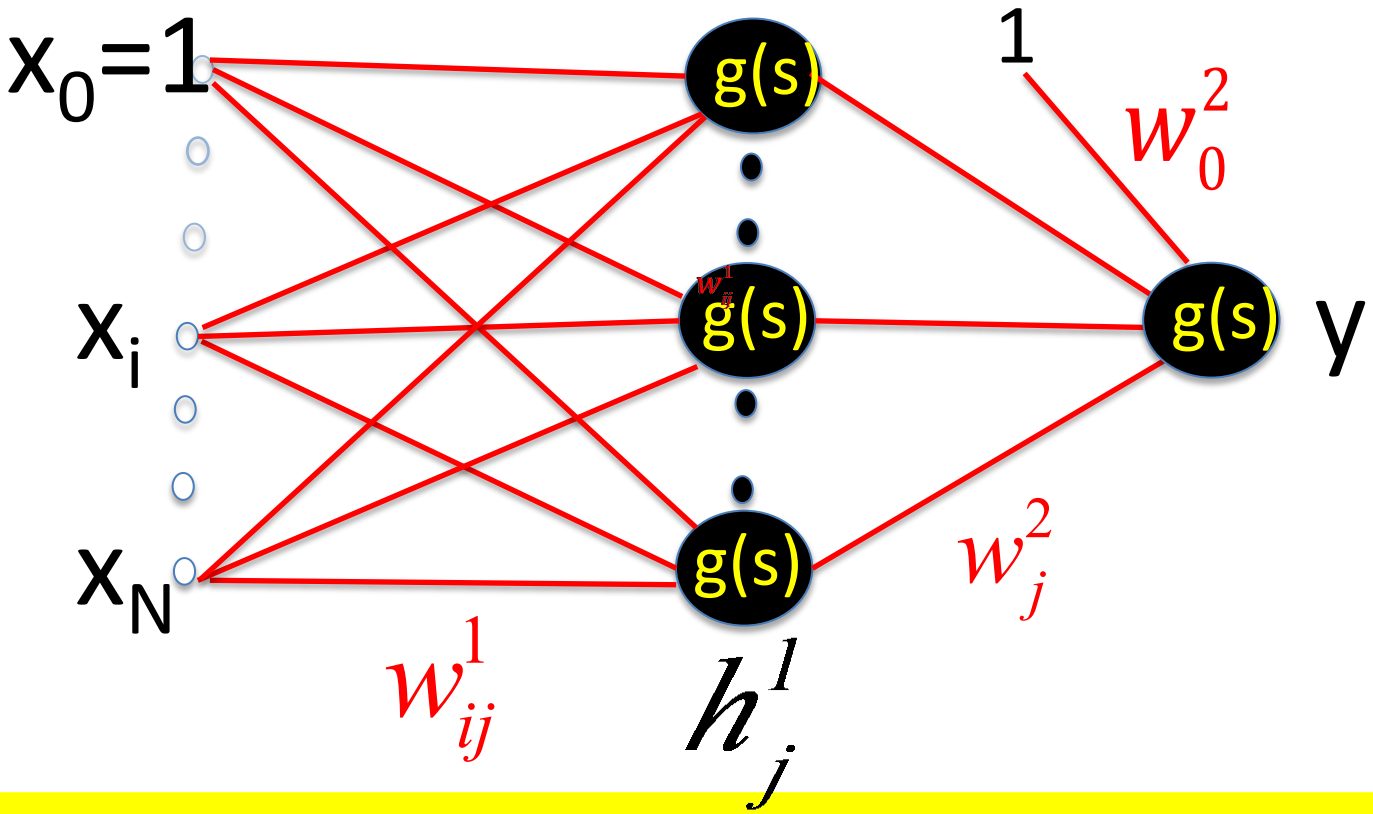
Lecture 5a

2-layer multiple outputs

Outline

1. Multiple categories
2. Associative memory
3. Number of HU’s; compression
4. Compress then recognize

Two layer network-single output



$$y = g\left(\sum_{j=0}^{N_1} w_j^2 h_j^1\right) \qquad h_j^1 = g\left(\sum_{i=0}^{N} w_{ij}^1 x_i\right)$$

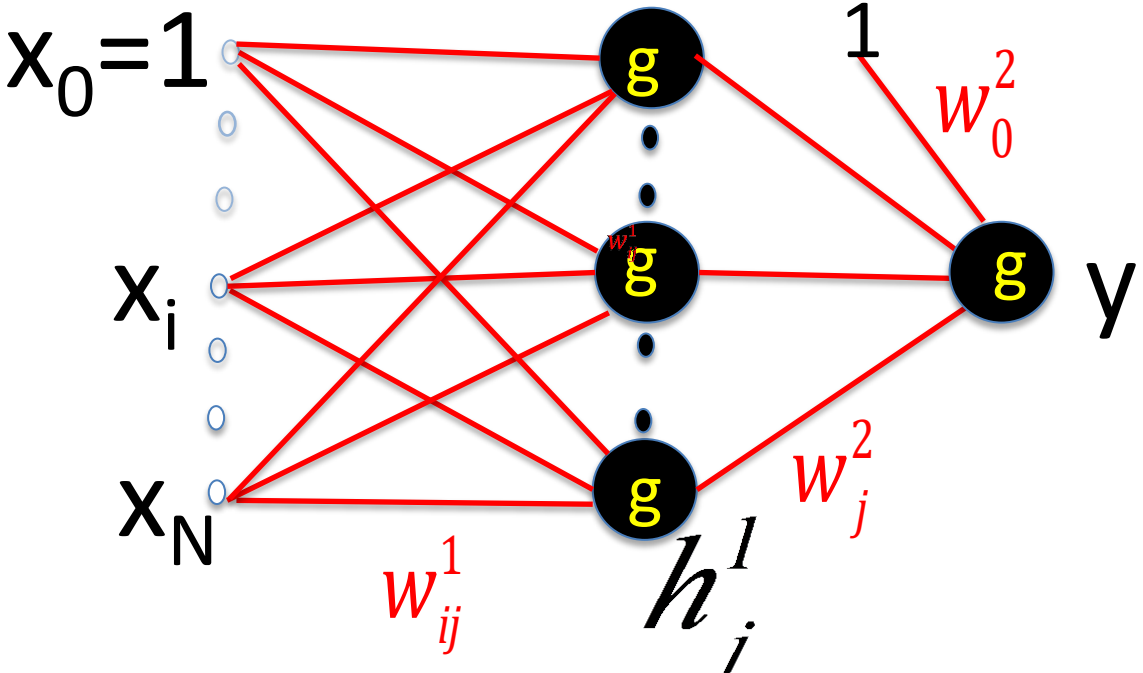
$$Training\ set: \left\{x^{(m)}, y^{(m)}\right\} \quad m = 1, M \quad y^{(m)} = label\,(0\ or\ 1, \pm 1)$$

Error Back Propagation

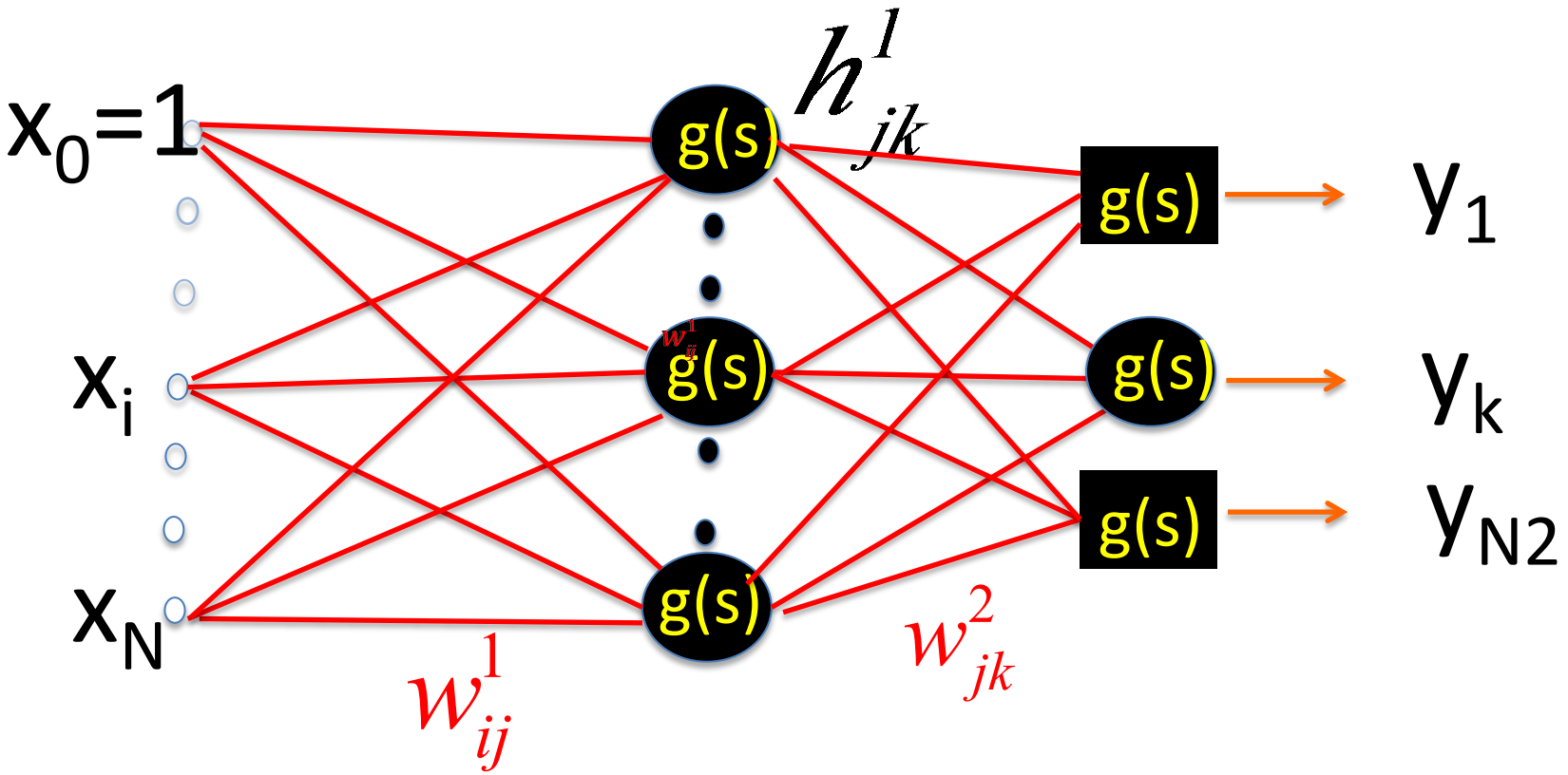
$$C(w_{ij}^1) = \sum_m^M (y^{(m)} - y)^2$$

$$\Delta w_{ij}^1 = -\alpha \frac{\partial C(w_{ij}^1)}{\partial w_{ij}^1} = -\alpha \sum_{m=1}^{M} 2(y - y^{(m)}) \frac{\partial y}{\partial w_{ij}^1}$$

$$\frac{\partial y}{\partial w_{ij}^1} = \frac{\partial g}{\partial s_2} \frac{\partial s_2}{\partial w_{ij}^1} = \frac{\partial g}{\partial s_2} w_j^2 \frac{\partial h_j}{\partial w_{ij}^1} = \frac{\partial g}{\partial s_2} w_j^2 \frac{\partial g}{\partial s_1} x_i^{(m)}$$



Two layer network



$$y_k = g\left(\sum_{j=0}^{N_1} w^2_{jk} h^1_j\right) \qquad h^1_j = g\left(\sum_{i=0}^{N} w^1_{ij} x_i\right)$$

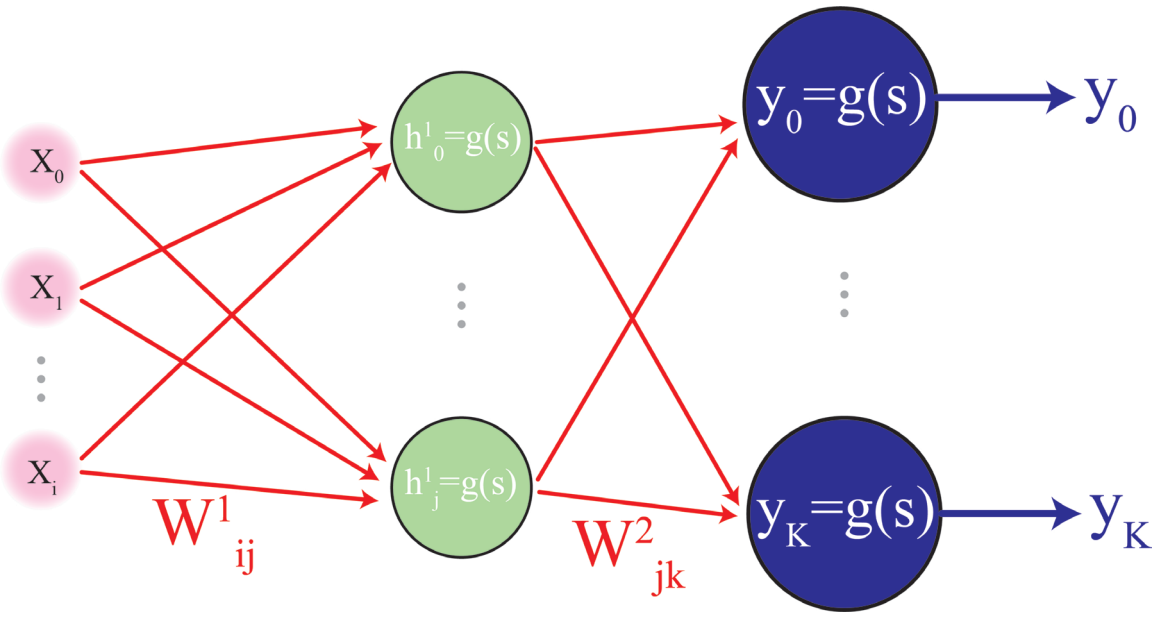
Training set: $\left\{x^{(m)}, y^{(m)}\right\} \quad m = 1, M \quad y^{(m)} = \pm 1$

Soft max

$$g(s_{k_0}) = \frac{e^{s_{k_0}}}{\sum_{k}^{N_2} e^{s_k}} \qquad \sum_{k_0}^{N_2} \frac{e^{s_{k_0}}}{\sum_{k}^{N_2} e^{s_k}} = 1$$

http://neuralnetworksanddeeplearning.com/chap3.html

Two layers: Cross entropy and softmax



$$s_j^1 = \sum_i^N x_i^{(m)} w_{ij}^1$$

$$h_j^1 = g_1(s_j^1) = \frac{1}{1+e^{-s_j^1}}$$

$$s_k^2 = \sum_j^{N_1} h_j^1 w_{jk}^2$$

$$y_k = g_2(s_k^2) = \frac{e^{s_k^2}}{\sum_p^K e^{s_p^2}}$$

Two layers: Cross entropy and softmax

The cross entropy cost function is: $$C = -\sum_{k}^{K} y_k^{(m)} \ln(y_k) \quad (1)$$

Weights of the second layer:

$$\frac{\partial C}{\partial s_n^2} = -\sum_{k}^{K} y_k^{(m)} \frac{\partial \ln(y_k)}{\partial s_n^2} \quad (2)$$

$$y_k = \frac{e^{s_k^2}}{\sum_{p}^{K} e^{s_p^2}} \Rightarrow \ln(y_k) = s_k^2 - \ln(\sum_{p}^{K} e^{s_p^2}) \Rightarrow \frac{\partial \ln(y_k)}{\partial s_n^2} = \delta_{kn} - \frac{e^{s_n^2}}{\sum_{p}^{K} e^{s_p^2}} = \delta_{kn} - y_n \quad (3)$$

$$(2),(3) \Rightarrow \frac{\partial C}{\partial s_n^2} = -\sum_{k}^{K} y_k^{(m)} (\delta_{kn} - y_n) = -y_n^{(m)} + y_n \sum_{k}^{K} y_k^{(m)} \quad (4)$$

$$\mathrm{y}^{(m)} \text{ is one-hot } \Rightarrow \sum_{k}^{K} y_k^{(m)} = 1 \quad (5)$$

$$(4),(5) \Rightarrow \frac{\partial C}{\partial s_n^2} = y_n - y_n^{(m)} \quad (6)$$

$$(6) \Rightarrow \frac{\partial C}{\partial s_k^2} = y_k - y_k^{(m)} \quad (7)$$

Two layers: Cross entropy and softmax

Weights of the second layer:

$$\frac{\partial C}{\partial w_{rs}^2} = \sum_k^K \frac{\partial C}{\partial s_k^2} \frac{\partial s_k^2}{\partial w_{rs}^2} \quad (8)$$

$$\frac{\partial s_k^2}{\partial w_{rs}^2} = \sum_j^{N_1} h_j^1 \frac{\partial w_{jk}^2}{\partial w_{rs}^2} = \sum_j^{N_1} h_j^1 \delta_{jr} \delta_{ks} = h_r^1 \delta_{ks} \quad (9)$$

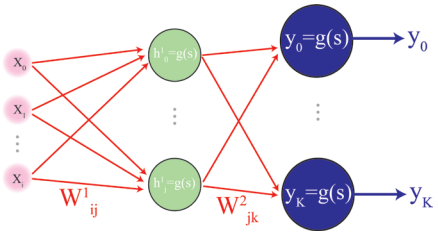
$$(7),(8),(9) \Rightarrow \frac{\partial C}{\partial w_{rs}^2} = \sum_k^K (y_k - y_k^{(m)}) h_r^1 \delta_{ks} = (y_s - y_s^{(m)}) h_r^1$$

$$\Rightarrow \frac{\partial C}{\partial w_{jk}^2} = (y_k - y_k^{(m)}) h_j^1 \quad (10)$$

The weights of the second layer are updated by the following equation:

Sum over m (samples) for batch update!!

$$w_{jk}^{2\ new} = w_{jk}^{2\ old} - \alpha \frac{\partial C}{\partial w_{jk}^2}$$



Two layers: Cross entropy and softmax

Weights of the first layer:

$$\frac{\partial C}{\partial w_{ij}^1} = \frac{\partial C}{\partial s_j^1}\frac{\partial s_j^1}{\partial w_{ij}^1} = \frac{\partial C}{\partial h_j^1}\frac{\partial h_j^1}{\partial s_j^1}\frac{\partial s_j^1}{\partial w_{ij}^1} = \left(\sum_k^K \frac{\partial C}{\partial s_k^2}\frac{\partial s_k^2}{\partial h_j^1}\right)\frac{\partial h_j^1}{\partial s_j^1}\frac{\partial s_j^1}{\partial w_{ij}^1} \quad (11)$$

$$\frac{\partial s_j^1}{\partial w_{ij}^1} = x_i^{(m)} \quad (12)$$

$$\frac{\partial h_j^1}{\partial s_j^1} = h_j^1(1-h_j^1) \quad (13)$$

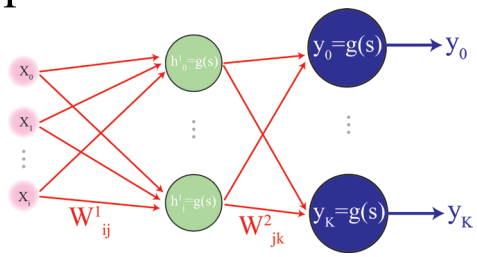
$$\frac{\partial s_k^2}{\partial h_j^1} = w_{jk}^2 \quad (14)$$

$$(7),(11),(12),(13),(14) \Rightarrow \frac{\partial C}{\partial w_{ij}^1} = \left(\sum_k^K (y_k - y_k^{(m)}) w_{jk}^2\right) h_j^1(1-h_j^1)x_i^{(m)}$$

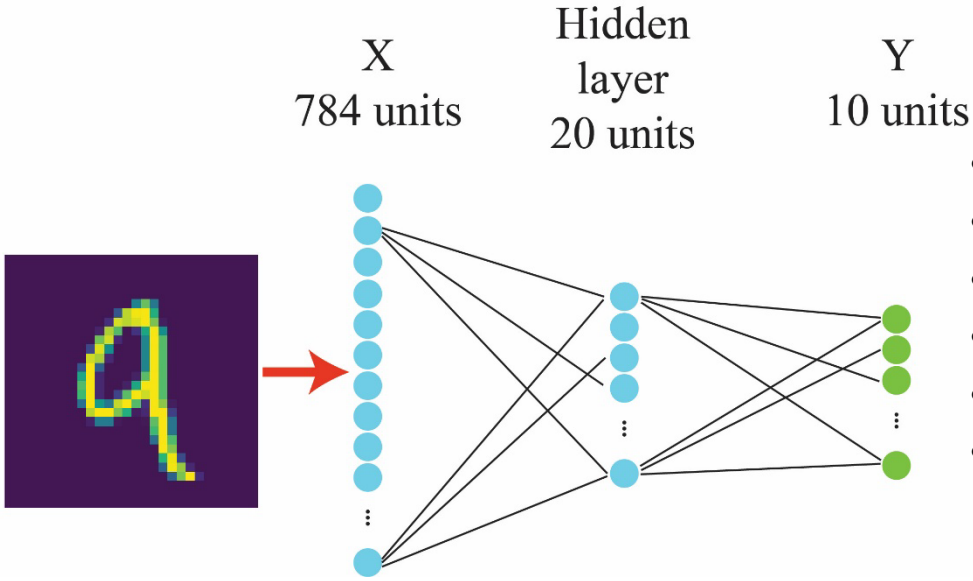
The weights of the first layer are updated by the following equation:

Sum over m (samples) for batch update!!

$$w_{ij}^{1\,new} = w_{ij}^{1\,old} - \alpha \frac{\partial C}{\partial w_{ij}^1}$$

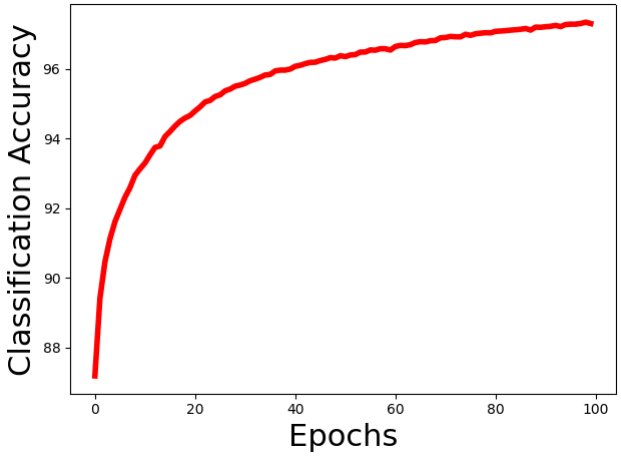


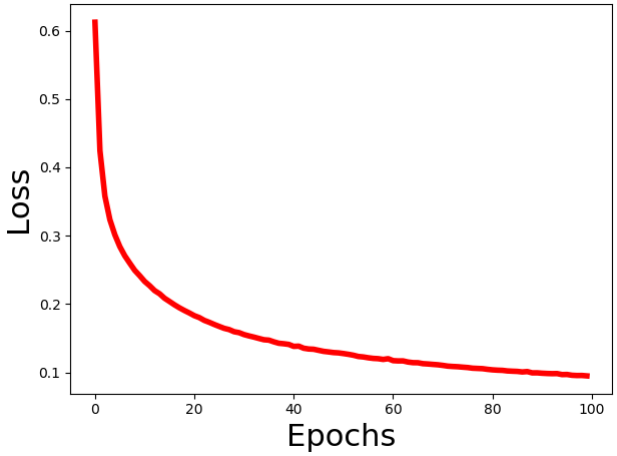
10 output units MNIST handwritten



- Cross entropy cost function
- Hidden layer activation function: Sigmoid
- Final layer activation function: Softmax
- Learning rate: 1
- Batch size: 1000
- Epochs: 100

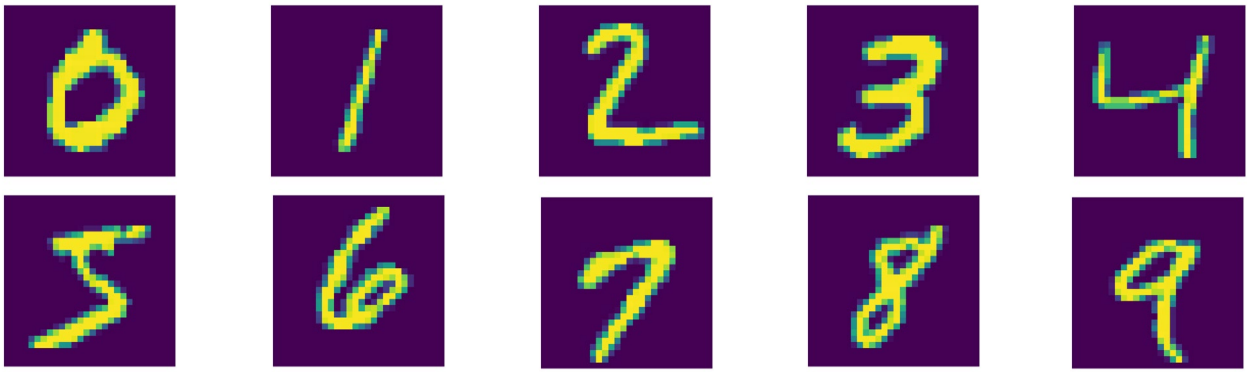
$$w_{jk}^{2\ new} = w_{jk}^{2\ old} - \alpha \frac{\partial C}{\partial w_{jk}^{2}}$$





Training accuracy: 97.31%
Test accuracy: 95.85%

10 output units MNIST handwritten: one layer



Ten neurons classification

10 ADALINE's – Training accuracy	85.00%
10 ADALINE's – Test accuracy	80.83%

Higher accuracy can be reached with more iterations. However, the training is very slow that needs a lot of iterations.

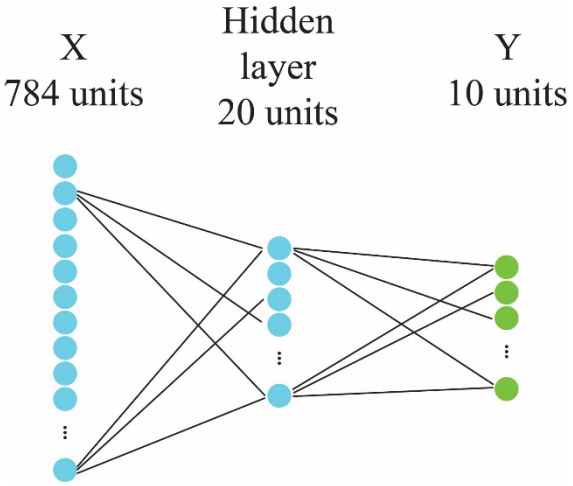
10000 training samples
10000 test samples

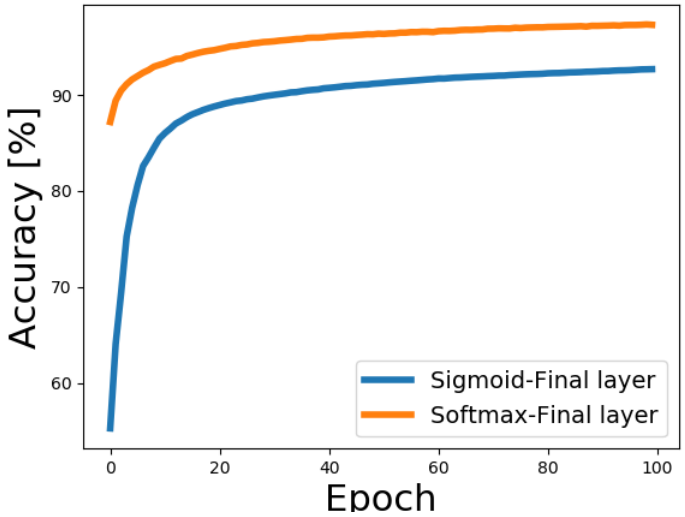
Sigmoid activation function

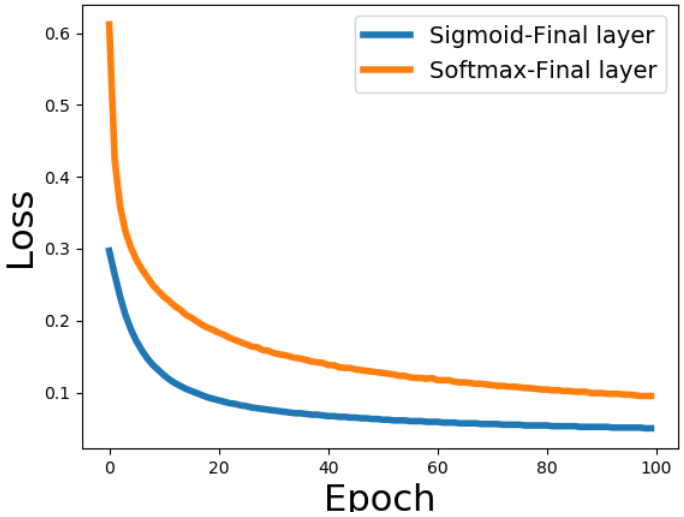
Softmax vs Sigmoid in final layer

- Cross entropy cost function
- Hidden layer activation function: Sigmoid
- Learning rate: 1
- Batch size: 1000
- Epochs: 100

- Final layer activation function: Softmax or Sigmoid



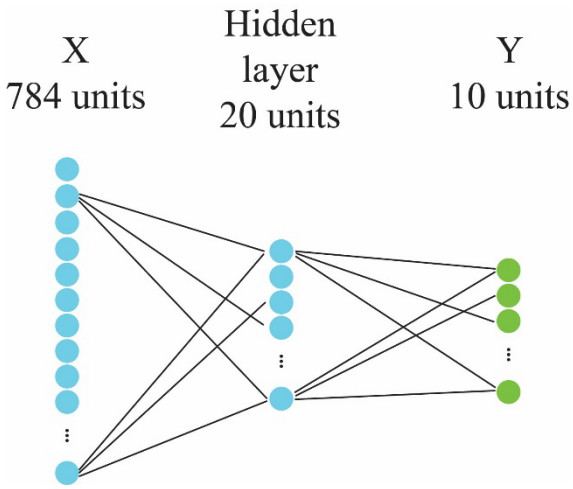


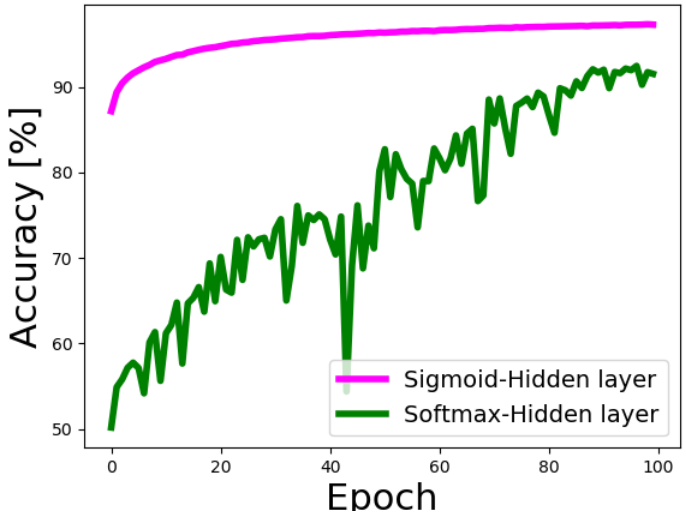


Softmax vs Sigmoid in hidden layer

- Cross entropy cost function
- Final layer activation function: Softmax
- Learning rate: 1
- Batch size: 1000
- Epochs: 100

- Hidden layer activation function: Sigmoid or Softmax





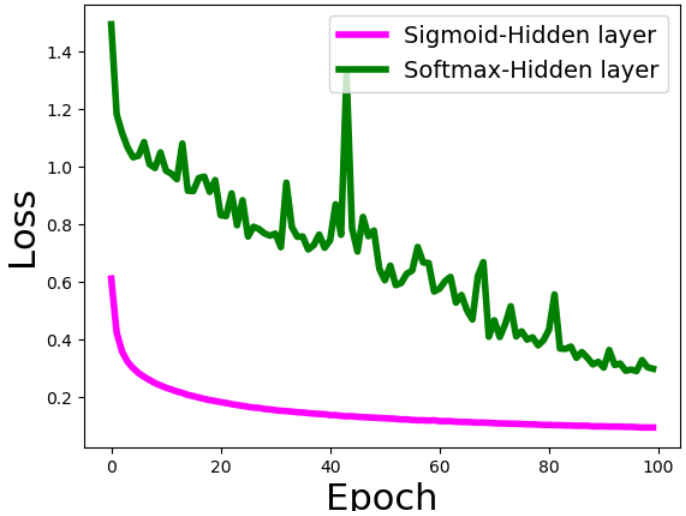
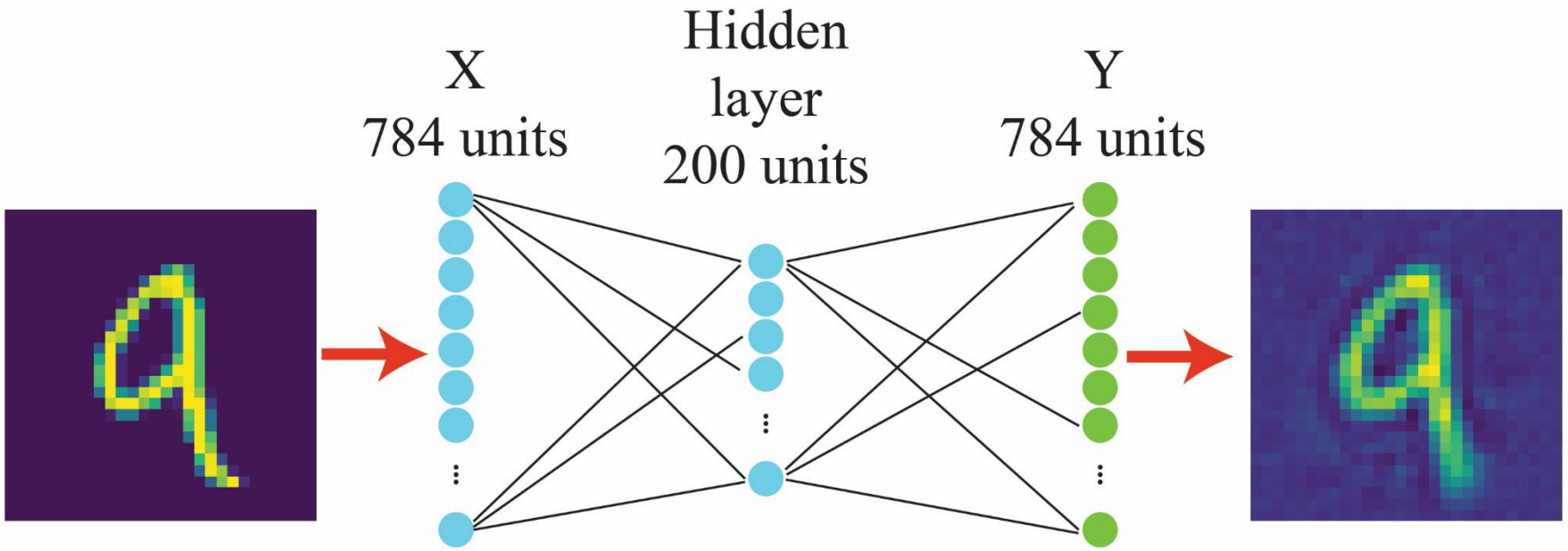
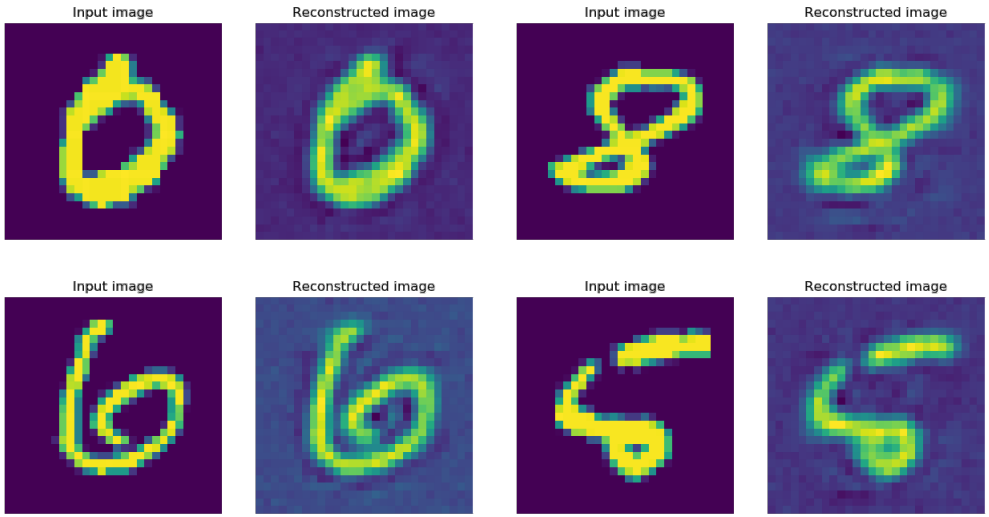


Image reconstruction



- Sigmoid activation function
- Mean squared error cost function
- 200 hidden units
- 60000 training sample

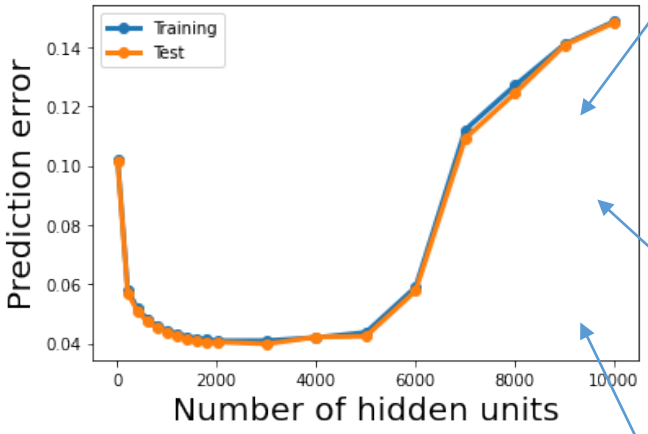


Number of hidden units

Average error of reconstruction as a function of number of hidden units after 10 epochs

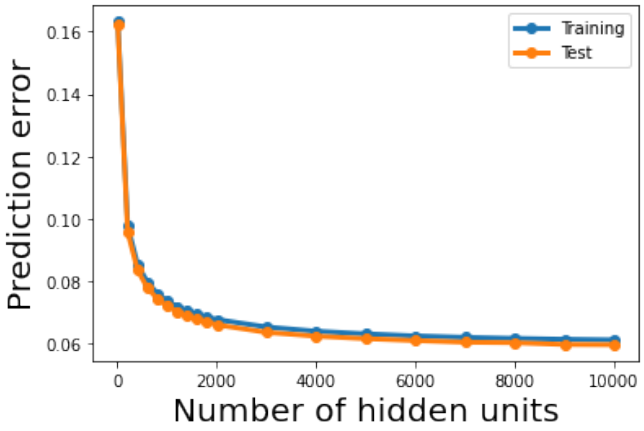
First layer activation: Sigmoid

Second layer: No activation

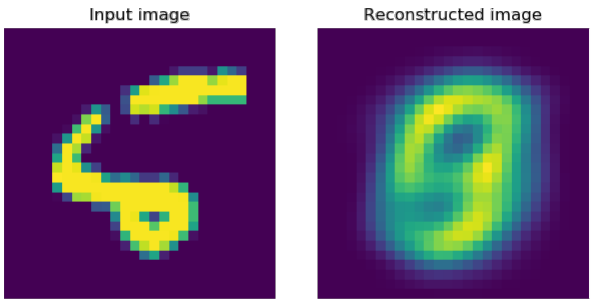


First layer activation: Sigmoid

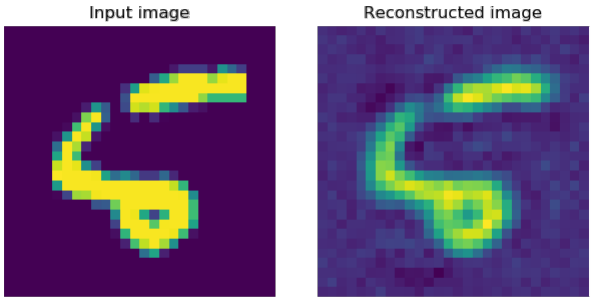
Second layer activation: Sigmoid



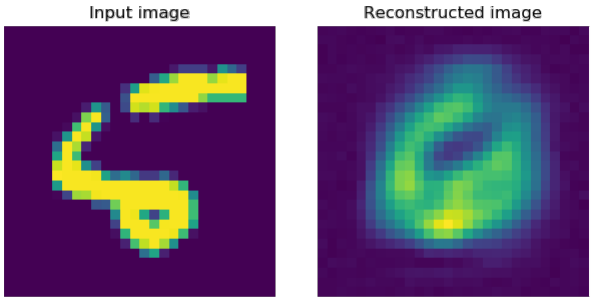
2 hidden units



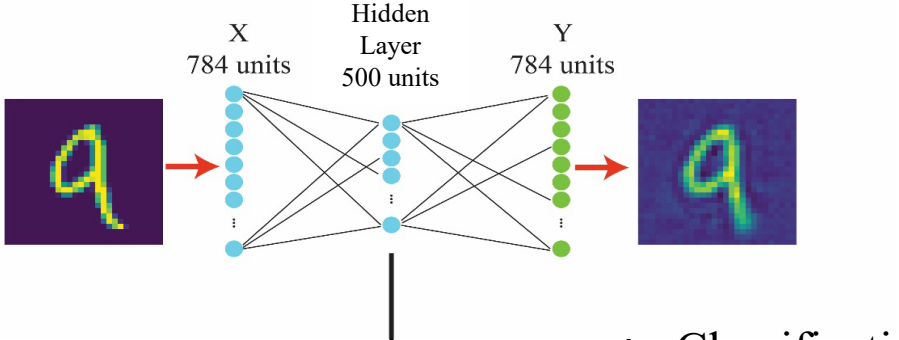
256 hidden units



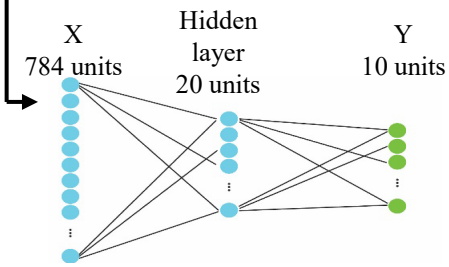
8000 hidden units



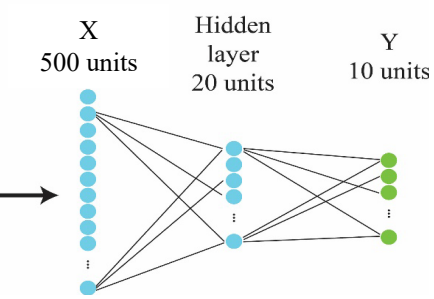
Classification using hidden layer

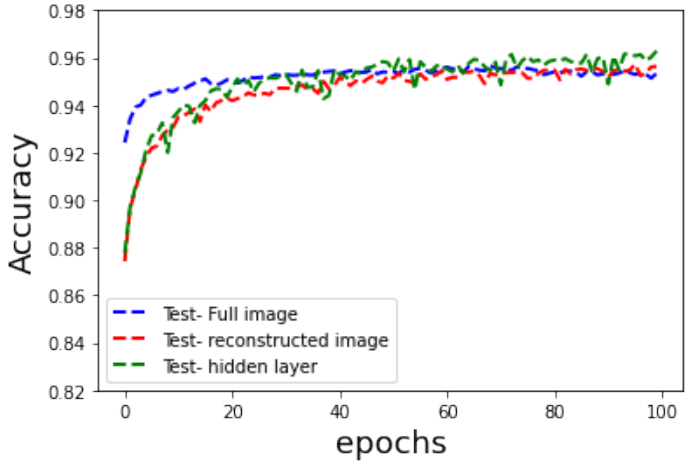


Classification using reconstructed image

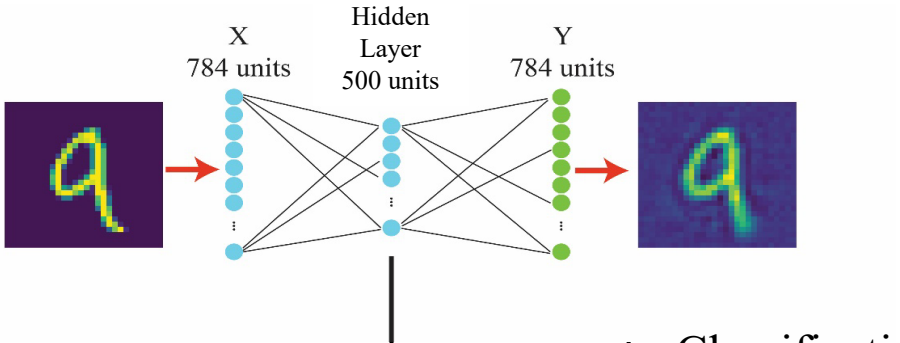


Classification using hidden layer

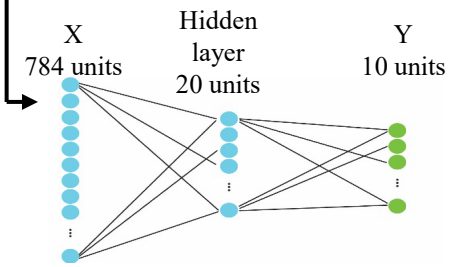




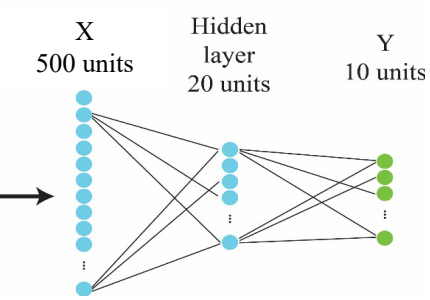
Classification using hidden layer

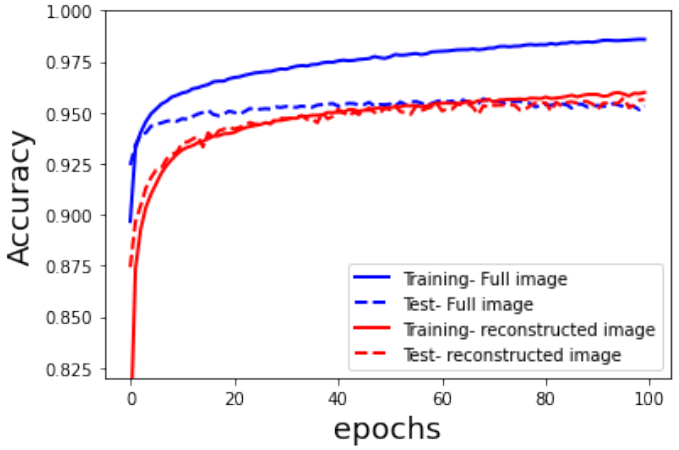


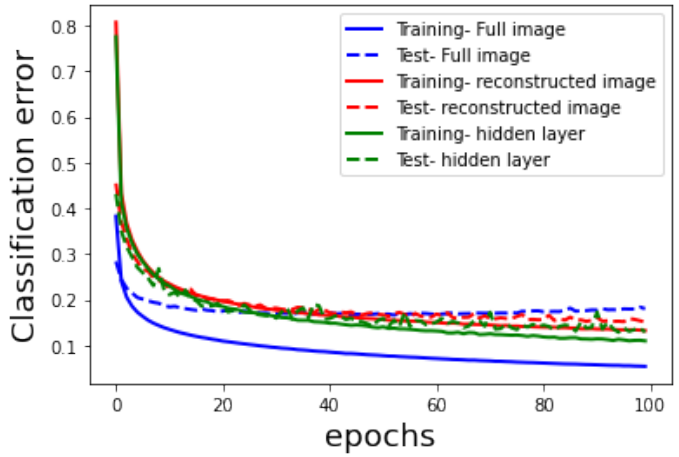
Classification using reconstructed image



Classification using hidden layer

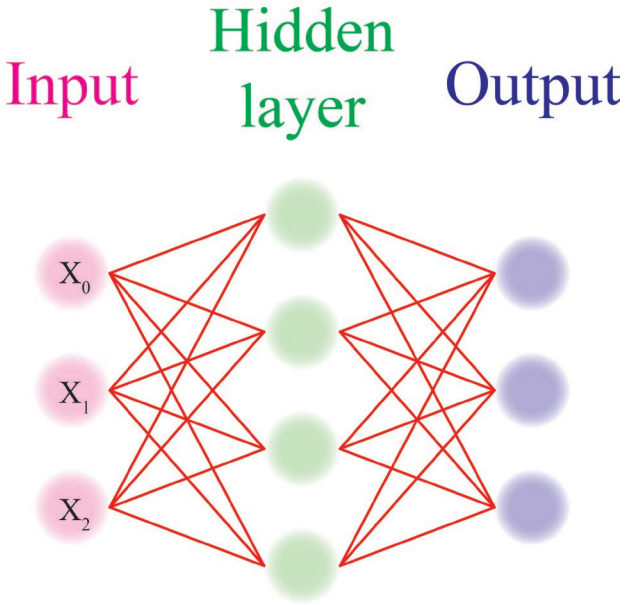




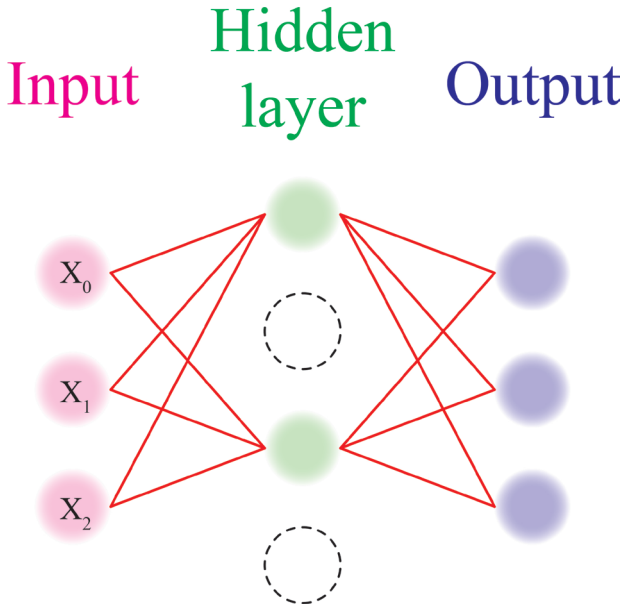


Dropout

Standard network



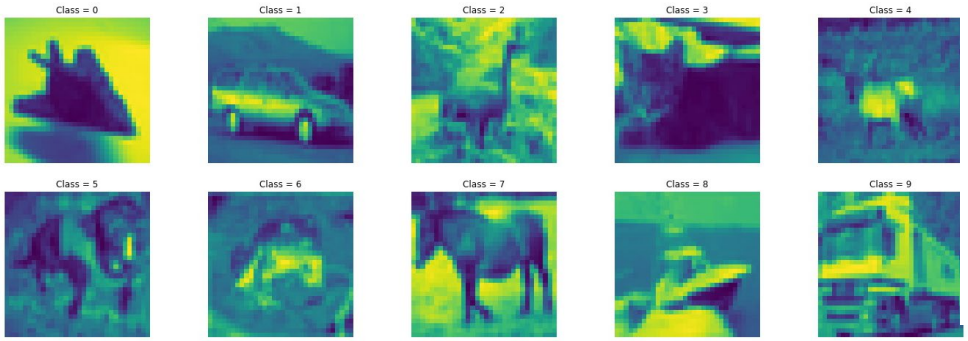
After applying dropout

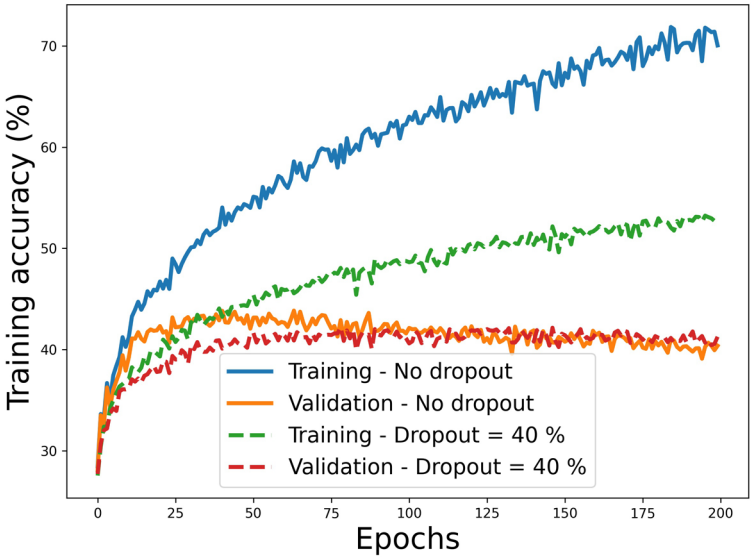


- Dropout: ignoring random units in layers during the training.
- We use dropout to prevent over-fitting.

Dropout

MNIST fashion

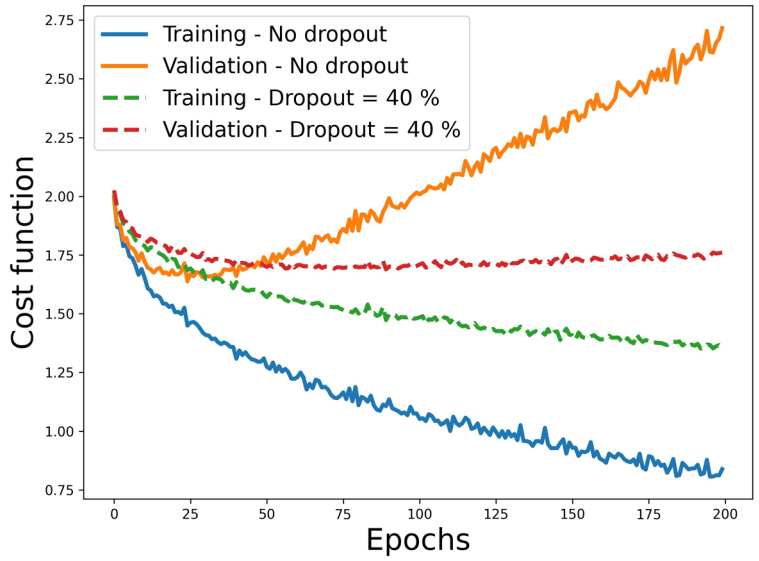




- Training samples: 40000
- Validation samples: 10000
- Test samples: 10000
- Number of pixels: 32x32

Two-layer network structure:

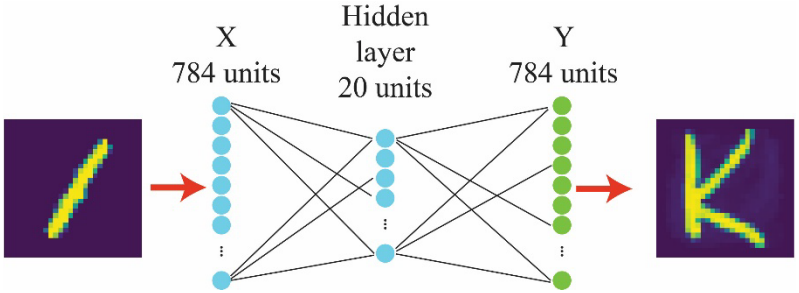
- Number of hidden units: 200
- Middle layer activation function: ReLU
- Final layer activation function: Softmax
- Optimizer: Adam
- Cost function: Cross entropy
- Learning rate: 0.001



Test accuracy (no dropout): 38.79 %

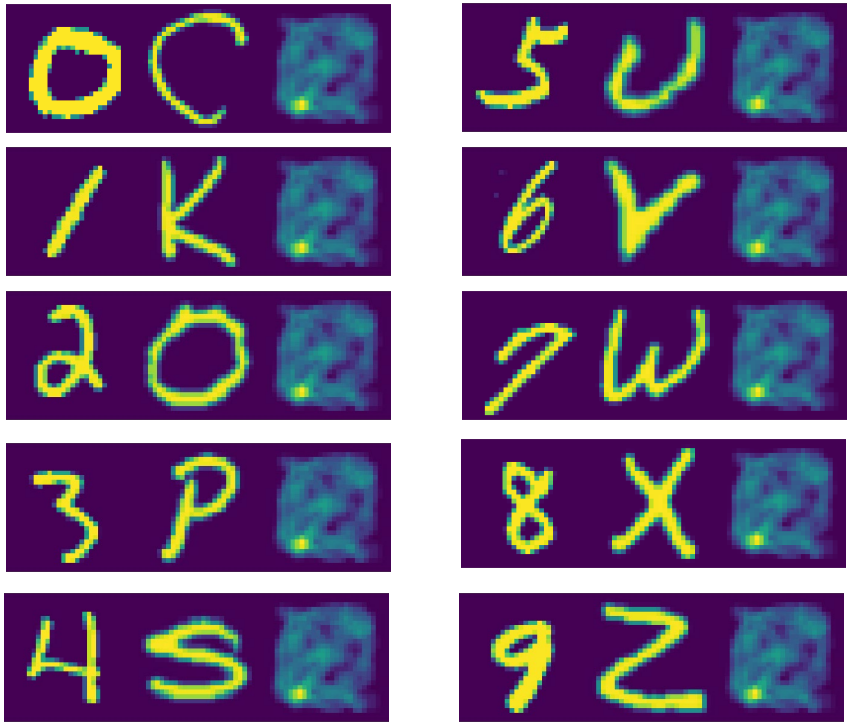
Test accuracy (dropout = 40 %): 39.42 %

Heteroassociative memory: 20 hidden units



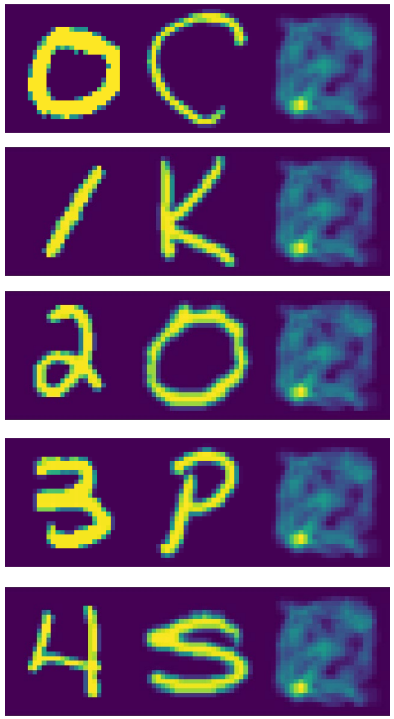
Dropout

Input Target Output

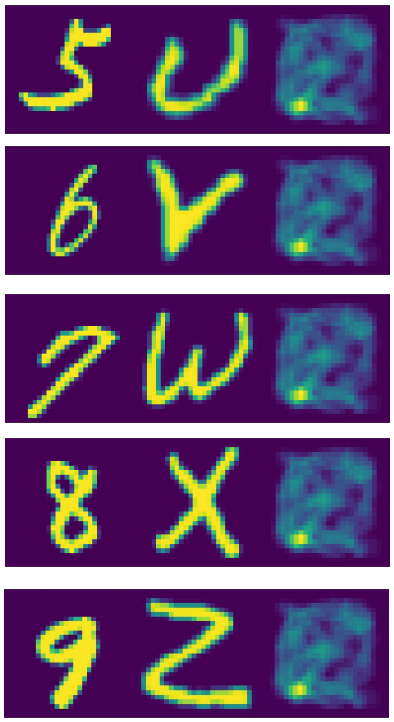


Input Target Output

Input Target Output



Input Target Output

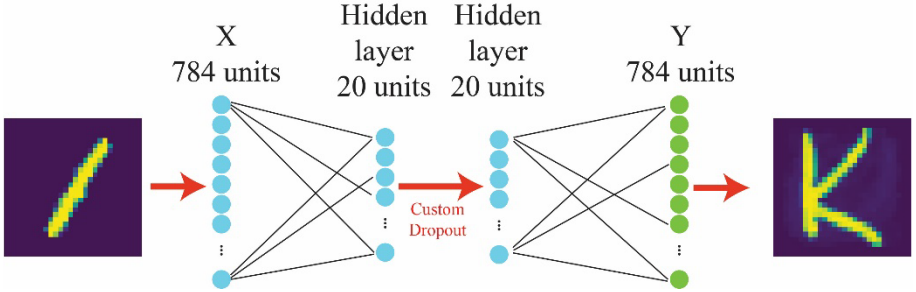


Training error: 41.39 %
Test error: 43.75 %

Training error: 45.42 %
Test error: 48.02 %

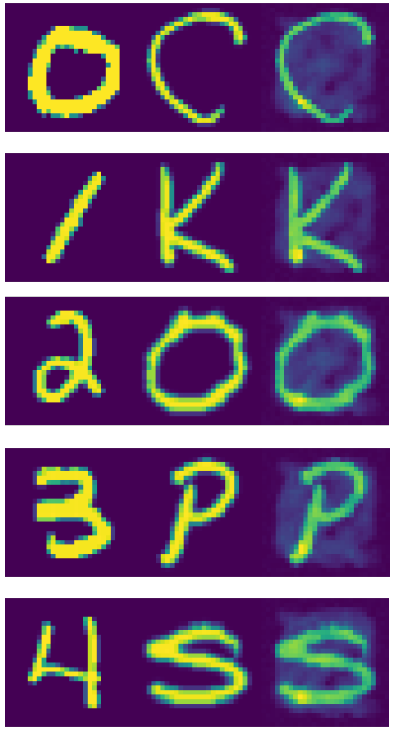
Heteroassociative memory: 20 hidden units

Custom Dropout

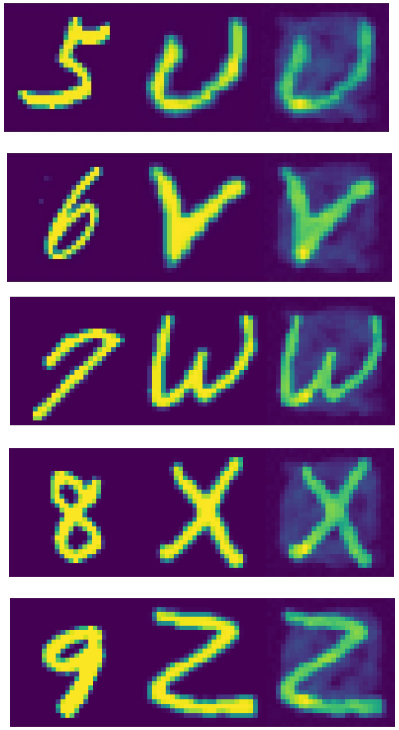


Use the weights from custom dropout as initial weights in the network without dropout

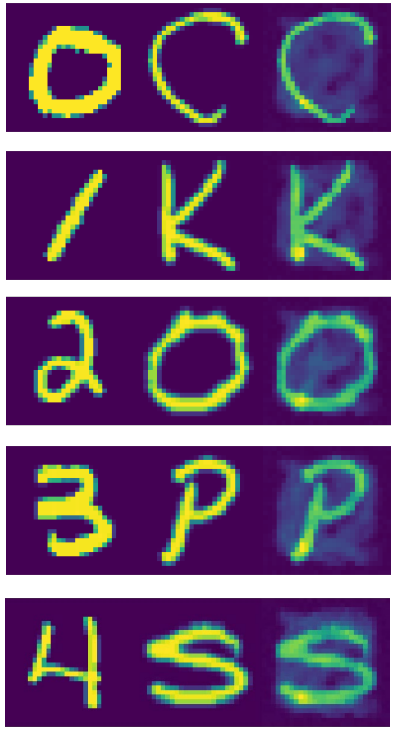
Input Target Output



Input Target Output



Input Target Output



Input Target Output