Autoencoders

CE/CZ4042 – Tutorial 10

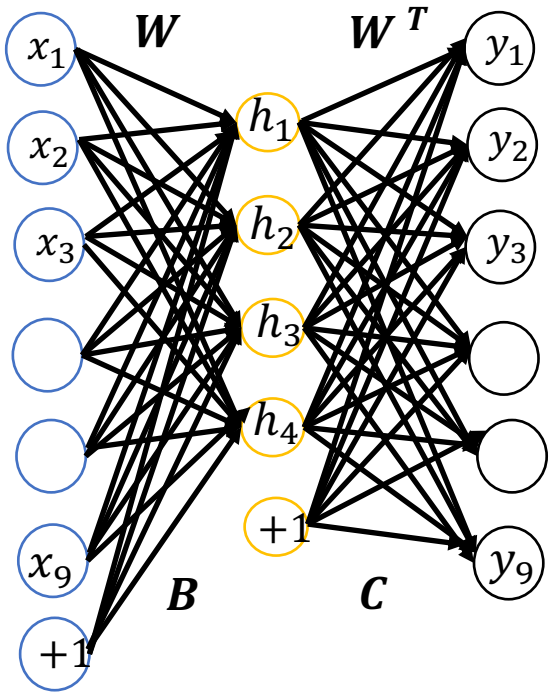
1. Given five binary patterns:

$$\boldsymbol{x}_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{x}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Design an autoencoder with four hidden neurons to reconstruct the patterns.

Use gradient descent learning with $\alpha = 0.1$ to learning curves. Plot the weights and the hidden-layer activations and reconstructions of the input patterns.

Repeat the above by introducing a sparsity constraint with a penalty parameter $\beta = 0.5$ and sparsity parameter $\rho = 0.1$.

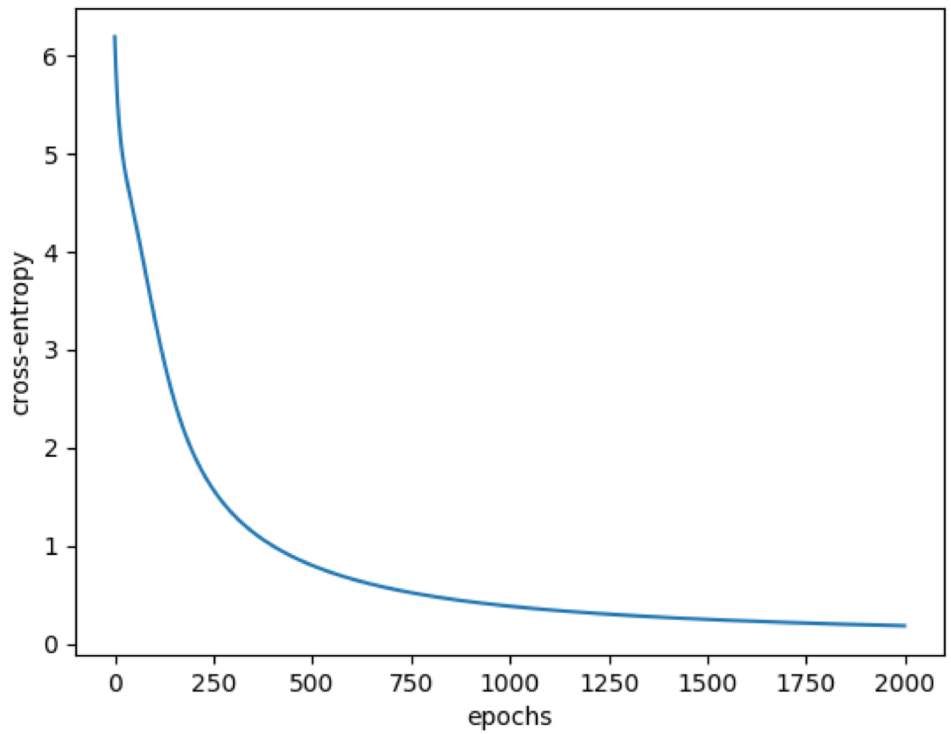


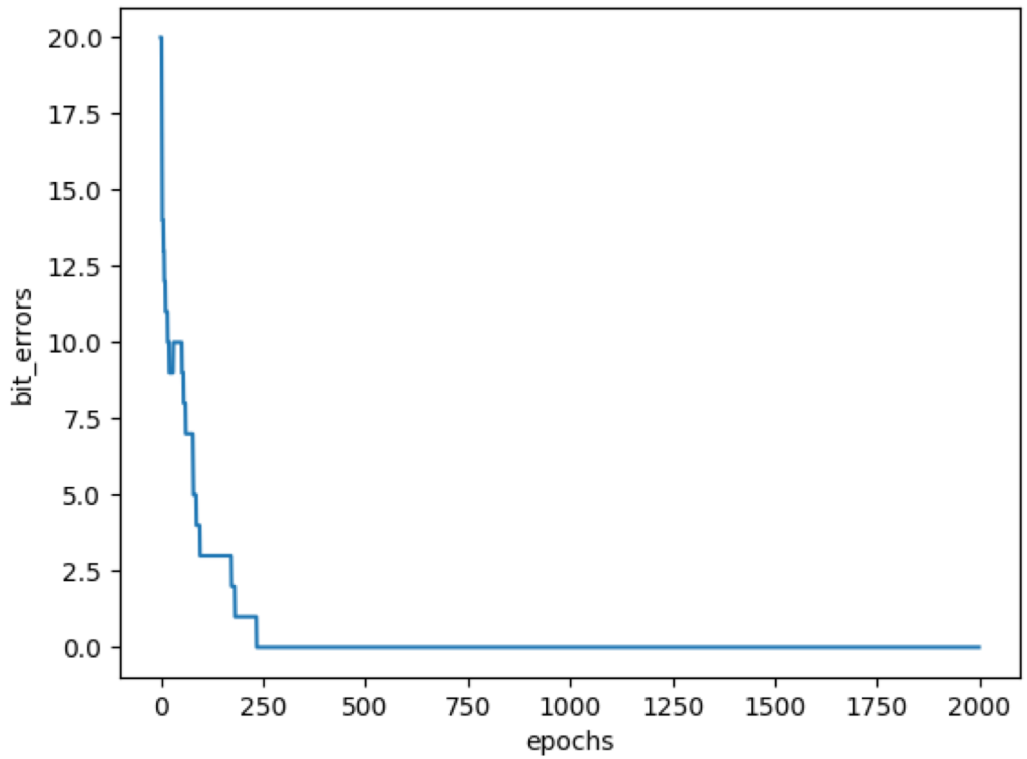
$$\boldsymbol{x}_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \boldsymbol{x}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{x}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\boldsymbol{H} = f(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{B})$$
$$\boldsymbol{Y} = f(\boldsymbol{H}\boldsymbol{W}^T + \boldsymbol{C})$$

$$J = -\sum_{p=1}^{P} \left(\boldsymbol{x}_p \log \boldsymbol{y}_p + \left(1 - \boldsymbol{x}_p\right) \log\left(1 - \boldsymbol{y}_p\right)\right)$$

$$\boldsymbol{W} = \boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} J$$
$$\boldsymbol{b} = \boldsymbol{b} - \alpha \nabla_{\boldsymbol{b}} J$$
$$\boldsymbol{c} = \boldsymbol{c} - \alpha \nabla_{\boldsymbol{c}} J$$

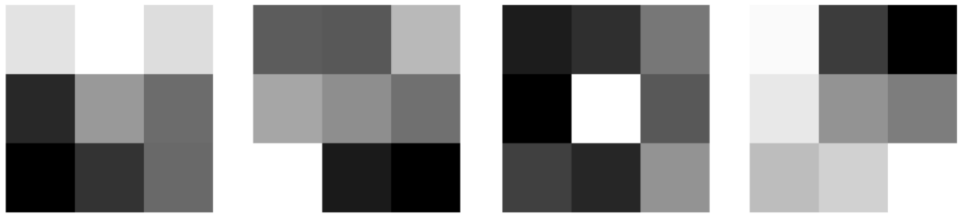

cross-entropy
epochs
0
1
2
3
4
5
6
0
250
500
750
1000
1250
1500
1750
2000


bit_errors
epochs
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
250
500
750
1000
1250
1500
1750
2000

Learned weights:

$$\boldsymbol{w}_1 = \begin{pmatrix} 2.71 & 3.76 & 2.48 \\ -4.26 & -0.05 & -1.72 \\ -5.75 & -3.82 & -1.79 \end{pmatrix}, \boldsymbol{w}_2 = \begin{pmatrix} -3.63 & -2.71 & 1.91 \\ 1.05 & 0.0 & -1.32 \\ 5.02 & -5.19 & -6.34 \end{pmatrix},$$

$$\boldsymbol{w}_3 = \begin{pmatrix} -3.63 & -2.71 & 0.83 \\ -5.02 & 7.47 & -0.72 \\ -1.88 & -3.14 & 2.24 \end{pmatrix}, \boldsymbol{w}_4 = \begin{pmatrix} 3.97 & -4.21 & -6.89 \\ 3.18 & -0.5 & -1.43 \\ 1.34 & 2.21 & 4.22 \end{pmatrix}$$



x	h	y
$x_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$h_1 = \begin{pmatrix} 1.00 \\ 0.01 \\ 0.00 \\ 0.00 \end{pmatrix}$	$y_1 = \begin{pmatrix} 0.99 & 0.96 & 0.96 \\ 0.02 & 0.03 & 0.0 \\ 0.03 & 0.01 & 0.03 \end{pmatrix}$
$x_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$h_2 = \begin{pmatrix} 0.0 \\ 0.99 \\ 0.0 \\ 0.99 \end{pmatrix}$	$y_2 = \begin{pmatrix} 0.98 & 0.0 & 0.01 \\ 0.98 & 0.02 & 0.0 \\ 1.0 & 0.06 & 0.04 \end{pmatrix}$
$x_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$h_3 = \begin{pmatrix} 0.08 \\ 1.0 \\ 1.0 \\ 0.0 \end{pmatrix}$	$y_3 = \begin{pmatrix} 0.03 & 0.0 & 0.97 \\ 0.02 & 0.98 & 0.0 \\ 0.99 & 0.0 & 0.0 \end{pmatrix}$
$x_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$h_4 = \begin{pmatrix} 0.85 \\ 0.0 \\ 0.99 \\ 1.0 \end{pmatrix}$	$y_4 = \begin{pmatrix} 0.99 & 0.02 & 0.03 \\ 0.01 & 0.97 & 0.0 \\ 0.02 & 0.02 & 0.99 \end{pmatrix}$
$x_5 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$h_5 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{pmatrix}$	$y_5 = \begin{pmatrix} 1.0 & 0.01 & 0.0 \\ 0.97 & 0.02 & 0.01 \\ 0.95 & 0.91 & 0.96 \end{pmatrix}$

Sparse autoencoder:

$$J = -\sum_{p=1}^{P} \left(\boldsymbol{x}_p \log \boldsymbol{y}_p + \left(1 - \boldsymbol{x}_p\right) \log\left(1 - \boldsymbol{y}_p\right) \right)$$

$$J_1 = J + \beta \Omega(\boldsymbol{H})$$

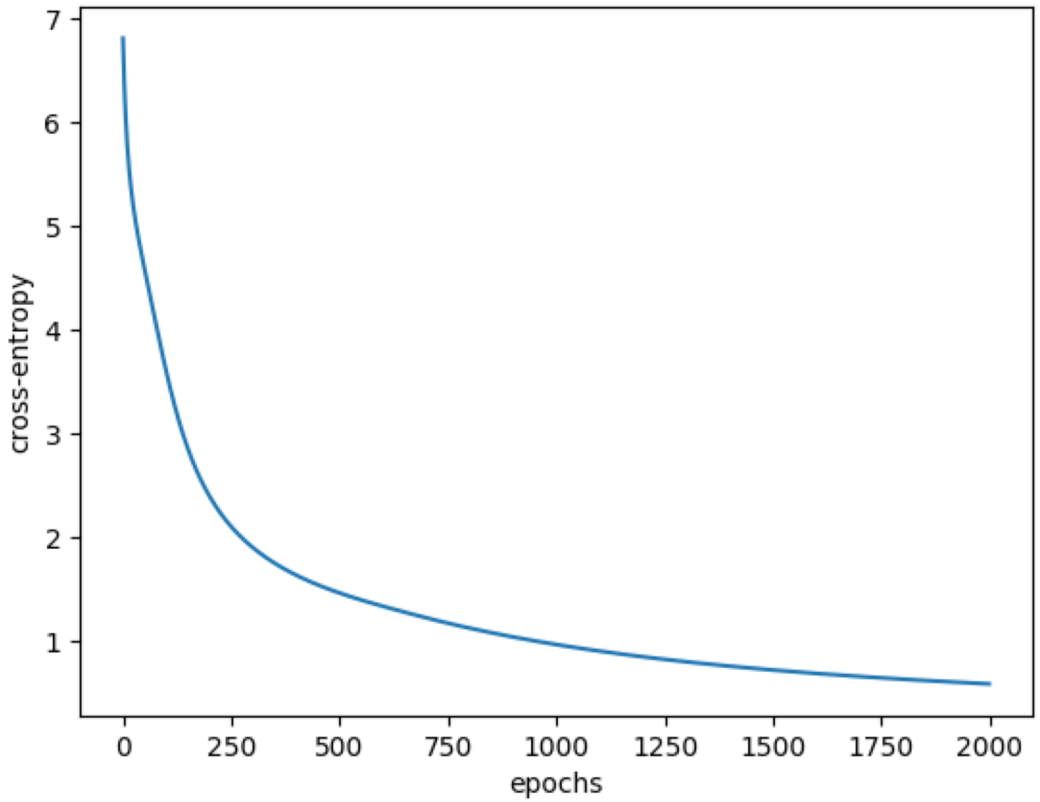
Where the sparsity constraint:

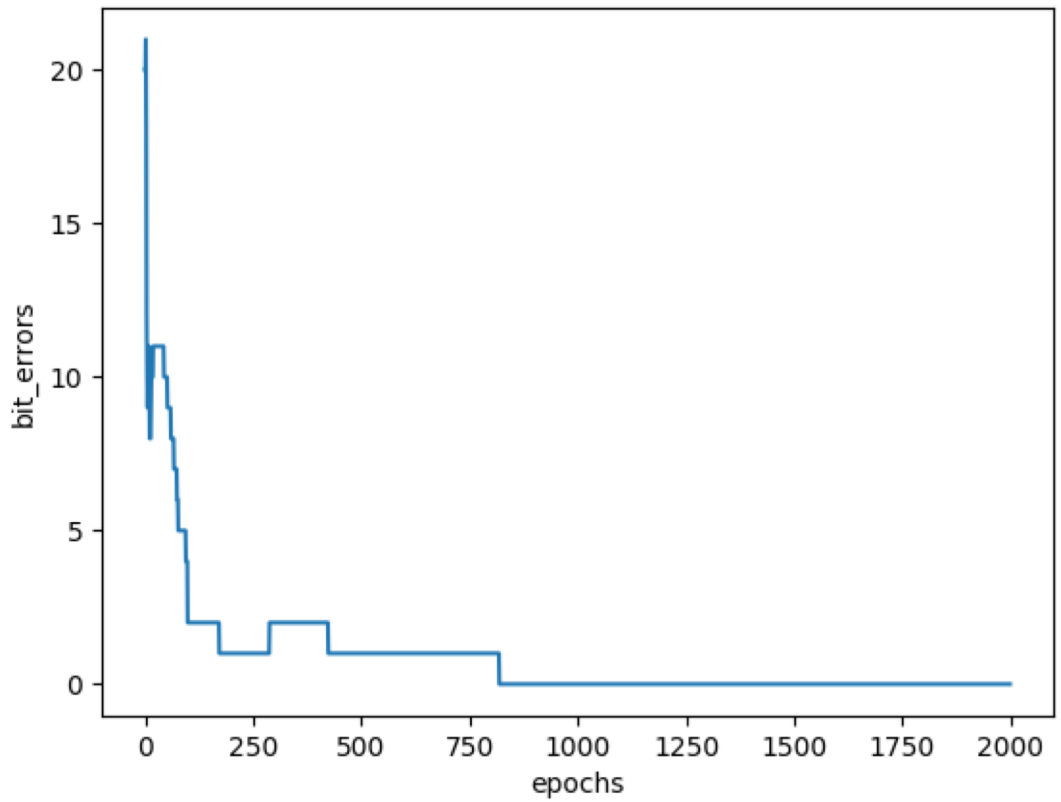
$$\Omega(\boldsymbol{H}) = \sum_{j=1}^{M} \rho \log \frac{\rho}{\rho_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \rho_j}$$

$$\rho_j = \frac{1}{P} \sum_{p=1}^{P} h_{pj} = \frac{1}{P} \sum_{p=1}^{P} f\left(\boldsymbol{x}_p^T \boldsymbol{w}_j + b_j\right)$$

```
entropy = -tf.reduce_mean(tf.reduce_sum(x * tf.log(y) + (1 - x) * tf.log(1 - y), axis=1))
divergence = tf.reduce_sum(rho*tf.log(rho/tf.reduce_mean(h, axis=0)) +
                           (1 - rho)*tf.log((1 - rho)/(1-tf.reduce_mean(h, axis=0))))

loss = entropy + beta*divergence
```

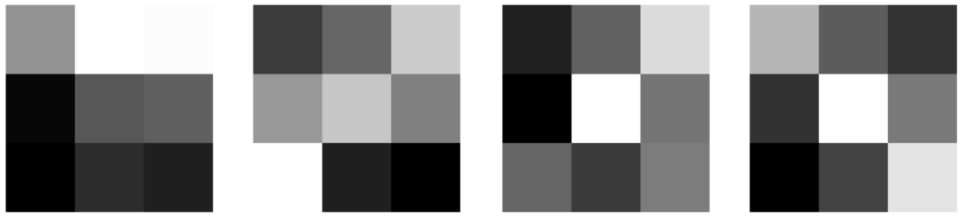

cross-entropy
epochs
7
6
5
4
3
2
1
0
250
500
750
1000
1250
1500
1750
2000


bit_errors
epochs
20
15
10
5
0
0
250
500
750
1000
1250
1500
1750
2000

Learned weights:

$$\boldsymbol{w}_1 = \begin{pmatrix} 1.18 & 6.19 & 6.01 \\ -5.45 & -1.69 & -1.32 \\ -5.75 & -3.69 & -4.26 \end{pmatrix}, \boldsymbol{w}_2 = \begin{pmatrix} -3.69 & -1.99 & 2.00 \\ -0.03 & 1.81 & -0.95 \\ 4.08 & -4.81 & -6.06 \end{pmatrix},$$

$$\boldsymbol{w}_3 = \begin{pmatrix} -4.13 & -1.36 & 3.77 \\ -5.55 & 5.31 & -0.62 \\ -1.24 & -3.07 & -0.28 \end{pmatrix}, \boldsymbol{w}_4 = \begin{pmatrix} 1.50 & -1.74 & -3.25 \\ -3.27 & 4.20 & -0.68 \\ -5.11 & -2.66 & 3.17 \end{pmatrix}$$



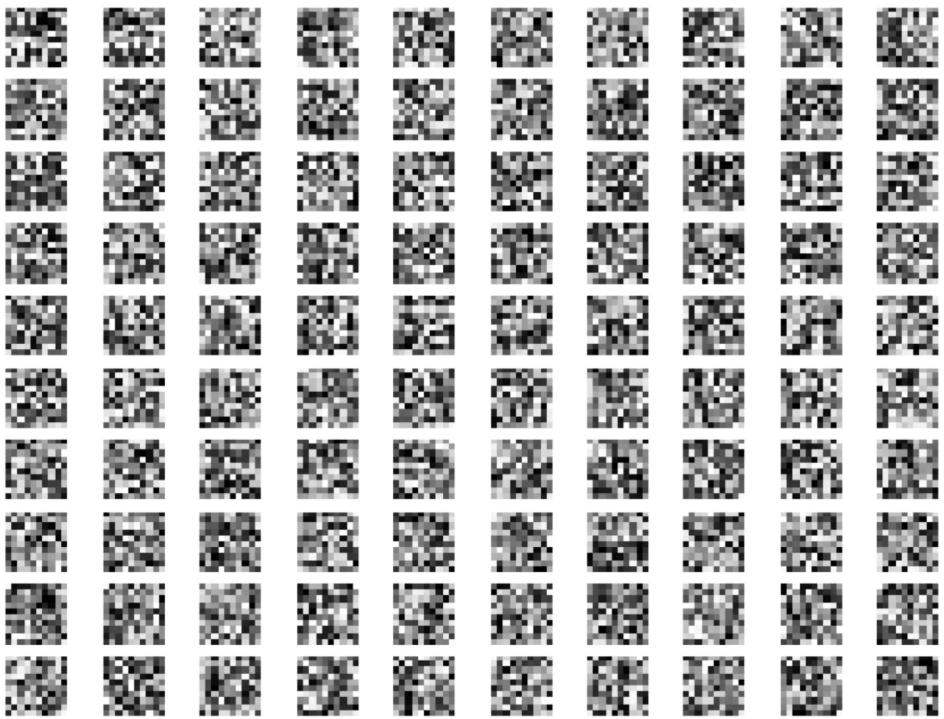
x	h	y
$x_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$h_1 = \begin{pmatrix} 1.00 \\ 0.03 \\ 0.07 \\ 0.00 \end{pmatrix}$	$y_1 = \begin{pmatrix} 1.0 & 0.95 & 0.94 \\ 0.04 & 0.0 & 0.0 \\ 0.04 & 0.05 & 0.05 \end{pmatrix}$
$x_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$h_2 = \begin{pmatrix} 0.0 \\ 0.65 \\ 0.0 \\ 0.0 \end{pmatrix}$	$y_2 = \begin{pmatrix} 0.92 & 0.01 & 0.1 \\ 0.94 & 0.07 & 0.0 \\ 0.99 & 0.12 & 0.09 \end{pmatrix}$
$x_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$h_3 = \begin{pmatrix} 0.11 \\ 1.0 \\ 1.0 \\ 0.0 \end{pmatrix}$	$y_3 = \begin{pmatrix} 0.06 & 0.0 & 0.95 \\ 0.03 & 0.96 & 0.0 \\ 0.99 & 0.0 & 0.0 \end{pmatrix}$
$x_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$h_4 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.52 \\ 1.0 \end{pmatrix}$	$y_4 = \begin{pmatrix} 0.98 & 0.0 & 0.0 \\ 0.03 & 0.96 & 0.0 \\ 0.03 & 0.04 & 0.99 \end{pmatrix}$
$x_5 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$h_5 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$y_5 = \begin{pmatrix} 0.99 & 0.05 & 0.03 \\ 0.94 & 0.02 & 0.01 \\ 0.92 & 0.76 & 0.83 \end{pmatrix}$

2. Create 100 images of 10x10 pixels by randomly generating pixel values between 0.0 and 1.0 from a uniform distribution.
   Design the following autoencoders to reconstruct the input patterns, using mean square error as the cost function:
   a. An undercomplete autoencoder with 49 hidden neurons
   b. An overcomplete autoencoder with 144 hidden neurons
   c. A sparse autoencoder with 144 hidden neurons and sparsity parameter $\rho = 0.05$ and penalty parameter $\beta = 0.5$.

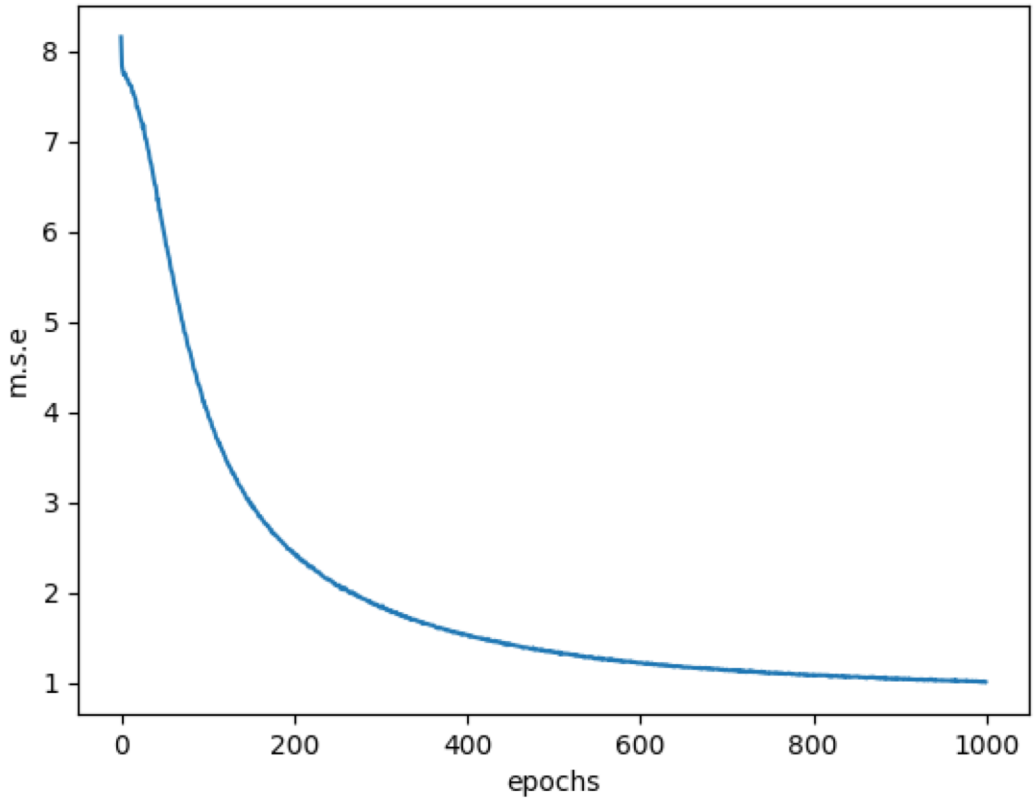
   Compare features learned by different autoencoders.

100 images of 10x10 size

X = **np.random.rand**(100, 10*10)

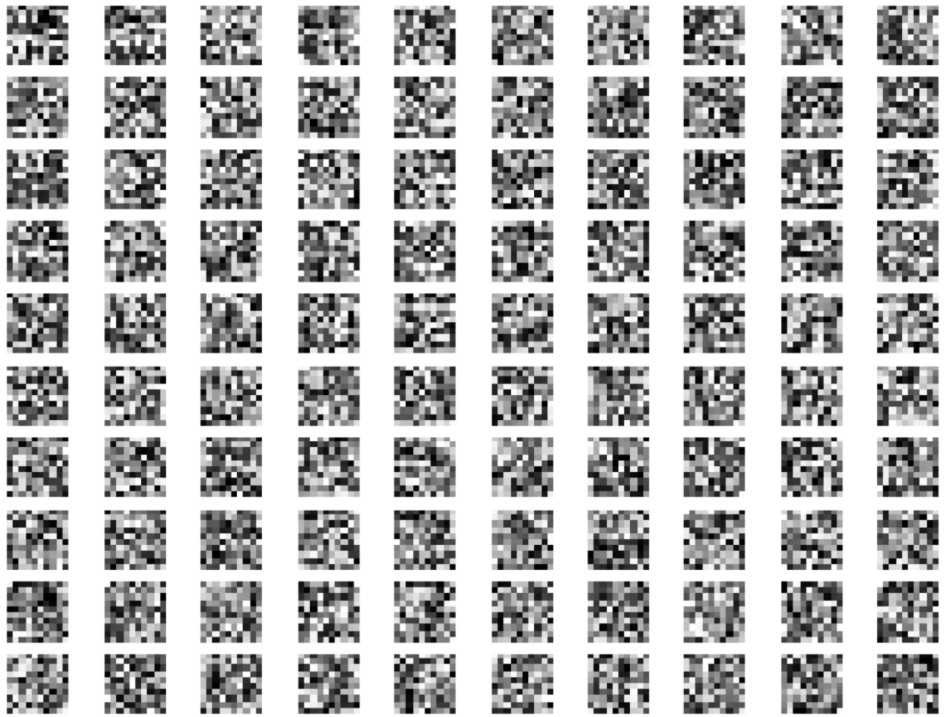


Undercomplete autoencoder: 49 hidden units

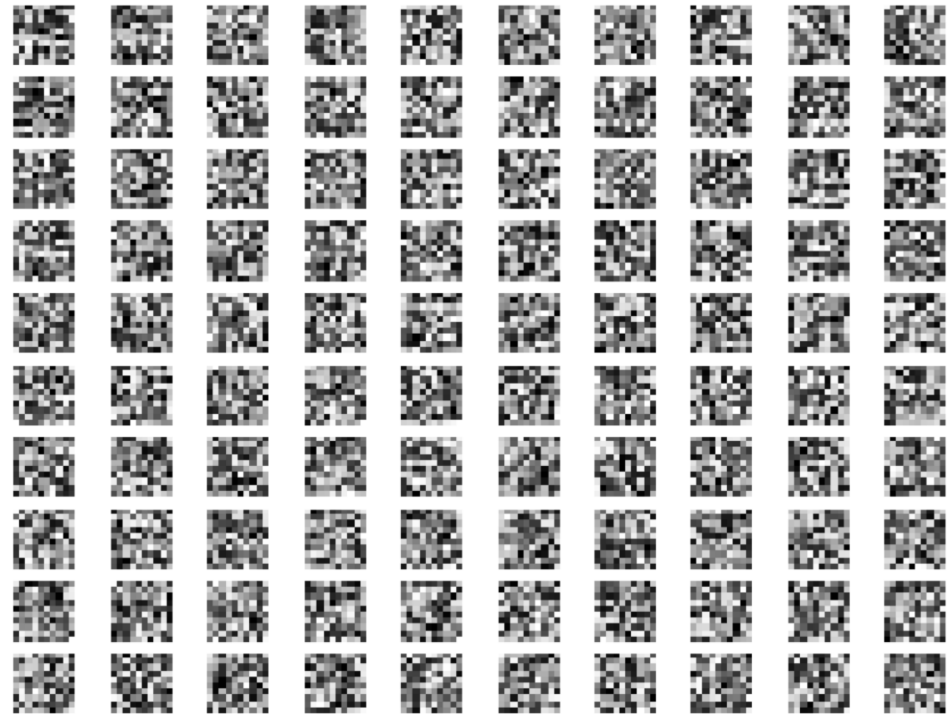


Weights learned

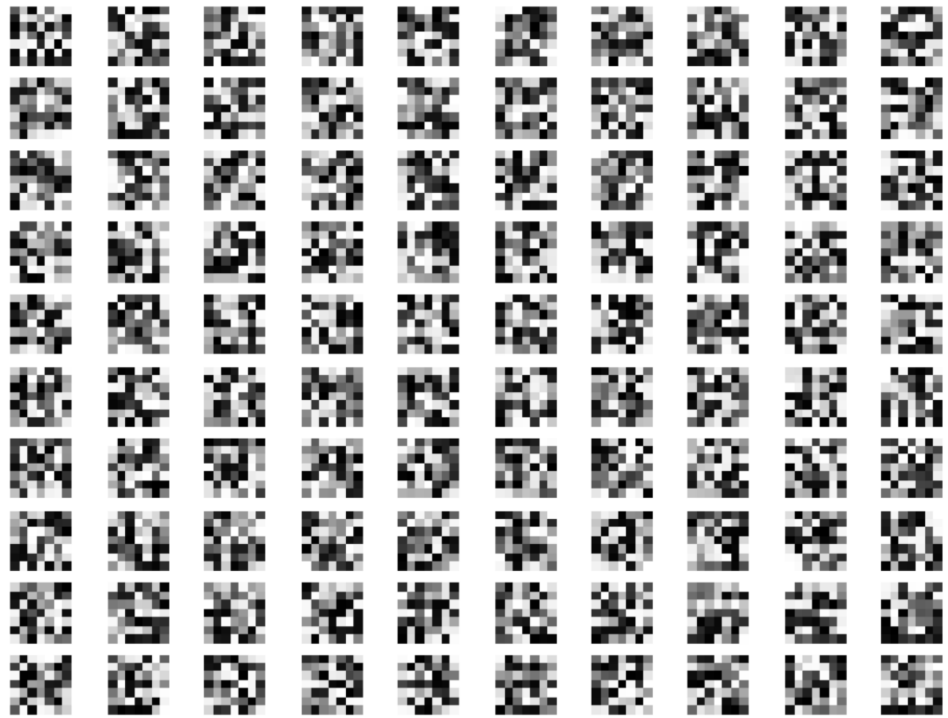
Input images



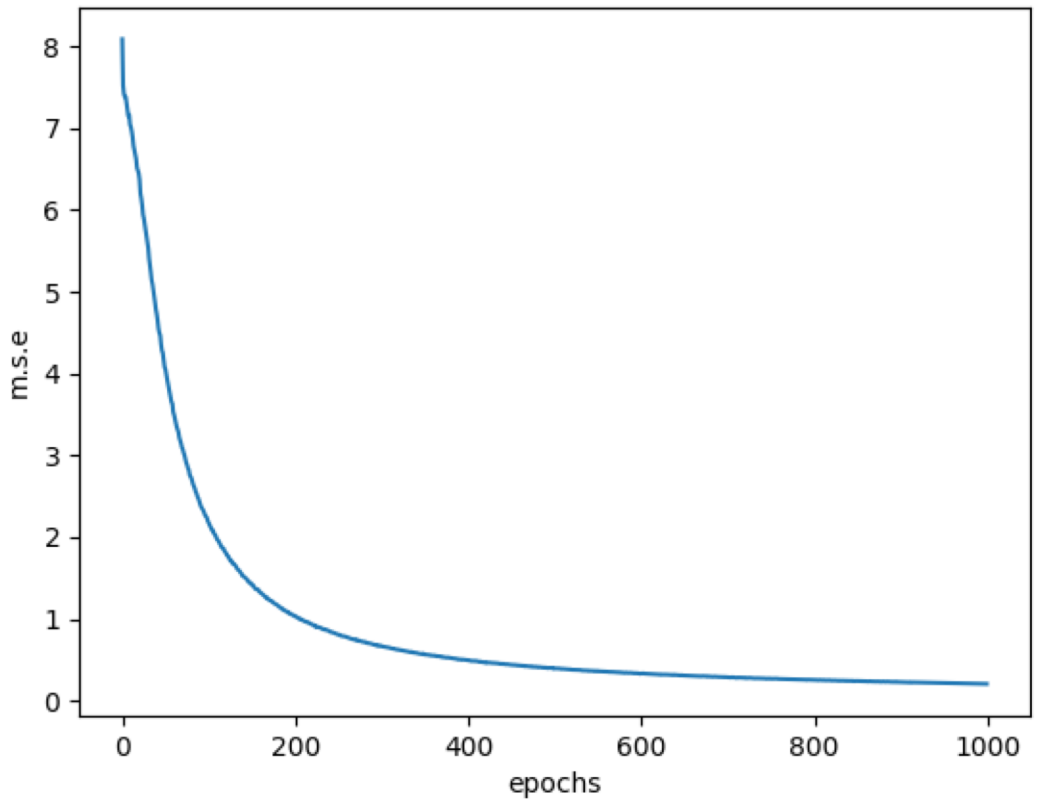
Output reconstructions



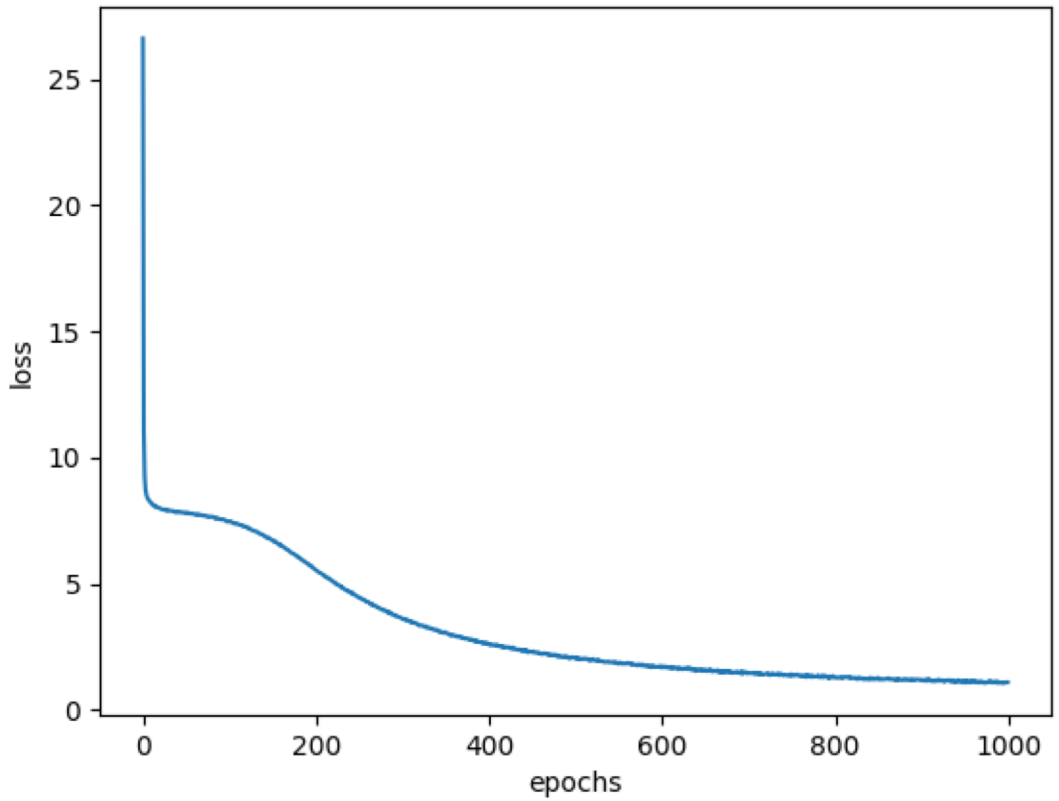
Hidden-layer activations



Overcomplete autoencoder: 144 hidden units

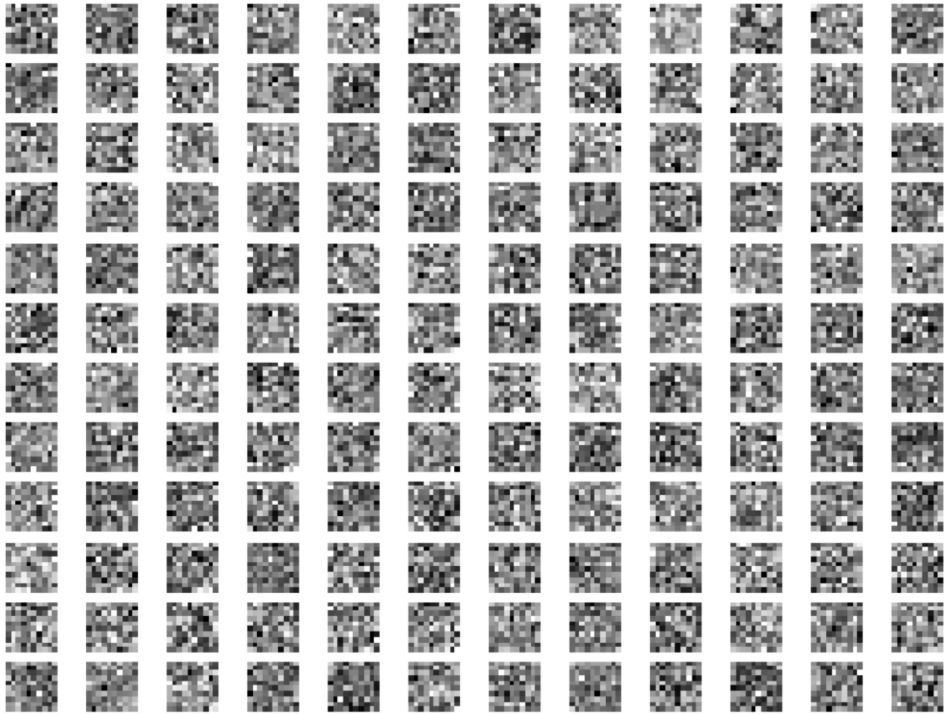


Sparse autoencoder: 144 hidden units

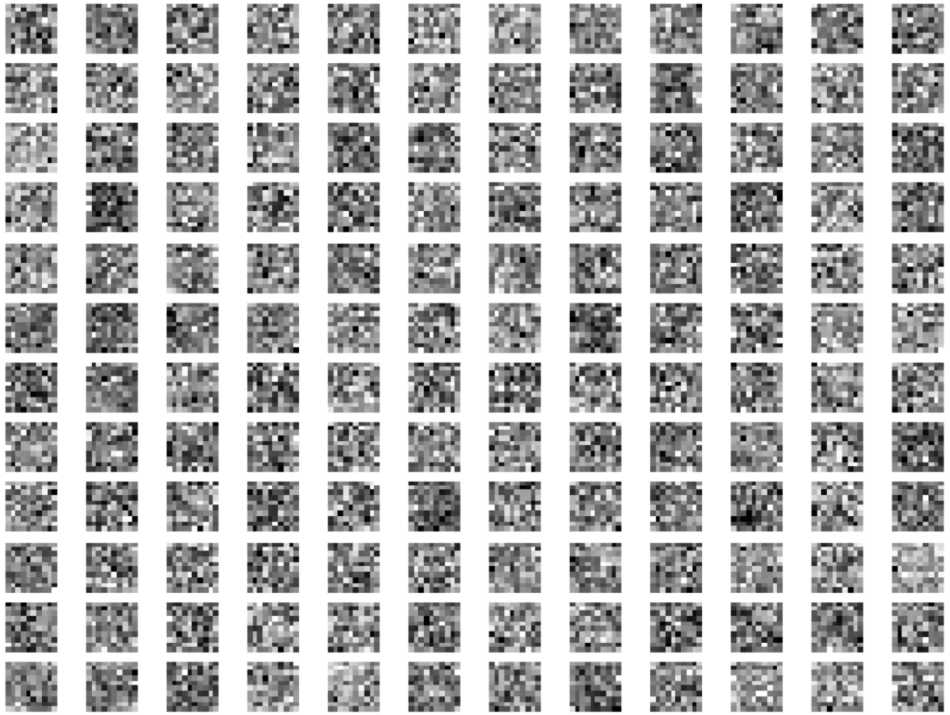


Weights learned by overcomplete autoencoders

Without sparsity constrained

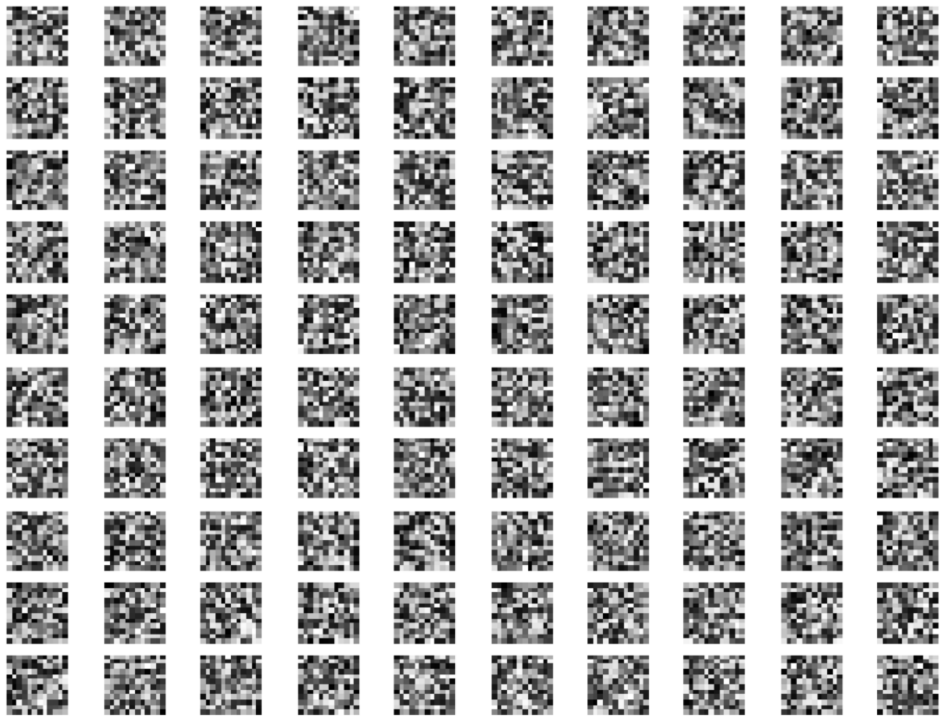


With sparsity constrained

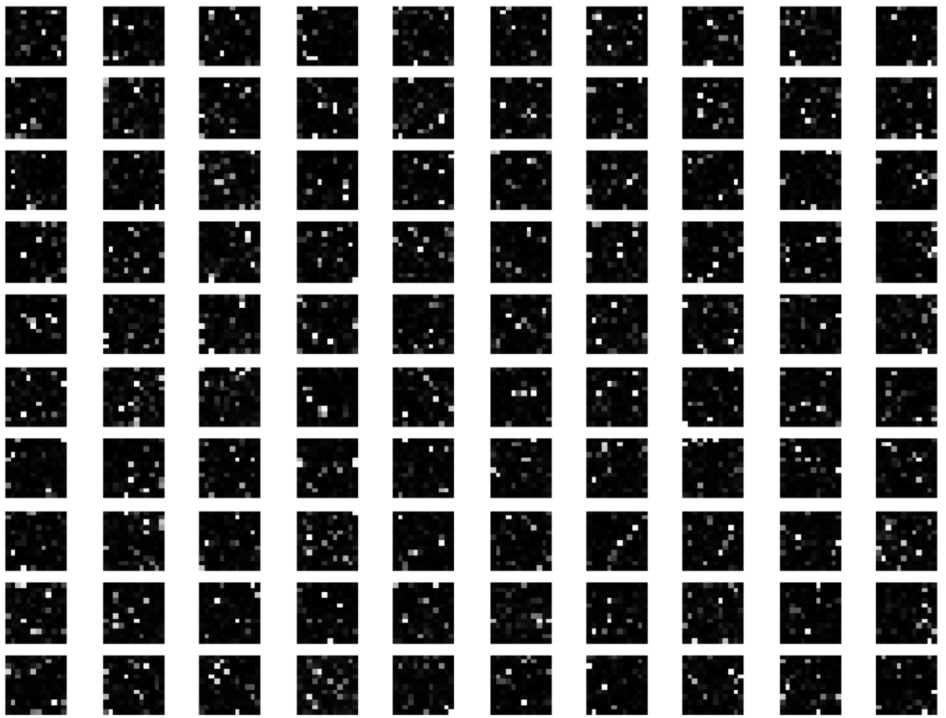


Hidden-layer activations of overcomplete autoencoders

Without sparsity constrained



With sparsity constrained

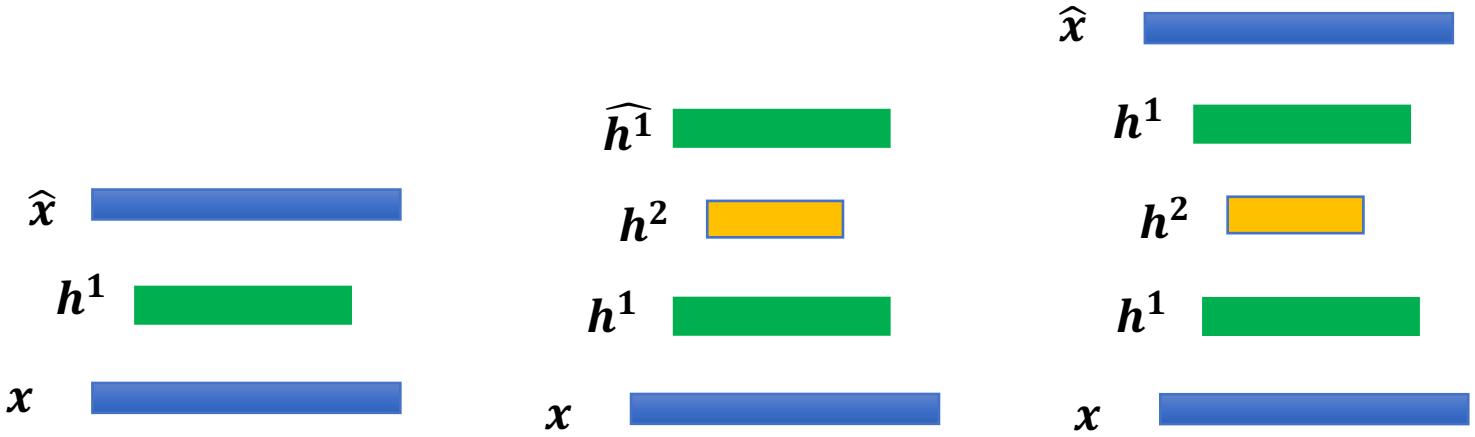


3. Design a denoising autoencoder to reconstruct MNIST images:
http://yann.lecun.com/exdb/mnist/

(a) Assume one hidden layer with 625 neurons, multiplicative noise, and cross-entropy cost function. Use 10% corruption level, learning factor $\alpha = 0.1$, batch size = 128, sparsity constant $\rho = 0.02$, and penalty parameter $\beta = 0.4$.
Plot the learning curves, the weights, and the hidden layer activations for sample test images.

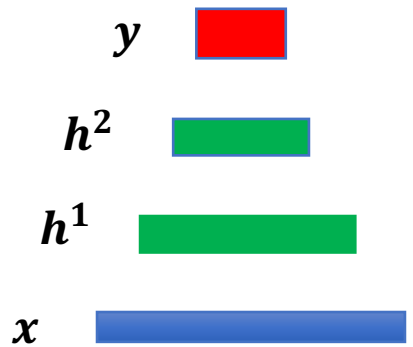
(b) Add another hidden layer with 100 neurons and train the autoencoder as before. Plot the feature maps.
Plot the learning curves, the weights, and the hidden layer activations for sample test images

(c) Add a softmax layer on top of the second hidden layer to design a classifier. Show learning curves and find the accuracy of the classifier.
Plot the learning curves and the weights and find the accuracy for test patterns.



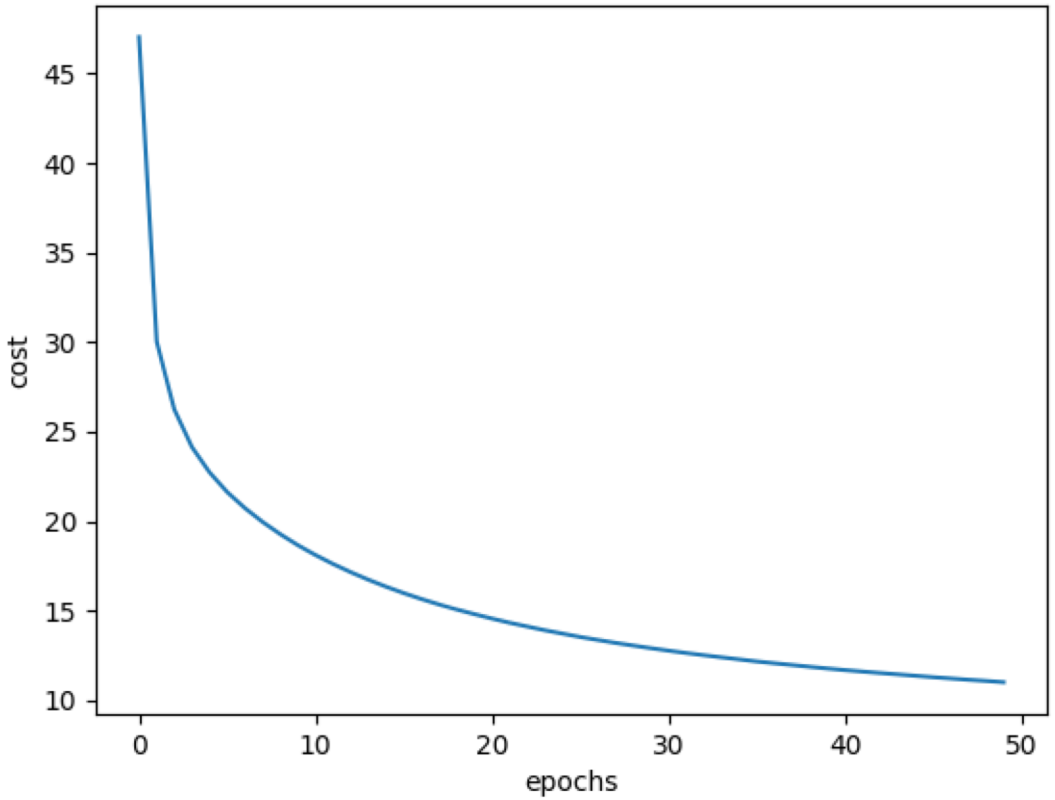
Input: 28x28
First hidden-layer: 625
Second hidden-layer: 100

Training with the sparsity constraints:
sparsity constant $\rho = 0.02$, and penalty parameter $\beta = 0.4$.

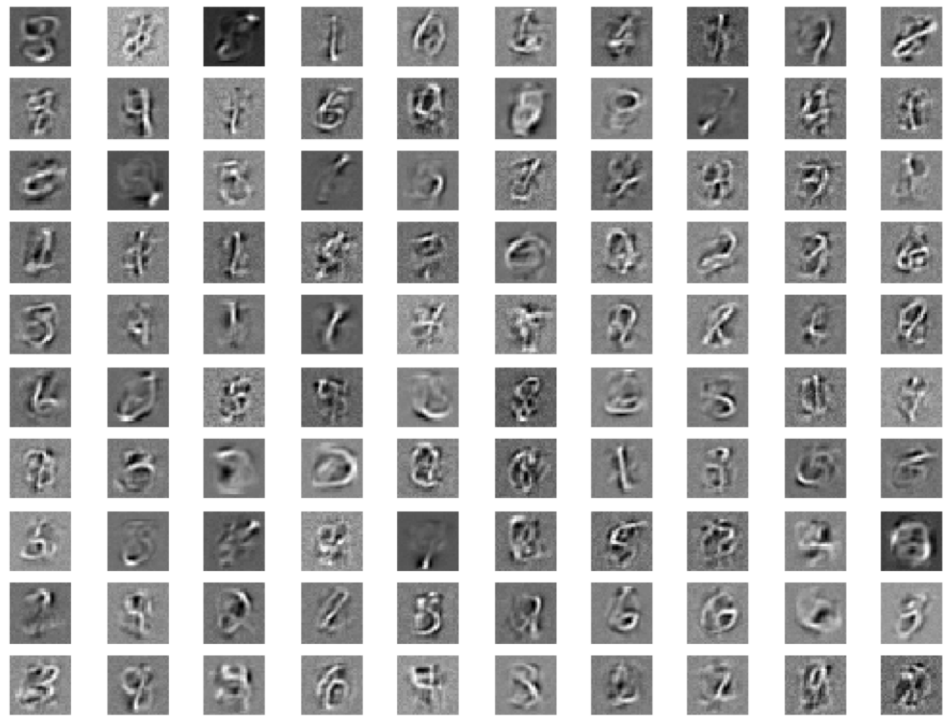


Training the classifier:
Output is a softmax layer
All weights are fined tuned.

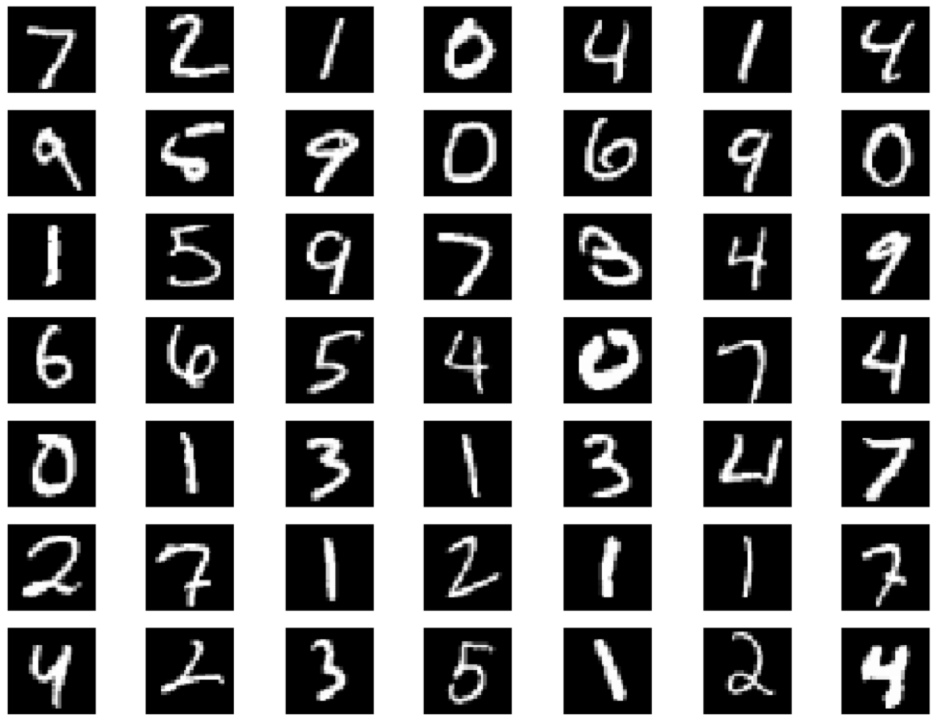
Training hidden-layer 1:



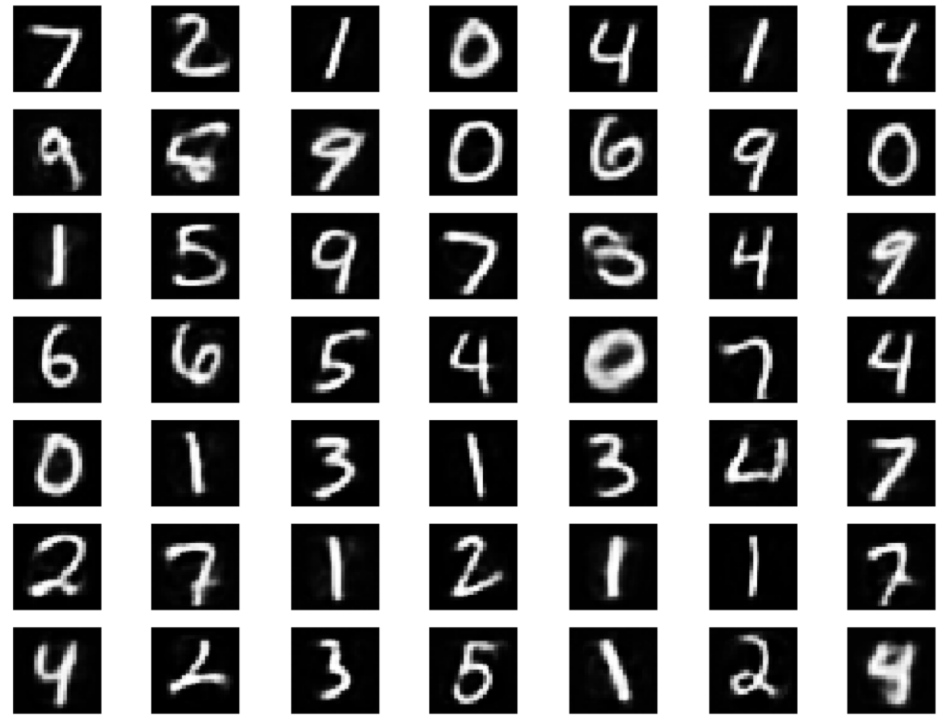
Weights learned by the hidden layer - 1



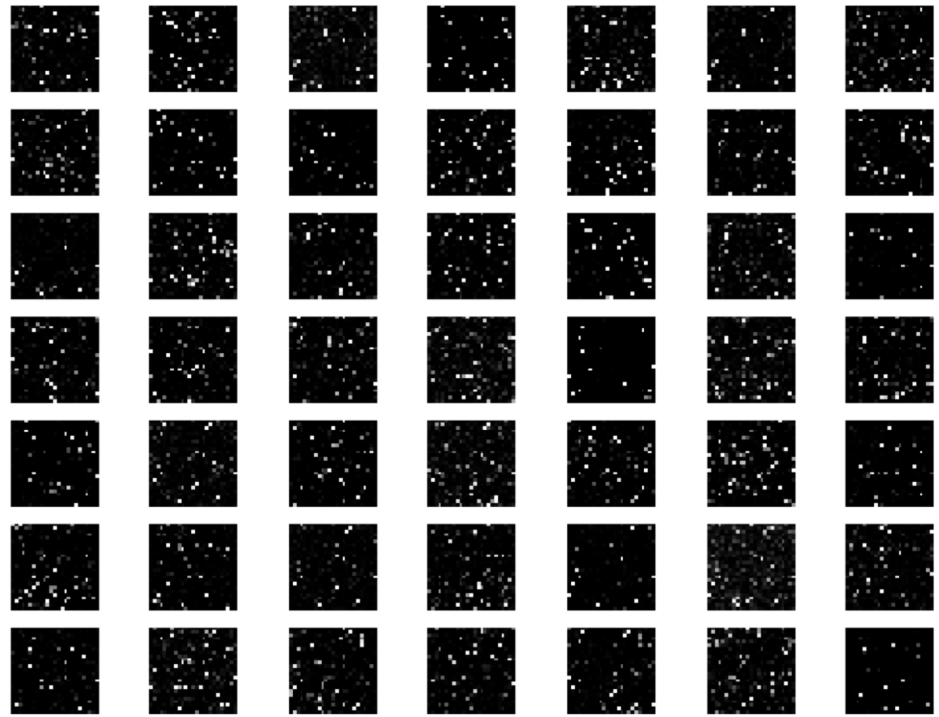
Test patterns



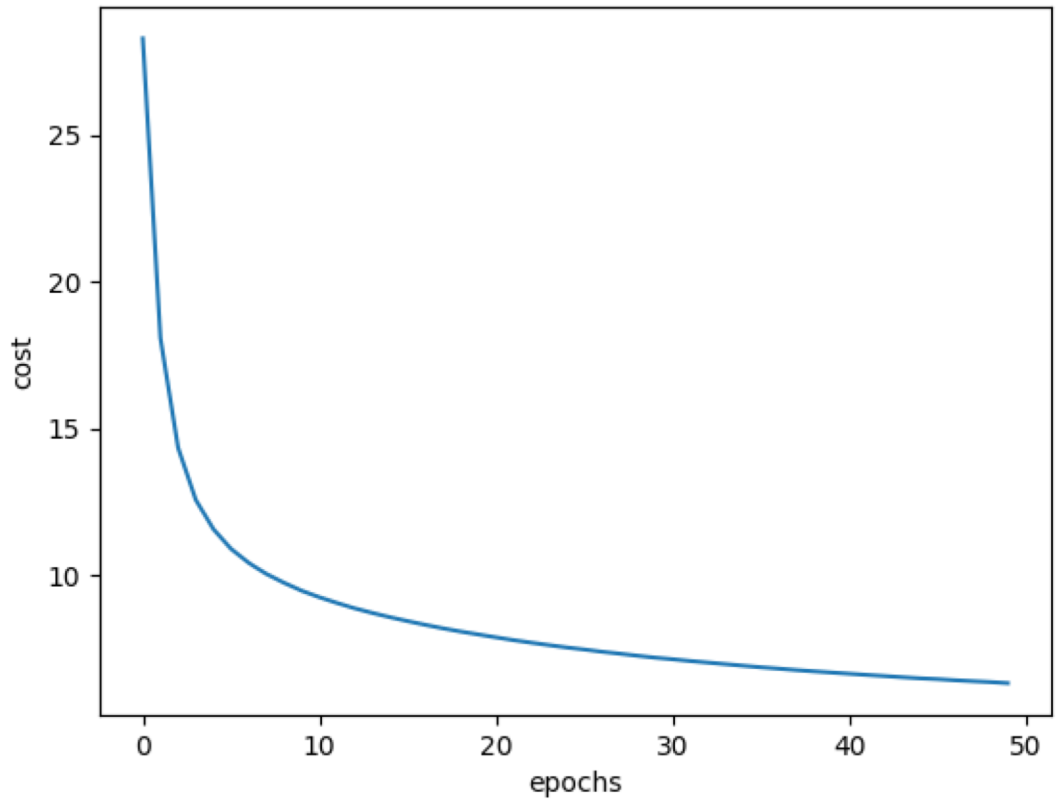
Output reconstructions



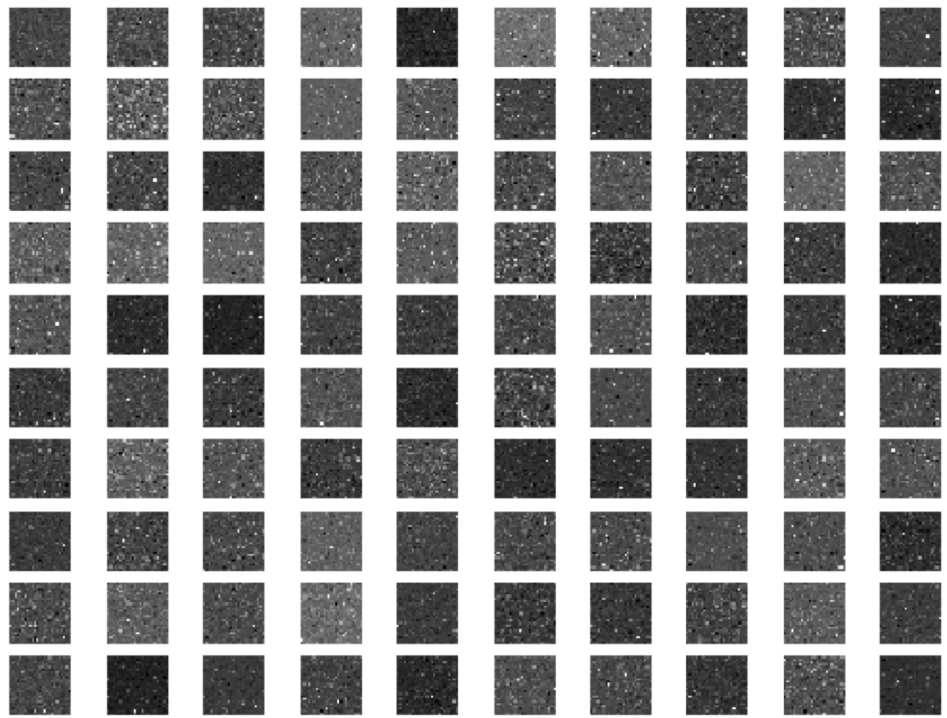
Hidden layer 1 activation on test patterns



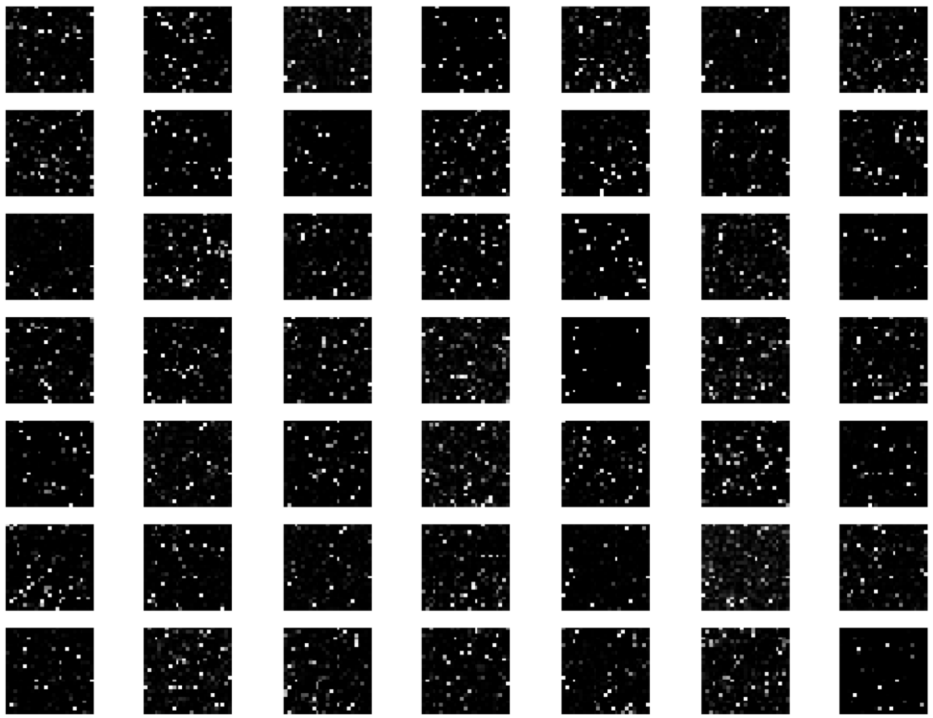
Training hidden-layer 2:



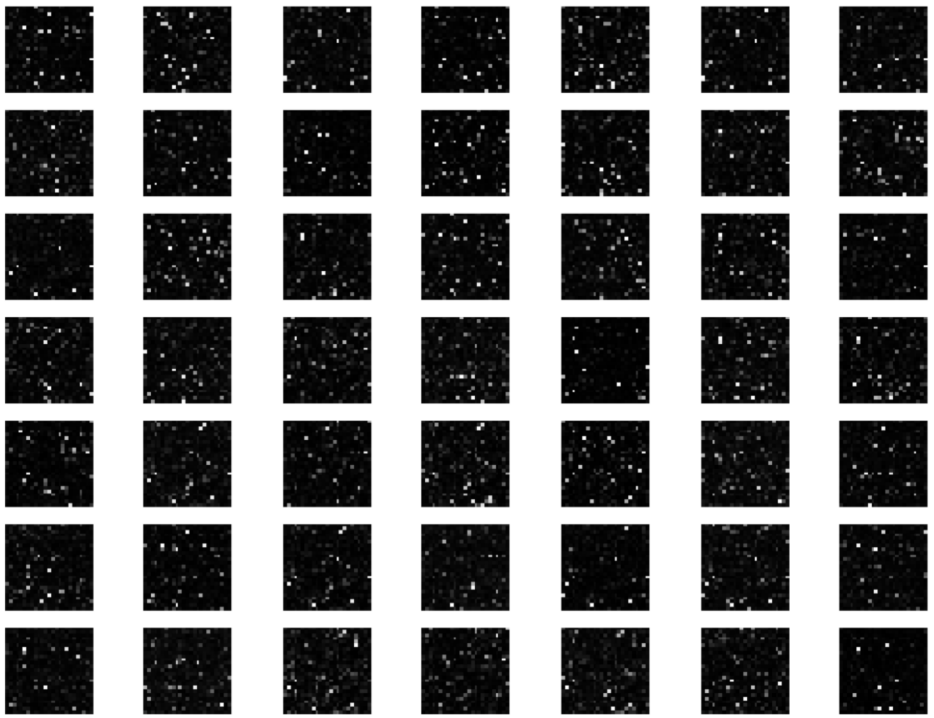
Weights learned by the hidden layer - 2



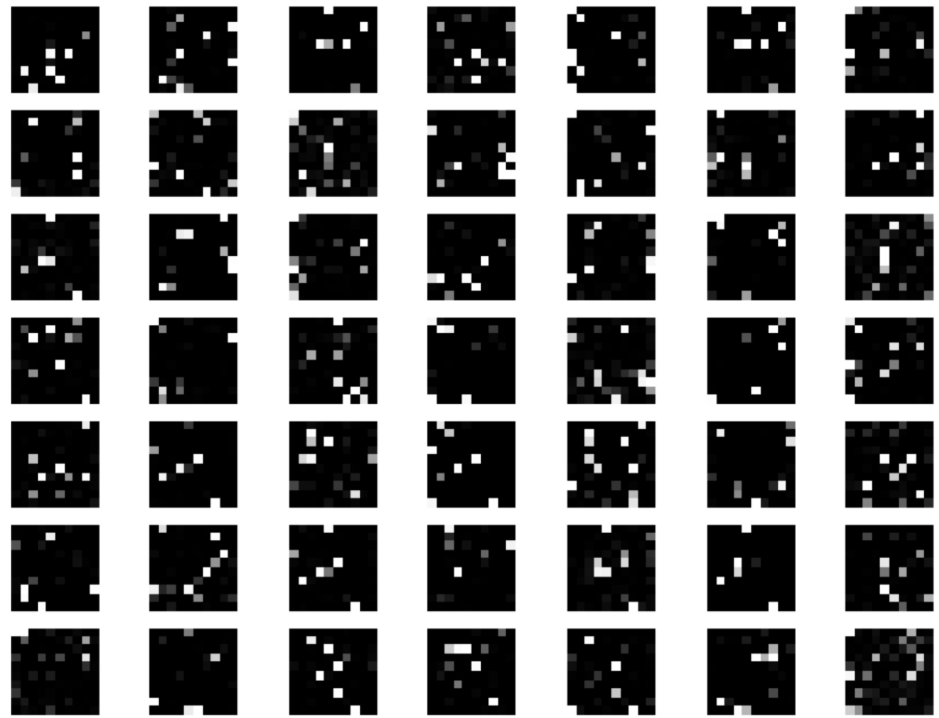
Output of the hidden layer 1



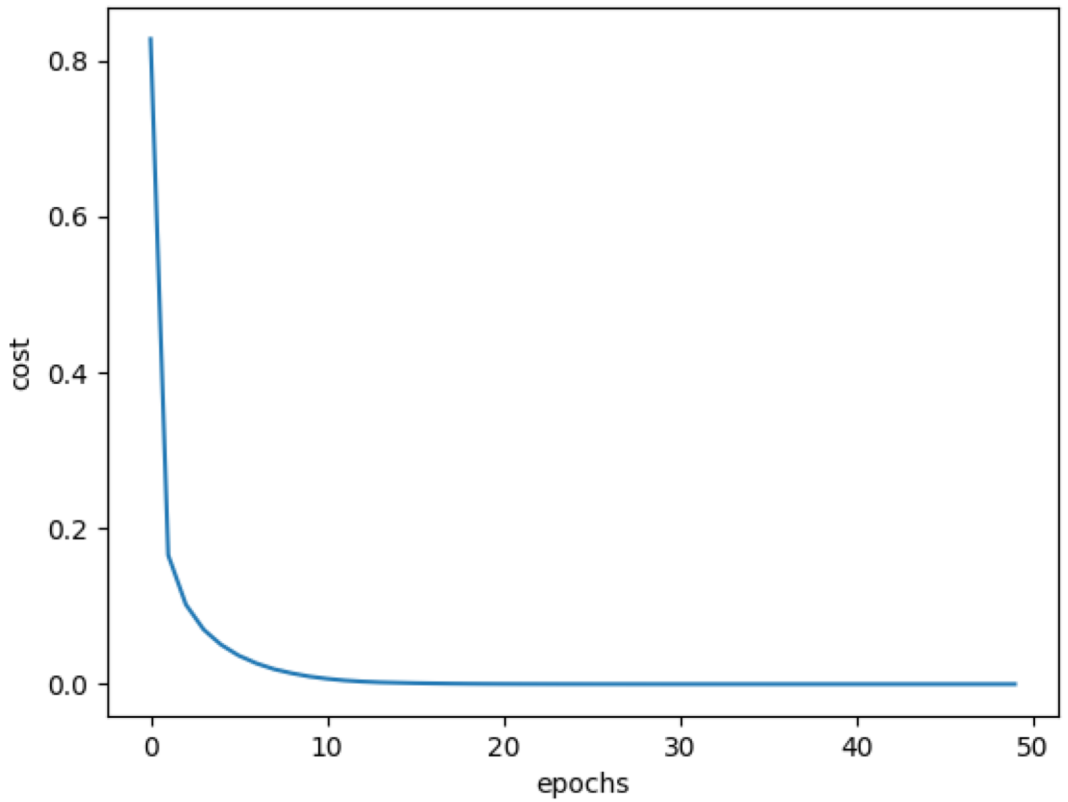
Reconstructed hidden-layer 1 output

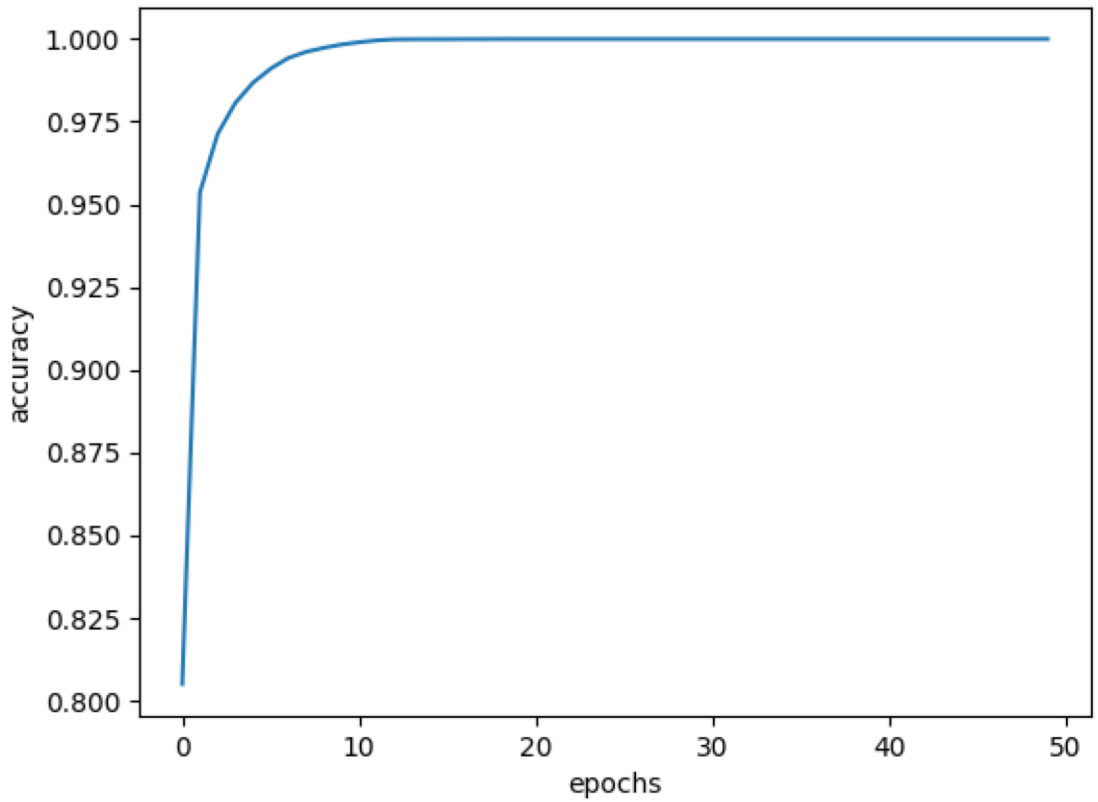


Hidden layer 2 activation on test patterns

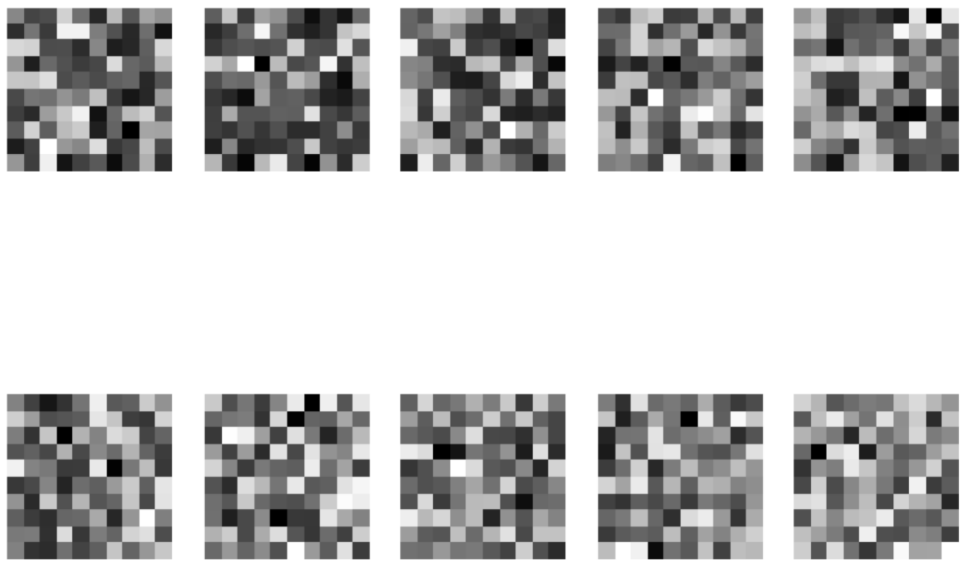


Training the classifier:

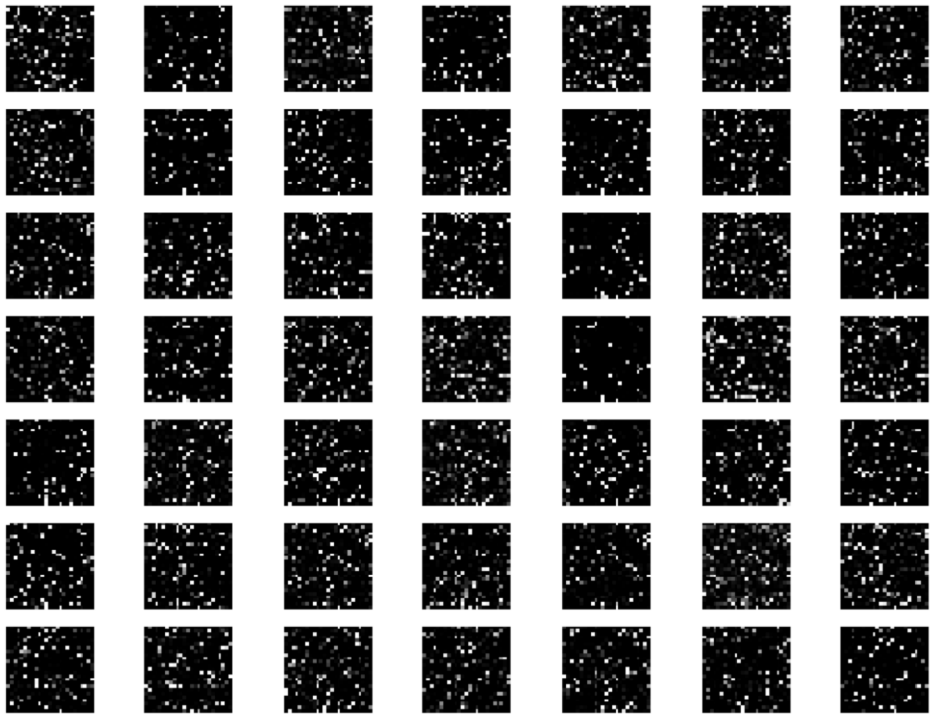




Weights learned by the output layer - 1



Hidden-layer 1 activations on test images



Hidden-layer 2 activations on test images