

SYDE 552 Assignment 3: Hippocampal Models

Due Friday, March 15, Anywhere on Earth

Value: 15% of total marks for the course

The purpose of this assignment is to give you experience working with associative memories. To do so, we'll be using pytorch to implement different associative memory models.

You can work in groups to do the assignment, but your answers and code must be original to you. Your submission will be a filled-out copy of this notebook (cells for code and written answers provided).

1. The Hippocampus

1.a) [2 marks] The hippocampus is implicated in spatial navigation and episodic memory. How do we know this? What are some of the neuroscience results that revealed these facilities of the hippocampus? (see Kandel *et al.* Ch. 65)

Role in spatial navigation

The numerous connections between the hippocampus and other areas of the brain cause neural connections to strengthen over time. Processes involved in synaptic plasticity and long-term potentiation of the hippocampus allow mammals to orient themselves and navigate within an environment [1].

The hippocampus has been implicated in spatial navigation by several experiments that studied the ability of healthy and lesioned rats to navigate the Morris water maze. The lesioned rats demonstrated significant deficits in their spatial navigation abilities [2].

Additionally, other experiments have demonstrated the existence of "place" and "grid" cells in the hippocampus, which have an important role in spatial navigation and orientation. Grid cells arranged in hexagonal patterns of varying spatial resolution allow an organism to know where it is in space [2]. Place cells store information about specific locations or landmarks that allow an organism to navigate a familiar environment.

Role in episodic memory

We know the hippocampus is implicated in episodic memory from the case study of HM, a patient who suffered severed damage to the hippocampus [3]. He was able to learn new motor tasks over several days (tracing between the outlines of a star) but was incapable of storing new information—conversations with people, or anything that happened after his accident. In other words, HM's ability to form *episodic memories* was severely impaired following his accident, but his *implicit memory* was unaffected. From these observations of HM, we can conclude that the hippocampus is deeply implicated in the formation/consolidation of episodic memories.

References

[1] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. J. Hudspeth, "Prefrontal Cortex, Hippocampus, and the Biology of Explicit Memory Storage," in *Principles of Neural Science, 5th Ed.* New York: McGraw Hill, 2013, ch. 67, pp. 1487-1521.

[2] T. Stewart and M. Furlong. SYDE 552. Class Lecture, Computational Neuroscience: "Lecture 13: Hippocampus 3 – Spatio-temporal Modelling." Systems Design Engineering, University of Waterloo, Waterloo, Canada, [DATE], 2024.

[3] T. Stewart and M. Furlong. SYDE 552. Class Lecture, Computational Neuroscience: "Lecture 11: Hippocampus 1 – Anatomy and Role in Cognition." Systems Design Engineering, University of Waterloo, Waterloo, Canada, [DATE], 2024.

1.b) [2 marks] The Gluck and Meyers model of hippocampus is a simple, effective model of hippocampus, and how representations may be constructed for the slow transfer to the neocortex. However, it still has its limitations. Explain some (two or more) of the limitations of the Gluck and Meyers model. (The Gluck and Meyers Ch.6 reading will be useful in answering this question.)

Gluck and Meyers Model

The model cortico-hippocampal model proposed by Gluck and Meyers uses a predictive autoencoder to approximate the hippocampal-region (HR) network. Information is then passed from the HR network to a separate cortico-cerebellar network for long term memory storage. Empirical results show that the autoencoder architecture of the model is strikingly consistent with the neurophysiological processes that take place during learning tasks [4].

The model is based on the finding that the hippocampus is necessary in types of learning that require modifying *representations*, but not their *associations*—in other words, subjects with hippocampal damage can still learn to map a conditioned stimulus (tone) onto a corresponding behavioural response [4]. Experiments have shown that this model accurately predicts the real behaviours seen in humans, rabbits, and rats when exposed to various conditioning experiments, including CS/UR pairing and discrimination reversal when the predictive/non-predictive stimuli are swapped [4].

Limitations

Though this model captures much of the behaviour that is observed in biological subjects, it does not capture all possible results seen in conditioning experiments. Extinction, for example, happens when the conditioned stimulus is presented alone, after the subject has learned an association. In this case, the conditioned response weakens, but the model does not contain a way to do this (even though the observed behaviour may be 'correct', the mechanism underlying it is an oversimplification). The extinction also occurs much more quickly in the model than it does in experimental trials.

Another one of the limitations of this model is that it has not been shown to generalize to other tasks that involve the hippocampus in a biological network [4]. The model cannot perform the delayed non-match to sample or declarative memory tasks, for example. The authors note that this is a limitation of the current implementation, however, and not necessarily the underlying structure of the model.

Finally, one other significant limitation of the cortico-hippocampal model is that it does not account for timing effects. In biological subjects with an intact hippocampus can be conditioned when a short delay is introduced between the conditioned stimulus and unconditioned response (known as *trace conditioning*). However, in lesioned subjects, this function is disrupted. Timing is therefore significant in understanding the role of the hippocampus in conditioning, which is unaccounted for in this model [4].

References

[4] M. A. Gluck and C. E. Meyers, "Cortico-Hippocampal Interaction in Associative Learning," in *Gateway to Memory*. Cambridge, MA: MIT Press, 2000, ch. 6, pp. 145-187.

1.c) [2 marks] Unsupervised pre-training is integral to the Gluck and Meyers model of Hippocampus, and unsupervised pre-training has been shown to accelerate reinforcement learning in rats navigating mazes. However, unsupervised pre-training is not common in deep learning techniques. Describe why that may be. (See [15.1 of Goodfellow et al.](#))

Greedy layer-wise unsupervised pretraining involves individually optimizing each layer before moving to the next, such that all previous layers are fixed while training the next.

According to Goodfellow et al., the popularity of this algorithm has declined as more modern techniques have started to outperform it [5]. Another contributing factor in its lack of popularity could have to do with efficiency. Greedy algorithms tend to be computationally intensive (each layer is trained individually) and for this reason do not scale well for deep learning applications. Additionally, while this approach ensures optimal solutions for the individual components of the network, it does not guarantee a globally optimal solution [5].

One other possible (and more biologically-motivated) reason would be that this is not how neural networks in the

One other possible (and more biologically motivated) reason would be that this is not how neural networks in the brain are trained. From what we know about the brain, there is no way to isolate layers during training or learning processes. In this sense, if we want to mimic the processes that take place in the brain, it would make sense to start with something that is more similar to the way we understand it to work.

[5] I. Goodfellow, Y. Bengio, and A. Courville, "Representation Learning," in *Deep Learning*. Cambridge, MA: MIT Press, 2016, ch. 15, pp. 524-554.

2. Hopfield Networks

Preliminaries

Although you should have installed them for prior assignments, we will require the pytorch and keras. Let's install those now.

```
pip install torch keras
```

Next, we will download the MNIST dataset. We will do this through the Keras library instead of torchvision.

In [71]:

```
import keras.datasets.mnist
import torch
import numpy as np
import matplotlib.pyplot as plt
```

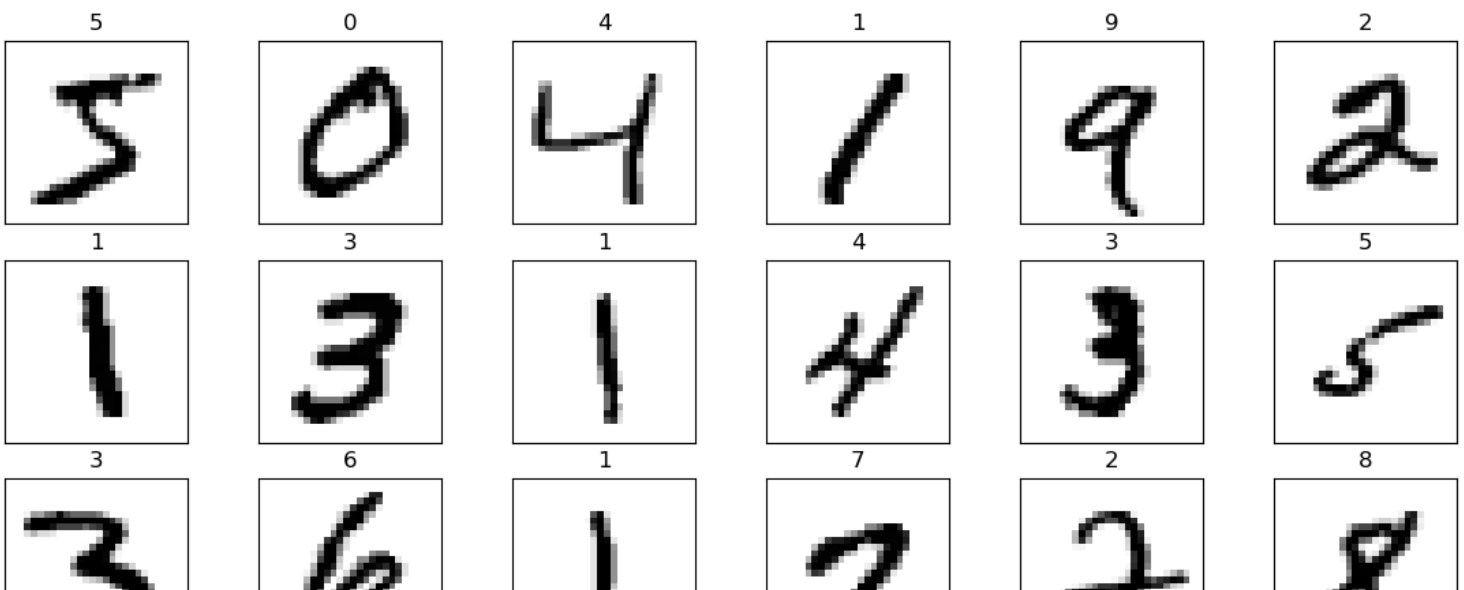
In [72]:

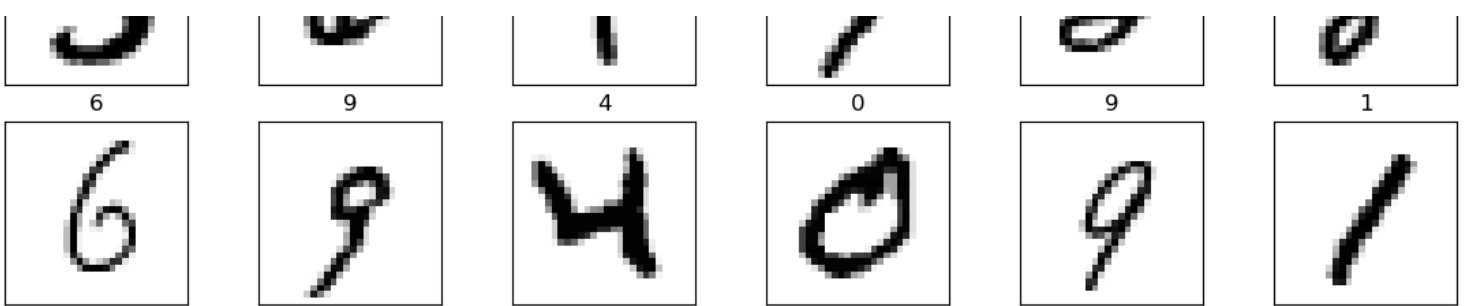
```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

As before, we will examine some of the images to make sure we got them right.

In [73]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(14,8))
for i in range(24):
    plt.subplot(4, 6, i+1)
    plt.imshow(x_train[i].reshape((28,28)), vmin=0, vmax=255, cmap='gray_r')
    plt.xticks([])
    plt.yticks([])
    plt.title(int(y_train[i]))
```





Normally, the MNIST dataset has images represented by values in the range $[0, 255]$. However, since we are dealing with Hopfield networks, we are going to binarize the data.

In [74]:

```
def binarize(xs):
    """
    xs : a num_samples by num_features array of images.
    """
    binary = (xs / 255) > 0.5
    integer = 2 * binary - 1
    return integer

binary_imgs = binarize(x_train)
```

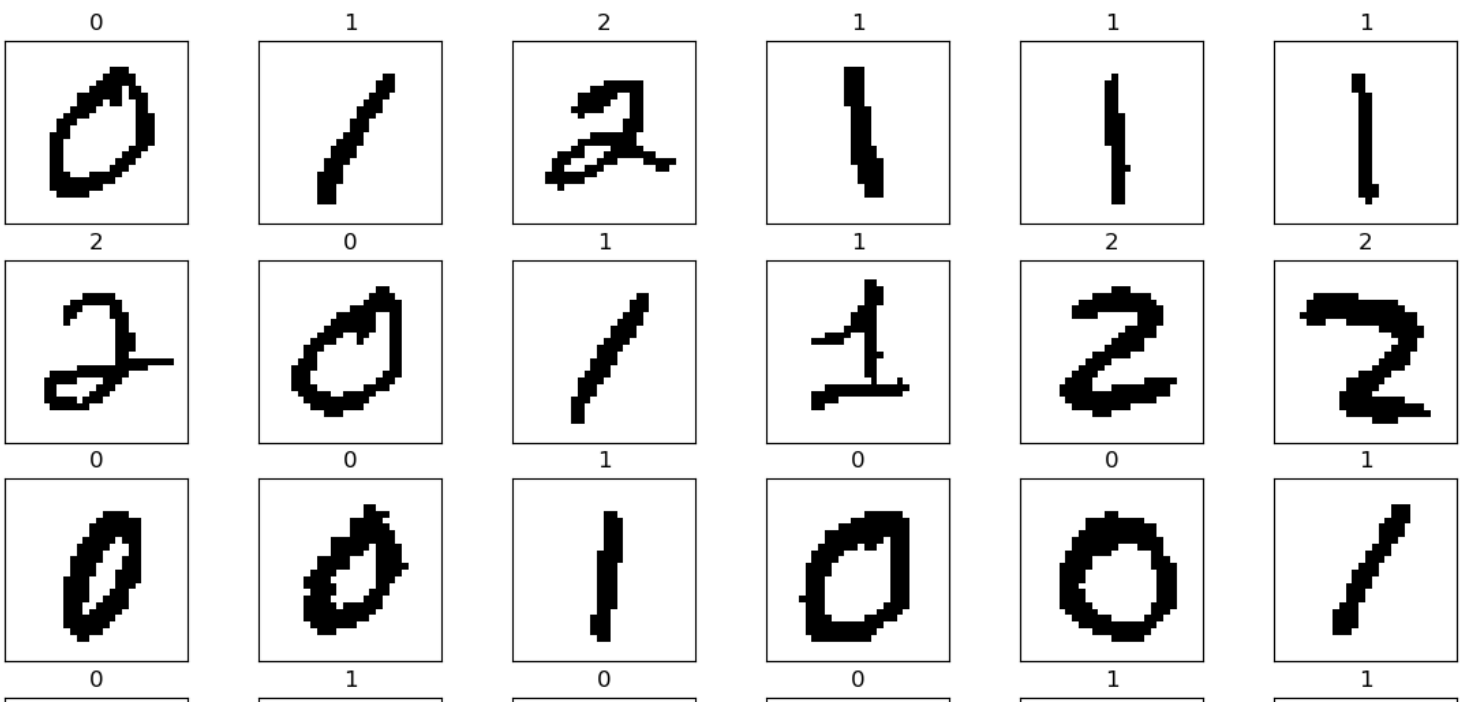
Further, to make things easy on our network, we are only going to look at images of the digits 0, 1, and 2

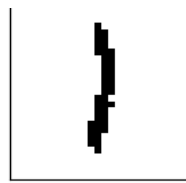
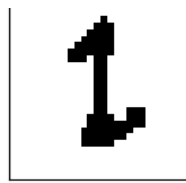
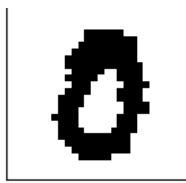
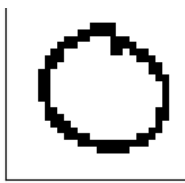
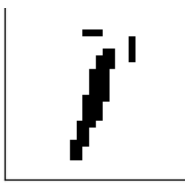
In [75]:

```
binary_imgs = binary_imgs[y_train < 3,:]
binary_labels = y_train[y_train < 3]
```

In [76]:

```
plt.figure(figsize=(14,8))
for i in range(24):
    plt.subplot(4, 6, i+1)
    plt.imshow(binary_imgs[i].reshape((28,28)), vmin=-1, vmax=1, cmap='gray_r')
    plt.xticks([])
    plt.yticks([])
    plt.title(int(binary_labels[i]))
```





We will also need to reshape the data into a vector representation, and then covert it to a pytorch tensor

In [77]:

```
binary_img_vecs = binary_imgs.reshape((-1,28*28))
binary_img_tensor = torch.from_numpy(binary_img_vecs).to(torch.float32)
```

In [78]:

```
print(f"Binary image tensor: {binary_img_tensor.shape}")
```

```
Binary image tensor: torch.Size([18623, 784])
```

2.a) [2 Marks] Implement the Hopfield Network using Hopfield's learning rule.

We will first implement the Hopfield network using the outer product formulation for the weight matrix. Given a set of patterns \mathbf{x}_i

, by first computing the matrix

$$D = \frac{1}{N} \sum_i^N (\mathbf{x}_i - \theta)(\mathbf{x}_i - \theta)^T$$

.

$$\text{where } \theta = \frac{1}{ND} \sum_i^N \sum_j^D x_{i,j}$$

, i.e., the average value of all elements in the training data.

Next we remove the diagonal element of the matrix, making the weight matrix:

$$W = D - \text{diag}(D),$$

where $\text{diag}(D)$

is the diagonal of the D

matrix. Note that when implementing this in numpy or pytorch we must apply the `diag` function twice, i.e.:

```
W = D - torch.diag(torch.diag(D))
```

Here is some code to implement the learning rule:

In [79]:

```
def outer_product_hopfield_matrix(training_patterns):
    """
    train_hopfield_matrix - produces a matrix for a (non-Modern) Hopfield network using the
    outer product rule.

    Parameters:
    -----
    training_patterns : torch.Tensor
        A Tensor of shape (num_patterns, num_features) that will be used to construct the weight
        matrix.

    Returns:
    -----
    W : torch.Tensor
        A (num_features, num_features) Tensor that stores the papers encoded in the network.
    """
```

```

    theta = torch.sum(training_patterns) / (training_patterns.shape[0] * training_patterns.
shape[1])
    D = torch.einsum('nd,ne->de', training_patterns - theta, training_patterns - theta) / f
loat(training_patterns.shape[0])
    W = (D - torch.diag(torch.diag(D)))
    return W

def evaluate_hopfield_network(W, input_pattern, training_patterns=None, num_iters=5, thresh
old=0):
    """
    Evaluates a Hopfield network with weight matrix W on a number of test patterns. Also com
putes the similarity.
    """
    assert input_pattern.shape[1] == 1, f'''This function assumes you are cleaning up one pa
tern at a time.

                                Expected the input to be shape (1,{input_patter
n.shape[1]}),

                                got {input_pattern.shape}'''

    s = input_pattern

    similarities = None
    if training_patterns is not None:
        similarities = torch.zeros((num_iters, training_patterns.shape[0]))
    ## end if

    for i in range(num_iters):
        s = torch.sign(W @ s - threshold)

        if training_patterns is not None:
            similarities[i,:] = torch.einsum('d,nd->n',s,training_patterns)
        ## end if
    ## end for
    return s, similarities

```

We want you to do the following things:

1. Compute the capacity of the network, using the expression $C \approx \frac{d}{2\log_2(d)}$, where d is the number of neurons.
2. For a training set of 10 patterns, plot the original image and the reconstructed image side-by-side.
3. Plot the training and test error (Mean squared error between the predicted and true values) of the Hopfield network as a function of the number of patterns stored in the network up to capacity, C , for five randomly selected training sets selected from `binary_img_tensor`.
4. For a training set of size 10 patterns, add salt and pepper noise (bit flips) to the testing images and compute the training error as the probability of noise increases. Note: Compare the reconstructed images to the non-noisy images, e.g., for any test image you would do something like:

```

noisy_image = corrupt(test_image, noise_level)
reconstructed_images = evaluate_hopfield_network(W, noisy_image.T)
error = mse(reconstructed_images, test_images)

```

In [80]:

```

## 2.a.1 - Compute network capacity.
d = 784 # d is the number of neurons in the network, 28x28 = 784
C = int(d / (2 * np.log2(d)))
print(f"Network capacity: {C}")

```

Network capacity: 40

In [81]:

```

# 2.a.2 - plot original and reconstructed images.

# create weight matrix, W
W = outer_product_hopfield_matrix(binary_img_tensor[:10,:])

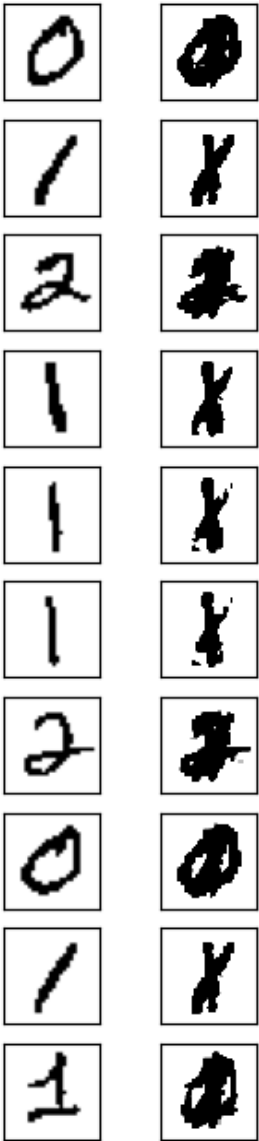
plt.figure(figsize=(2,8))

for i in range(10):
    # reconstruct training image, i:
    reconstructed_img = W @ binary_img_tensor[i,:].reshape((1,-1)).T

    # plot original image
    plt.subplot(10,2,1+2*i)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(binary_imgs[i].reshape((28,28)), vmin=-1, vmax=1, cmap='gray_r')

    # plot reconstructed image
    plt.subplot(10,2,1+2*i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(reconstructed_img.reshape((28,28)), vmin=-1, vmax=1, cmap='gray_r')

```



In []:

```

def mse(true_vals, pred_val):
    """
    Calculates the mean squared error between an image and the corresponding predicted image

```

```

'''
return torch.mean((true_vals - pred_val)**2)

def generate_permutation(imgs, num_patterns):
    '''
    Returns a permutation of a list of images (imgs) of the specified length (num_patterns)
    '''
    perm = np.random.choice(imgs.shape[0], num_patterns, replace=False)
    return imgs[perm,:]

```

In [82]:

```

def test_network(training_patterns, test_patterns, C, method='outer_product'):
    num_trials = 5
    assert training_patterns.shape[0] == C, f'''Error: Expected training patterns to contain {C} patterns'''
    training_error = torch.zeros(len(range(2, C)))
    test_error = torch.zeros(len(range(2, C)))

    # repeat for 5 trials
    for i in range(num_trials):
        print(f"Trial {i+1}")

        # vary num_patterns up to C
        for pattern_idx, num_patterns in enumerate(range(2, C)):

            # permute training patterns
            permuted = generate_permutation(training_patterns, num_patterns)

            # create weight matrix, W
            if method == 'outer_product':
                W = outer_product_hopfield_matrix(permuted[:num_patterns,:])
            elif method == 'pseudo_inv':
                W = pseudoinverse_hopfield_matrix(permuted[:num_patterns,:])
            else:
                raise ValueError(f"Invalid method: {method}")

            # for each image in the training and testing sets, compute the mse:
            reconstructed_train = W @ permuted[pattern_idx,:].reshape(-1,1)
            training_error[pattern_idx] += mse(permuted[pattern_idx,:), reconstructed_train)

        print(f"Train error with {num_patterns} patterns: {training_error[pattern_idx]}")

        reconstructed_test = W @ test_patterns[pattern_idx,:].reshape(-1,1)
        test_error[pattern_idx] += mse(test_patterns[i,:], reconstructed_test)
        print(f"Test error with {num_patterns} patterns: {test_error[pattern_idx]}")

        training_error[pattern_idx] /= num_patterns
        test_error[pattern_idx] /= num_patterns

    training_error /= num_trials
    test_error /= num_trials

    return training_error, test_error

```

In [83]:

```

train_imgs = binary_img_tensor
test_imgs = binary_img_tensor[:40,:] # use the same testing set each time
train_avg, test_avg = test_network(train_imgs, test_imgs, C)

```

```

Trial 1
Train error with 2 patterns: 14723.7470703125
Test error with 2 patterns: 18266.03515625
Train error with 3 patterns: 16548.62109375
Test error with 3 patterns: 1768.3544921875
Train error with 4 patterns: 2806.4556640625

```


Train error with 4 patterns: 2808.43338840625
Test error with 4 patterns: 809.6845092773438
Train error with 5 patterns: 4617.66162109375
Test error with 5 patterns: 12326.06640625
Train error with 6 patterns: 5232.9541015625
Test error with 6 patterns: 1662.32470703125
Train error with 7 patterns: 8133.642578125
Test error with 7 patterns: 2322.42724609375
Train error with 8 patterns: 4744.0458984375
Test error with 8 patterns: 2368.52490234375
Train error with 9 patterns: 1447.4847412109375
Test error with 9 patterns: 508.4510192871094
Train error with 10 patterns: 4351.00390625
Test error with 10 patterns: 2003.624755859375
Train error with 11 patterns: 10533.0400390625
Test error with 11 patterns: 363.53961181640625
Train error with 12 patterns: 2063.65869140625
Test error with 12 patterns: 893.3114624023438
Train error with 13 patterns: 1142.0592041015625
Test error with 13 patterns: 1380.093505859375
Train error with 14 patterns: 1888.42529296875
Test error with 14 patterns: 791.1902465820312
Train error with 15 patterns: 3827.854736328125
Test error with 15 patterns: 1040.928955078125
Train error with 16 patterns: 1701.4876708984375
Test error with 16 patterns: 2902.1416015625
Train error with 17 patterns: 1669.7271728515625
Test error with 17 patterns: 3329.168212890625
Train error with 18 patterns: 8154.822265625
Test error with 18 patterns: 3293.956298828125
Train error with 19 patterns: 1598.515869140625
Test error with 19 patterns: 982.452392578125
Train error with 20 patterns: 2363.28955078125
Test error with 20 patterns: 2190.85888671875
Train error with 21 patterns: 605.7230834960938
Test error with 21 patterns: 353.22442626953125
Train error with 22 patterns: 664.6589965820312
Test error with 22 patterns: 745.052490234375
Train error with 23 patterns: 1284.022705078125
Test error with 23 patterns: 975.0233154296875
Train error with 24 patterns: 1656.21533203125
Test error with 24 patterns: 2948.959228515625
Train error with 25 patterns: 1466.470947265625
Test error with 25 patterns: 1568.8017578125
Train error with 26 patterns: 702.1287841796875
Test error with 26 patterns: 1473.7379150390625
Train error with 27 patterns: 1076.603271484375
Test error with 27 patterns: 2002.7518310546875
Train error with 28 patterns: 1986.044921875
Test error with 28 patterns: 1339.662841796875
Train error with 29 patterns: 2461.687255859375
Test error with 29 patterns: 1218.8602294921875
Train error with 30 patterns: 366.3486633300781
Test error with 30 patterns: 1443.3037109375
Train error with 31 patterns: 1761.245361328125
Test error with 31 patterns: 1032.8916015625
Train error with 32 patterns: 1221.2607421875
Test error with 32 patterns: 1503.9486083984375
Train error with 33 patterns: 5581.46875
Test error with 33 patterns: 1214.1710205078125
Train error with 34 patterns: 5141.5283203125
Test error with 34 patterns: 1291.941650390625
Train error with 35 patterns: 1485.9798583984375
Test error with 35 patterns: 529.0741577148438
Train error with 36 patterns: 1599.749755859375
Test error with 36 patterns: 1577.1539306640625
Train error with 37 patterns: 4989.90673828125
Test error with 37 patterns: 1558.8231201171875

Train error with 38 patterns: 3858.92138671875
Test error with 38 patterns: 1537.660888671875
Train error with 39 patterns: 1615.84033203125
Test error with 39 patterns: 933.9187622070312
Trial 2
Train error with 2 patterns: 53818.86328125
Test error with 2 patterns: 19948.513671875
Train error with 3 patterns: 28129.904296875
Test error with 3 patterns: 10016.14453125
Train error with 4 patterns: 3747.135986328125
Test error with 4 patterns: 3434.246826171875
Train error with 5 patterns: 6925.35791015625
Test error with 5 patterns: 13945.451171875
Train error with 6 patterns: 12789.25
Test error with 6 patterns: 4685.5869140625
Train error with 7 patterns: 10803.7626953125
Test error with 7 patterns: 5894.07421875
Train error with 8 patterns: 10333.896484375
Test error with 8 patterns: 3926.416015625
Train error with 9 patterns: 3847.89306640625
Test error with 9 patterns: 2212.184326171875
Train error with 10 patterns: 6418.3154296875
Test error with 10 patterns: 3706.98193359375
Train error with 11 patterns: 15152.978515625
Test error with 11 patterns: 725.8504638671875
Train error with 12 patterns: 2908.28271484375
Test error with 12 patterns: 2114.0625
Train error with 13 patterns: 3000.7783203125
Test error with 13 patterns: 2843.81640625
Train error with 14 patterns: 2400.8232421875
Test error with 14 patterns: 1282.383056640625
Train error with 15 patterns: 6643.3427734375
Test error with 15 patterns: 3004.9375
Train error with 16 patterns: 7557.78515625
Test error with 16 patterns: 5497.296875
Train error with 17 patterns: 3202.6142578125
Test error with 17 patterns: 7783.0966796875
Train error with 18 patterns: 14767.3583984375
Test error with 18 patterns: 7242.2919921875
Train error with 19 patterns: 3053.98681640625
Test error with 19 patterns: 1798.119384765625
Train error with 20 patterns: 8738.46875
Test error with 20 patterns: 3684.67626953125
Train error with 21 patterns: 9594.2822265625
Test error with 21 patterns: 1337.324951171875
Train error with 22 patterns: 1848.876708984375
Test error with 22 patterns: 1908.998779296875
Train error with 23 patterns: 2447.06591796875
Test error with 23 patterns: 1745.1591796875
Train error with 24 patterns: 3005.365234375
Test error with 24 patterns: 5083.708984375
Train error with 25 patterns: 4548.43212890625
Test error with 25 patterns: 3062.40087890625
Train error with 26 patterns: 4613.21142578125
Test error with 26 patterns: 3246.033447265625
Train error with 27 patterns: 7975.5009765625
Test error with 27 patterns: 4802.982421875
Train error with 28 patterns: 3060.5537109375
Test error with 28 patterns: 2399.64404296875
Train error with 29 patterns: 5924.9033203125
Test error with 29 patterns: 2350.4208984375
Train error with 30 patterns: 1589.7874755859375
Test error with 30 patterns: 2671.95263671875
Train error with 31 patterns: 2699.2646484375
Test error with 31 patterns: 1978.20849609375
Train error with 32 patterns: 4282.35498046875
Test error with 32 patterns: 2878.452392578125
Train error with 33 patterns: 6738.0537109375

Test error with 33 patterns: 2697.419921875
Train error with 34 patterns: 10131.1103515625
Test error with 34 patterns: 2730.474609375
Train error with 35 patterns: 2615.045654296875
Test error with 35 patterns: 1065.921875
Train error with 36 patterns: 2999.898681640625
Test error with 36 patterns: 3056.85888671875
Train error with 37 patterns: 6227.025390625
Test error with 37 patterns: 3200.0361328125
Train error with 38 patterns: 7717.01220703125
Test error with 38 patterns: 2900.14599609375
Train error with 39 patterns: 2560.15771484375
Test error with 39 patterns: 949.931640625

Trial 3

Train error with 2 patterns: 58507.3515625
Test error with 2 patterns: 24977.01953125
Train error with 3 patterns: 48801.828125
Test error with 3 patterns: 17161.443359375
Train error with 4 patterns: 6178.7021484375
Test error with 4 patterns: 3528.045166015625
Train error with 5 patterns: 8504.6201171875
Test error with 5 patterns: 16586.453125
Train error with 6 patterns: 16407.326171875
Test error with 6 patterns: 6957.5263671875
Train error with 7 patterns: 12956.7626953125
Test error with 7 patterns: 8252.5322265625
Train error with 8 patterns: 11929.5234375
Test error with 8 patterns: 4777.7099609375
Train error with 9 patterns: 5242.8447265625
Test error with 9 patterns: 3222.042724609375
Train error with 10 patterns: 7758.93408203125
Test error with 10 patterns: 6517.7578125
Train error with 11 patterns: 16110.115234375
Test error with 11 patterns: 1117.626220703125
Train error with 12 patterns: 4888.73974609375
Test error with 12 patterns: 3088.962890625
Train error with 13 patterns: 3832.865234375
Test error with 13 patterns: 3683.080322265625
Train error with 14 patterns: 3898.56005859375
Test error with 14 patterns: 1791.1536865234375
Train error with 15 patterns: 7697.1884765625
Test error with 15 patterns: 4040.168701171875
Train error with 16 patterns: 8475.154296875
Test error with 16 patterns: 8192.25390625
Train error with 17 patterns: 8445.037109375
Test error with 17 patterns: 15739.380859375
Train error with 18 patterns: 17709.8828125
Test error with 18 patterns: 8934.234375
Train error with 19 patterns: 6116.783203125
Test error with 19 patterns: 2796.44189453125
Train error with 20 patterns: 9403.16796875
Test error with 20 patterns: 5138.716796875
Train error with 21 patterns: 11668.06640625
Test error with 21 patterns: 1859.62109375
Train error with 22 patterns: 6915.7919921875
Test error with 22 patterns: 3095.029296875
Train error with 23 patterns: 3677.17138671875
Test error with 23 patterns: 2985.42578125
Train error with 24 patterns: 4128.24169921875
Test error with 24 patterns: 7244.51953125
Train error with 25 patterns: 4989.24951171875
Test error with 25 patterns: 4976.2265625
Train error with 26 patterns: 5125.68359375
Test error with 26 patterns: 4488.9638671875
Train error with 27 patterns: 8993.677734375
Test error with 27 patterns: 7566.896484375
Train error with 28 patterns: 4451.6748046875
Test error with 28 patterns: 2477.888404296875

Test error with 28 patterns: 3477.889404298875
Train error with 29 patterns: 9421.0830078125
Test error with 29 patterns: 3396.69873046875
Train error with 30 patterns: 5571.03759765625
Test error with 30 patterns: 3638.56787109375
Train error with 31 patterns: 3394.82373046875
Test error with 31 patterns: 2973.326171875
Train error with 32 patterns: 5678.0400390625
Test error with 32 patterns: 4518.3583984375
Train error with 33 patterns: 8968.6220703125
Test error with 33 patterns: 4253.6171875
Train error with 34 patterns: 10810.640625
Test error with 34 patterns: 4165.5458984375
Train error with 35 patterns: 6924.5244140625
Test error with 35 patterns: 1719.745361328125
Train error with 36 patterns: 4623.5673828125
Test error with 36 patterns: 4675.1025390625
Train error with 37 patterns: 7498.2890625
Test error with 37 patterns: 4563.900390625
Train error with 38 patterns: 8365.1767578125
Test error with 38 patterns: 4404.42626953125
Train error with 39 patterns: 1509.3875732421875
Test error with 39 patterns: 949.9705200195312

Trial 4

Train error with 2 patterns: 60381.0
Test error with 2 patterns: 25228.626953125
Train error with 3 patterns: 53838.296875
Test error with 3 patterns: 18284.08984375
Train error with 4 patterns: 19812.44140625
Test error with 4 patterns: 6954.54296875
Train error with 5 patterns: 18478.984375
Test error with 5 patterns: 21960.669921875
Train error with 6 patterns: 20142.40234375
Test error with 6 patterns: 8672.4521484375
Train error with 7 patterns: 16721.701171875
Test error with 7 patterns: 9925.6552734375
Train error with 8 patterns: 15787.2939453125
Test error with 8 patterns: 5435.14404296875
Train error with 9 patterns: 10932.5517578125
Test error with 9 patterns: 3926.557861328125
Train error with 10 patterns: 9807.4140625
Test error with 10 patterns: 8164.25732421875
Train error with 11 patterns: 27219.515625
Test error with 11 patterns: 1652.935791015625
Train error with 12 patterns: 6321.9482421875
Test error with 12 patterns: 3952.04296875
Train error with 13 patterns: 5893.40869140625
Test error with 13 patterns: 4470.9296875
Train error with 14 patterns: 5322.8798828125
Test error with 14 patterns: 2932.23828125
Train error with 15 patterns: 9079.18359375
Test error with 15 patterns: 5371.9326171875
Train error with 16 patterns: 11825.35546875
Test error with 16 patterns: 10353.4765625
Train error with 17 patterns: 9295.7236328125
Test error with 17 patterns: 20229.6015625
Train error with 18 patterns: 18890.609375
Test error with 18 patterns: 11066.1689453125
Train error with 19 patterns: 12056.796875
Test error with 19 patterns: 3577.7021484375
Train error with 20 patterns: 13632.1806640625
Test error with 20 patterns: 7215.9892578125
Train error with 21 patterns: 15164.6962890625
Test error with 21 patterns: 2741.3779296875
Train error with 22 patterns: 13590.0107421875
Test error with 22 patterns: 3912.096923828125
Train error with 23 patterns: 4932.98974609375
Test error with 23 patterns: 4079.3681640625

Train error with 24 patterns: 5936.9365234375
Test error with 24 patterns: 9171.0048828125
Train error with 25 patterns: 6209.1015625
Test error with 25 patterns: 6632.2451171875
Train error with 26 patterns: 5981.96484375
Test error with 26 patterns: 5680.79052734375
Train error with 27 patterns: 10429.01171875
Test error with 27 patterns: 9210.4853515625
Train error with 28 patterns: 6963.130859375
Test error with 28 patterns: 4527.4140625
Train error with 29 patterns: 12744.9990234375
Test error with 29 patterns: 4484.5166015625
Train error with 30 patterns: 9636.5966796875
Test error with 30 patterns: 4816.953125
Train error with 31 patterns: 9606.978515625
Test error with 31 patterns: 4392.248046875
Train error with 32 patterns: 9320.095703125
Test error with 32 patterns: 6348.62109375
Train error with 33 patterns: 12206.408203125
Test error with 33 patterns: 5573.958984375
Train error with 34 patterns: 11297.130859375
Test error with 34 patterns: 5891.748046875
Train error with 35 patterns: 7811.86181640625
Test error with 35 patterns: 2364.4775390625
Train error with 36 patterns: 5358.9482421875
Test error with 36 patterns: 6302.36181640625
Train error with 37 patterns: 8947.861328125
Test error with 37 patterns: 6007.73046875
Train error with 38 patterns: 10947.841796875
Test error with 38 patterns: 5706.92138671875
Train error with 39 patterns: 1134.1771240234375
Test error with 39 patterns: 850.5533447265625
Trial 5

Train error with 2 patterns: 63898.09765625
Test error with 2 patterns: 34690.375
Train error with 3 patterns: 65218.9375
Test error with 3 patterns: 21003.84765625
Train error with 4 patterns: 28732.2109375
Test error with 4 patterns: 10768.08984375
Train error with 5 patterns: 20832.18359375
Test error with 5 patterns: 27505.693359375
Train error with 6 patterns: 24238.73828125
Test error with 6 patterns: 10104.833984375
Train error with 7 patterns: 19018.884765625
Test error with 7 patterns: 12544.08203125
Train error with 8 patterns: 20948.22265625
Test error with 8 patterns: 6475.3046875
Train error with 9 patterns: 14094.6220703125
Test error with 9 patterns: 5284.48583984375
Train error with 10 patterns: 10503.0205078125
Test error with 10 patterns: 9548.1669921875
Train error with 11 patterns: 28127.177734375
Test error with 11 patterns: 1993.1527099609375
Train error with 12 patterns: 8128.10498046875
Test error with 12 patterns: 4912.525390625
Train error with 13 patterns: 11428.490234375
Test error with 13 patterns: 5801.79150390625
Train error with 14 patterns: 6075.21826171875
Test error with 14 patterns: 3178.3720703125
Train error with 15 patterns: 10573.625
Test error with 15 patterns: 6684.4140625
Train error with 16 patterns: 19996.66796875
Test error with 16 patterns: 13189.158203125
Train error with 17 patterns: 15531.962890625
Test error with 17 patterns: 25778.296875
Train error with 18 patterns: 20530.79296875
Test error with 18 patterns: 13672.4833984375
Train error with 19 patterns: 13050.4375

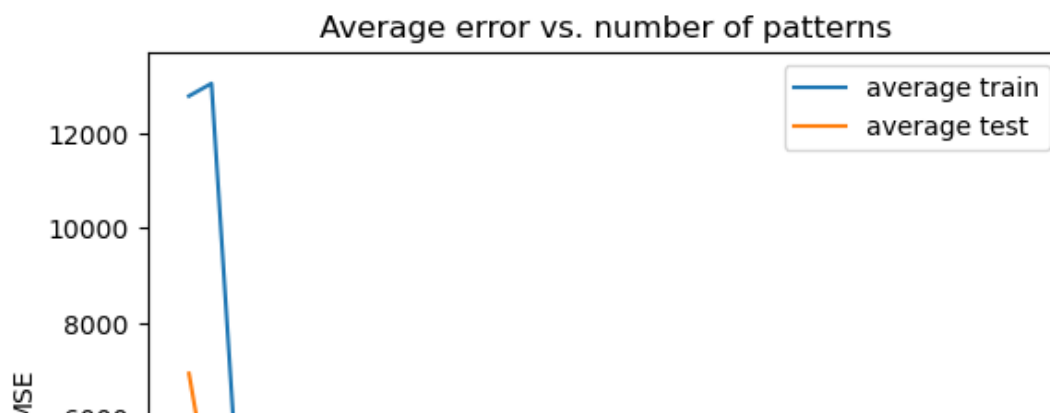
Test error with 19 patterns: 4674.68408203125
 Train error with 20 patterns: 18960.65234375
 Test error with 20 patterns: 8963.3671875
 Train error with 21 patterns: 15683.140625
 Test error with 21 patterns: 3586.12060546875
 Train error with 22 patterns: 15792.4970703125
 Test error with 22 patterns: 5479.46875
 Train error with 23 patterns: 10285.845703125
 Test error with 23 patterns: 4982.17578125
 Train error with 24 patterns: 7261.9248046875
 Test error with 24 patterns: 12118.1171875
 Train error with 25 patterns: 9720.7001953125
 Test error with 25 patterns: 8415.8193359375
 Train error with 26 patterns: 7599.20458984375
 Test error with 26 patterns: 6876.41357421875
 Train error with 27 patterns: 15695.080078125
 Test error with 27 patterns: 12705.3671875
 Train error with 28 patterns: 8609.4697265625
 Test error with 28 patterns: 5383.0263671875
 Train error with 29 patterns: 13608.5400390625
 Test error with 29 patterns: 5611.10791015625
 Train error with 30 patterns: 15334.0947265625
 Test error with 30 patterns: 5788.36181640625
 Train error with 31 patterns: 12357.8056640625
 Test error with 31 patterns: 5385.45703125
 Train error with 32 patterns: 10277.8994140625
 Test error with 32 patterns: 7812.1396484375
 Train error with 33 patterns: 13881.60546875
 Test error with 33 patterns: 6915.35888671875
 Train error with 34 patterns: 12496.1455078125
 Test error with 34 patterns: 7214.2626953125
 Train error with 35 patterns: 13666.98828125
 Test error with 35 patterns: 2976.70166015625
 Train error with 36 patterns: 6605.09912109375
 Test error with 36 patterns: 7730.828125
 Train error with 37 patterns: 12154.0537109375
 Test error with 37 patterns: 7376.33740234375
 Train error with 38 patterns: 16428.73046875
 Test error with 38 patterns: 7234.958984375
 Train error with 39 patterns: 2786.41845703125
 Test error with 39 patterns: 902.1381225585938

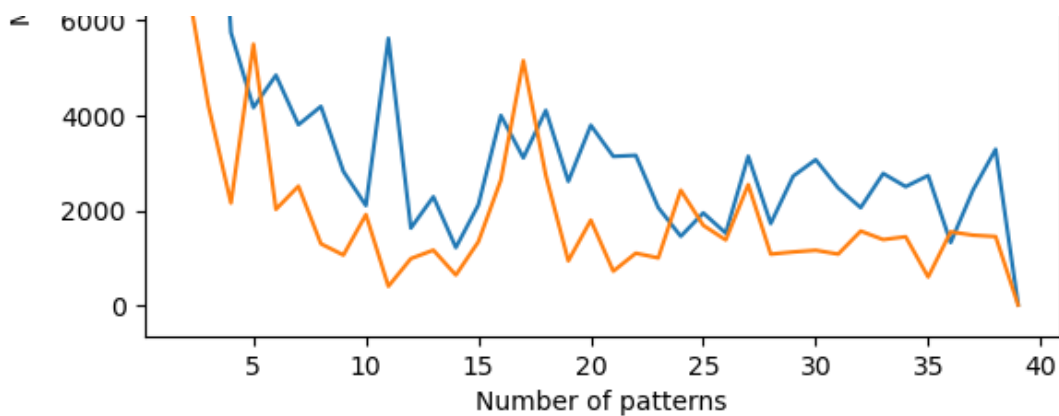
In [84]:

```

# 2.a.3 - Plot training and testing error.
plt.figure()
plt.plot(range(2,C), train_avg, label='average train')
plt.plot(range(2,C), test_avg, label='average test')
plt.xlabel('Number of patterns')
plt.ylabel('MSE')
plt.title('Average error vs. number of patterns')
plt.legend()
plt.show()

```





In [95]:

```
def salt_and_pepper(data, prob):
    """
    This method guarantees that the same number of pixels are flipped in each image
    """
    for img in range(data.shape[0]):
        # get number of pixels to flip for image size and selected probability:
        n_pixels = int(data.shape[1] * prob)

        # select a random sample of n_pixels:
        permuted = np.random.permutation(np.arange(0, data.shape[1], 1))
        pixels = permuted[:n_pixels]

        # flip the selected pixels:
        for p in pixels:
            data[img, p] = 1 - data[img, p]

    return data
```

In [96]:

```
# 2.a.4 - Test salt-and-pepper noise.
num_steps = 10
prob_flip = torch.linspace(0.01, 0.5, num_steps)
train_data = binary_img_tensor[:10, :]
test_data = binary_img_tensor[10:20, :]
mses = np.zeros((num_steps))

W = outer_product_hopfield_matrix(train_data)

# vary bit flip probability for each step in prob_flip:
for n, p in enumerate(prob_flip):
    noisy_test = test_data.clone()

    # randomly flip pixels in the training data:
    noisy_test = salt_and_pepper(noisy_test, p)

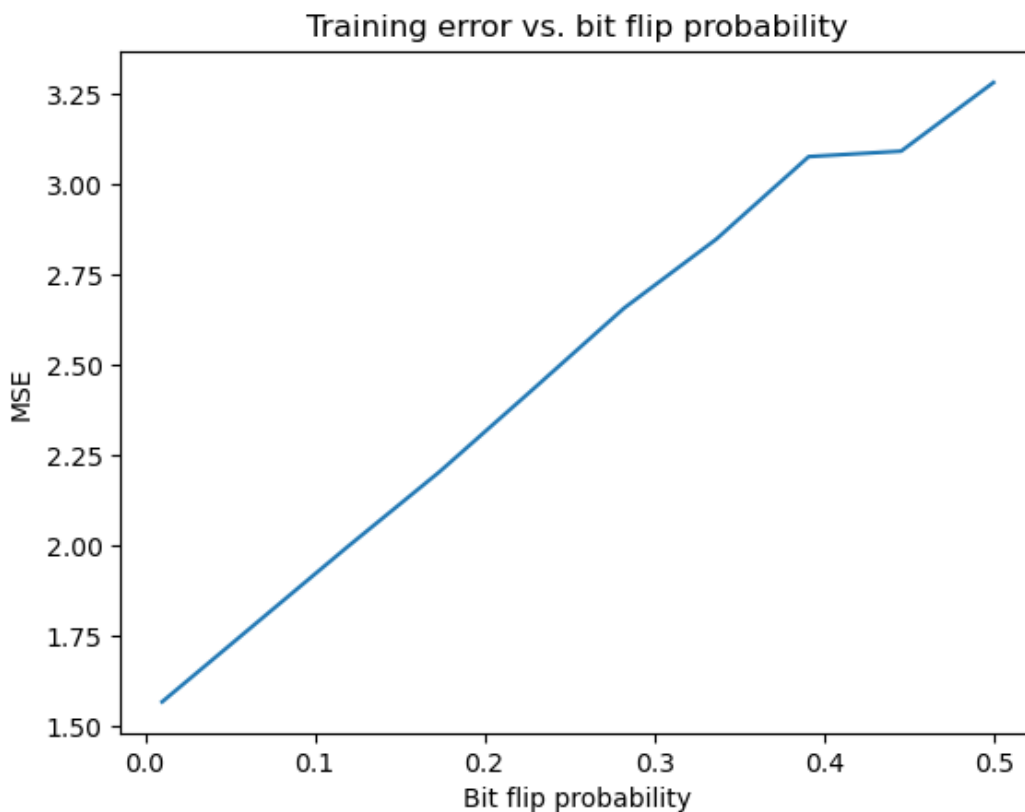
    # compute test error for each noisy pattern:
    for i in range(noisy_test.shape[0]):
        reconstructed_img, similarities = evaluate_hopfield_network(W, noisy_test[i, :].reshape(1, -1).T)
        mses[n] += mse(noisy_test[i, :], reconstructed_img)

    mses[n] /= noisy_test.shape[0]
```

In [97]:

```
# plot the training error vs. bit flip probability:
plt.figure()
plt.plot(prob_flip, mses)
plt.xlabel('Bit flip probability')
```

```
plt.ylabel('MSE')
plt.title('Training error vs. bit flip probability')
plt.show()
```



2.b) [2 Mark] Train Hopfield network using the pseudo-inverse and repeat the tasks 2.a) 2-4, however, for this example, ensure the threshold value is 0. Even though the pseudo-inverse has a greater capacity than the hopfield network, we will still test up to the theoretical capacity of the Hopfield learning rule.

In [98]:

```
def pseudo_inv(X, lamb=0.01):
    """
    pseudo_inv - Implements the pseudoinverse from the previous assignment.

    Parameters:
    -----
    X : torch.Tensor
        A (num_patterns, num_features) Tensor holding the training data.
    lamb : float
        The regularization term for the pseudoinverse

    Returns:
    -----
    The pseudoinverse of X
    """
    return torch.inverse(X.T@X + lamb * torch.eye(X.shape[1]).float()) @ X.T

def pseudoinverse_hopfield_matrix(training_patterns, lamb=0.01):
    """
    pseudoinverse_hopfield_matrix - Uses the regularized pseudoinverse to construct a weight
    matrix for
        a non-modern Hopfield network.

    Parameters:
    -----
    training_patterns : torch.Tensor
        The (num_patterns, num_features) Tensor containing the training data.
    lamb : float
```


The regularization term for the pseudo-inverse.

Returns:

The weight matrix compatible with the evaluate_hopfield_network function
'''

```
W = pseudo_inv(training_patterns, lamb=lamb) @ training_patterns
return W / training_patterns.shape[0]
```

In [99]:

```
# repeat 2.a.2 with the pseudoinverse weight matrix:

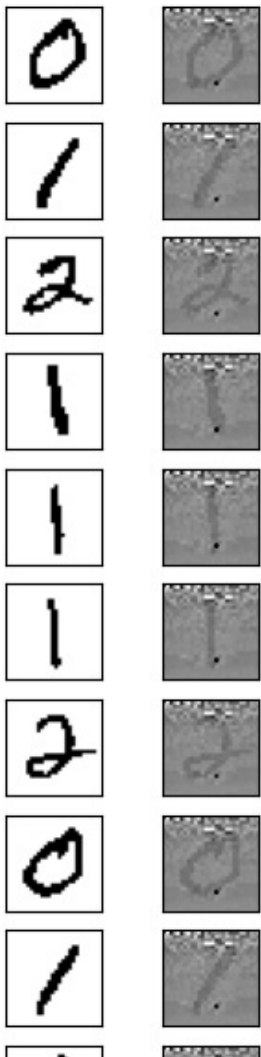
# create weight matrix, W
W_pseudo = pseudoinverse_hopfield_matrix(binary_img_tensor[:10,:])

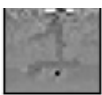
plt.figure(figsize=(2,8))

for i in range(10):
    # reconstruct training image, i.
    reconstructed_img = W_pseudo @ binary_img_tensor[i,:].reshape((1,-1)).T

    # plot original image
    plt.subplot(10,2,1+2*i)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(binary_imgs[i].reshape((28,28)), vmin=-1, vmax=1, cmap='gray_r')

    # plot reconstructed image
    plt.subplot(10,2,1+2*i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(reconstructed_img.reshape((28,28)), vmin=-1, vmax=1, cmap='gray_r')
```





In [100]:

```
# repeat 2.a.3 with the pseudoinverse weight matrix:
train_avg_pseudo, test_avg_pseudo = test_network(binary_img_tensor[:40,:], binary_img_tensor[40:80,:], C, method='pseudo_inv')
```

```
Trial 1
Train error with 2 patterns: 18.99361228942871
Test error with 2 patterns: 6.499236583709717
Train error with 3 patterns: 3.5399248600006104
Test error with 3 patterns: 2.5406882762908936
Train error with 4 patterns: 1.5302963256835938
Test error with 4 patterns: 1.5757850408554077
Train error with 5 patterns: 1.4271588325500488
Test error with 5 patterns: 1.2561595439910889
Train error with 6 patterns: 3.0607070922851562
Test error with 6 patterns: 2.235905647277832
Train error with 7 patterns: 4.004777431488037
Test error with 7 patterns: 2.5507469177246094
Train error with 8 patterns: 1.1753640174865723
Test error with 8 patterns: 1.1699212789535522
Train error with 9 patterns: 3.473083019256592
Test error with 9 patterns: 4.564286231994629
Train error with 10 patterns: 2.1821584701538086
Test error with 10 patterns: 2.176326274871826
Train error with 11 patterns: 10.056584358215332
Test error with 11 patterns: 6.681079864501953
Train error with 12 patterns: 1.4214050769805908
Test error with 12 patterns: 1.323349118232727
Train error with 13 patterns: 2.233267068862915
Test error with 13 patterns: 2.1453535556793213
Train error with 14 patterns: 1.3009603023529053
Test error with 14 patterns: 1.2703211307525635
Train error with 15 patterns: 3.762040138244629
Test error with 15 patterns: 3.226418972015381
Train error with 16 patterns: 1.3671079874038696
Test error with 16 patterns: 1.415519118309021
Train error with 17 patterns: 4.661015033721924
Test error with 17 patterns: 5.077094554901123
Train error with 18 patterns: 1.2912507057189941
Test error with 18 patterns: 1.2473961114883423
Train error with 19 patterns: 1.3537806272506714
Test error with 19 patterns: 1.3982475996017456
Train error with 20 patterns: 2.425196886062622
Test error with 20 patterns: 2.491262912750244
Train error with 21 patterns: 3.4241859912872314
Test error with 21 patterns: 2.750239610671997
Train error with 22 patterns: 3.3450655937194824
Test error with 22 patterns: 3.418970823287964
Train error with 23 patterns: 3.7796337604522705
Test error with 23 patterns: 3.9377541542053223
Train error with 24 patterns: 2.5093605518341064
Test error with 24 patterns: 2.2815444469451904
Train error with 25 patterns: 3.984649896621704
Test error with 25 patterns: 3.161036968231201
Train error with 26 patterns: 1.9538573026657104
Test error with 26 patterns: 1.7201552391052246
Train error with 27 patterns: 6.950435638427734
Test error with 27 patterns: 5.507430076599121
Train error with 28 patterns: 3.646420955657959
Test error with 28 patterns: 2.909663200378418
Train error with 29 patterns: 4.360316276550293
Test error with 29 patterns: 4.839378356933594
Train error with 30 patterns: 4.581050205065576
```

Train error with 30 patterns: 4.581050395965576
Test error with 30 patterns: 4.4998087882995605
Train error with 31 patterns: 1.9530658721923828
Test error with 31 patterns: 2.1455860137939453
Train error with 32 patterns: 3.8395776748657227
Test error with 32 patterns: 2.7654762268066406
Train error with 33 patterns: 3.7681500911712646
Test error with 33 patterns: 4.138830661773682
Train error with 34 patterns: 1.9126112461090088
Test error with 34 patterns: 1.5922499895095825
Train error with 35 patterns: 3.3966281414031982
Test error with 35 patterns: 3.53224515914917
Train error with 36 patterns: 3.1010448932647705
Test error with 36 patterns: 3.2340686321258545
Train error with 37 patterns: 12.384049415588379
Test error with 37 patterns: 8.557955741882324
Train error with 38 patterns: 1.4499298334121704
Test error with 38 patterns: 1.3527988195419312
Train error with 39 patterns: 7.669066905975342
Test error with 39 patterns: 7.303956985473633

Trial 2

Train error with 2 patterns: 23.791650772094727
Test error with 2 patterns: 8.991059303283691
Train error with 3 patterns: 6.6845293045043945
Test error with 3 patterns: 5.195241451263428
Train error with 4 patterns: 3.234156608581543
Test error with 4 patterns: 3.2122621536254883
Train error with 5 patterns: 2.3710107803344727
Test error with 5 patterns: 2.3110907077789307
Train error with 6 patterns: 4.255857467651367
Test error with 6 patterns: 3.4540841579437256
Train error with 7 patterns: 6.78385066986084
Test error with 7 patterns: 4.265451431274414
Train error with 8 patterns: 2.6221301555633545
Test error with 8 patterns: 2.542571783065796
Train error with 9 patterns: 5.973381042480469
Test error with 9 patterns: 7.46022367477417
Train error with 10 patterns: 6.768260955810547
Test error with 10 patterns: 6.645397186279297
Train error with 11 patterns: 11.145301818847656
Test error with 11 patterns: 7.753080368041992
Train error with 12 patterns: 2.9314751625061035
Test error with 12 patterns: 2.7888131141662598
Train error with 13 patterns: 3.676769256591797
Test error with 13 patterns: 3.6605916023254395
Train error with 14 patterns: 3.6758954524993896
Test error with 14 patterns: 3.394597053527832
Train error with 15 patterns: 6.6732988357543945
Test error with 15 patterns: 5.66718053817749
Train error with 16 patterns: 3.0658111572265625
Test error with 16 patterns: 3.06009578704834
Train error with 17 patterns: 6.728035926818848
Test error with 17 patterns: 7.021824359893799
Train error with 18 patterns: 2.8890578746795654
Test error with 18 patterns: 2.768437147140503
Train error with 19 patterns: 2.7122154235839844
Test error with 19 patterns: 2.7918009757995605
Train error with 20 patterns: 4.294790267944336
Test error with 20 patterns: 4.412018775939941
Train error with 21 patterns: 4.679637908935547
Test error with 21 patterns: 3.96022367477417
Train error with 22 patterns: 5.322826862335205
Test error with 22 patterns: 5.600087642669678
Train error with 23 patterns: 7.006855010986328
Test error with 23 patterns: 7.1535539627075195
Train error with 24 patterns: 4.9103593826293945
Test error with 24 patterns: 4.233780860900879
Train error with 25 patterns: 10.178986549377441

Test error with 25 patterns: 7.940831661224365
Train error with 26 patterns: 3.990907669067383
Test error with 26 patterns: 3.6040382385253906
Train error with 27 patterns: 9.63485336303711
Test error with 27 patterns: 8.197696685791016
Train error with 28 patterns: 5.112870216369629
Test error with 28 patterns: 4.2848944664001465
Train error with 29 patterns: 8.535173416137695
Test error with 29 patterns: 8.803304672241211
Train error with 30 patterns: 10.662391662597656
Test error with 30 patterns: 12.436904907226562
Train error with 31 patterns: 6.424538612365723
Test error with 31 patterns: 7.042075157165527
Train error with 32 patterns: 12.946603775024414
Test error with 32 patterns: 10.108423233032227
Train error with 33 patterns: 5.576669692993164
Test error with 33 patterns: 6.0723090171813965
Train error with 34 patterns: 4.658988952636719
Test error with 34 patterns: 3.636569023132324
Train error with 35 patterns: 6.447815895080566
Test error with 35 patterns: 7.545795917510986
Train error with 36 patterns: 7.875162124633789
Test error with 36 patterns: 8.029867172241211
Train error with 37 patterns: 17.519489288330078
Test error with 37 patterns: 12.185343742370605
Train error with 38 patterns: 13.91700267791748
Test error with 38 patterns: 11.40699577331543
Train error with 39 patterns: 7.313145637512207
Test error with 39 patterns: 4.799083709716797

Trial 3

Train error with 2 patterns: 30.43768310546875
Test error with 2 patterns: 11.781071662902832
Train error with 3 patterns: 8.310308456420898
Test error with 3 patterns: 6.418981552124023
Train error with 4 patterns: 4.823490142822266
Test error with 4 patterns: 4.701641082763672
Train error with 5 patterns: 9.237737655639648
Test error with 5 patterns: 5.757145404815674
Train error with 6 patterns: 14.076397895812988
Test error with 6 patterns: 9.593771934509277
Train error with 7 patterns: 9.072526931762695
Test error with 7 patterns: 6.008246898651123
Train error with 8 patterns: 4.2214508056640625
Test error with 8 patterns: 4.033987045288086
Train error with 9 patterns: 8.811503410339355
Test error with 9 patterns: 10.180694580078125
Train error with 10 patterns: 7.886525630950928
Test error with 10 patterns: 7.75232458114624
Train error with 11 patterns: 12.468419075012207
Test error with 11 patterns: 8.949881553649902
Train error with 12 patterns: 4.224926471710205
Test error with 12 patterns: 4.067413806915283
Train error with 13 patterns: 5.298290252685547
Test error with 13 patterns: 5.432851314544678
Train error with 14 patterns: 5.905872344970703
Test error with 14 patterns: 5.259888172149658
Train error with 15 patterns: 8.154095649719238
Test error with 15 patterns: 7.258509635925293
Train error with 16 patterns: 4.447973728179932
Test error with 16 patterns: 4.360064506530762
Train error with 17 patterns: 8.656591415405273
Test error with 17 patterns: 9.132406234741211
Train error with 18 patterns: 4.064434051513672
Test error with 18 patterns: 3.9697623252868652
Train error with 19 patterns: 5.222080230712891
Test error with 19 patterns: 5.766783714294434
Train error with 20 patterns: 5.979790687561035
Test error with 20 patterns: 6.460988521575928

Train error with 21 patterns: 6.536507606506348
Test error with 21 patterns: 5.702358245849609
Train error with 22 patterns: 7.447958946228027
Test error with 22 patterns: 7.716407775878906
Train error with 23 patterns: 9.590059280395508
Test error with 23 patterns: 9.873826026916504
Train error with 24 patterns: 7.5049896240234375
Test error with 24 patterns: 6.991917610168457
Train error with 25 patterns: 13.636098861694336
Test error with 25 patterns: 11.15259838104248
Train error with 26 patterns: 5.975358963012695
Test error with 26 patterns: 5.657688140869141
Train error with 27 patterns: 19.674667358398438
Test error with 27 patterns: 15.943475723266602
Train error with 28 patterns: 7.911473751068115
Test error with 28 patterns: 7.039003372192383
Train error with 29 patterns: 16.51219940185547
Test error with 29 patterns: 18.59542465209961
Train error with 30 patterns: 12.070241928100586
Test error with 30 patterns: 13.86870288848877
Train error with 31 patterns: 7.649663925170898
Test error with 31 patterns: 8.271610260009766
Train error with 32 patterns: 17.083969116210938
Test error with 32 patterns: 13.125863075256348
Train error with 33 patterns: 7.815369129180908
Test error with 33 patterns: 8.289281845092773
Train error with 34 patterns: 8.392959594726562
Test error with 34 patterns: 6.7948384284973145
Train error with 35 patterns: 7.774797439575195
Test error with 35 patterns: 8.896560668945312
Train error with 36 patterns: 21.83182716369629
Test error with 36 patterns: 22.50494384765625
Train error with 37 patterns: 21.487459182739258
Test error with 37 patterns: 16.185441970825195
Train error with 38 patterns: 22.18851089477539
Test error with 38 patterns: 19.10326385498047
Train error with 39 patterns: 3.59614634513855
Test error with 39 patterns: 2.8062210083007812

Trial 4

Train error with 2 patterns: 37.305809020996094
Test error with 2 patterns: 16.94283676147461
Train error with 3 patterns: 11.121260643005371
Test error with 3 patterns: 8.572504043579102
Train error with 4 patterns: 6.9114484786987305
Test error with 4 patterns: 6.438920021057129
Train error with 5 patterns: 13.415855407714844
Test error with 5 patterns: 8.603767395019531
Train error with 6 patterns: 16.67186164855957
Test error with 6 patterns: 11.759039878845215
Train error with 7 patterns: 11.079361915588379
Test error with 7 patterns: 7.664592742919922
Train error with 8 patterns: 6.260951042175293
Test error with 8 patterns: 5.87349271774292
Train error with 9 patterns: 10.497200012207031
Test error with 9 patterns: 12.117331504821777
Train error with 10 patterns: 9.484588623046875
Test error with 10 patterns: 9.408445358276367
Train error with 11 patterns: 13.83496379852295
Test error with 11 patterns: 10.317437171936035
Train error with 12 patterns: 5.693351745605469
Test error with 12 patterns: 5.545499324798584
Train error with 13 patterns: 6.4710307121276855
Test error with 13 patterns: 6.632607936859131
Train error with 14 patterns: 8.33853530883789
Test error with 14 patterns: 7.4031662940979
Train error with 15 patterns: 9.52743148803711
Test error with 15 patterns: 8.585936546325684
Train error with 16 patterns: 7.016025542212201

Train error with 16 patterns: 7.016025543212891
Test error with 16 patterns: 6.9094438552856445
Train error with 17 patterns: 10.761678695678711
Test error with 17 patterns: 10.919517517089844
Train error with 18 patterns: 5.460628509521484
Test error with 18 patterns: 5.329776287078857
Train error with 19 patterns: 7.891156196594238
Test error with 19 patterns: 8.571233749389648
Train error with 20 patterns: 7.238903045654297
Test error with 20 patterns: 7.800136566162109
Train error with 21 patterns: 7.929653644561768
Test error with 21 patterns: 7.058406352996826
Train error with 22 patterns: 17.84796905517578
Test error with 22 patterns: 21.787431716918945
Train error with 23 patterns: 12.654867172241211
Test error with 23 patterns: 13.340099334716797
Train error with 24 patterns: 8.65139102935791
Test error with 24 patterns: 8.10820198059082
Train error with 25 patterns: 16.575969696044922
Test error with 25 patterns: 13.593351364135742
Train error with 26 patterns: 9.448139190673828
Test error with 26 patterns: 8.607161521911621
Train error with 27 patterns: 21.97686004638672
Test error with 27 patterns: 18.410255432128906
Train error with 28 patterns: 12.050600051879883
Test error with 28 patterns: 11.85367202758789
Train error with 29 patterns: 27.35295867919922
Test error with 29 patterns: 31.985607147216797
Train error with 30 patterns: 14.303590774536133
Test error with 30 patterns: 16.55257225036621
Train error with 31 patterns: 13.269428253173828
Test error with 31 patterns: 13.72207260131836
Train error with 32 patterns: 18.737377166748047
Test error with 32 patterns: 14.586321830749512
Train error with 33 patterns: 26.01718521118164
Test error with 33 patterns: 26.739574432373047
Train error with 34 patterns: 9.756746292114258
Test error with 34 patterns: 8.028924942016602
Train error with 35 patterns: 17.367813110351562
Test error with 35 patterns: 20.470415115356445
Train error with 36 patterns: 24.186309814453125
Test error with 36 patterns: 25.498193740844727
Train error with 37 patterns: 37.73538589477539
Test error with 37 patterns: 27.019058227539062
Train error with 38 patterns: 25.55539894104004
Test error with 38 patterns: 22.794414520263672
Train error with 39 patterns: 6.22249174118042
Test error with 39 patterns: 6.002554416656494

Trial 5

Train error with 2 patterns: 41.457000732421875
Test error with 2 patterns: 19.462011337280273
Train error with 3 patterns: 16.204692840576172
Test error with 3 patterns: 12.547893524169922
Train error with 4 patterns: 10.034789085388184
Test error with 4 patterns: 9.207764625549316
Train error with 5 patterns: 14.518681526184082
Test error with 5 patterns: 9.721640586853027
Train error with 6 patterns: 20.690317153930664
Test error with 6 patterns: 14.694652557373047
Train error with 7 patterns: 12.660417556762695
Test error with 7 patterns: 9.00688362121582
Train error with 8 patterns: 7.788530349731445
Test error with 8 patterns: 7.240262985229492
Train error with 9 patterns: 11.707806587219238
Test error with 9 patterns: 13.297087669372559
Train error with 10 patterns: 10.689693450927734
Test error with 10 patterns: 10.651429176330566
Train error with 11 patterns: 15.38919448852539

Test error with 11 patterns: 11.805849075317383
 Train error with 12 patterns: 7.820155143737793
 Test error with 12 patterns: 7.444737911224365
 Train error with 13 patterns: 8.171658515930176
 Test error with 13 patterns: 8.343305587768555
 Train error with 14 patterns: 9.40735149383545
 Test error with 14 patterns: 8.477835655212402
 Train error with 15 patterns: 10.824149131774902
 Test error with 15 patterns: 9.847485542297363
 Train error with 16 patterns: 11.656670570373535
 Test error with 16 patterns: 10.879741668701172
 Train error with 17 patterns: 12.853742599487305
 Test error with 17 patterns: 12.89273738861084
 Train error with 18 patterns: 7.077268123626709
 Test error with 18 patterns: 6.743849754333496
 Train error with 19 patterns: 9.987785339355469
 Test error with 19 patterns: 10.881474494934082
 Train error with 20 patterns: 8.514314651489258
 Test error with 20 patterns: 9.091981887817383
 Train error with 21 patterns: 9.201909065246582
 Test error with 21 patterns: 8.286593437194824
 Train error with 22 patterns: 21.67560577392578
 Test error with 22 patterns: 26.37179183959961
 Train error with 23 patterns: 16.495311737060547
 Test error with 23 patterns: 17.82758331298828
 Train error with 24 patterns: 12.68526840209961
 Test error with 24 patterns: 11.373497009277344
 Train error with 25 patterns: 24.375471115112305
 Test error with 25 patterns: 19.722129821777344
 Train error with 26 patterns: 59.87852096557617
 Test error with 26 patterns: 68.45317840576172
 Train error with 27 patterns: 25.595212936401367
 Test error with 27 patterns: 22.008392333984375
 Train error with 28 patterns: 13.454939842224121
 Test error with 28 patterns: 13.23831558227539
 Train error with 29 patterns: 34.88291931152344
 Test error with 29 patterns: 40.41469955444336
 Train error with 30 patterns: 17.39556884765625
 Test error with 30 patterns: 20.874107360839844
 Train error with 31 patterns: 15.418939590454102
 Test error with 31 patterns: 16.332143783569336
 Train error with 32 patterns: 21.765207290649414
 Test error with 32 patterns: 17.148447036743164
 Train error with 33 patterns: 30.04388427734375
 Test error with 33 patterns: 30.873245239257812
 Train error with 34 patterns: 11.30770206451416
 Test error with 34 patterns: 9.43504810333252
 Train error with 35 patterns: 19.952245712280273
 Test error with 35 patterns: 22.969003677368164
 Train error with 36 patterns: 25.50362205505371
 Test error with 36 patterns: 26.89909553527832
 Train error with 37 patterns: 42.44252395629883
 Test error with 37 patterns: 30.412641525268555
 Train error with 38 patterns: 27.760879516601562
 Test error with 38 patterns: 24.7841796875
 Train error with 39 patterns: 2.3043694496154785
 Test error with 39 patterns: 2.164994955062866

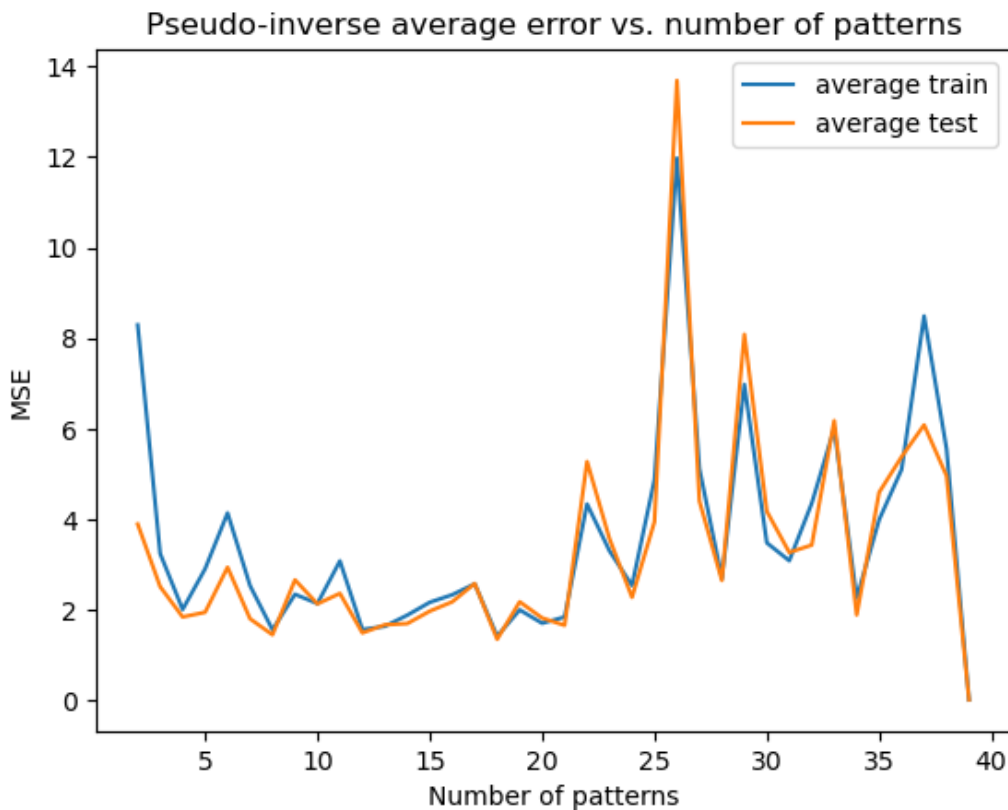
In [101]:

```

plt.figure()
plt.plot(range(2,C), train_avg_pseudo, label='average train')
plt.plot(range(2,C), test_avg_pseudo, label='average test')
plt.xlabel('Number of patterns')
plt.ylabel('MSE')
plt.title('Pseudo-inverse average error vs. number of patterns')
plt.legend()

```

```
plt.show()
```



```
In [ ]:
```

```
# repeat 2.a.4 with the pseudoinverse weight matrix and salt-and-pepper noise:
num_steps = 10
prob_flip = torch.linspace(0.01,0.5,num_steps)
train_data = binary_img_tensor[:10,:]
test_data = binary_img_tensor[10:20,:]
mses = np.zeros((num_steps))

W = pseudoinverse_hopfield_matrix(train_data)

# vary bit flip probability for each step in prob_flip:
for n, p in enumerate(prob_flip):
    noisy_test = test_data.clone()

    # randomly flip pixels in the training data:
    noisy_test = salt_and_pepper(noisy_test, p)

    # compute test error for each noisy pattern:
    for i in range(noisy_test.shape[0]):
        reconstructed_img, similarities = evaluate_hopfield_network(W, noisy_test[i,:].reshape(1, -1).T, threshold=0)
        mses[n] += mse(noisy_test[i,:], reconstructed_img)

    mses[n] /= noisy_test.shape[0]
```

2.c) [1 Mark] Show how the network behaves when an entire region of the image is corrupted.

For your best performing network, take three test images and set half of the inputs to be equal to zero. Plot the original images, the corrupted images, and the reconstructed images, side-by-side.

```
In [102]:
```

```
test_imgs = binary_img_tensor[:10,:]
corrupted_imgs = test_imgs.clone()
reconstructed_imgs = torch.zeros((test_imgs.shape[0],test_imgs.shape[1]))
```



```
W = outer_product_hopfield_matrix(test_imgs)
```

```
# corrupt half of the image:
```

```
for i in range(corrupted_imgs.shape[0]):  
    corrupted_imgs[i,392:] = 0 # set half of the image pixels to 0
```

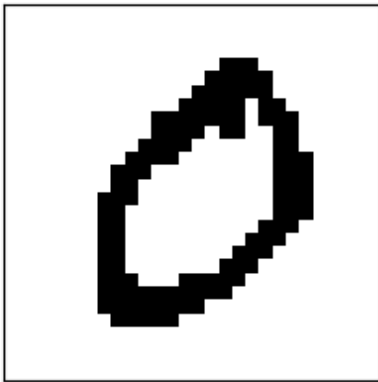
```
In [103]:
```

```
# plot original, corrupted, and reconstructed images
```

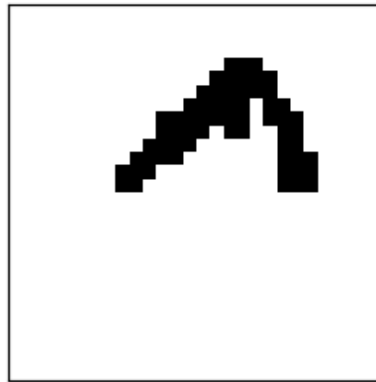
```
plt.subplots(3, 3, figsize=(10,9))
```

```
for i in range(3):  
    plt.subplot(3,3,1+3*i)  
    plt.imshow(test_imgs[i].reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')  
    plt.xticks([])  
    plt.yticks([])  
    plt.title('Original')  
  
    reconstructed_img = W @ corrupted_imgs[i,:].reshape(1, -1).T  
    plt.subplot(3,3,2+3*i)  
    plt.imshow(corrupted_imgs[i].reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')  
    plt.xticks([])  
    plt.yticks([])  
    plt.title('Corrupted')  
  
    plt.subplot(3,3,3+3*i)  
    plt.imshow(reconstructed_img.reshape((28,28)), cmap='gray_r')  
    plt.xticks([])  
    plt.yticks([])  
    plt.title('Reconstructed')
```

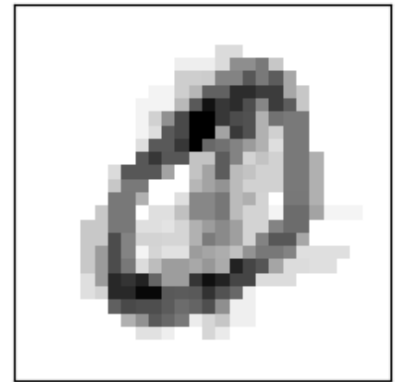
Original



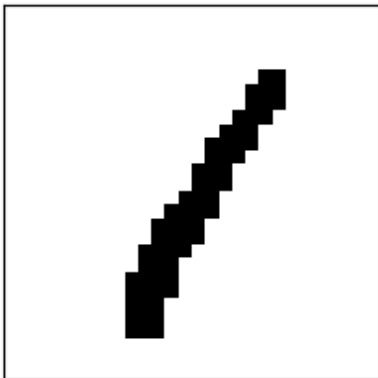
Corrupted



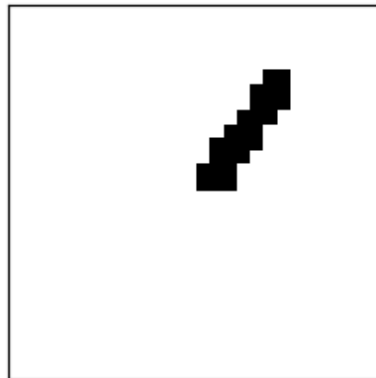
Reconstructed



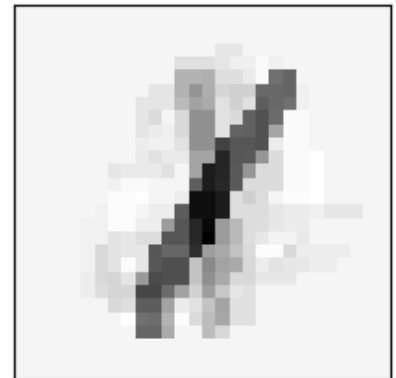
Original



Corrupted



Reconstructed



Original

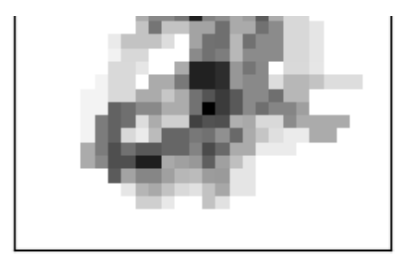
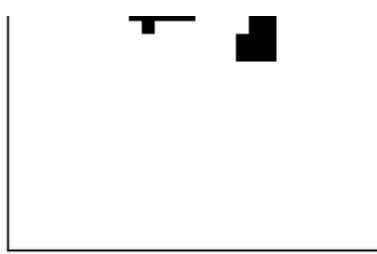
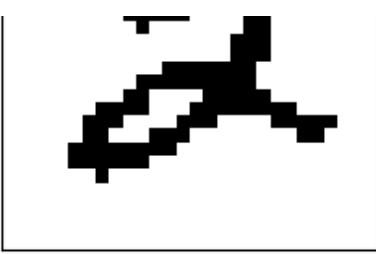


Corrupted



Reconstructed





3. Autoencoders

Now we are going to train an autoencoder to perform the same associative task that we explored above. We are going to define our Autoencoder using sigmoid neurons, which shouldn't be too terrible, since it is a relatively shallow network, but one is not obligated to use those.

Because we are going to use a sigmoidal output function, so we should be concerned with values in the range $[0, 1]$, instead of $\{-1, 1\}$

. Training will be done using the original MNIST images, **not** the binary images.

In [104]:

```
from torch.utils.data import TensorDataset, DataLoader

x_train_tensor = torch.from_numpy(x_train.reshape((-1, 28*28)).astype(np.float32) / 255)
x_test_tensor = torch.from_numpy(x_test.reshape((-1, 28*28)).astype(np.float32) / 255)

train_dataset = TensorDataset(x_train_tensor, x_train_tensor)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=1000, shuffle=True)

test_dataset = TensorDataset(x_test_tensor, x_test_tensor)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=1000, shuffle=True)
```

In [105]:

```
class Autoencoder(torch.nn.Module):
    def __init__(self, num_inputs, num_hidden):
        super().__init__()

        # Building an linear encoder with Linear layer followed by Sigmoid activation function
        self.encoder = torch.nn.Sequential(
            torch.nn.Linear(num_inputs, num_hidden),
            torch.nn.Sigmoid(),
        )

        # Building an linear decoder with Linear layer followed by Sigmoid activation function
        # The Sigmoid activation function outputs the value between 0 and 1
        self.decoder = torch.nn.Sequential(
            torch.nn.Linear(num_hidden, num_inputs),
            torch.nn.Sigmoid(),
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

In [106]:

```
import torch.nn as nn
import torch.nn.functional as F

def continue_training(training_loss, testing_loss):
```

```

network.train()          # configure the network for training
for i in range(10):      # train the network 10 times
    epoch_loss = []
    for data, target in train_loader:          # working in batches of 1000
        optimizer.zero_grad()                 # initialize the learning system
        output = network(data)                 # feed in the data
        loss = F.mse_loss(output, target)      # compute how wrong the output is
        loss.backward()                        # change the weights to reduce error
        optimizer.step()                       # update the learning rule
        epoch_loss.append(loss.detach().numpy())

    # update the list of training accuracy values
    training_loss.append(np.mean(epoch_loss)) # store the loss for later.
    print('Iteration', len(training_loss), 'Training loss:', training_loss[-1])

correct = 0
network.eval()
test_set_loss = []
for data, target in test_loader:              # go through the test data once (in groups of 1000)
    output = network(data)                     # feed in the data
    loss = F.mse_loss(output, target)
    test_set_loss.append(loss.detach().numpy())

# update the list of testing accuracy values
testing_loss.append(np.mean(test_set_loss))
print('Iteration', len(testing_loss), 'Testing accuracy:', testing_loss[-1])

```

3.1 [2 Marks] For the autoencoder we will do the following:

1. Train the network and plot the testing and training losses (repeated trials) for 5 different different values of the hidden layer size. Ensure the number of hidden layers is always less than the number of input features . Select a good number of training iterations (i.e., not overfitting) and a good number of hidden neurons
2. Plot 10 input and reconstructed images from the training set and 10 from the testing set. How do these compare to the reconstructions of from the Hopfield networks you constructed above?
3. With your trained network, compare the loss on inputs corrupted salt and pepper noise. Sweep through a range of number of pixels corrupted from 0 to 75%. Because the image data is back in the range [0,1] , this time corrupt the image by setting pixels to equal 0 or 1 with 50% probability.

In [107]:

```

# create the autoencoder here
network = Autoencoder(784, 3)

# We're all hip, fashionable people here, let's use the Adam optimizer.
optimizer = torch.optim.Adam(network.parameters(), lr = 1e-1)

```

In [152]:

```

# 3.1.1 - Train the network
hidden_layer_sizes = [4, 8, 16, 32, 64, 128, 256] # num units in hidden layer (< num features)
num_epochs = 20
train_mean = np.zeros((len(hidden_layer_sizes), num_epochs)) # row for each hidden layer size; col for each epoch
test_mean = np.zeros((len(hidden_layer_sizes), num_epochs))
autoencoders = [] # store a list of networks so the best one can be used later

# train the network, for 5 different values of hidden layer size
for i, sz in enumerate(hidden_layer_sizes):
    print(f"Training network with {sz} hidden units")

    # average over 5 trials
    for j in range(5):
        training_loss = []

```

```

testing_loss = []

# create autoencoder
network = Autoencoder(784, sz)
autoencoders.append(network)
optimizer = torch.optim.Adam(network.parameters(), lr=1e-1)

# train network for 10 epochs
for iter in range(num_epochs):
    continue_training(training_loss, testing_loss)

    train_mean[i] += np.array(training_loss)
    test_mean[i] += np.array(testing_loss)

train_mean[i] /= 5
test_mean[i] /= 5

```

```

Training network with 4 hidden units
Iteration 1 Training loss: 0.06733804
Iteration 1 Testing accuracy: 0.0675423
Iteration 2 Training loss: 0.06730151
Iteration 2 Testing accuracy: 0.06753073
Iteration 3 Training loss: 0.06729693
Iteration 3 Testing accuracy: 0.06749482
Iteration 4 Training loss: 0.06729359
Iteration 4 Testing accuracy: 0.06753801
Iteration 5 Training loss: 0.067299575
Iteration 5 Testing accuracy: 0.067507885
Iteration 6 Training loss: 0.067294
Iteration 6 Testing accuracy: 0.06749748
Iteration 7 Training loss: 0.06728947
Iteration 7 Testing accuracy: 0.06750034
Iteration 8 Training loss: 0.06729066
Iteration 8 Testing accuracy: 0.067566656
Iteration 9 Training loss: 0.06729678
Iteration 9 Testing accuracy: 0.06749187
Iteration 10 Training loss: 0.06729246
Iteration 10 Testing accuracy: 0.067474976
Iteration 11 Training loss: 0.067289755
Iteration 11 Testing accuracy: 0.06749278
Iteration 12 Training loss: 0.06728892
Iteration 12 Testing accuracy: 0.06749015
Iteration 13 Training loss: 0.067293
Iteration 13 Testing accuracy: 0.06748706
Iteration 14 Training loss: 0.06729453
Iteration 14 Testing accuracy: 0.067497656
Iteration 15 Training loss: 0.06729346
Iteration 15 Testing accuracy: 0.06749268
Iteration 16 Training loss: 0.06729171
Iteration 16 Testing accuracy: 0.06751486
Iteration 17 Training loss: 0.06729656
Iteration 17 Testing accuracy: 0.067514986
Iteration 18 Training loss: 0.06729783
Iteration 18 Testing accuracy: 0.067519225
Iteration 19 Training loss: 0.06731378
Iteration 19 Testing accuracy: 0.06749339
Iteration 20 Training loss: 0.06728936
Iteration 20 Testing accuracy: 0.06749457
Iteration 1 Training loss: 0.067328826
Iteration 1 Testing accuracy: 0.067519866
Iteration 2 Training loss: 0.06732252
Iteration 2 Testing accuracy: 0.06755288
Iteration 3 Training loss: 0.067326136
Iteration 3 Testing accuracy: 0.06753904
Iteration 4 Training loss: 0.06733009
Iteration 4 Testing accuracy: 0.067542344
Iteration 5 Training loss: 0.067317754
Iteration 5 Testing accuracy: 0.06761752

```

Iteration 6 Training loss: 0.067328826
Iteration 6 Testing accuracy: 0.06756618
Iteration 7 Training loss: 0.06733423
Iteration 7 Testing accuracy: 0.06750584
Iteration 8 Training loss: 0.0673295
Iteration 8 Testing accuracy: 0.06752731
Iteration 9 Training loss: 0.06732576
Iteration 9 Testing accuracy: 0.067482814
Iteration 10 Training loss: 0.0673252
Iteration 10 Testing accuracy: 0.06751189
Iteration 11 Training loss: 0.067331366
Iteration 11 Testing accuracy: 0.067565456
Iteration 12 Training loss: 0.06732364
Iteration 12 Testing accuracy: 0.067586705
Iteration 13 Training loss: 0.06731536
Iteration 13 Testing accuracy: 0.06751647
Iteration 14 Training loss: 0.06732778
Iteration 14 Testing accuracy: 0.06752709
Iteration 15 Training loss: 0.06731808
Iteration 15 Testing accuracy: 0.06751877
Iteration 16 Training loss: 0.06732333
Iteration 16 Testing accuracy: 0.067532495
Iteration 17 Training loss: 0.06733482
Iteration 17 Testing accuracy: 0.067536056
Iteration 18 Training loss: 0.067322455
Iteration 18 Testing accuracy: 0.067579746
Iteration 19 Training loss: 0.06733139
Iteration 19 Testing accuracy: 0.067522086
Iteration 20 Training loss: 0.06733193
Iteration 20 Testing accuracy: 0.06752695
Iteration 1 Training loss: 0.06734495
Iteration 1 Testing accuracy: 0.06755341
Iteration 2 Training loss: 0.06730814
Iteration 2 Testing accuracy: 0.06748812
Iteration 3 Training loss: 0.06729238
Iteration 3 Testing accuracy: 0.06753419
Iteration 4 Training loss: 0.067290954
Iteration 4 Testing accuracy: 0.06752048
Iteration 5 Training loss: 0.067294516
Iteration 5 Testing accuracy: 0.06750607
Iteration 6 Training loss: 0.06729066
Iteration 6 Testing accuracy: 0.06748782
Iteration 7 Training loss: 0.0672925
Iteration 7 Testing accuracy: 0.06750448
Iteration 8 Training loss: 0.06728612
Iteration 8 Testing accuracy: 0.06748574
Iteration 9 Training loss: 0.06729633
Iteration 9 Testing accuracy: 0.06750047
Iteration 10 Training loss: 0.06729062
Iteration 10 Testing accuracy: 0.06749285
Iteration 11 Training loss: 0.06728753
Iteration 11 Testing accuracy: 0.06753019
Iteration 12 Training loss: 0.067288846
Iteration 12 Testing accuracy: 0.06747038
Iteration 13 Training loss: 0.067295656
Iteration 13 Testing accuracy: 0.06753011
Iteration 14 Training loss: 0.067294285
Iteration 14 Testing accuracy: 0.06756148
Iteration 15 Training loss: 0.06729437
Iteration 15 Testing accuracy: 0.067515925
Iteration 16 Training loss: 0.067297556
Iteration 16 Testing accuracy: 0.067471325
Iteration 17 Training loss: 0.067295074
Iteration 17 Testing accuracy: 0.067519546
Iteration 18 Training loss: 0.06728848
Iteration 18 Testing accuracy: 0.067505635
Iteration 19 Training loss: 0.06729214
Iteration 19 Testing accuracy: 0.06747524

Iteration 19 Training loss: 0.06747324
Iteration 20 Training loss: 0.06729325
Iteration 20 Testing accuracy: 0.06756085
Iteration 1 Training loss: 0.06734566
Iteration 1 Testing accuracy: 0.06752448
Iteration 2 Training loss: 0.06730281
Iteration 2 Testing accuracy: 0.0675201
Iteration 3 Training loss: 0.06729117
Iteration 3 Testing accuracy: 0.067535415
Iteration 4 Training loss: 0.06729293
Iteration 4 Testing accuracy: 0.06754185
Iteration 5 Training loss: 0.06728497
Iteration 5 Testing accuracy: 0.067493916
Iteration 6 Training loss: 0.067287646
Iteration 6 Testing accuracy: 0.067500874
Iteration 7 Training loss: 0.0672934
Iteration 7 Testing accuracy: 0.067538604
Iteration 8 Training loss: 0.067289405
Iteration 8 Testing accuracy: 0.06750549
Iteration 9 Training loss: 0.063286714
Iteration 9 Testing accuracy: 0.06332843
Iteration 10 Training loss: 0.06301577
Iteration 10 Testing accuracy: 0.062960215
Iteration 11 Training loss: 0.06252686
Iteration 11 Testing accuracy: 0.06245617
Iteration 12 Training loss: 0.062063098
Iteration 12 Testing accuracy: 0.06174951
Iteration 13 Training loss: 0.061746832
Iteration 13 Testing accuracy: 0.061205108
Iteration 14 Training loss: 0.06075599
Iteration 14 Testing accuracy: 0.060589384
Iteration 15 Training loss: 0.06060308
Iteration 15 Testing accuracy: 0.060985528
Iteration 16 Training loss: 0.060573857
Iteration 16 Testing accuracy: 0.06035021
Iteration 17 Training loss: 0.060938347
Iteration 17 Testing accuracy: 0.06028076
Iteration 18 Training loss: 0.060494628
Iteration 18 Testing accuracy: 0.061087657
Iteration 19 Training loss: 0.060545925
Iteration 19 Testing accuracy: 0.06030897
Iteration 20 Training loss: 0.060915176
Iteration 20 Testing accuracy: 0.060470812
Iteration 1 Training loss: 0.067343146
Iteration 1 Testing accuracy: 0.0675473
Iteration 2 Training loss: 0.06730795
Iteration 2 Testing accuracy: 0.06753196
Iteration 3 Training loss: 0.06729515
Iteration 3 Testing accuracy: 0.06749577
Iteration 4 Training loss: 0.06729753
Iteration 4 Testing accuracy: 0.06749989
Iteration 5 Training loss: 0.06729477
Iteration 5 Testing accuracy: 0.0675302
Iteration 6 Training loss: 0.06729266
Iteration 6 Testing accuracy: 0.06749674
Iteration 7 Training loss: 0.06729627
Iteration 7 Testing accuracy: 0.0675633
Iteration 8 Training loss: 0.06729364
Iteration 8 Testing accuracy: 0.06749117
Iteration 9 Training loss: 0.0672957
Iteration 9 Testing accuracy: 0.067505255
Iteration 10 Training loss: 0.0672965
Iteration 10 Testing accuracy: 0.06749358
Iteration 11 Training loss: 0.06728908
Iteration 11 Testing accuracy: 0.06751762
Iteration 12 Training loss: 0.06730225
Iteration 12 Testing accuracy: 0.06750704
Iteration 13 Training loss: 0.06728837

Iteration 13 Testing accuracy: 0.067523584
Iteration 14 Training loss: 0.06729028
Iteration 14 Testing accuracy: 0.06749129
Iteration 15 Training loss: 0.067287885
Iteration 15 Testing accuracy: 0.067501545
Iteration 16 Training loss: 0.067292236
Iteration 16 Testing accuracy: 0.06751553
Iteration 17 Training loss: 0.06729443
Iteration 17 Testing accuracy: 0.067520335
Iteration 18 Training loss: 0.06729672
Iteration 18 Testing accuracy: 0.067551
Iteration 19 Training loss: 0.06728764
Iteration 19 Testing accuracy: 0.06749997
Iteration 20 Training loss: 0.06420985
Iteration 20 Testing accuracy: 0.06408358
Training network with 8 hidden units
Iteration 1 Training loss: 0.067333356
Iteration 1 Testing accuracy: 0.06753849
Iteration 2 Training loss: 0.0673217
Iteration 2 Testing accuracy: 0.0675232
Iteration 3 Training loss: 0.06731921
Iteration 3 Testing accuracy: 0.06755887
Iteration 4 Training loss: 0.06732351
Iteration 4 Testing accuracy: 0.067561895
Iteration 5 Training loss: 0.067326345
Iteration 5 Testing accuracy: 0.067555964
Iteration 6 Training loss: 0.063571185
Iteration 6 Testing accuracy: 0.06362841
Iteration 7 Training loss: 0.06301259
Iteration 7 Testing accuracy: 0.06288352
Iteration 8 Training loss: 0.06279261
Iteration 8 Testing accuracy: 0.06286403
Iteration 9 Training loss: 0.06246039
Iteration 9 Testing accuracy: 0.06250073
Iteration 10 Training loss: 0.06208218
Iteration 10 Testing accuracy: 0.062440746
Iteration 11 Training loss: 0.0613242
Iteration 11 Testing accuracy: 0.060697623
Iteration 12 Training loss: 0.061163846
Iteration 12 Testing accuracy: 0.06113561
Iteration 13 Training loss: 0.060790826
Iteration 13 Testing accuracy: 0.06210388
Iteration 14 Training loss: 0.060519118
Iteration 14 Testing accuracy: 0.060500927
Iteration 15 Training loss: 0.060727835
Iteration 15 Testing accuracy: 0.061790973
Iteration 16 Training loss: 0.060581613
Iteration 16 Testing accuracy: 0.060328025
Iteration 17 Training loss: 0.060482904
Iteration 17 Testing accuracy: 0.060213707
Iteration 18 Training loss: 0.06048043
Iteration 18 Testing accuracy: 0.060316186
Iteration 19 Training loss: 0.057247423
Iteration 19 Testing accuracy: 0.056788463
Iteration 20 Training loss: 0.05661458
Iteration 20 Testing accuracy: 0.056205362
Iteration 1 Training loss: 0.067345165
Iteration 1 Testing accuracy: 0.06759165
Iteration 2 Training loss: 0.06730297
Iteration 2 Testing accuracy: 0.06751484
Iteration 3 Training loss: 0.06729923
Iteration 3 Testing accuracy: 0.067519605
Iteration 4 Training loss: 0.06729381
Iteration 4 Testing accuracy: 0.06746783
Iteration 5 Training loss: 0.06729169
Iteration 5 Testing accuracy: 0.067544356
Iteration 6 Training loss: 0.067298785
Iteration 6 Testing accuracy: 0.06749688

Iteration 7 Training loss: 0.067301795
Iteration 7 Testing accuracy: 0.06748458
Iteration 8 Training loss: 0.06729122
Iteration 8 Testing accuracy: 0.06752749
Iteration 9 Training loss: 0.06729623
Iteration 9 Testing accuracy: 0.06749748
Iteration 10 Training loss: 0.06729642
Iteration 10 Testing accuracy: 0.0675509
Iteration 11 Training loss: 0.06729011
Iteration 11 Testing accuracy: 0.06748219
Iteration 12 Training loss: 0.06729353
Iteration 12 Testing accuracy: 0.06750507
Iteration 13 Training loss: 0.06729479
Iteration 13 Testing accuracy: 0.067511395
Iteration 14 Training loss: 0.067291856
Iteration 14 Testing accuracy: 0.0674812
Iteration 15 Training loss: 0.06729362
Iteration 15 Testing accuracy: 0.0675111
Iteration 16 Training loss: 0.063566096
Iteration 16 Testing accuracy: 0.06364816
Iteration 17 Training loss: 0.06303974
Iteration 17 Testing accuracy: 0.06306358
Iteration 18 Training loss: 0.06273249
Iteration 18 Testing accuracy: 0.06272967
Iteration 19 Training loss: 0.06206858
Iteration 19 Testing accuracy: 0.06171304
Iteration 20 Training loss: 0.061625294
Iteration 20 Testing accuracy: 0.061576497
Iteration 1 Training loss: 0.06323303
Iteration 1 Testing accuracy: 0.0631747
Iteration 2 Training loss: 0.0621353
Iteration 2 Testing accuracy: 0.06246511
Iteration 3 Training loss: 0.061502755
Iteration 3 Testing accuracy: 0.06150369
Iteration 4 Training loss: 0.060926504
Iteration 4 Testing accuracy: 0.060644932
Iteration 5 Training loss: 0.060734354
Iteration 5 Testing accuracy: 0.060612164
Iteration 6 Training loss: 0.060944516
Iteration 6 Testing accuracy: 0.06093464
Iteration 7 Training loss: 0.060625684
Iteration 7 Testing accuracy: 0.060474597
Iteration 8 Training loss: 0.06065652
Iteration 8 Testing accuracy: 0.06041349
Iteration 9 Training loss: 0.06066083
Iteration 9 Testing accuracy: 0.060412884
Iteration 10 Training loss: 0.060533214
Iteration 10 Testing accuracy: 0.06056236
Iteration 11 Training loss: 0.060557105
Iteration 11 Testing accuracy: 0.060227327
Iteration 12 Training loss: 0.06056874
Iteration 12 Testing accuracy: 0.060229264
Iteration 13 Training loss: 0.06042185
Iteration 13 Testing accuracy: 0.060448695
Iteration 14 Training loss: 0.060486976
Iteration 14 Testing accuracy: 0.060437005
Iteration 15 Training loss: 0.0604984
Iteration 15 Testing accuracy: 0.06059773
Iteration 16 Training loss: 0.061397467
Iteration 16 Testing accuracy: 0.060361784
Iteration 17 Training loss: 0.060456704
Iteration 17 Testing accuracy: 0.060213365
Iteration 18 Training loss: 0.060680006
Iteration 18 Testing accuracy: 0.06038786
Iteration 19 Training loss: 0.056872044
Iteration 19 Testing accuracy: 0.05630032
Iteration 20 Training loss: 0.056610815
Iteration 20 Testing accuracy: 0.056186758

Iteration 20 Training loss: 0.067318730
Iteration 20 Testing accuracy: 0.06752484
Iteration 1 Training loss: 0.06731734
Iteration 1 Testing accuracy: 0.06752484
Iteration 2 Training loss: 0.067314774
Iteration 2 Testing accuracy: 0.06749563
Iteration 3 Training loss: 0.06731837
Iteration 3 Testing accuracy: 0.067594275
Iteration 4 Training loss: 0.067308895
Iteration 4 Testing accuracy: 0.06756049
Iteration 5 Training loss: 0.06734114
Iteration 5 Testing accuracy: 0.06752007
Iteration 6 Training loss: 0.067325115
Iteration 6 Testing accuracy: 0.06752755
Iteration 7 Training loss: 0.067321286
Iteration 7 Testing accuracy: 0.067533255
Iteration 8 Training loss: 0.067331746
Iteration 8 Testing accuracy: 0.067518696
Iteration 9 Training loss: 0.06733465
Iteration 9 Testing accuracy: 0.06751923
Iteration 10 Training loss: 0.06731944
Iteration 10 Testing accuracy: 0.06749867
Iteration 11 Training loss: 0.06733694
Iteration 11 Testing accuracy: 0.067492336
Iteration 12 Training loss: 0.06733654
Iteration 12 Testing accuracy: 0.06760858
Iteration 13 Training loss: 0.0673233
Iteration 13 Testing accuracy: 0.067537375
Iteration 14 Training loss: 0.06733134
Iteration 14 Testing accuracy: 0.0675403
Iteration 15 Training loss: 0.06733266
Iteration 15 Testing accuracy: 0.06761314
Iteration 16 Training loss: 0.06733702
Iteration 16 Testing accuracy: 0.0674992
Iteration 17 Training loss: 0.06733063
Iteration 17 Testing accuracy: 0.06761346
Iteration 18 Training loss: 0.06732727
Iteration 18 Testing accuracy: 0.06755022
Iteration 19 Training loss: 0.06371062
Iteration 19 Testing accuracy: 0.063851535
Iteration 20 Training loss: 0.063166484
Iteration 20 Testing accuracy: 0.063202456
Iteration 1 Training loss: 0.06734356
Iteration 1 Testing accuracy: 0.06756784
Iteration 2 Training loss: 0.06730317
Iteration 2 Testing accuracy: 0.06750815
Iteration 3 Training loss: 0.06729933
Iteration 3 Testing accuracy: 0.067513935
Iteration 4 Training loss: 0.06729203
Iteration 4 Testing accuracy: 0.06751411
Iteration 5 Training loss: 0.067291364
Iteration 5 Testing accuracy: 0.06751539
Iteration 6 Training loss: 0.067292176
Iteration 6 Testing accuracy: 0.067541
Iteration 7 Training loss: 0.067288354
Iteration 7 Testing accuracy: 0.06748992
Iteration 8 Training loss: 0.06729492
Iteration 8 Testing accuracy: 0.067520045
Iteration 9 Training loss: 0.063336685
Iteration 9 Testing accuracy: 0.063333675
Iteration 10 Training loss: 0.0629977
Iteration 10 Testing accuracy: 0.06294507
Iteration 11 Training loss: 0.06285429
Iteration 11 Testing accuracy: 0.06295397
Iteration 12 Training loss: 0.06270592
Iteration 12 Testing accuracy: 0.062673755
Iteration 13 Training loss: 0.062247958
Iteration 13 Testing accuracy: 0.06243067
Iteration 14 Training loss: 0.061607536

Iteration 14 Testing accuracy: 0.06459503
Iteration 15 Training loss: 0.06089766
Iteration 15 Testing accuracy: 0.06115254
Iteration 16 Training loss: 0.06077536
Iteration 16 Testing accuracy: 0.060403258
Iteration 17 Training loss: 0.060928337
Iteration 17 Testing accuracy: 0.06078388
Iteration 18 Training loss: 0.060579028
Iteration 18 Testing accuracy: 0.060334366
Iteration 19 Training loss: 0.060146667
Iteration 19 Testing accuracy: 0.059553813
Iteration 20 Training loss: 0.056807168
Iteration 20 Testing accuracy: 0.056354605
Training network with 16 hidden units
Iteration 1 Training loss: 0.06156334
Iteration 1 Testing accuracy: 0.062089037
Iteration 2 Training loss: 0.061222468
Iteration 2 Testing accuracy: 0.060853362
Iteration 3 Training loss: 0.058162972
Iteration 3 Testing accuracy: 0.05746571
Iteration 4 Training loss: 0.05704112
Iteration 4 Testing accuracy: 0.058467466
Iteration 5 Training loss: 0.057094064
Iteration 5 Testing accuracy: 0.05671941
Iteration 6 Training loss: 0.05694429
Iteration 6 Testing accuracy: 0.056613296
Iteration 7 Training loss: 0.05705202
Iteration 7 Testing accuracy: 0.056433357
Iteration 8 Training loss: 0.05423476
Iteration 8 Testing accuracy: 0.05377227
Iteration 9 Training loss: 0.05384714
Iteration 9 Testing accuracy: 0.054177128
Iteration 10 Training loss: 0.051272213
Iteration 10 Testing accuracy: 0.05055636
Iteration 11 Training loss: 0.051390853
Iteration 11 Testing accuracy: 0.050755214
Iteration 12 Training loss: 0.048448544
Iteration 12 Testing accuracy: 0.048348043
Iteration 13 Training loss: 0.04909923
Iteration 13 Testing accuracy: 0.0481077
Iteration 14 Training loss: 0.04808019
Iteration 14 Testing accuracy: 0.04790424
Iteration 15 Training loss: 0.048002344
Iteration 15 Testing accuracy: 0.048006568
Iteration 16 Training loss: 0.048097912
Iteration 16 Testing accuracy: 0.048015416
Iteration 17 Training loss: 0.046370227
Iteration 17 Testing accuracy: 0.046149384
Iteration 18 Training loss: 0.04558336
Iteration 18 Testing accuracy: 0.0450053
Iteration 19 Training loss: 0.044936076
Iteration 19 Testing accuracy: 0.044495128
Iteration 20 Training loss: 0.043731313
Iteration 20 Testing accuracy: 0.04366982
Iteration 1 Training loss: 0.06741042
Iteration 1 Testing accuracy: 0.06765175
Iteration 2 Training loss: 0.06740712
Iteration 2 Testing accuracy: 0.06767754
Iteration 3 Training loss: 0.067419715
Iteration 3 Testing accuracy: 0.06758641
Iteration 4 Training loss: 0.06741722
Iteration 4 Testing accuracy: 0.06753671
Iteration 5 Training loss: 0.06742906
Iteration 5 Testing accuracy: 0.06771899
Iteration 6 Training loss: 0.06327491
Iteration 6 Testing accuracy: 0.063225575
Iteration 7 Training loss: 0.06294797
Iteration 7 Testing accuracy: 0.06303717

Iteration 8 Training loss: 0.062635355
Iteration 8 Testing accuracy: 0.06253789
Iteration 9 Training loss: 0.058484074
Iteration 9 Testing accuracy: 0.058161445
Iteration 10 Training loss: 0.058266934
Iteration 10 Testing accuracy: 0.057983994
Iteration 11 Training loss: 0.0582812
Iteration 11 Testing accuracy: 0.05761708
Iteration 12 Training loss: 0.058031578
Iteration 12 Testing accuracy: 0.05736366
Iteration 13 Training loss: 0.05476934
Iteration 13 Testing accuracy: 0.054343782
Iteration 14 Training loss: 0.054441523
Iteration 14 Testing accuracy: 0.054182768
Iteration 15 Training loss: 0.054626152
Iteration 15 Testing accuracy: 0.054297604
Iteration 16 Training loss: 0.05441909
Iteration 16 Testing accuracy: 0.05409091
Iteration 17 Training loss: 0.054751303
Iteration 17 Testing accuracy: 0.054646857
Iteration 18 Training loss: 0.05433011
Iteration 18 Testing accuracy: 0.05405621
Iteration 19 Training loss: 0.051428076
Iteration 19 Testing accuracy: 0.051023196
Iteration 20 Training loss: 0.050931472
Iteration 20 Testing accuracy: 0.052039403
Iteration 1 Training loss: 0.067369975
Iteration 1 Testing accuracy: 0.06755693
Iteration 2 Training loss: 0.06736685
Iteration 2 Testing accuracy: 0.06761141
Iteration 3 Training loss: 0.067402855
Iteration 3 Testing accuracy: 0.06780254
Iteration 4 Training loss: 0.06739677
Iteration 4 Testing accuracy: 0.06752511
Iteration 5 Training loss: 0.067381404
Iteration 5 Testing accuracy: 0.06753959
Iteration 6 Training loss: 0.06737471
Iteration 6 Testing accuracy: 0.06767831
Iteration 7 Training loss: 0.06739173
Iteration 7 Testing accuracy: 0.06755194
Iteration 8 Training loss: 0.067391515
Iteration 8 Testing accuracy: 0.06759019
Iteration 9 Training loss: 0.06736903
Iteration 9 Testing accuracy: 0.06764417
Iteration 10 Training loss: 0.06396155
Iteration 10 Testing accuracy: 0.063872024
Iteration 11 Training loss: 0.06317633
Iteration 11 Testing accuracy: 0.06316571
Iteration 12 Training loss: 0.0627663
Iteration 12 Testing accuracy: 0.06271093
Iteration 13 Training loss: 0.062231004
Iteration 13 Testing accuracy: 0.06194987
Iteration 14 Training loss: 0.061402757
Iteration 14 Testing accuracy: 0.06103931
Iteration 15 Training loss: 0.061140306
Iteration 15 Testing accuracy: 0.060892306
Iteration 16 Training loss: 0.06106018
Iteration 16 Testing accuracy: 0.060970854
Iteration 17 Training loss: 0.060852226
Iteration 17 Testing accuracy: 0.06182889
Iteration 18 Training loss: 0.0572888
Iteration 18 Testing accuracy: 0.05661968
Iteration 19 Training loss: 0.053394616
Iteration 19 Testing accuracy: 0.05290758
Iteration 20 Training loss: 0.05117387
Iteration 20 Testing accuracy: 0.05067574
Iteration 1 Training loss: 0.06734677
Iteration 1 Testing accuracy: 0.067575075

Iteration 1 Training loss: 0.067373073
Iteration 1 Testing accuracy: 0.06752755
Iteration 2 Training loss: 0.06735547
Iteration 2 Testing accuracy: 0.06752755
Iteration 3 Training loss: 0.06734662
Iteration 3 Testing accuracy: 0.06756295
Iteration 4 Training loss: 0.06734453
Iteration 4 Testing accuracy: 0.06752469
Iteration 5 Training loss: 0.067367576
Iteration 5 Testing accuracy: 0.06764809
Iteration 6 Training loss: 0.0673627
Iteration 6 Testing accuracy: 0.0675724
Iteration 7 Training loss: 0.067353494
Iteration 7 Testing accuracy: 0.06755148
Iteration 8 Training loss: 0.067375615
Iteration 8 Testing accuracy: 0.067552574
Iteration 9 Training loss: 0.06736367
Iteration 9 Testing accuracy: 0.06755796
Iteration 10 Training loss: 0.067348845
Iteration 10 Testing accuracy: 0.067571
Iteration 11 Training loss: 0.06734714
Iteration 11 Testing accuracy: 0.06753657
Iteration 12 Training loss: 0.06737444
Iteration 12 Testing accuracy: 0.06758231
Iteration 13 Training loss: 0.067366414
Iteration 13 Testing accuracy: 0.06759499
Iteration 14 Training loss: 0.067358226
Iteration 14 Testing accuracy: 0.06754928
Iteration 15 Training loss: 0.06360961
Iteration 15 Testing accuracy: 0.063769296
Iteration 16 Training loss: 0.06313903
Iteration 16 Testing accuracy: 0.06312587
Iteration 17 Training loss: 0.058840115
Iteration 17 Testing accuracy: 0.058542173
Iteration 18 Training loss: 0.057350256
Iteration 18 Testing accuracy: 0.056911986
Iteration 19 Training loss: 0.055520352
Iteration 19 Testing accuracy: 0.054575108
Iteration 20 Training loss: 0.0518005
Iteration 20 Testing accuracy: 0.052117467
Iteration 1 Training loss: 0.06735359
Iteration 1 Testing accuracy: 0.06755273
Iteration 2 Training loss: 0.06734955
Iteration 2 Testing accuracy: 0.06757333
Iteration 3 Training loss: 0.06736127
Iteration 3 Testing accuracy: 0.067621574
Iteration 4 Training loss: 0.067352295
Iteration 4 Testing accuracy: 0.06755039
Iteration 5 Training loss: 0.06735305
Iteration 5 Testing accuracy: 0.0676027
Iteration 6 Training loss: 0.067358084
Iteration 6 Testing accuracy: 0.06760685
Iteration 7 Training loss: 0.06737722
Iteration 7 Testing accuracy: 0.06752534
Iteration 8 Training loss: 0.067357324
Iteration 8 Testing accuracy: 0.06763012
Iteration 9 Training loss: 0.067356154
Iteration 9 Testing accuracy: 0.06753437
Iteration 10 Training loss: 0.06734265
Iteration 10 Testing accuracy: 0.067531414
Iteration 11 Training loss: 0.06735628
Iteration 11 Testing accuracy: 0.06759181
Iteration 12 Training loss: 0.067348175
Iteration 12 Testing accuracy: 0.06766188
Iteration 13 Training loss: 0.06736411
Iteration 13 Testing accuracy: 0.06747894
Iteration 14 Training loss: 0.067369536
Iteration 14 Testing accuracy: 0.06756159
Iteration 15 Training loss: 0.06051659

Iteration 15 Testing accuracy: 0.059508722
Iteration 16 Training loss: 0.059043597
Iteration 16 Testing accuracy: 0.058710646
Iteration 17 Training loss: 0.05873363
Iteration 17 Testing accuracy: 0.058384173
Iteration 18 Training loss: 0.058569048
Iteration 18 Testing accuracy: 0.05816748
Iteration 19 Training loss: 0.05606909
Iteration 19 Testing accuracy: 0.05469674
Iteration 20 Training loss: 0.055983976
Iteration 20 Testing accuracy: 0.055278152
Training network with 32 hidden units
Iteration 1 Training loss: 0.053503014
Iteration 1 Testing accuracy: 0.053440493
Iteration 2 Training loss: 0.05310807
Iteration 2 Testing accuracy: 0.051828593
Iteration 3 Training loss: 0.052202642
Iteration 3 Testing accuracy: 0.051703524
Iteration 4 Training loss: 0.051903017
Iteration 4 Testing accuracy: 0.05289964
Iteration 5 Training loss: 0.05133436
Iteration 5 Testing accuracy: 0.051042933
Iteration 6 Training loss: 0.05139137
Iteration 6 Testing accuracy: 0.051154815
Iteration 7 Training loss: 0.050816763
Iteration 7 Testing accuracy: 0.05005203
Iteration 8 Training loss: 0.050761774
Iteration 8 Testing accuracy: 0.05001499
Iteration 9 Training loss: 0.050788864
Iteration 9 Testing accuracy: 0.050451316
Iteration 10 Training loss: 0.047597602
Iteration 10 Testing accuracy: 0.047303155
Iteration 11 Training loss: 0.04751496
Iteration 11 Testing accuracy: 0.048320726
Iteration 12 Training loss: 0.045870867
Iteration 12 Testing accuracy: 0.0466576
Iteration 13 Training loss: 0.043712404
Iteration 13 Testing accuracy: 0.043794997
Iteration 14 Training loss: 0.043476034
Iteration 14 Testing accuracy: 0.043426618
Iteration 15 Training loss: 0.043206766
Iteration 15 Testing accuracy: 0.04280981
Iteration 16 Training loss: 0.04222207
Iteration 16 Testing accuracy: 0.043785296
Iteration 17 Training loss: 0.04166399
Iteration 17 Testing accuracy: 0.041088585
Iteration 18 Training loss: 0.041491497
Iteration 18 Testing accuracy: 0.0412197
Iteration 19 Training loss: 0.041333225
Iteration 19 Testing accuracy: 0.040871568
Iteration 20 Training loss: 0.03905461
Iteration 20 Testing accuracy: 0.03918391
Iteration 1 Training loss: 0.05381201
Iteration 1 Testing accuracy: 0.053593833
Iteration 2 Training loss: 0.05303975
Iteration 2 Testing accuracy: 0.051832624
Iteration 3 Training loss: 0.05262721
Iteration 3 Testing accuracy: 0.051822472
Iteration 4 Training loss: 0.05177517
Iteration 4 Testing accuracy: 0.05089588
Iteration 5 Training loss: 0.05171556
Iteration 5 Testing accuracy: 0.05175654
Iteration 6 Training loss: 0.051088292
Iteration 6 Testing accuracy: 0.050970614
Iteration 7 Training loss: 0.05163477
Iteration 7 Testing accuracy: 0.052443992
Iteration 8 Training loss: 0.051748376
Iteration 8 Testing accuracy: 0.050470643

Iteration 9 Training loss: 0.050943922
Iteration 9 Testing accuracy: 0.052644722
Iteration 10 Training loss: 0.0511857
Iteration 10 Testing accuracy: 0.05064811
Iteration 11 Training loss: 0.050954297
Iteration 11 Testing accuracy: 0.051193856
Iteration 12 Training loss: 0.050801802
Iteration 12 Testing accuracy: 0.051602643
Iteration 13 Training loss: 0.047634244
Iteration 13 Testing accuracy: 0.04656156
Iteration 14 Training loss: 0.045133427
Iteration 14 Testing accuracy: 0.045543447
Iteration 15 Training loss: 0.04247002
Iteration 15 Testing accuracy: 0.04146769
Iteration 16 Training loss: 0.04070824
Iteration 16 Testing accuracy: 0.040633567
Iteration 17 Training loss: 0.040649045
Iteration 17 Testing accuracy: 0.04002601
Iteration 18 Training loss: 0.04048837
Iteration 18 Testing accuracy: 0.039738648
Iteration 19 Training loss: 0.040006693
Iteration 19 Testing accuracy: 0.03989665
Iteration 20 Training loss: 0.0396811
Iteration 20 Testing accuracy: 0.03975941
Iteration 1 Training loss: 0.057591494
Iteration 1 Testing accuracy: 0.056497544
Iteration 2 Training loss: 0.056700956
Iteration 2 Testing accuracy: 0.05646692
Iteration 3 Training loss: 0.0576331
Iteration 3 Testing accuracy: 0.056323152
Iteration 4 Training loss: 0.056364145
Iteration 4 Testing accuracy: 0.057085507
Iteration 5 Training loss: 0.056381676
Iteration 5 Testing accuracy: 0.056138553
Iteration 6 Training loss: 0.055813085
Iteration 6 Testing accuracy: 0.05504328
Iteration 7 Training loss: 0.05273091
Iteration 7 Testing accuracy: 0.052204926
Iteration 8 Training loss: 0.052452385
Iteration 8 Testing accuracy: 0.05265242
Iteration 9 Training loss: 0.05252771
Iteration 9 Testing accuracy: 0.052425243
Iteration 10 Training loss: 0.052356873
Iteration 10 Testing accuracy: 0.052987944
Iteration 11 Training loss: 0.0525176
Iteration 11 Testing accuracy: 0.053049643
Iteration 12 Training loss: 0.052348148
Iteration 12 Testing accuracy: 0.052117985
Iteration 13 Training loss: 0.050031055
Iteration 13 Testing accuracy: 0.049686436
Iteration 14 Training loss: 0.0493743
Iteration 14 Testing accuracy: 0.049130727
Iteration 15 Training loss: 0.047257084
Iteration 15 Testing accuracy: 0.047006764
Iteration 16 Training loss: 0.045554888
Iteration 16 Testing accuracy: 0.04500237
Iteration 17 Training loss: 0.04448344
Iteration 17 Testing accuracy: 0.04382928
Iteration 18 Training loss: 0.04381116
Iteration 18 Testing accuracy: 0.04326145
Iteration 19 Training loss: 0.042853355
Iteration 19 Testing accuracy: 0.042390324
Iteration 20 Training loss: 0.04289157
Iteration 20 Testing accuracy: 0.04237517
Iteration 1 Training loss: 0.05383175
Iteration 1 Testing accuracy: 0.05278061
Iteration 2 Training loss: 0.05308755
Iteration 2 Testing accuracy: 0.052164745

Iteration 2 Training loss: 0.052104743
Iteration 2 Testing accuracy: 0.052165247
Iteration 3 Training loss: 0.052110676
Iteration 3 Testing accuracy: 0.052165247
Iteration 4 Training loss: 0.051967856
Iteration 4 Testing accuracy: 0.05172609
Iteration 5 Training loss: 0.05130023
Iteration 5 Testing accuracy: 0.0542395
Iteration 6 Training loss: 0.051091183
Iteration 6 Testing accuracy: 0.05145549
Iteration 7 Training loss: 0.05100328
Iteration 7 Testing accuracy: 0.05149133
Iteration 8 Training loss: 0.050867178
Iteration 8 Testing accuracy: 0.05106563
Iteration 9 Training loss: 0.051358316
Iteration 9 Testing accuracy: 0.050191343
Iteration 10 Training loss: 0.051546954
Iteration 10 Testing accuracy: 0.05082649
Iteration 11 Training loss: 0.051166587
Iteration 11 Testing accuracy: 0.05043011
Iteration 12 Training loss: 0.050820157
Iteration 12 Testing accuracy: 0.050144274
Iteration 13 Training loss: 0.05099843
Iteration 13 Testing accuracy: 0.05067707
Iteration 14 Training loss: 0.0506502
Iteration 14 Testing accuracy: 0.050111897
Iteration 15 Training loss: 0.050832625
Iteration 15 Testing accuracy: 0.050256472
Iteration 16 Training loss: 0.050467864
Iteration 16 Testing accuracy: 0.05044943
Iteration 17 Training loss: 0.049400404
Iteration 17 Testing accuracy: 0.049028676
Iteration 18 Training loss: 0.046335477
Iteration 18 Testing accuracy: 0.046677757
Iteration 19 Training loss: 0.045891047
Iteration 19 Testing accuracy: 0.04511564
Iteration 20 Training loss: 0.045405243
Iteration 20 Testing accuracy: 0.044838153
Iteration 1 Training loss: 0.048068795
Iteration 1 Testing accuracy: 0.048953194
Iteration 2 Training loss: 0.046685673
Iteration 2 Testing accuracy: 0.044972908
Iteration 3 Training loss: 0.04507696
Iteration 3 Testing accuracy: 0.04382906
Iteration 4 Training loss: 0.046040315
Iteration 4 Testing accuracy: 0.04557598
Iteration 5 Training loss: 0.04466963
Iteration 5 Testing accuracy: 0.044786472
Iteration 6 Training loss: 0.044390608
Iteration 6 Testing accuracy: 0.045263484
Iteration 7 Training loss: 0.044503585
Iteration 7 Testing accuracy: 0.045137383
Iteration 8 Training loss: 0.04404737
Iteration 8 Testing accuracy: 0.042861294
Iteration 9 Training loss: 0.044219848
Iteration 9 Testing accuracy: 0.04263419
Iteration 10 Training loss: 0.04387045
Iteration 10 Testing accuracy: 0.04681478
Iteration 11 Training loss: 0.043970894
Iteration 11 Testing accuracy: 0.042746916
Iteration 12 Training loss: 0.04358937
Iteration 12 Testing accuracy: 0.04227175
Iteration 13 Training loss: 0.043695357
Iteration 13 Testing accuracy: 0.04276757
Iteration 14 Training loss: 0.043842353
Iteration 14 Testing accuracy: 0.042649426
Iteration 15 Training loss: 0.043491587
Iteration 15 Testing accuracy: 0.042701814
Iteration 16 Training loss: 0.043477103

Iteration 16 Testing accuracy: 0.044060476
Iteration 17 Training loss: 0.04346653
Iteration 17 Testing accuracy: 0.042974256
Iteration 18 Training loss: 0.043393396
Iteration 18 Testing accuracy: 0.04379695
Iteration 19 Training loss: 0.042265795
Iteration 19 Testing accuracy: 0.04138134
Iteration 20 Training loss: 0.04161596
Iteration 20 Testing accuracy: 0.041020934
Training network with 64 hidden units
Iteration 1 Training loss: 0.046001896
Iteration 1 Testing accuracy: 0.045804597
Iteration 2 Training loss: 0.043943338
Iteration 2 Testing accuracy: 0.043543976
Iteration 3 Training loss: 0.042761907
Iteration 3 Testing accuracy: 0.042911608
Iteration 4 Training loss: 0.04196603
Iteration 4 Testing accuracy: 0.041140296
Iteration 5 Training loss: 0.041177906
Iteration 5 Testing accuracy: 0.039348
Iteration 6 Training loss: 0.04018838
Iteration 6 Testing accuracy: 0.039189856
Iteration 7 Training loss: 0.040515658
Iteration 7 Testing accuracy: 0.04001563
Iteration 8 Training loss: 0.03995473
Iteration 8 Testing accuracy: 0.038641326
Iteration 9 Training loss: 0.039587453
Iteration 9 Testing accuracy: 0.039124966
Iteration 10 Training loss: 0.03927897
Iteration 10 Testing accuracy: 0.038955316
Iteration 11 Training loss: 0.039287373
Iteration 11 Testing accuracy: 0.038839944
Iteration 12 Training loss: 0.03878406
Iteration 12 Testing accuracy: 0.037713744
Iteration 13 Training loss: 0.037501063
Iteration 13 Testing accuracy: 0.037255704
Iteration 14 Training loss: 0.03723382
Iteration 14 Testing accuracy: 0.03580737
Iteration 15 Training loss: 0.037385292
Iteration 15 Testing accuracy: 0.03684018
Iteration 16 Training loss: 0.03712168
Iteration 16 Testing accuracy: 0.03683522
Iteration 17 Training loss: 0.036934827
Iteration 17 Testing accuracy: 0.037398633
Iteration 18 Training loss: 0.03719776
Iteration 18 Testing accuracy: 0.036842845
Iteration 19 Training loss: 0.036945306
Iteration 19 Testing accuracy: 0.03605939
Iteration 20 Training loss: 0.037118185
Iteration 20 Testing accuracy: 0.037470277
Iteration 1 Training loss: 0.0556918
Iteration 1 Testing accuracy: 0.05460415
Iteration 2 Training loss: 0.055229273
Iteration 2 Testing accuracy: 0.055282306
Iteration 3 Training loss: 0.05474662
Iteration 3 Testing accuracy: 0.05501572
Iteration 4 Training loss: 0.05436084
Iteration 4 Testing accuracy: 0.054682244
Iteration 5 Training loss: 0.054054994
Iteration 5 Testing accuracy: 0.05356912
Iteration 6 Training loss: 0.053906657
Iteration 6 Testing accuracy: 0.053155296
Iteration 7 Training loss: 0.05352291
Iteration 7 Testing accuracy: 0.05510386
Iteration 8 Training loss: 0.054051403
Iteration 8 Testing accuracy: 0.05346139
Iteration 9 Training loss: 0.053866968
Iteration 9 Testing accuracy: 0.05333223

Iteration 10 Training loss: 0.053480104
Iteration 10 Testing accuracy: 0.05265098
Iteration 11 Training loss: 0.0537465
Iteration 11 Testing accuracy: 0.054219227
Iteration 12 Training loss: 0.054093312
Iteration 12 Testing accuracy: 0.05260433
Iteration 13 Training loss: 0.05336792
Iteration 13 Testing accuracy: 0.0558128
Iteration 14 Training loss: 0.053378817
Iteration 14 Testing accuracy: 0.05349074
Iteration 15 Training loss: 0.049061906
Iteration 15 Testing accuracy: 0.049698137
Iteration 16 Training loss: 0.049245786
Iteration 16 Testing accuracy: 0.04882404
Iteration 17 Training loss: 0.04732001
Iteration 17 Testing accuracy: 0.04659105
Iteration 18 Training loss: 0.046837073
Iteration 18 Testing accuracy: 0.046639245
Iteration 19 Training loss: 0.04667057
Iteration 19 Testing accuracy: 0.045538276
Iteration 20 Training loss: 0.04546219
Iteration 20 Testing accuracy: 0.044795785
Iteration 1 Training loss: 0.03873114
Iteration 1 Testing accuracy: 0.037408777
Iteration 2 Training loss: 0.037046142
Iteration 2 Testing accuracy: 0.03643907
Iteration 3 Training loss: 0.036162388
Iteration 3 Testing accuracy: 0.035041984
Iteration 4 Training loss: 0.034960482
Iteration 4 Testing accuracy: 0.033816166
Iteration 5 Training loss: 0.03258986
Iteration 5 Testing accuracy: 0.031372055
Iteration 6 Training loss: 0.03252836
Iteration 6 Testing accuracy: 0.032037232
Iteration 7 Training loss: 0.032314587
Iteration 7 Testing accuracy: 0.03152505
Iteration 8 Training loss: 0.0317633
Iteration 8 Testing accuracy: 0.031203162
Iteration 9 Training loss: 0.031638462
Iteration 9 Testing accuracy: 0.030373145
Iteration 10 Training loss: 0.031164356
Iteration 10 Testing accuracy: 0.030588344
Iteration 11 Training loss: 0.029385379
Iteration 11 Testing accuracy: 0.028215727
Iteration 12 Training loss: 0.029458275
Iteration 12 Testing accuracy: 0.02843464
Iteration 13 Training loss: 0.029289352
Iteration 13 Testing accuracy: 0.027759045
Iteration 14 Training loss: 0.028880643
Iteration 14 Testing accuracy: 0.028312225
Iteration 15 Training loss: 0.02871791
Iteration 15 Testing accuracy: 0.028161097
Iteration 16 Training loss: 0.028902097
Iteration 16 Testing accuracy: 0.028756816
Iteration 17 Training loss: 0.02780465
Iteration 17 Testing accuracy: 0.0275531
Iteration 18 Training loss: 0.027743822
Iteration 18 Testing accuracy: 0.027503774
Iteration 19 Training loss: 0.027425366
Iteration 19 Testing accuracy: 0.026356865
Iteration 20 Training loss: 0.027383812
Iteration 20 Testing accuracy: 0.026294809
Iteration 1 Training loss: 0.047102783
Iteration 1 Testing accuracy: 0.0473674
Iteration 2 Training loss: 0.045889296
Iteration 2 Testing accuracy: 0.04399835
Iteration 3 Training loss: 0.043880586
Iteration 3 Testing accuracy: 0.04447717

Iteration 3 Training loss: 0.0447717
Iteration 4 Training loss: 0.043892678
Iteration 4 Testing accuracy: 0.043076996
Iteration 5 Training loss: 0.043008197
Iteration 5 Testing accuracy: 0.042778123
Iteration 6 Training loss: 0.04131571
Iteration 6 Testing accuracy: 0.04512112
Iteration 7 Training loss: 0.041634466
Iteration 7 Testing accuracy: 0.040304773
Iteration 8 Training loss: 0.041228645
Iteration 8 Testing accuracy: 0.040385954
Iteration 9 Training loss: 0.04131908
Iteration 9 Testing accuracy: 0.04029237
Iteration 10 Training loss: 0.041057583
Iteration 10 Testing accuracy: 0.04050227
Iteration 11 Training loss: 0.04084029
Iteration 11 Testing accuracy: 0.040242996
Iteration 12 Training loss: 0.04082714
Iteration 12 Testing accuracy: 0.042936645
Iteration 13 Training loss: 0.040622994
Iteration 13 Testing accuracy: 0.040513184
Iteration 14 Training loss: 0.040644743
Iteration 14 Testing accuracy: 0.040381737
Iteration 15 Training loss: 0.040445916
Iteration 15 Testing accuracy: 0.03904415
Iteration 16 Training loss: 0.041011646
Iteration 16 Testing accuracy: 0.040550373
Iteration 17 Training loss: 0.041110877
Iteration 17 Testing accuracy: 0.040073115
Iteration 18 Training loss: 0.04039168
Iteration 18 Testing accuracy: 0.039652344
Iteration 19 Training loss: 0.0405141
Iteration 19 Testing accuracy: 0.040394656
Iteration 20 Training loss: 0.04038017
Iteration 20 Testing accuracy: 0.039713867
Iteration 1 Training loss: 0.043630764
Iteration 1 Testing accuracy: 0.044258315
Iteration 2 Training loss: 0.041051064
Iteration 2 Testing accuracy: 0.039305072
Iteration 3 Training loss: 0.039162885
Iteration 3 Testing accuracy: 0.043731816
Iteration 4 Training loss: 0.038890995
Iteration 4 Testing accuracy: 0.039847743
Iteration 5 Training loss: 0.038617603
Iteration 5 Testing accuracy: 0.037906453
Iteration 6 Training loss: 0.03792906
Iteration 6 Testing accuracy: 0.036771867
Iteration 7 Training loss: 0.037652016
Iteration 7 Testing accuracy: 0.037009083
Iteration 8 Training loss: 0.03763844
Iteration 8 Testing accuracy: 0.0365265
Iteration 9 Training loss: 0.037005108
Iteration 9 Testing accuracy: 0.036934018
Iteration 10 Training loss: 0.037511695
Iteration 10 Testing accuracy: 0.03602987
Iteration 11 Training loss: 0.037112612
Iteration 11 Testing accuracy: 0.036276422
Iteration 12 Training loss: 0.03662662
Iteration 12 Testing accuracy: 0.035661474
Iteration 13 Training loss: 0.037034646
Iteration 13 Testing accuracy: 0.036721677
Iteration 14 Training loss: 0.03690838
Iteration 14 Testing accuracy: 0.036130022
Iteration 15 Training loss: 0.037050452
Iteration 15 Testing accuracy: 0.03645165
Iteration 16 Training loss: 0.037257746
Iteration 16 Testing accuracy: 0.035802357
Iteration 17 Training loss: 0.03670435

Iteration 17 Testing accuracy: 0.036568604
Iteration 18 Training loss: 0.036651183
Iteration 18 Testing accuracy: 0.03583225
Iteration 19 Training loss: 0.036629375
Iteration 19 Testing accuracy: 0.036863524
Iteration 20 Training loss: 0.036460735
Iteration 20 Testing accuracy: 0.03534279
Training network with 128 hidden units
Iteration 1 Training loss: 0.069446094
Iteration 1 Testing accuracy: 0.07143211
Iteration 2 Training loss: 0.06991902
Iteration 2 Testing accuracy: 0.06959366
Iteration 3 Training loss: 0.07211876
Iteration 3 Testing accuracy: 0.07169094
Iteration 4 Training loss: 0.07329009
Iteration 4 Testing accuracy: 0.072983265
Iteration 5 Training loss: 0.07451299
Iteration 5 Testing accuracy: 0.076056875
Iteration 6 Training loss: 0.07673064
Iteration 6 Testing accuracy: 0.07655288
Iteration 7 Training loss: 0.0743922
Iteration 7 Testing accuracy: 0.07407875
Iteration 8 Training loss: 0.074742354
Iteration 8 Testing accuracy: 0.07448287
Iteration 9 Training loss: 0.07526538
Iteration 9 Testing accuracy: 0.07422358
Iteration 10 Training loss: 0.07524644
Iteration 10 Testing accuracy: 0.07515135
Iteration 11 Training loss: 0.07380479
Iteration 11 Testing accuracy: 0.07363583
Iteration 12 Training loss: 0.07321031
Iteration 12 Testing accuracy: 0.072899364
Iteration 13 Training loss: 0.07321802
Iteration 13 Testing accuracy: 0.07266905
Iteration 14 Training loss: 0.07448674
Iteration 14 Testing accuracy: 0.07379024
Iteration 15 Training loss: 0.07486758
Iteration 15 Testing accuracy: 0.075920485
Iteration 16 Training loss: 0.07681484
Iteration 16 Testing accuracy: 0.0757261
Iteration 17 Training loss: 0.07668789
Iteration 17 Testing accuracy: 0.07559831
Iteration 18 Training loss: 0.07914759
Iteration 18 Testing accuracy: 0.07925277
Iteration 19 Training loss: 0.079218045
Iteration 19 Testing accuracy: 0.078609444
Iteration 20 Training loss: 0.08167129
Iteration 20 Testing accuracy: 0.080692634
Iteration 1 Training loss: 0.079039335
Iteration 1 Testing accuracy: 0.08058755
Iteration 2 Training loss: 0.0789365
Iteration 2 Testing accuracy: 0.07951276
Iteration 3 Training loss: 0.078317426
Iteration 3 Testing accuracy: 0.078934565
Iteration 4 Training loss: 0.07778294
Iteration 4 Testing accuracy: 0.07760117
Iteration 5 Training loss: 0.07754435
Iteration 5 Testing accuracy: 0.07988212
Iteration 6 Training loss: 0.078574926
Iteration 6 Testing accuracy: 0.07915582
Iteration 7 Training loss: 0.08114477
Iteration 7 Testing accuracy: 0.082214825
Iteration 8 Training loss: 0.0815421
Iteration 8 Testing accuracy: 0.08165085
Iteration 9 Training loss: 0.081352316
Iteration 9 Testing accuracy: 0.08189009
Iteration 10 Training loss: 0.081102625
Iteration 10 Testing accuracy: 0.0813034

Iteration 11 Training loss: 0.08211983
Iteration 11 Testing accuracy: 0.08198579
Iteration 12 Training loss: 0.08223602
Iteration 12 Testing accuracy: 0.082273595
Iteration 13 Training loss: 0.08362925
Iteration 13 Testing accuracy: 0.08430026
Iteration 14 Training loss: 0.08524985
Iteration 14 Testing accuracy: 0.085079424
Iteration 15 Training loss: 0.08497644
Iteration 15 Testing accuracy: 0.08528982
Iteration 16 Training loss: 0.08602466
Iteration 16 Testing accuracy: 0.08581066
Iteration 17 Training loss: 0.087311216
Iteration 17 Testing accuracy: 0.08744441
Iteration 18 Training loss: 0.08762043
Iteration 18 Testing accuracy: 0.08774392
Iteration 19 Training loss: 0.08830884
Iteration 19 Testing accuracy: 0.08812629
Iteration 20 Training loss: 0.08819114
Iteration 20 Testing accuracy: 0.088298455
Iteration 1 Training loss: 0.075165555
Iteration 1 Testing accuracy: 0.07561177
Iteration 2 Training loss: 0.07645047
Iteration 2 Testing accuracy: 0.07937319
Iteration 3 Training loss: 0.07681951
Iteration 3 Testing accuracy: 0.077027455
Iteration 4 Training loss: 0.07862142
Iteration 4 Testing accuracy: 0.07838066
Iteration 5 Training loss: 0.07926727
Iteration 5 Testing accuracy: 0.079636134
Iteration 6 Training loss: 0.080101006
Iteration 6 Testing accuracy: 0.08108801
Iteration 7 Training loss: 0.080208145
Iteration 7 Testing accuracy: 0.07994376
Iteration 8 Training loss: 0.07999382
Iteration 8 Testing accuracy: 0.07997297
Iteration 9 Training loss: 0.08177766
Iteration 9 Testing accuracy: 0.08224761
Iteration 10 Training loss: 0.08222746
Iteration 10 Testing accuracy: 0.082038224
Iteration 11 Training loss: 0.084539294
Iteration 11 Testing accuracy: 0.08442556
Iteration 12 Training loss: 0.084388755
Iteration 12 Testing accuracy: 0.08507566
Iteration 13 Training loss: 0.08476214
Iteration 13 Testing accuracy: 0.08425423
Iteration 14 Training loss: 0.084617026
Iteration 14 Testing accuracy: 0.08458445
Iteration 15 Training loss: 0.08550263
Iteration 15 Testing accuracy: 0.0853222
Iteration 16 Training loss: 0.08667665
Iteration 16 Testing accuracy: 0.08614618
Iteration 17 Training loss: 0.088201895
Iteration 17 Testing accuracy: 0.08823029
Iteration 18 Training loss: 0.08975811
Iteration 18 Testing accuracy: 0.09007982
Iteration 19 Training loss: 0.09156709
Iteration 19 Testing accuracy: 0.091043435
Iteration 20 Training loss: 0.091578364
Iteration 20 Testing accuracy: 0.09201473
Iteration 1 Training loss: 0.05679932
Iteration 1 Testing accuracy: 0.055957925
Iteration 2 Training loss: 0.055679936
Iteration 2 Testing accuracy: 0.05525299
Iteration 3 Training loss: 0.05547545
Iteration 3 Testing accuracy: 0.054390103
Iteration 4 Training loss: 0.055518694
Iteration 4 Testing accuracy: 0.05470719

Iteration 4 Training loss: 0.05470719
Iteration 4 Testing accuracy: 0.055969633
Iteration 5 Training loss: 0.056492984
Iteration 5 Testing accuracy: 0.055969633
Iteration 6 Training loss: 0.055696845
Iteration 6 Testing accuracy: 0.054599278
Iteration 7 Training loss: 0.056644984
Iteration 7 Testing accuracy: 0.055708993
Iteration 8 Training loss: 0.05669187
Iteration 8 Testing accuracy: 0.056034457
Iteration 9 Training loss: 0.05898718
Iteration 9 Testing accuracy: 0.059213728
Iteration 10 Training loss: 0.059931975
Iteration 10 Testing accuracy: 0.059068132
Iteration 11 Training loss: 0.06086666
Iteration 11 Testing accuracy: 0.0607898
Iteration 12 Training loss: 0.061082993
Iteration 12 Testing accuracy: 0.060430624
Iteration 13 Training loss: 0.061733507
Iteration 13 Testing accuracy: 0.061638884
Iteration 14 Training loss: 0.06157173
Iteration 14 Testing accuracy: 0.062047374
Iteration 15 Training loss: 0.0626186
Iteration 15 Testing accuracy: 0.061780524
Iteration 16 Training loss: 0.06219375
Iteration 16 Testing accuracy: 0.062292516
Iteration 17 Training loss: 0.06260051
Iteration 17 Testing accuracy: 0.062192924
Iteration 18 Training loss: 0.06253953
Iteration 18 Testing accuracy: 0.06260571
Iteration 19 Training loss: 0.06234809
Iteration 19 Testing accuracy: 0.061129104
Iteration 20 Training loss: 0.061728712
Iteration 20 Testing accuracy: 0.061098717
Iteration 1 Training loss: 0.07231123
Iteration 1 Testing accuracy: 0.072157785
Iteration 2 Training loss: 0.07287089
Iteration 2 Testing accuracy: 0.073308
Iteration 3 Training loss: 0.073992625
Iteration 3 Testing accuracy: 0.07528774
Iteration 4 Training loss: 0.07444405
Iteration 4 Testing accuracy: 0.07425485
Iteration 5 Training loss: 0.07521784
Iteration 5 Testing accuracy: 0.075460926
Iteration 6 Training loss: 0.076843336
Iteration 6 Testing accuracy: 0.076750964
Iteration 7 Training loss: 0.07620705
Iteration 7 Testing accuracy: 0.07582133
Iteration 8 Training loss: 0.07725323
Iteration 8 Testing accuracy: 0.076812476
Iteration 9 Training loss: 0.078463815
Iteration 9 Testing accuracy: 0.078493714
Iteration 10 Training loss: 0.07972017
Iteration 10 Testing accuracy: 0.079164654
Iteration 11 Training loss: 0.081053175
Iteration 11 Testing accuracy: 0.08061298
Iteration 12 Training loss: 0.082260676
Iteration 12 Testing accuracy: 0.08180887
Iteration 13 Training loss: 0.083603315
Iteration 13 Testing accuracy: 0.083027415
Iteration 14 Training loss: 0.086090475
Iteration 14 Testing accuracy: 0.08510306
Iteration 15 Training loss: 0.08824066
Iteration 15 Testing accuracy: 0.08860769
Iteration 16 Training loss: 0.08943288
Iteration 16 Testing accuracy: 0.08872695
Iteration 17 Training loss: 0.09043
Iteration 17 Testing accuracy: 0.089869946
Iteration 18 Training loss: 0.09209121

Iteration 18 Testing accuracy: 0.09156909
Iteration 19 Training loss: 0.09213455
Iteration 19 Testing accuracy: 0.091458514
Iteration 20 Training loss: 0.09384238
Iteration 20 Testing accuracy: 0.09379475
Training network with 256 hidden units
Iteration 1 Training loss: 0.11437055
Iteration 1 Testing accuracy: 0.11518595
Iteration 2 Training loss: 0.11437054
Iteration 2 Testing accuracy: 0.11518594
Iteration 3 Training loss: 0.11437053
Iteration 3 Testing accuracy: 0.11518595
Iteration 4 Training loss: 0.11437053
Iteration 4 Testing accuracy: 0.11518594
Iteration 5 Training loss: 0.11437053
Iteration 5 Testing accuracy: 0.11518595
Iteration 6 Training loss: 0.11437052
Iteration 6 Testing accuracy: 0.11518595
Iteration 7 Training loss: 0.11437053
Iteration 7 Testing accuracy: 0.11518595
Iteration 8 Training loss: 0.11437054
Iteration 8 Testing accuracy: 0.11518595
Iteration 9 Training loss: 0.11437055
Iteration 9 Testing accuracy: 0.11518595
Iteration 10 Training loss: 0.11437053
Iteration 10 Testing accuracy: 0.11518595
Iteration 11 Training loss: 0.11437054
Iteration 11 Testing accuracy: 0.11518594
Iteration 12 Training loss: 0.11437054
Iteration 12 Testing accuracy: 0.11518595
Iteration 13 Training loss: 0.11437053
Iteration 13 Testing accuracy: 0.11518595
Iteration 14 Training loss: 0.11437055
Iteration 14 Testing accuracy: 0.11518594
Iteration 15 Training loss: 0.11437052
Iteration 15 Testing accuracy: 0.11518595
Iteration 16 Training loss: 0.11437052
Iteration 16 Testing accuracy: 0.11518595
Iteration 17 Training loss: 0.11437054
Iteration 17 Testing accuracy: 0.11518594
Iteration 18 Training loss: 0.11437053
Iteration 18 Testing accuracy: 0.11518595
Iteration 19 Training loss: 0.11437055
Iteration 19 Testing accuracy: 0.11518595
Iteration 20 Training loss: 0.11437053
Iteration 20 Testing accuracy: 0.11518595
Iteration 1 Training loss: 0.11256162
Iteration 1 Testing accuracy: 0.113806464
Iteration 2 Training loss: 0.112567805
Iteration 2 Testing accuracy: 0.11382347
Iteration 3 Training loss: 0.112560526
Iteration 3 Testing accuracy: 0.1138049
Iteration 4 Training loss: 0.11318936
Iteration 4 Testing accuracy: 0.11443093
Iteration 5 Training loss: 0.11318569
Iteration 5 Testing accuracy: 0.11441815
Iteration 6 Training loss: 0.11318681
Iteration 6 Testing accuracy: 0.11441777
Iteration 7 Training loss: 0.11318481
Iteration 7 Testing accuracy: 0.11441739
Iteration 8 Training loss: 0.113186024
Iteration 8 Testing accuracy: 0.11441978
Iteration 9 Training loss: 0.11318461
Iteration 9 Testing accuracy: 0.11441735
Iteration 10 Training loss: 0.11318385
Iteration 10 Testing accuracy: 0.11441888
Iteration 11 Training loss: 0.11318399
Iteration 11 Testing accuracy: 0.11441704

Iteration 12 Training loss: 0.11318656
Iteration 12 Testing accuracy: 0.114420846
Iteration 13 Training loss: 0.11318883
Iteration 13 Testing accuracy: 0.11442101
Iteration 14 Training loss: 0.11318802
Iteration 14 Testing accuracy: 0.11442256
Iteration 15 Training loss: 0.113185726
Iteration 15 Testing accuracy: 0.11441805
Iteration 16 Training loss: 0.11319196
Iteration 16 Testing accuracy: 0.11442566
Iteration 17 Training loss: 0.11318531
Iteration 17 Testing accuracy: 0.11441757
Iteration 18 Training loss: 0.11318717
Iteration 18 Testing accuracy: 0.114419594
Iteration 19 Training loss: 0.11318621
Iteration 19 Testing accuracy: 0.11441815
Iteration 20 Training loss: 0.11318941
Iteration 20 Testing accuracy: 0.11442208
Iteration 1 Training loss: 0.11458512
Iteration 1 Testing accuracy: 0.11552056
Iteration 2 Training loss: 0.11458511
Iteration 2 Testing accuracy: 0.11552055
Iteration 3 Training loss: 0.11458511
Iteration 3 Testing accuracy: 0.11552055
Iteration 4 Training loss: 0.114585124
Iteration 4 Testing accuracy: 0.11552056
Iteration 5 Training loss: 0.11458512
Iteration 5 Testing accuracy: 0.11552056
Iteration 6 Training loss: 0.11458511
Iteration 6 Testing accuracy: 0.11552056
Iteration 7 Training loss: 0.11458511
Iteration 7 Testing accuracy: 0.11552055
Iteration 8 Training loss: 0.11458512
Iteration 8 Testing accuracy: 0.11552055
Iteration 9 Training loss: 0.11458511
Iteration 9 Testing accuracy: 0.11552056
Iteration 10 Training loss: 0.11458512
Iteration 10 Testing accuracy: 0.115520574
Iteration 11 Training loss: 0.11458512
Iteration 11 Testing accuracy: 0.11552056
Iteration 12 Training loss: 0.11458512
Iteration 12 Testing accuracy: 0.11552056
Iteration 13 Training loss: 0.11458512
Iteration 13 Testing accuracy: 0.11552055
Iteration 14 Training loss: 0.11458512
Iteration 14 Testing accuracy: 0.11552056
Iteration 15 Training loss: 0.114585094
Iteration 15 Testing accuracy: 0.11552055
Iteration 16 Training loss: 0.114585124
Iteration 16 Testing accuracy: 0.11552056
Iteration 17 Training loss: 0.11458511
Iteration 17 Testing accuracy: 0.11552056
Iteration 18 Training loss: 0.11458512
Iteration 18 Testing accuracy: 0.11552056
Iteration 19 Training loss: 0.11458512
Iteration 19 Testing accuracy: 0.11552056
Iteration 20 Training loss: 0.11458511
Iteration 20 Testing accuracy: 0.11552056
Iteration 1 Training loss: 0.115247816
Iteration 1 Testing accuracy: 0.116180375
Iteration 2 Training loss: 0.115247816
Iteration 2 Testing accuracy: 0.116180375
Iteration 3 Training loss: 0.11524782
Iteration 3 Testing accuracy: 0.116180375
Iteration 4 Training loss: 0.11524782
Iteration 4 Testing accuracy: 0.116180375
Iteration 5 Training loss: 0.11524782
Iteration 5 Testing accuracy: 0.11618036

Iteration 5 Training loss: 0.11524782
Iteration 5 Testing accuracy: 0.11618036
Iteration 6 Training loss: 0.11524782
Iteration 6 Testing accuracy: 0.11618036
Iteration 7 Training loss: 0.115247816
Iteration 7 Testing accuracy: 0.116180375
Iteration 8 Training loss: 0.11524782
Iteration 8 Testing accuracy: 0.116180375
Iteration 9 Training loss: 0.11524781
Iteration 9 Testing accuracy: 0.116180375
Iteration 10 Training loss: 0.115247816
Iteration 10 Testing accuracy: 0.11618036
Iteration 11 Training loss: 0.115247816
Iteration 11 Testing accuracy: 0.116180375
Iteration 12 Training loss: 0.11524782
Iteration 12 Testing accuracy: 0.11618036
Iteration 13 Training loss: 0.11524782
Iteration 13 Testing accuracy: 0.11618036
Iteration 14 Training loss: 0.11524782
Iteration 14 Testing accuracy: 0.116180375
Iteration 15 Training loss: 0.115247816
Iteration 15 Testing accuracy: 0.11618036
Iteration 16 Training loss: 0.11524781
Iteration 16 Testing accuracy: 0.116180345
Iteration 17 Training loss: 0.11524782
Iteration 17 Testing accuracy: 0.116180375
Iteration 18 Training loss: 0.11524782
Iteration 18 Testing accuracy: 0.116180375
Iteration 19 Training loss: 0.115247816
Iteration 19 Testing accuracy: 0.116180375
Iteration 20 Training loss: 0.115247816
Iteration 20 Testing accuracy: 0.116180375
Iteration 1 Training loss: 0.11340048
Iteration 1 Testing accuracy: 0.11461649
Iteration 2 Training loss: 0.11337884
Iteration 2 Testing accuracy: 0.11460404
Iteration 3 Training loss: 0.11338095
Iteration 3 Testing accuracy: 0.11460676
Iteration 4 Training loss: 0.1134061
Iteration 4 Testing accuracy: 0.11462624
Iteration 5 Training loss: 0.11337803
Iteration 5 Testing accuracy: 0.11460434
Iteration 6 Training loss: 0.11338411
Iteration 6 Testing accuracy: 0.11460824
Iteration 7 Training loss: 0.113913015
Iteration 7 Testing accuracy: 0.115157686
Iteration 8 Training loss: 0.113913015
Iteration 8 Testing accuracy: 0.115157686
Iteration 9 Training loss: 0.113913015
Iteration 9 Testing accuracy: 0.115157686
Iteration 10 Training loss: 0.11391301
Iteration 10 Testing accuracy: 0.115157686
Iteration 11 Training loss: 0.11391301
Iteration 11 Testing accuracy: 0.1151577
Iteration 12 Training loss: 0.113913015
Iteration 12 Testing accuracy: 0.115157686
Iteration 13 Training loss: 0.11391301
Iteration 13 Testing accuracy: 0.115157686
Iteration 14 Training loss: 0.11391301
Iteration 14 Testing accuracy: 0.115157686
Iteration 15 Training loss: 0.11391301
Iteration 15 Testing accuracy: 0.115157686
Iteration 16 Training loss: 0.113913015
Iteration 16 Testing accuracy: 0.115157686
Iteration 17 Training loss: 0.11391301
Iteration 17 Testing accuracy: 0.1151577
Iteration 18 Training loss: 0.11391301
Iteration 18 Testing accuracy: 0.115157686
Iteration 19 Training loss: 0.113913015

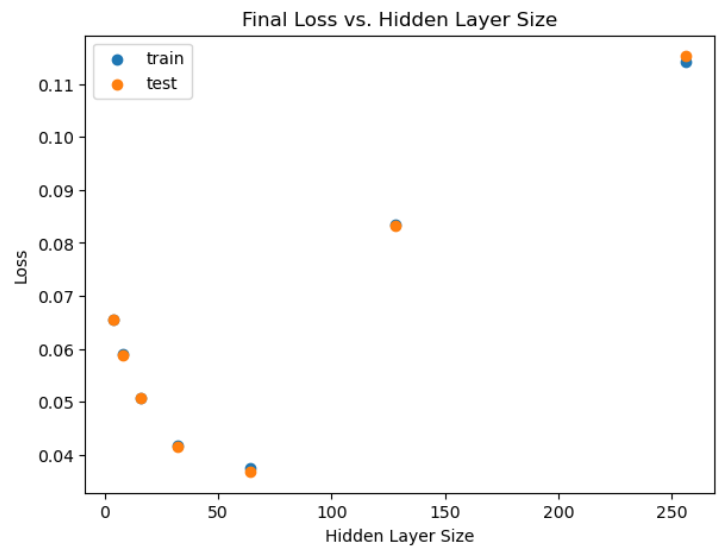
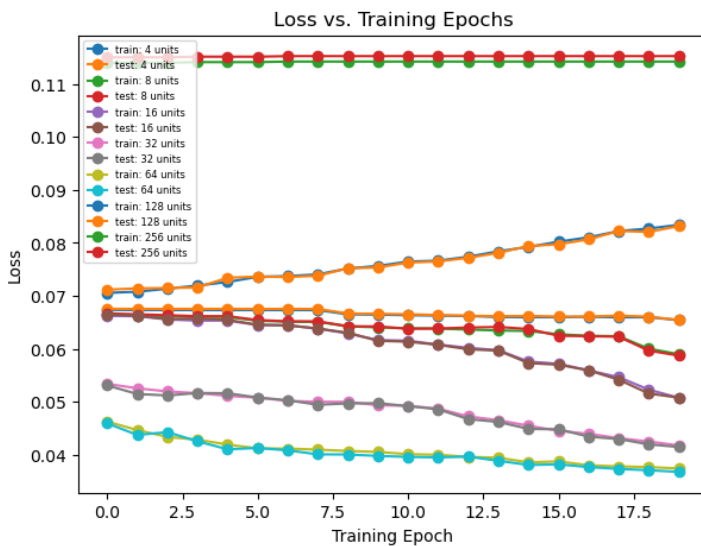
Iteration 19 Testing accuracy: 0.115157686
 Iteration 20 Training loss: 0.113913015
 Iteration 20 Testing accuracy: 0.115157686

In [153]:

```
# plot training and testing loss (averaged over 5 trials):
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))

# plot 1: loss vs. training epochs
for i, sz in enumerate(hidden_layer_sizes):
    ax1.plot(train_mean[i], 'o-', label=f'train: {sz} units')
    ax1.plot(test_mean[i], 'o-', label=f'test: {sz} units')
ax1.set(title='Loss vs. Training Epochs', xlabel='Training Epoch', ylabel='Loss')
ax1.legend(prop={'size': 6})

# plot 2: final loss vs. hidden layer size
ax2.scatter(hidden_layer_sizes, np.array(train_mean[:, -1]), marker='o', label=f'train')
ax2.scatter(hidden_layer_sizes, np.array(test_mean[:, -1]), marker='o', label=f'test')
ax2.set(title='Final Loss vs. Hidden Layer Size', xlabel='Hidden Layer Size', ylabel='Loss')
ax2.legend()
plt.show()
```



Result:

After 20 training epochs, the best performing hidden layer size is 64. Beyond this, the network with 128 nodes has a significantly larger starting loss which trends upwards with more training, not down. It is possible that some number between 64 and 128 would be even better, or that more training would lead to a different conclusion. For the next question, I will use the 64-node network.

In [221]:

```
# 3.1.2 - Plot 10 original and reconstructed images using the best hidden layer size from the previous step
plt.subplots(4, 10, figsize=(6, 8))
for i in range(10):
    # plot original image from training dataset
    plt.subplot(10, 4, 1+4*i)
    plt.xticks([])
    plt.yticks([])
    if i == 0: plt.title('Train')
    plt.imshow(train_dataset[i][0].reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')

    # plot reconstructed image
    reconstructed = autoencoders[20](train_dataset[i][0]).detach().numpy()
```

```

plt.subplot(10,4,2+4*i)
plt.xticks([])
plt.yticks([])
if i == 0: plt.title('Reconstructed')
plt.imshow(reconstructed.reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')

# plot original image from testing dataset
plt.subplot(10,4,3+4*i)
plt.xticks([])
plt.yticks([])
if i == 0: plt.title('Test')
plt.imshow(test_dataset[i][0].reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')

# plot reconstructed image
reconstructed = autoencoders[20](test_dataset[i][0]).detach().numpy()
plt.subplot(10,4,4+4*i)
plt.xticks([])
plt.yticks([])
if i == 0: plt.title('Reconstructed')
plt.imshow(reconstructed.reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')

```

/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3698652392.py:5: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.

```
plt.subplot(10,4,1+4*i)
```

/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3698652392.py:13: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.







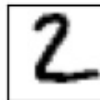











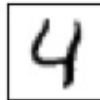







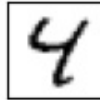



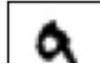

```
plt.subplot(10,4,2+4*i)
```

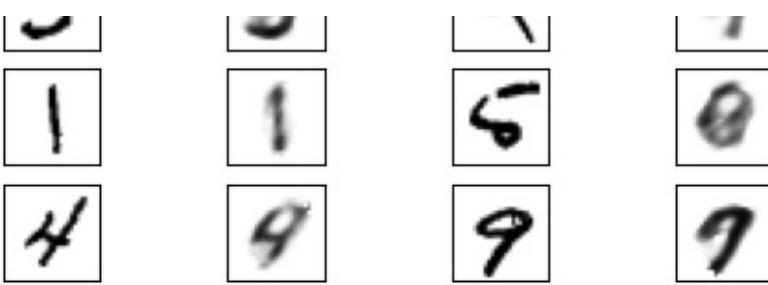
/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3698652392.py:20: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.

```
plt.subplot(10,4,3+4*i)
```

/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3698652392.py:28: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.

```
plt.subplot(10,4,4+4*i)
```

Train	Reconstructed	Test	Reconstructed
			
			
			
			
			
			
			
			



How do these compare to the reconstructions of the Hopfield networks above?

Qualitativeley, the reconstructions of the autoencoder are more legible than those of the Hopfield network because the images are greyscaled (rather than binary 0/1). The uncertainty in the reconstruction is shown as a fuzzy digit with lighter grey pixels, rather than the collection of dark black pixels generated by the Hopfield network. Overall, this makes for digits that are slightly easier to interpret.

In [222]:

```
def corrupt(data, prob):
    for img in range(data.shape[0]):

        # get number of pixels to flip for image size and selected probability:
        n_pixels = int(data.shape[1] * prob)

        # select a random sample of n_pixels:
        permuted = np.random.permutation(np.arange(0, data.shape[1], 1))
        pixels = permuted[:n_pixels]

        # set pixels to 0 or 1 with equal probability:
        for p in pixels:
            flip = np.random.rand()
            if flip < 0.5:
                data[img, p] = 0
            else:
                data[img, p] = 1

    return data
```

In [233]:

```
# 3.1.3 - Compare loss on inputs with salt-and-pepper noise
prob_flip = torch.linspace(0.01, 0.75, 10)
test_data = test_dataset[:10,:][0]
mses = np.zeros(test_data.shape[0])

for n, p in enumerate(prob_flip):
    noisy_test = test_data.clone()
    noisy_test = corrupt(noisy_test, p)

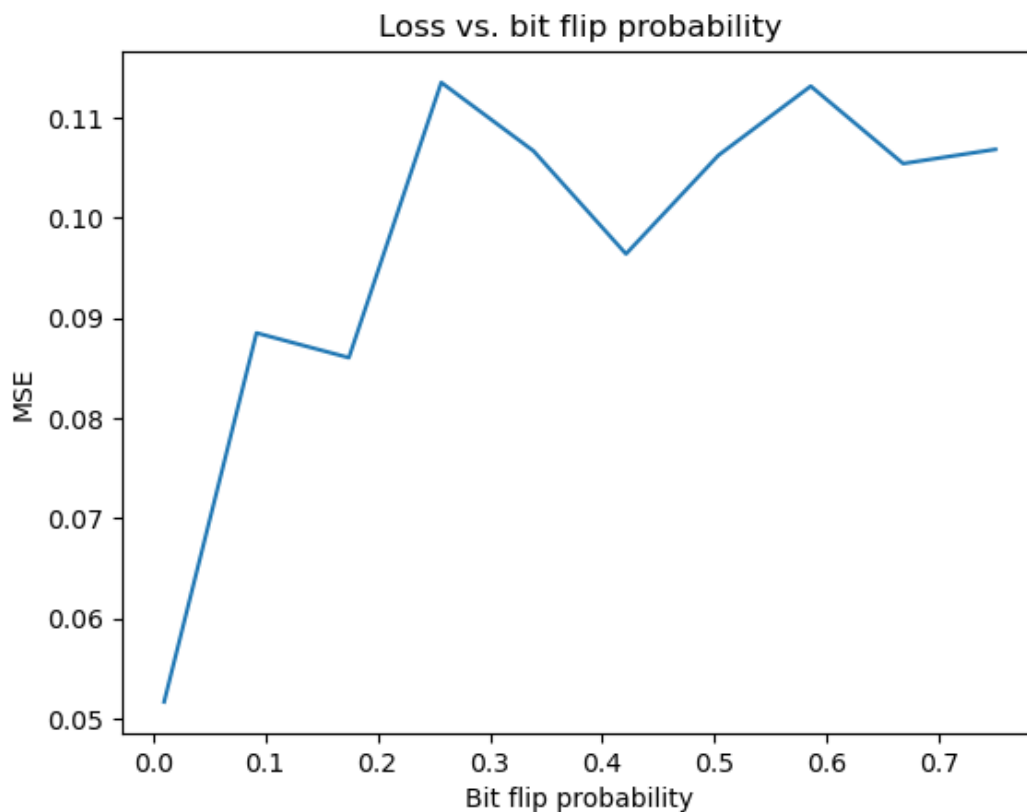
    # compute test error with autoencoder for each noisy pattern:
    for i in range(noisy_test.shape[0]):
        reconstr = autoencoders[20](noisy_test[i])
        mses[n] += mse(reconstr, test_data[i])

    mses[n] /= noisy_test.shape[0]
```

In [234]:

```
# plot loss:
plt.figure()
plt.plot(prob_flip, mses)
plt.xlabel('Bit flip probability')
plt.ylabel('MSE')
plt.title('Loss vs. bit flip probability')
```

```
plt.show()
```



In [238]:

3.1.2 - Plot 10 original and reconstructed images using the best hidden layer size from the previous step

```
plt.subplots(2, 10, figsize=(6, 8))
```

```
for i in range(10):
```

```
    # plot original image from test dataset
```

```
    plt.subplot(10,2,1+2*i)
```

```
    plt.xticks([])
```

```
    plt.yticks([])
```

```
    if i == 0: plt.title('Test')
```

```
    plt.imshow(test_data[i].reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')
```

```
    # plot reconstructed image
```

```
    reconstructed = autoencoders[20](test_data[i]).detach().numpy()
```

```
    plt.subplot(10,2,2+2*i)
```

```
    plt.xticks([])
```

```
    plt.yticks([])
```

```
    if i == 0: plt.title('Reconstructed')
```

```
    plt.imshow(reconstructed.reshape((28,28)), vmin=0, vmax=1, cmap='gray_r')
```

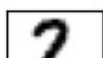
/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3993531560.py:6: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call `ax.remove()` as needed.

```
    plt.subplot(10,2,1+2*i)
```

/var/folders/rx/5_fd7v5s5dbc3yr3cx7bw9m40000gn/T/ipykernel_54013/3993531560.py:14: Matplotlib DeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call `ax.remove()` as needed.

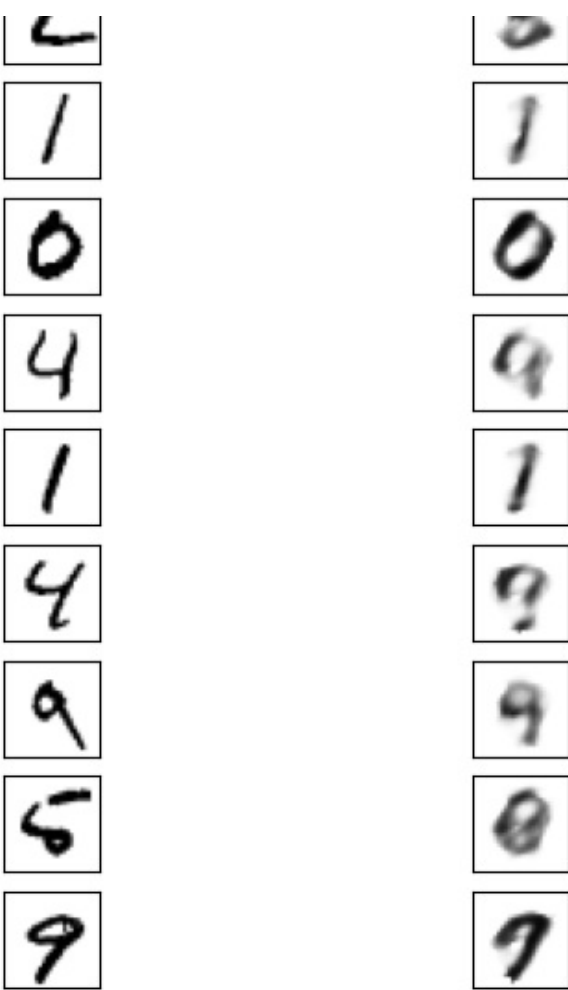
```
    plt.subplot(10,2,2+2*i)
```

Test



Reconstructed





3.2 [1 Mark] Described what you've observed about the results of the autoencoder, compared to the Hopfield networks.

TO DO

The autoencoder is less

3.3 [1 Marks] Sensory Preconditioning - Now we are going to look at small datasets. We are going to use the sensory preconditioning protocol, discussed by Gluck and Meyers.

We will also need to construct a data set that performs the preconditioning. We will break this down into three phases:

1. Do the sensory preconditioning. Plot the training loss vs number of epochs, describe the resultant behaviour.

In [146]:

```
# Phase 1: Train the network
# features: s1, s2, context1, context2
# we are teaching this network that these inputs only occur together, regardless of the context.
stimuli = np.array([[0,0,0,0],
                    [0,0,0,0],
                    [0,0,1,0],
                    [0,0,0,1],
                    [0,0,1,1],
                    [1,1,0,0],
                    [1,1,0,0],
                    [1,1,1,0],
                    [1,1,0,1],
```

```
[1,1,1,1])).astype(np.float32)
```

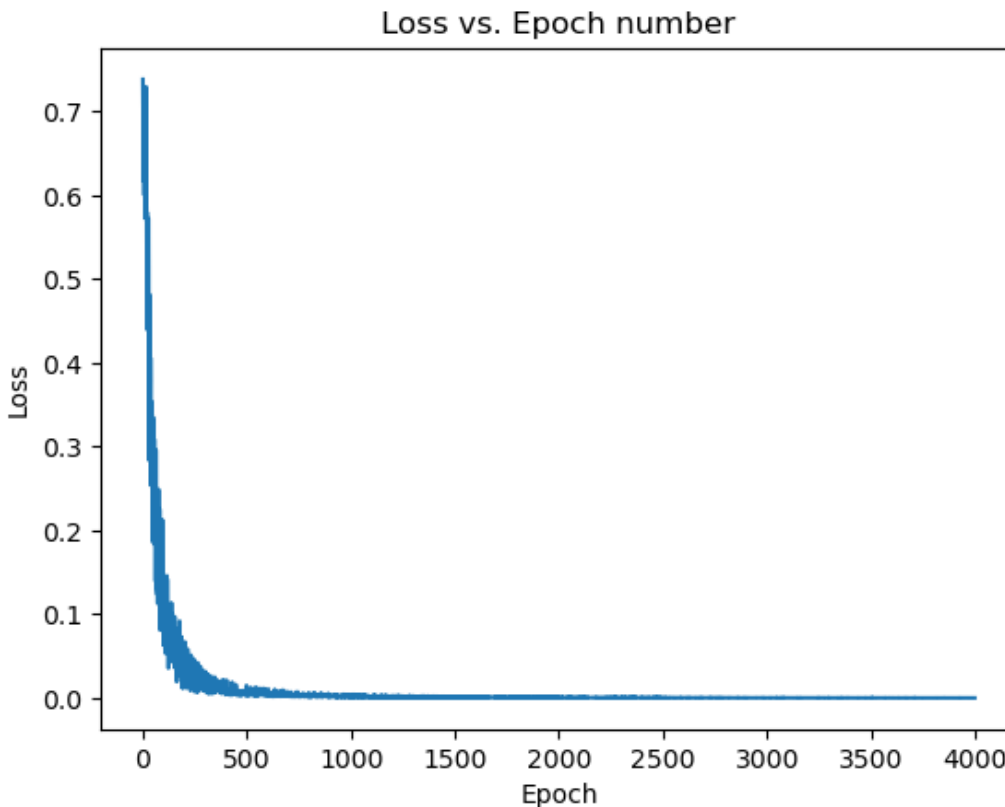
```
dataset = TensorDataset(torch.from_numpy(stimuli), torch.from_numpy(stimuli))  
data_loader = torch.utils.data.DataLoader(dataset, batch_size=8, shuffle=True)
```

```
epochs = 2000  
outputs = []  
losses = []
```

```
model = Autoencoder(4, 3) # must be compatible with shape (8x4)  
optimizer = torch.optim.Adam(model.parameters(), lr = 1e-1)  
loss_function = torch.nn.BCELoss()  
  
for epoch in range(epochs):  
    for (inp, out) in data_loader:  
        # print(inp.shape, out.shape)  
  
        reconstructed = model(inp) # output of autoencoder  
        loss = loss_function(reconstructed, out) # calculating the loss function  
  
        # The gradients are set to zero, the gradient is computed and stored.  
        optimizer.zero_grad()  
        loss.backward()  
        optimizer.step() # .step() performs parameter update  
  
        # Storing the losses in a list for plotting  
        losses.append(loss.detach().numpy())  
    pass  
    outputs.append((epoch, inp, reconstructed))  
pass
```

In [147]:

```
# Plot loss performance  
plt.plot(losses)  
plt.xlabel('Epoch')  
plt.ylabel('Loss')  
plt.title('Loss vs. Epoch number')  
plt.show()
```



Discussion

The resultant behaviour is

1. Train the conditioned response to one stimulus. We will train this using simple linear regression where we map the the hidden state to the desired output. You can reuse the linear regression code from above.

Lets call the features (columns) of the stimuli data as s1, s2, c1, and c2.

In [240]:

```
conditioning_stimuli = torch.from_numpy(np.array([[0,0,0,0],
          [0,0,1,0],
          [0,0,0,1],
          [0,0,1,1],
          [1,0,0,0],
          [1,0,1,0],
          [1,0,0,1],
          [1,0,1,1]]).astype(np.float32))
conditioning_outputs = torch.from_numpy(np.array([[0,0,0,0,1,1,1,1]]).astype(np.float32).T)
conditioning_features = model.encoder(conditioning_stimuli)

def get_weights(features, values, lamb=0.01):
    W = pseudo_inv(features, lamb=lamb) @ values
    return W

def evaluate(weights, features):
    return torch.sign(features @ weights)

W = get_weights(conditioning_features, conditioning_outputs)
conditioned_response = evaluate(W, conditioning_features[4:,:])
```

1. Report the strength of the conditioned response to the other stimulus. Has the network transferred learning from one stimuli to the other?

In [241]:

```
test_stimuli = torch.from_numpy(np.array([
          [0,1,0,0],
          [0,1,1,0],
          [0,1,0,1],
          [0,1,1,1]]).astype(np.float32))

testing_features = model.encoder(test_stimuli)
transferred_response = evaluate(W, testing_features)
```

1. Compare to the response of the system without either of the preconditioned inputs

In [242]:

```
unconditioned_responses = evaluate(W, conditioning_features[:4,:])
```

1. Plot graphs showing the strength of the response to the conditioned stimuli, the pre-conditioned stimuli, and the unconditioned stimuli (*i.e.*, s1 = 0, s2 = 0, and context in {(0,0),(0,1),(1,0),(1,1)}
)

In [247]:

```
# plot conditioned, transferred, and unconditioned responses
fig, ax = plt.subplots()
```

```

ax.bar([1,2,3], [conditioned_response.detach().numpy().mean(), transferred_response.detach(
).numpy().mean(), unconditioned_responses.detach().numpy().mean()])
ax.set_ylabel('Conditioned Responses')
ax.set_xticks([1,2,3], labels=['s1+', 's2+', 's1-,s2-'])

# response_mu = torch.mean([conditioned_response, transferred_response, unconditioned_respon
ses], axis=1)
# response_std = torch.std([conditioned_response, transferred_response, unconditioned_respon
ses], axis=1)

# fig, ax = plt.subplots()
# ax.bar([1,2,3], response_mu, yerr=response_std, align='center', alpha=0.5, ecolor='black',
capsize=10)
# ax.set_ylabel('Conditioned Responses')
# ax.set_xticks([1,2,3], labels=['s1+', 's2+', 's1-,s2-'])

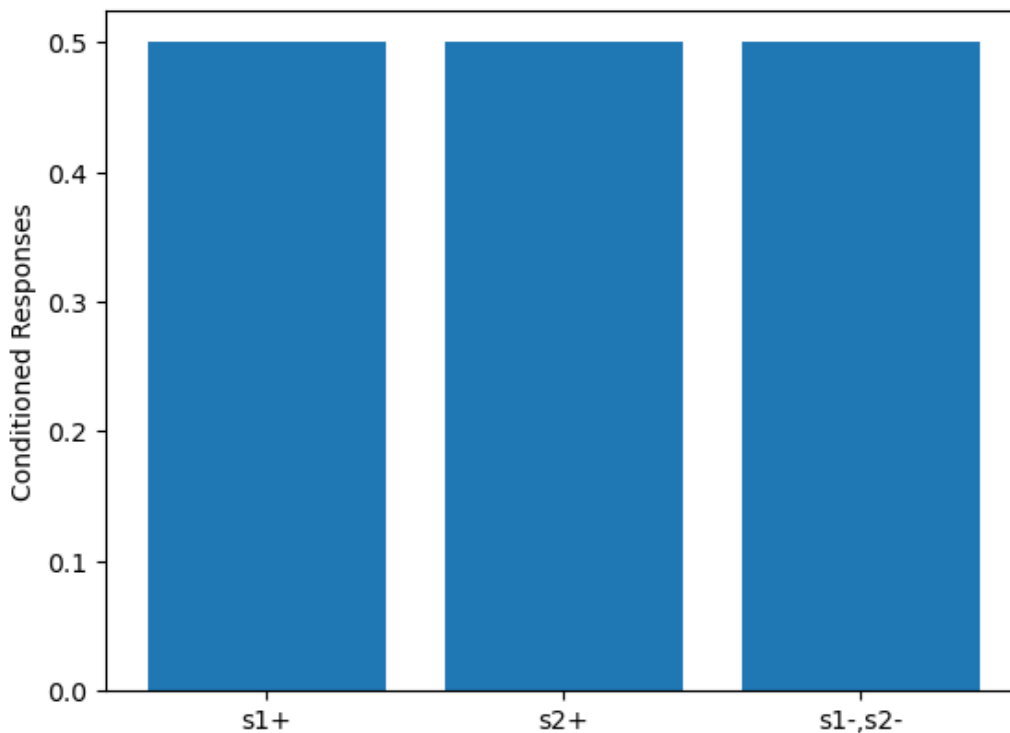
```

Out[247]:

```

[<matplotlib.axis.XTick at 0x17f10d990>,
 <matplotlib.axis.XTick at 0x17e95d110>,
 <matplotlib.axis.XTick at 0x14468ba10>]

```



[BONUS] [1 Mark] Repeat the sensory preconditioning, but use an conditioning stimulus that is not 100% correlated with the unconditioned stimulus. That is: change phase 2 so that it is not possible to perfectly predict the output.