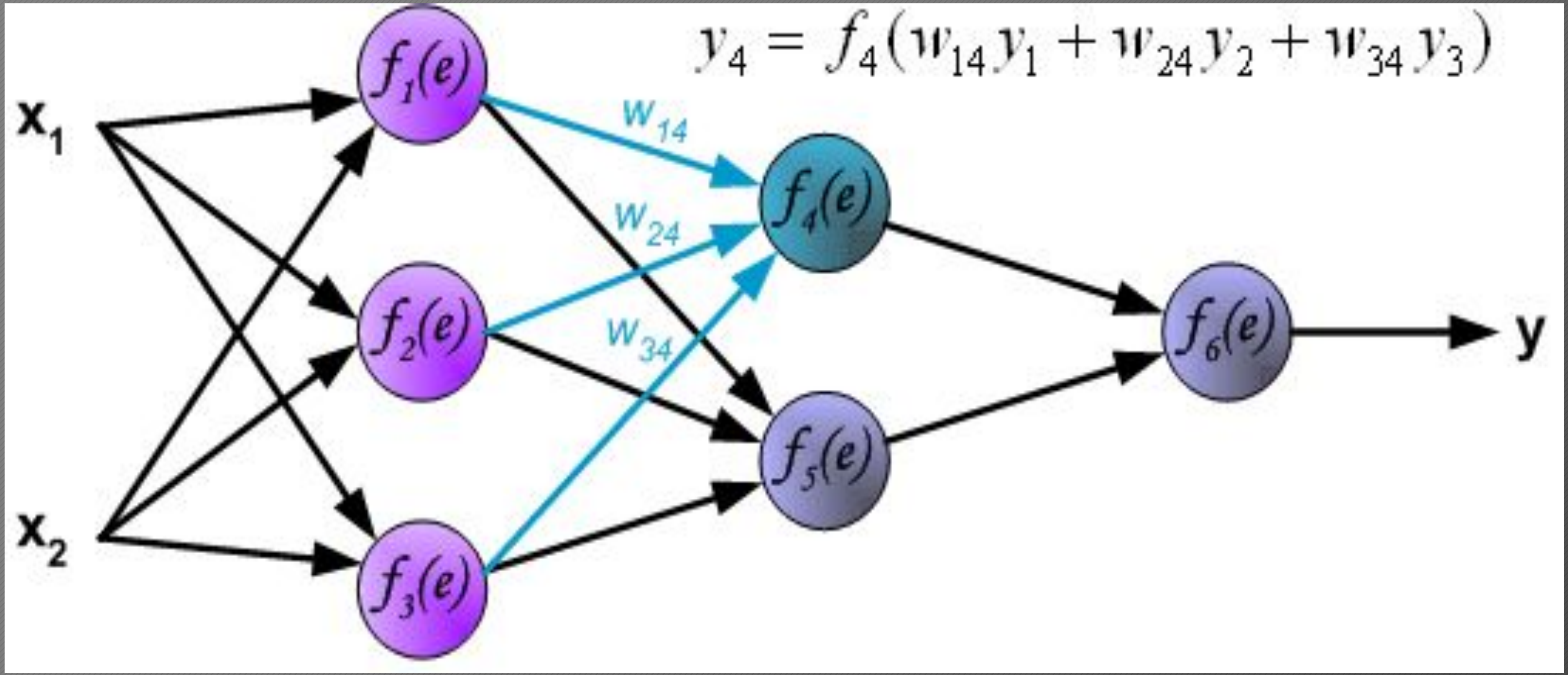
$$y_2 = f_2(w_{(x1)2} x_1 + w_{(x2)2} x_2)$$

1. Forward



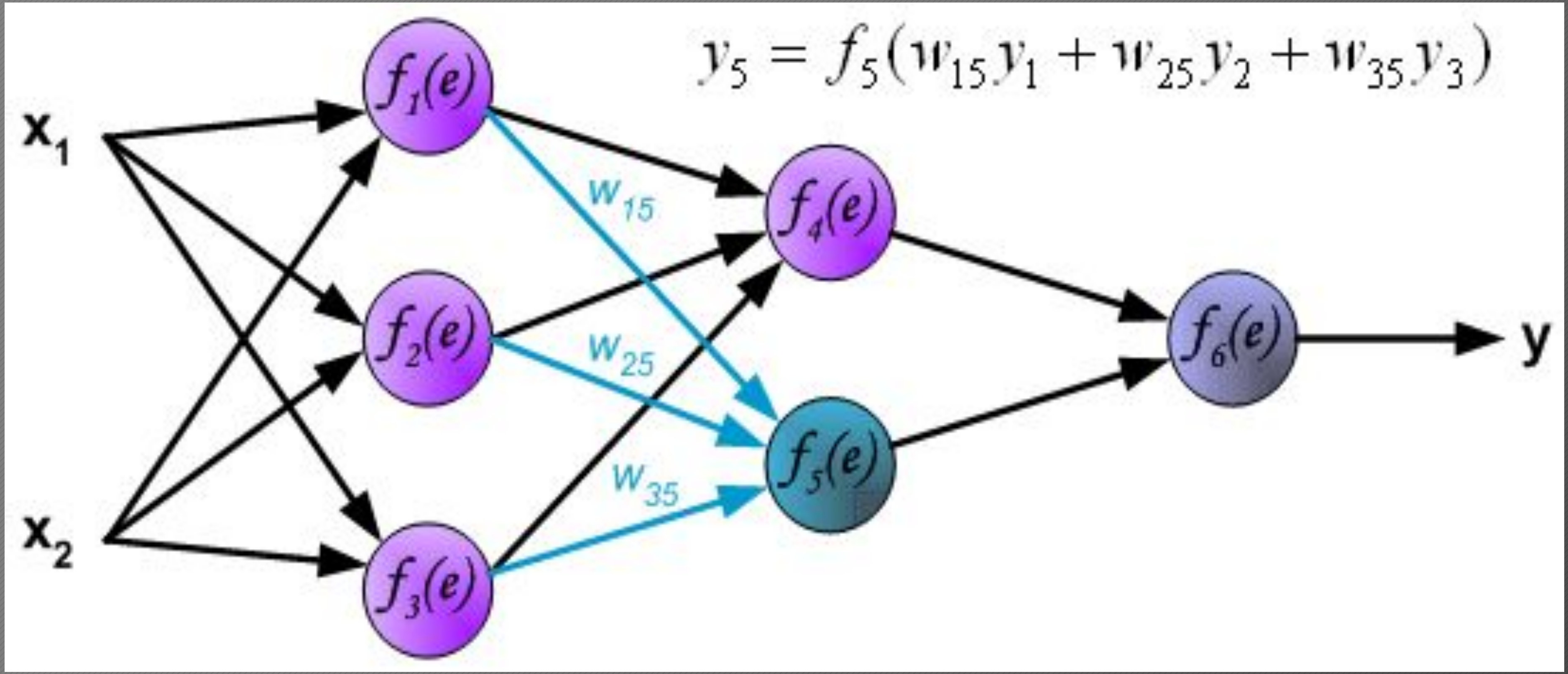
$$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$$

1. Forward

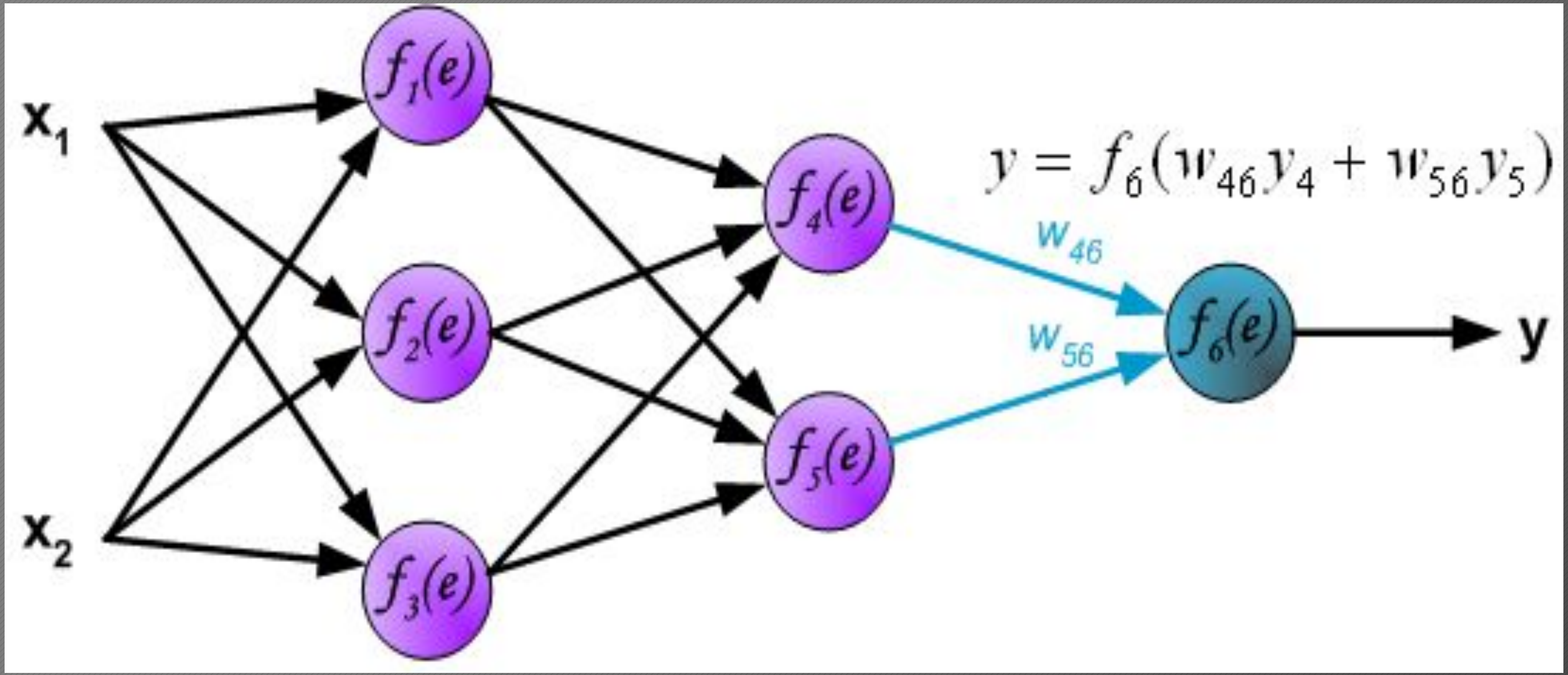


$$y_4 = f_4(w_{14} y_1 + w_{24} y_2 + w_{34} y_3)$$

1. Forward

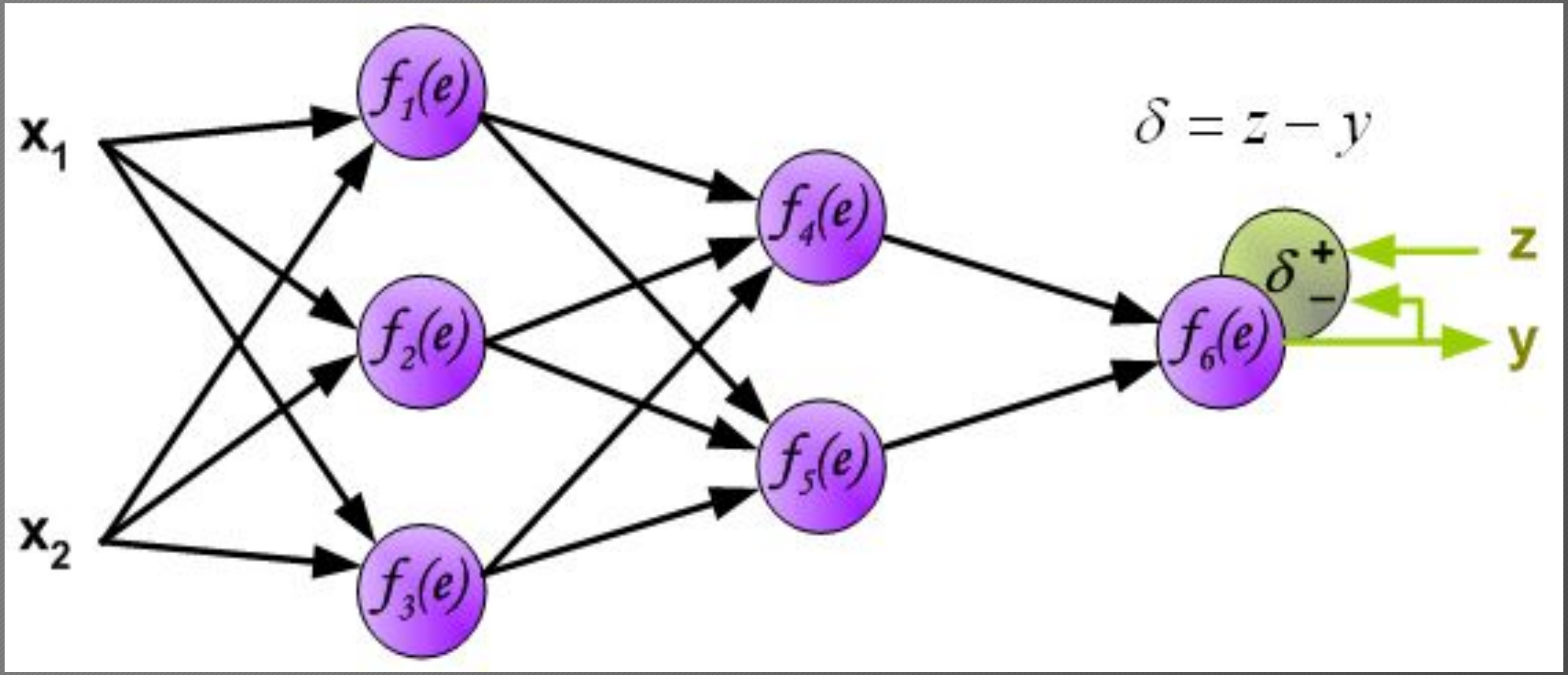

$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
w_{15}
w_{25}
w_{35}
$f_4(e)$
$f_5(e)$
$f_6(e)$
y

1. Forward

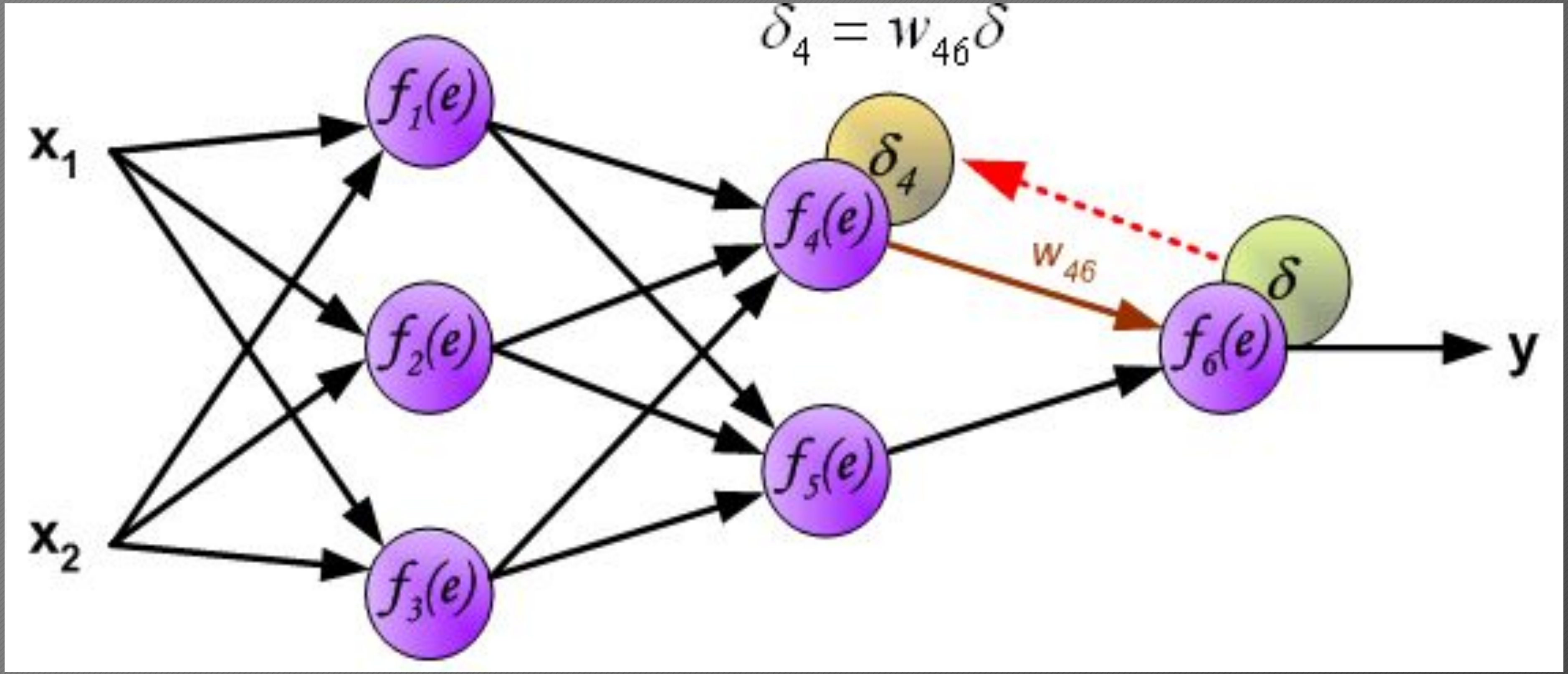


$$y = f_6(w_{46}y_4 + w_{56}y_5)$$

2. Compute errors

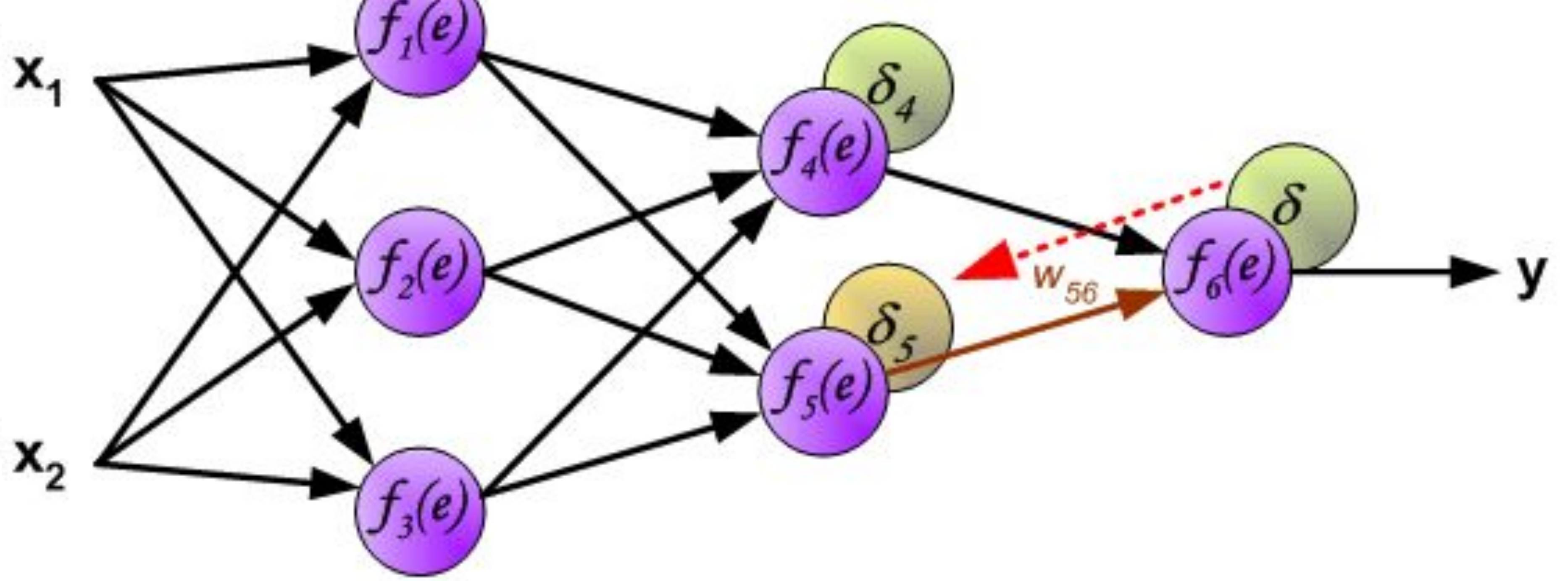


3. Backpropagation

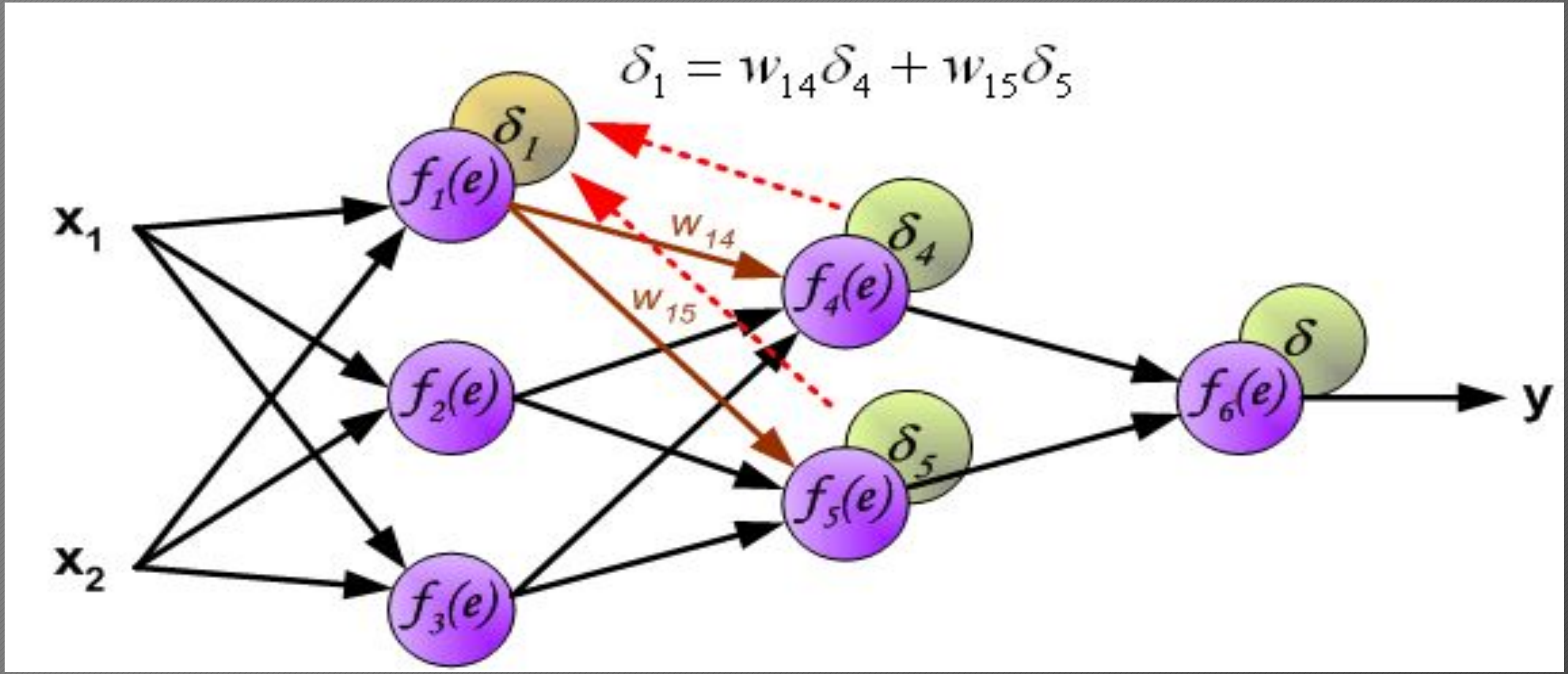


3. Backpropagation

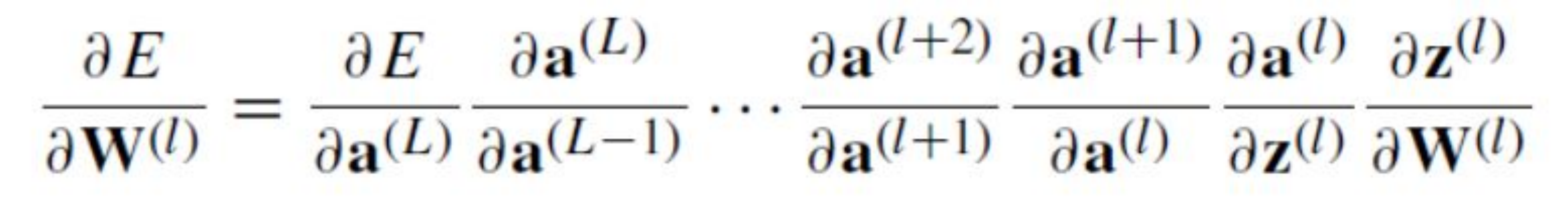
$$\delta_5 = w_{56}\delta$$



3. Backpropagation

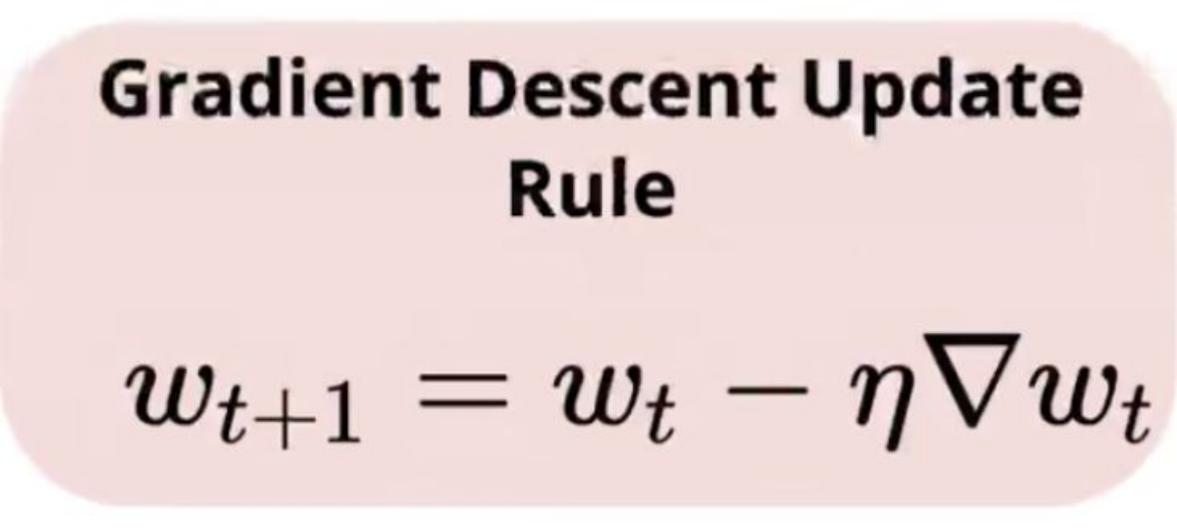


3. Backpropagation



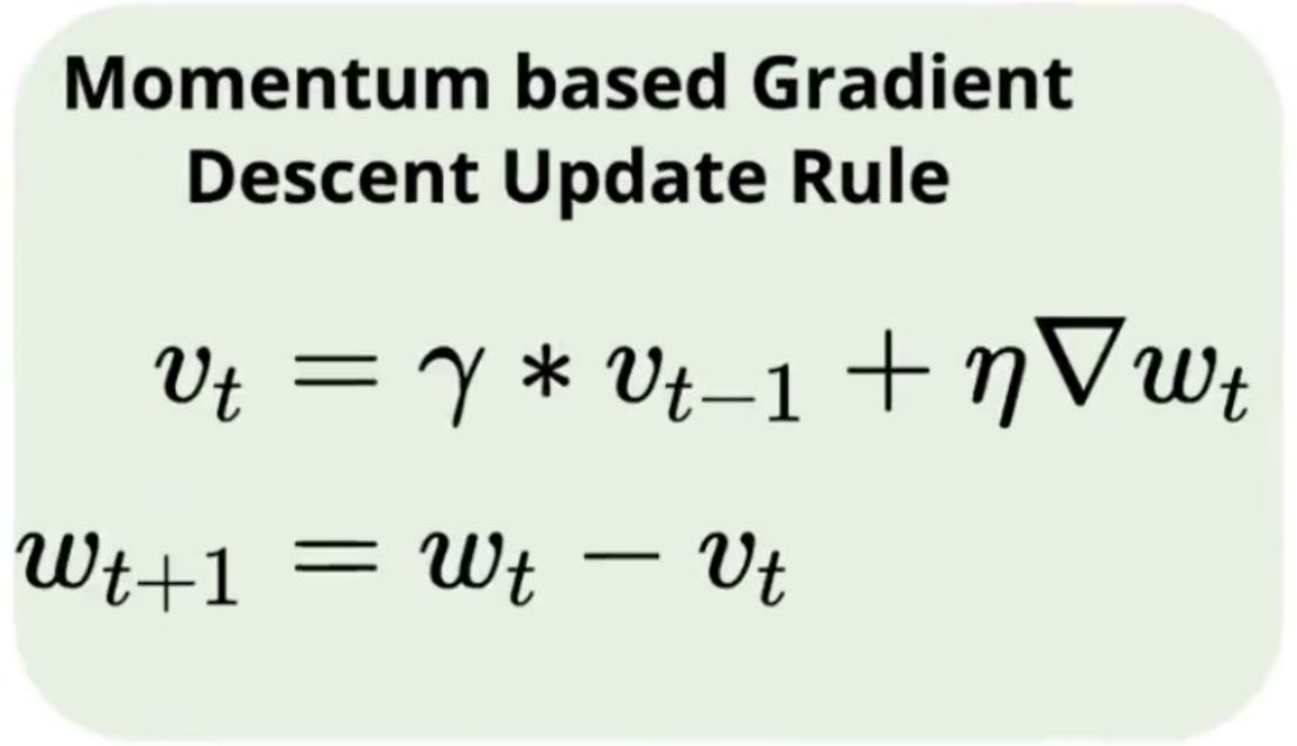
$$\frac{\partial E}{\partial \mathbf{W}^{(l)}} = \frac{\partial E}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{a}^{(L-1)}} \cdots \frac{\partial \mathbf{a}^{(l+2)}}{\partial \mathbf{a}^{(l+1)}} \frac{\partial \mathbf{a}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}}$$

4. Parameters update



Gradient Descent Update Rule

$$w_{t+1} = w_t - \eta \nabla w_t$$



Momentum based Gradient Descent Update Rule

$$v_t = \gamma * v_{t-1} + \eta \nabla w_t$$

$$w_{t+1} = w_t - v_t$$

Layers

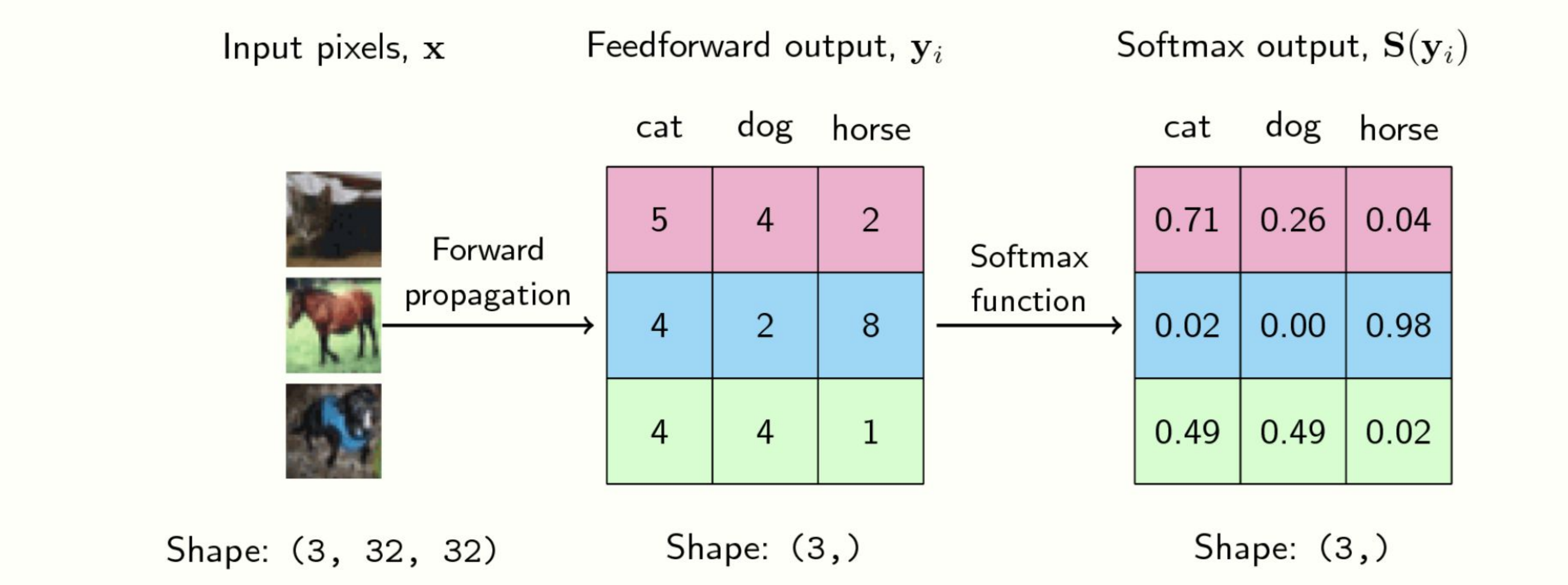
- **Rețea - Linear -> ReLU -> Linear**
- **Linear**
  - Forward: y = x * weight + bias
  - Backward
    - dweight = x.T * dy
    - dbias = dy
    - => dy * weight.T
- **ReLU**
  - Forward: max (0,x)
  - Backward: => dy * (x > 0)

Cost function

- Cross entropy - combină ***softmax*** cu ***negative log-likelihood***
- Softmax – transformă rezultatele generate de rețea într-o distribuție de probabilitate

$$S(f_{y_i}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

Cost function



Input pixels, $\mathbf{x}$

Shape: (3, 32, 32)

Feedforward output, $\mathbf{y}_i$

cat	dog	horse
5	4	2
4	2	8
4	4	1

Shape: (3,)

Softmax output, $\mathbf{S}(\mathbf{y}_i)$

cat	dog	horse
0.71	0.26	0.04
0.02	0.00	0.98
0.49	0.49	0.02

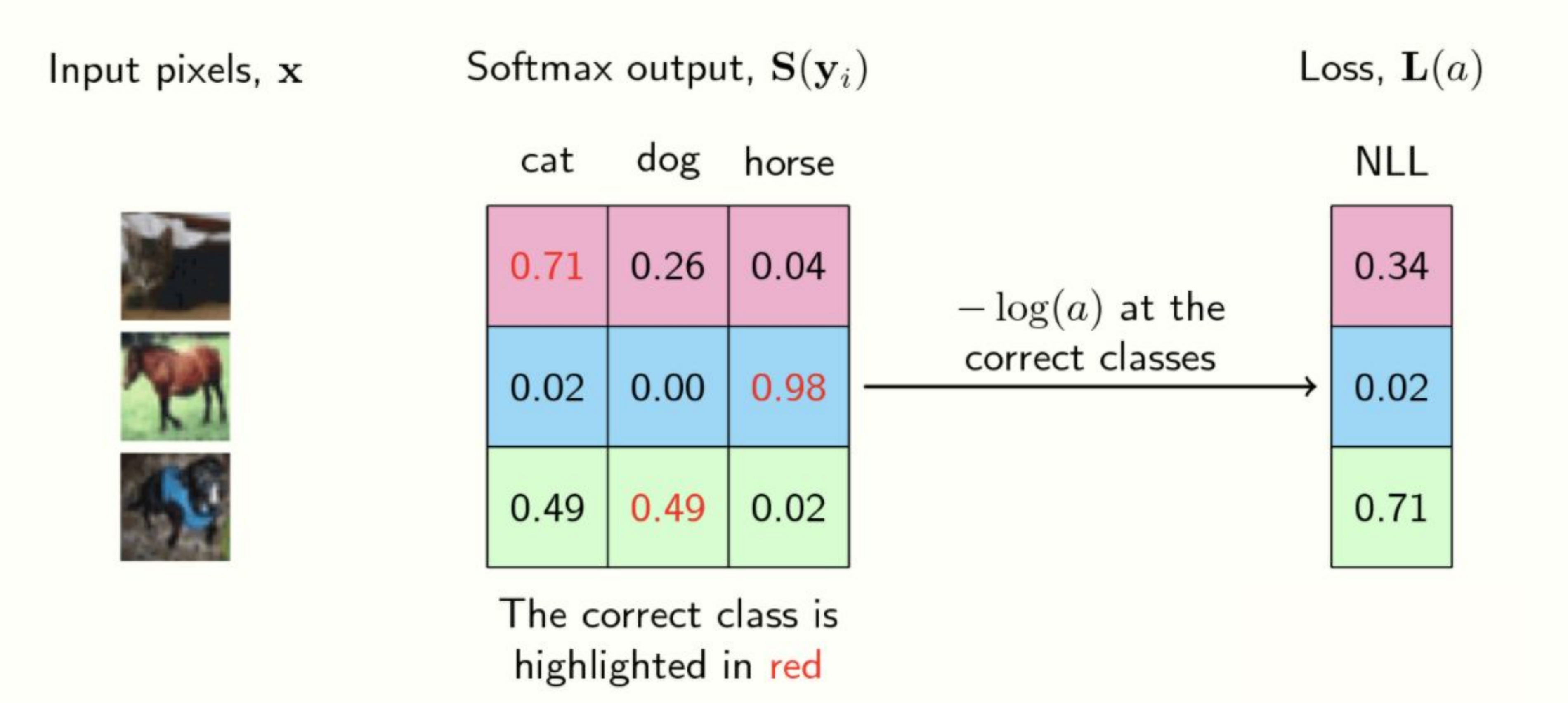
Shape: (3,)

Cost function

- **Cross entropy** - combină ***softmax*** cu ***negative log-likelihood***
- **Negative log-likelihood**

$$L(\mathbf{y}) = -\log(\mathbf{y})$$

Cost function



Input pixels, $\mathbf{x}$

Softmax output, $\mathbf{S}(\mathbf{y}_i)$

cat	dog	horse
0.71	0.26	0.04
0.02	0.00	0.98
0.49	0.49	0.02

$-\log(a)$ at the correct classes

Loss, $\mathbf{L}(a)$

NLL
0.34
0.02
0.71

The correct class is highlighted in red

Cost function

- **Cross entropy**
  - Forward:

$$p_k = \frac{e^{f_k}}{\sum_j e^{f_j}}$$

$$L_i = -log(p_{y_i})$$

  - Backward:

$$p_k = \frac{e^{f_k}}{\sum_j e^{f_j}}$$

$$\frac{\partial L_i}{\partial f_k} = p_k - \mathbb{1}(y_i = k)$$