Assignment 3
Restricted Boltzmann Machines on MNIST

Tarasco Pier Paolo
ID: 619622
Code: https://github.com/Pier297/ISPR/tree/main/Midterm%202

Code

```
function [W, bias_v, bias_h] = mini_batch(X, v, h, W, bias_v, bias_h, eps, k)
    delta_W = zeros(size(v, 1), size(h, 1));
    delta_bias_v = zeros(size(v, 1), 1);
    delta_bias_h = zeros(size(h, 1), 1);
    % For each data point of the mini-batch, apply CD training
    for i = 1:size(X, 1)
        v_0 = X(i, :)';
        h_0 = logistic(v_0' * W + bias_h')' > rand(size(h, 1), 1);
        % k step gibbs sampling
        h_k = h_0;
        for s = 1:k
            v_k = logistic(W * h_k + bias_v);
            p_j = logistic(v_k' * W + bias_h')';
            h_k = p_j > rand(size(h, 1), 1);
        end
        h_k = p_j;
        % When computing the gradient use p_j instead of h_j,
        % this reduces the sampling noise -> leads to faster training
        % -- [Hinton, A Practical Guide to Training Restricted Boltzmann Machines]
        delta_W = delta_W + (v_0 * h_0') - (v_k * h_k');
        delta_bias_v = delta_bias_v + (v_0 - v_k);
        delta_bias_h = delta_bias_h + (h_0 - h_k);
    end
    % Update W
    W = W + (eps / size(X, 1)) * delta_W;
    % Update biases
    bias_v = bias_v + (eps / size(X, 1)) * delta_bias_v;
    bias_h = bias_h + (eps / size(X, 1)) * delta_bias_h;
end
```

$$h \sim P(h_j = 1|v)$$

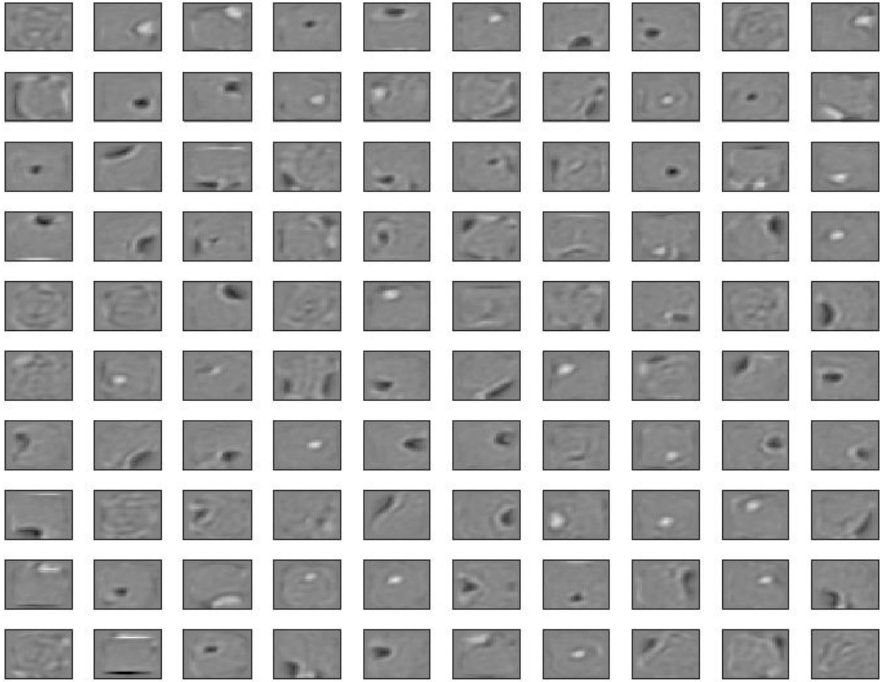
$$P(v_i = 1|h)$$

$$P(h_j = 1|v)$$

$$h \sim P(h_j = 1|v)$$

Results

100 features of the RBM



```
n_hidden_units = 100;
k = 1;
BATCH_SIZE = 20;
eps = 0.1;
MAX_EPOCHS_RBM = 20;
```

Softmax Layer

# EPOCHS	TR accuracy	TS accuracy	Time [s]
10	0.921	0.919	240
20	0.923	0.919	450

Logistic Regression (with ‘`fitcnet`’)

# Hidden Layers	Structure Hidden Layer	TR acc.	TS acc.	Time [s]
0	-	0.811	0.814	35
1	[10]	0.932	0.929	36
1	[100]	1	0.963	36
2	[100, 50]	1	0.964	50
2	[100, 75]	1	0.966	54

Results: `fitcnet(enc_X_train, Y_train, "LayerSizes", [100 75]);`

Accuracy = 1

Train Confusion Matrix

True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	5923									
1		6742								
2			5958							
3				6131						
4					5842					
5						5421				
6							5918			
7								6265		
8									5851	
9										5949

Accuracy = 0.966

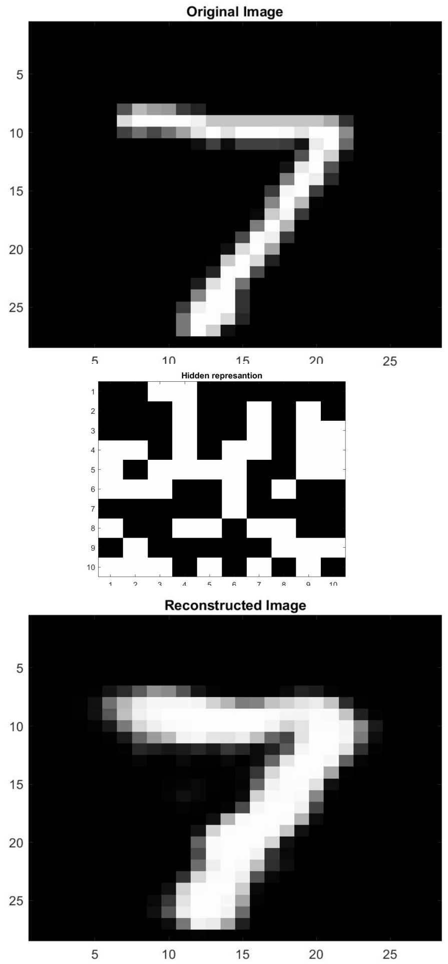
Test Confusion Matrix

True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	959		3	1	1	3	5	3	4	1
1		1121	7	1			2	2	2	
2	4	6	999	7			1	8	6	1
3		1	9	962		18		6	11	3
4	3	2	2		942		6	4	2	21
5	2		4	22		839	14	2	4	5
6	8	3	1		4	9	931		2	
7		4	8	4	6	1		996	2	7
8	2		5	14	1	11	1	2	929	9
9		2	1	4	13	7	1	9	7	965

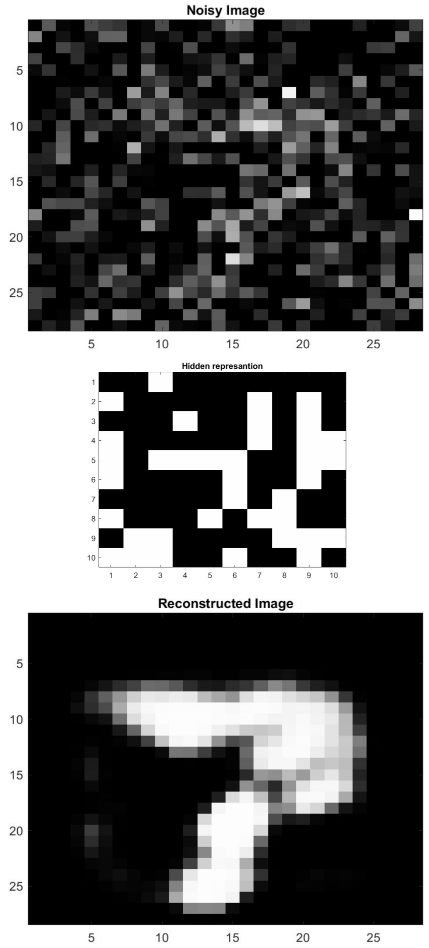
What if we didn’t use the RBM representation? **TR acc. = 1** **TS acc. = 0.975**

- **1% performance tradeoff versus 90% compression rate**

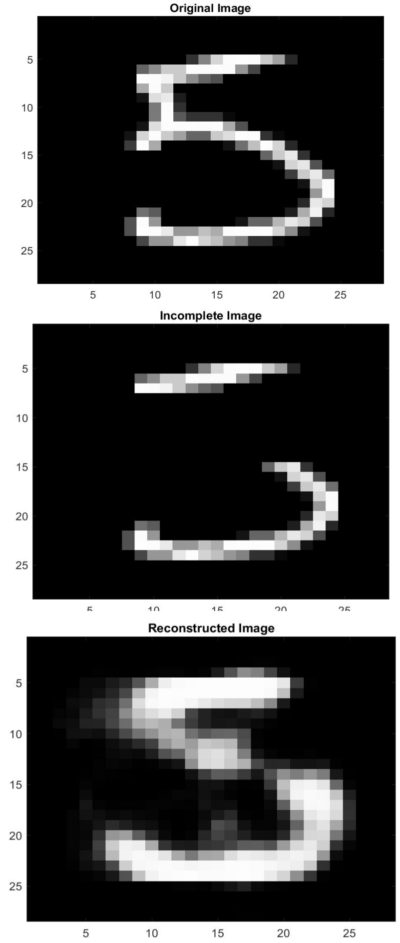
Reconstruction



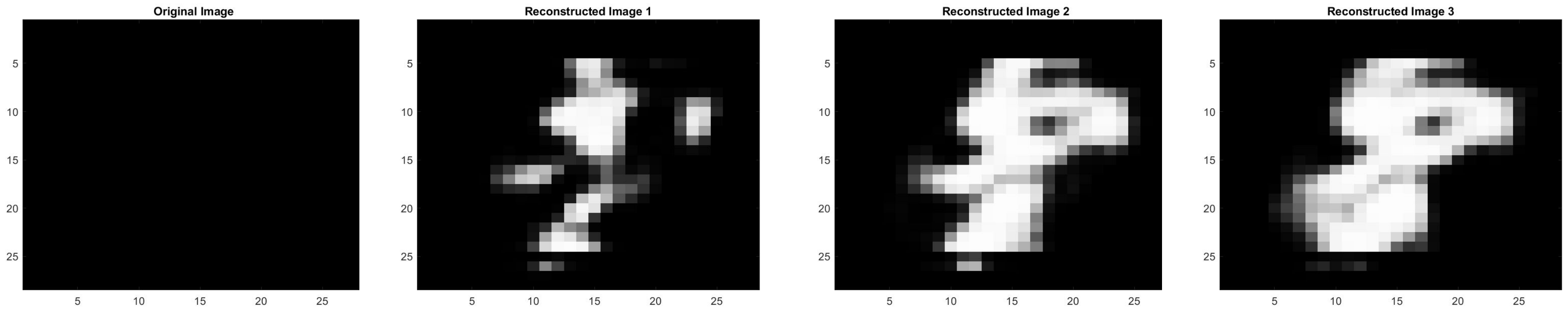
Denoising

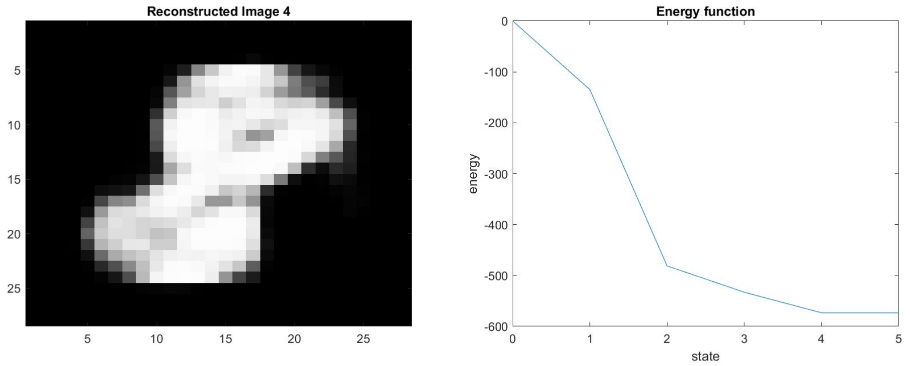


Completion



Dreaming





Final considerations:

- Training an RBM takes time and the performance on the classification task decreases, but:
- We can achieve almost the same performance with a 90% compression rate!
- We learn the probability distribution P(v, h)