Homework 3: Report

Anh Nguyen
nguye2aq@mail.uc.edu

Problem 1:

Problem Statement: A simple feed forward neural network was trained on 4000 images of digits from 0 to 9 (400 images each) from the MNIST dataset, then tested on another 1000 images (100 images each). Input is a flattened image array of 784 pixels, put into a input layer of 784 weights, output layer being 10 neurons for 10 classes.

System Specification:

The hyperparameters were tested and evaluated as below, and others unchanged across experiments are hidden activation as tanh, output activation as sigmoid, and loss as mean squared error. The model was trained with stochastic gradient descent. I did not change number of early stopping epochs as 2, and minimum test error fraction improvement being 0.03, as the training progress converges quickly and straightforwardly.

- *Exp0*: Beginning training with random default hyper-params

{'H': 0.75, 'L': 0.25, 'hidden_neuron': 200, 'lr': 0.01, 'momentum': 0.5, 'num_epochs': 24},
=> Training error fraction: 0.0965, testing error fraction: 0.139, early stopped at 32 epochs

- *Exp2*: I toggled a few hyper-parms, for exp1,2 and 7, decreased the number of hidden neurons to 150 but did not manage to increase performance on the test set

{'H': 0.1, 'L': 0.9, 'hidden_neuron': 200, 'lr': 0.005, 'momentum': 0.5, 'num_epochs': 70},
=> Training error fraction: 0.1065, testing error fraction: 0.146, early stopped at 58 epochs

- *Exp4*: I changed thresholds to be harsher for the model, but it failed to improve.

{'H': 0.1, 'L': 0.9, 'hidden_neuron': 200, 'lr': 0.01, 'momentum': 0.5, 'num_epochs': 80},
=> Training error fraction: 0.1475, testing error fraction: 0.211, early stopped at 48 epochs

- *Exp5*: I decreased learning rate and momentum, but it also failed to improve the model.

{'H': 0.25, 'L': 0.75, 'hidden_neuron': 200, 'lr': 0.0025, 'momentum': 0.3, 'num_epochs': 80},
=> Training error fraction: 0.14175, testing error fraction: 0.162, early stopped at 58 epochs

⇒ The set of hyper-parameters from exp0 provides the maximum model performance in terms of error fractions for both train and test set.

Results:

True\Pred	0	1	2	3	4	5	6	7	8	9
0	11	4	54	54	18	24	12	26	131	66
1	28	0	2	41	0	1	221	0	27	80
2	25	0	11	121	27	4	41	5	100	66
3	3	0	9	82	7	27	28	0	44	200
4	19	5	22	14	13	30	60	0	212	25
5	7	7	32	89	9	76	24	5	90	61
6	24	1	18	62	18	17	14	3	160	83
7	34	8	41	48	19	23	67	10	134	16
8	2	1	9	95	12	30	73	3	110	65
9	8	32	8	47	8	30	97	0	152	18

True\Pred	0	1	2	3	4	5	6	7	8	9
0	1	0	14	17	1	5	2	5	38	17
1	9	0	0	11	1	0	51	0	12	16
2	7	0	3	19	9	2	11	2	26	21
3	0	1	2	20	1	7	7	0	13	49
4	4	1	4	8	2	10	17	0	49	5
5	1	0	16	17	2	21	5	2	16	20
6	6	0	3	18	5	7	5	2	40	14
7	7	5	2	14	5	8	10	2	40	7
8	0	0	1	21	7	8	17	2	29	15
9	1	10	1	8	3	11	25	0	36	5

Figure 1a: *Confusion matrices of the untrained model on train set (left) and test set (right), with the vertical axis as ground truth*

True/Pred	0	1	2	3	4	5	6	7	8	9
0	388	0	0	1	0	2	3	0	6	0
1	0	390	3	1	0	0	4	0	2	0
2	4	1	365	2	4	3	7	5	8	1
3	0	0	10	356	1	9	2	5	10	7
4	0	1	1	1	365	0	5	5	8	14
5	4	2	3	14	9	340	5	4	16	3
6	9	2	3	0	5	3	375	0	2	1
7	0	5	10	1	5	0	0	360	5	14
8	5	3	7	7	4	16	4	5	346	3
9	1	6	1	9	16	6	3	8	10	340

True/Pred	0	1	2	3	4	5	6	7	8	9
0	96	0	1	1	0	2	0	0	0	0
1	0	95	0	3	1	0	0	0	1	0
2	0	4	74	4	6	0	1	5	5	1
3	1	1	3	84	1	1	0	3	6	0
4	0	2	1	0	83	1	5	0	1	7
5	1	0	0	4	0	85	0	2	5	3
6	2	1	2	0	0	3	90	1	1	0
7	0	4	1	0	4	0	0	87	1	3
8	1	3	1	3	2	3	2	3	81	1
9	1	1	0	1	6	3	0	3	3	82

Figure 1b: *Confusion matrices of the trained model on train set (left) and test set (right), with the vertical axis as ground truth*

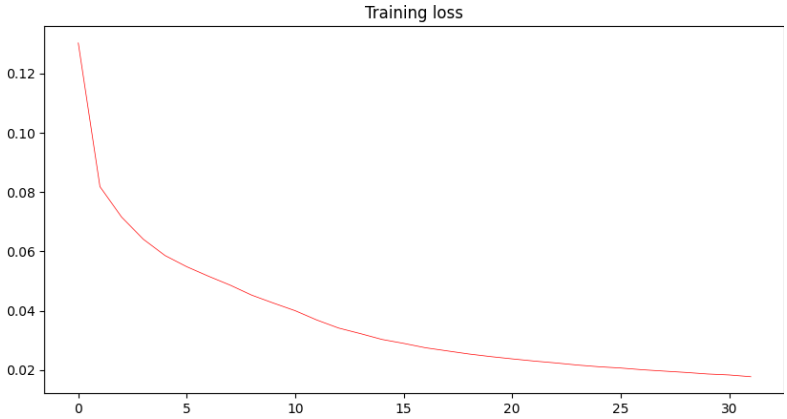

Figure 1c*: Loss curve of the model with best hyperparameters*

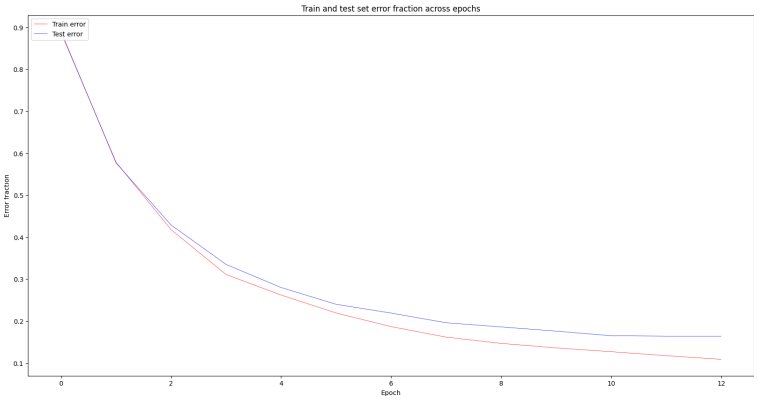

Figure 1d*: Error fraction for train set and test set curve of the model with best hyperparameters*

Discussion: The loss while training did plateau (so did the error fractions), and the model early stopped, therefore for the configuration we are allowed, we have optimized the model's performance on this train set, before overfitting the test set.

Problem 2:

Problem Statement: An autoencoder was trained to reconstruct 4000 images of digits from the MNIST dataset. The model has one hidden layer of 200 neurons, trained with stochastic gradient descent and mean squared error loss, and has the same output size as input size..

System Specification:

The hyperparameters were tested and evaluated as below, and the unchanged hyper-parameter set up was the same as problem 1. As it was not allowed to change the number of neurons in the hidden layers, the only hyper-param that is toggled was learning rate and epochs

- *Exp0-1*: Beginning training with random default hyper-params. After training for 40 epochs and not getting good reconstruction output, I trained for 210 epochs, and got really good results. It still did not early-stop, so I trained for a further 400 epochs, for a total of 610 epochs.

{'hidden_neuron': 200, 'lr': 0.01, 'momentum': 0.5, 'num_epochs': 210},
=> Test loss: 0.03674194261557957, 'train loss: 0.03477365953857899

- *Exp2*: I saw that in exp0-1 the training plateaued with really small decrease in loss, so I decrease learning rate, hoping it will better optimize, but it failed to beat exp0-1 within the first 410. Its loss has also plateaued, so I doubt it can have any significant improvement compared to exp0-1
- { 'hidden_neuron': 200, 'lr': 0.0025, 'momentum': 0.5, 'num_epochs': 810},

=> Training loss: 0.06394220283596991, testing lost: 0.06595196121177048

⇒ The set of hyper-parameters from exp0-1 provides the maximum model performance in terms of test loss for both train and test set.

Results:

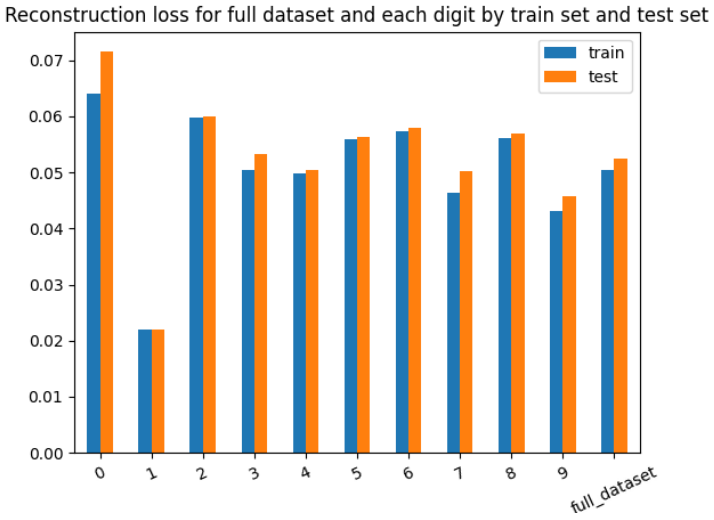

Figure 2a: *Reconstruction loss on the full dataset and each digit in both test and train set after training*

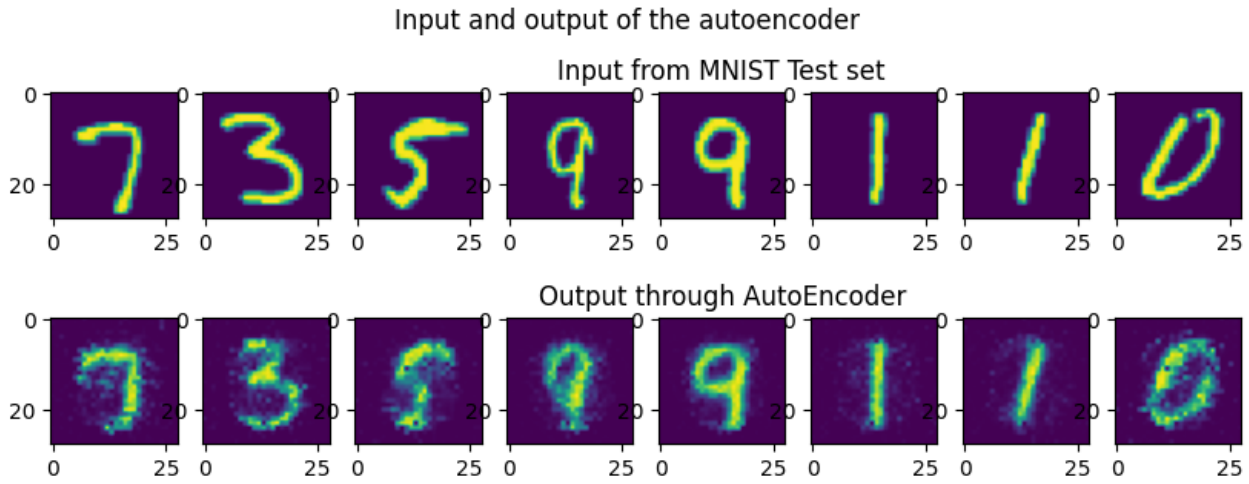

Figure 2b: *Random 8 samples and its reconstructed output of the autoencoder*

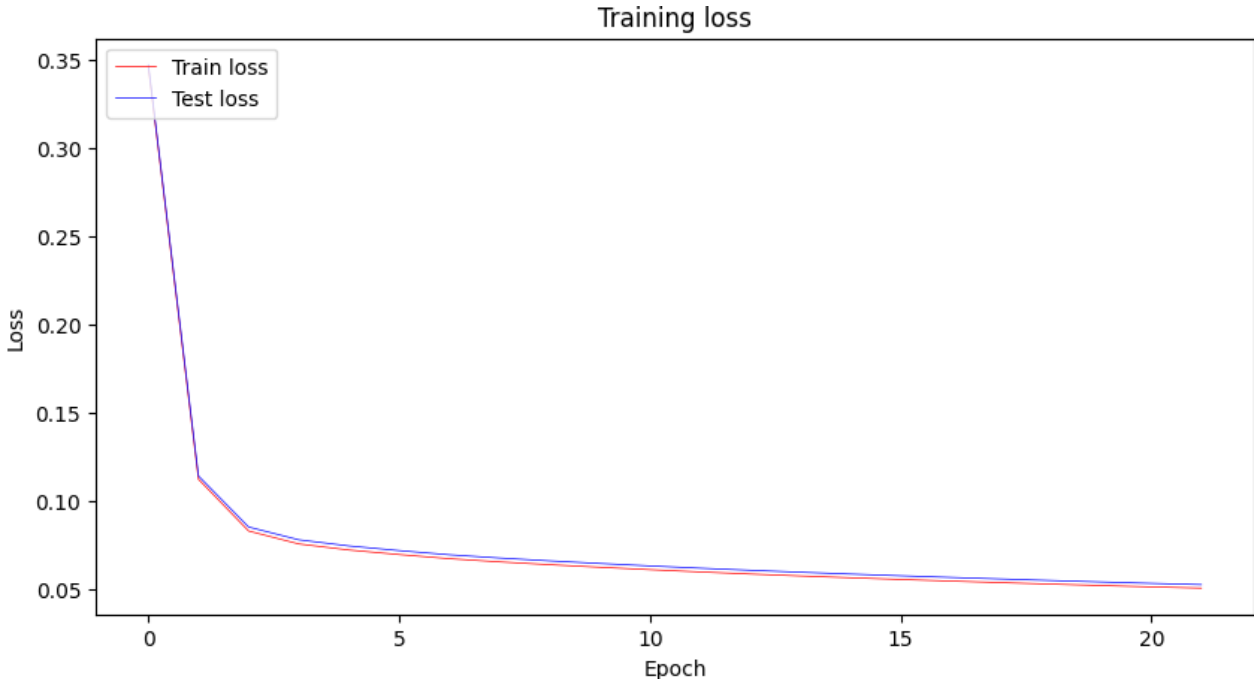

Figure 2c*: Loss during the training process by epochs, tested on the train and test set at every 10th epoch*

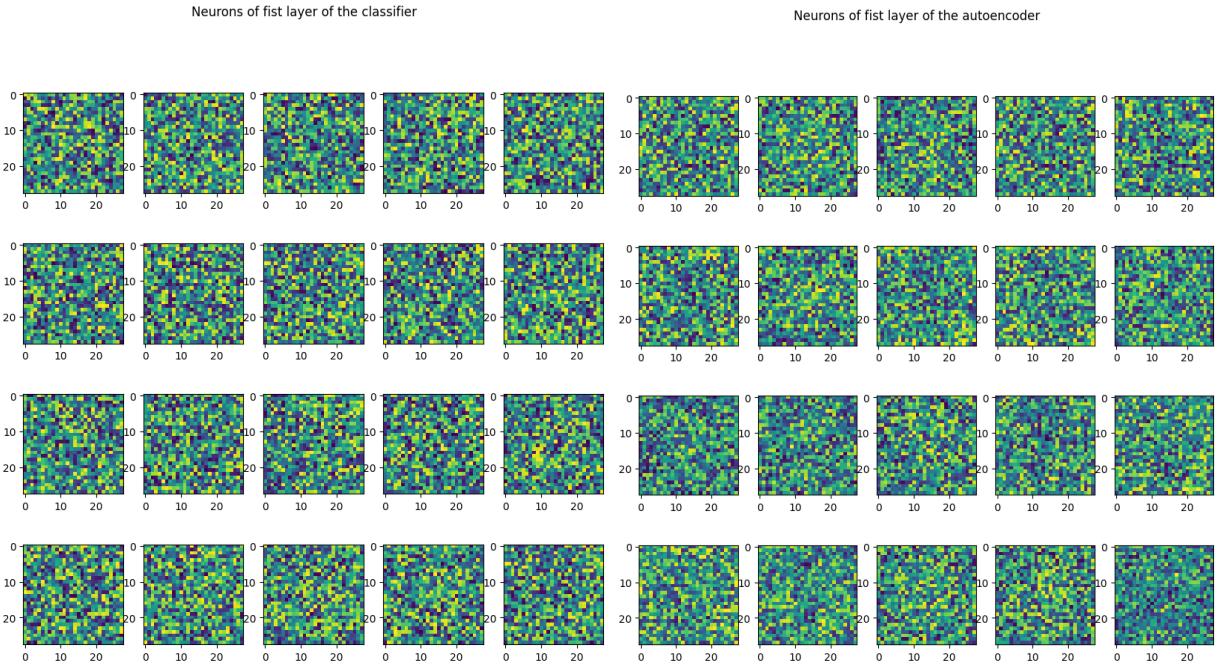

Figure 2d*: 20 random neuron visualized, from the first weight matrix (fully-connected) from Problem1's classifier (left) and Problem2's autoencoder (right)*

Discussion: The loss curves on both train and test set both plateaued and match each other really well (fig. 2c), which suggests that the model generalized really well on all data, and we have chosen a train set and a test set with similar distribution. Since the training plateaued, it suggests that although the model reconstructed the samples fairly well (fig. 2b), this was the best we could do with 1 hidden layer of 200 neurons. A larger hidden layer (Or more hidden layers) will allow more information to be compressed, and thus better reconstruction. We see that the number 1 is the easiest to reconstruct, as seen in its loss is comparatively the least (fig. 2a), because 1s have a very distinct shape, while others can easily be mistaken for other numbers, and thus require the model to have learned more (has more learning capacity, be trained longer,...). I surprisingly was not able to randomly find distinct meaningful weight representations in both problems (fig. 2d).