EE 689 Deep Learning Neural Networks Assignment 2

Oğuzhan Sevim
December 4, 2020

I. INTRODUCTION

In this assignment, different neural network (NN) structures are trained for modeling the MNIST dataset which consists of hand-written digits with labels. From simplest to relatively more complex structures (with at least one convolutional layer), different network structures are experimented.

The MNIST dataset is directly obtained from the Keras, and it consists of 70000 image samples where dimension of each sample is 28×28. This dataset is portioned into training and test sets by $60000 - 10000$, respectively.

Implementations are done by using Keras, which is a high level NN API of Python, on the Jupyter Notebook.

II. TASK 1: A LINEAR MODEL

In this section, we use a simple NN model with 2 layers. First layer feeds the weighted sum of input values to the output layer. This means that we have a linear classifier. In order to feed the $2D$ data into the network, each sample is reshaped as a vector. Also, SGD optimizer is used for in this part.

Weights are updated by SGD with learning rate 0.01. The relation between the epoch number and the accuracies (training&test) is shown in Figure 1.

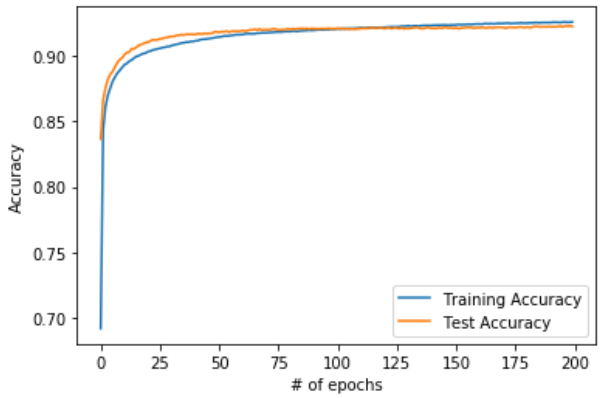

Fig. 1. (Task 1) Training and test accuracies vs. epochs. Each accuracies reach above 92.2%

Figure 1 also illustrates how the model overfits as the number of epochs increases. When the network is trained with 200 epochs, classification accuracy of test data is aroung 92.2%. However, after 100 epochs, there is almost no improvement in the classification accuracy. Instead, the network overfits more to the training data when epoch number exceeds 100 (roughly).

The confusion matrix C of the test results is calculated by confusion_matrix() function. C matrix is given below, where

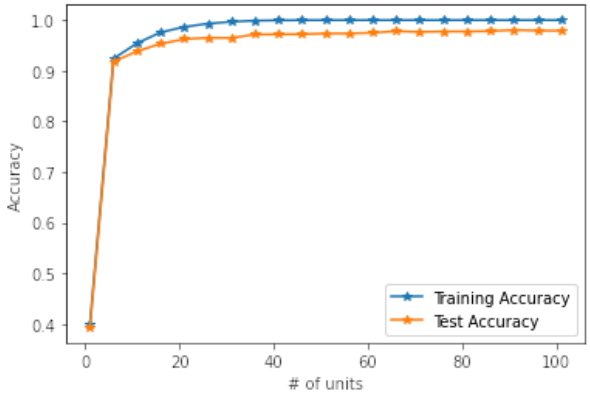

Fig. 2. Training and test accuracies (after 100 epoch) vs. Number of units in hidden layer (Task 2)

C_{ij} element denotes the number of outputs with true class i and categorized as class j:

$$\begin{bmatrix} 960 & 0 & 5 & 3 & 1 & 10 & 12 & 1 & 7 & 10 \\ 0 & 1110 & 8 & 1 & 2 & 3 & 3 & 6 & 7 & 7 \\ 2 & 2 & 924 & 22 & 5 & 3 & 5 & 21 & 7 & 2 \\ 2 & 2 & 15 & 919 & 1 & 33 & 1 & 10 & 24 & 11 \\ 0 & 0 & 11 & 0 & 913 & 9 & 9 & 7 & 9 & 31 \\ 4 & 1 & 6 & 28 & 0 & 772 & 11 & 1 & 26 & 7 \\ 9 & 4 & 11 & 1 & 12 & 18 & 912 & 0 & 10 & 0 \\ 1 & 2 & 10 & 11 & 2 & 7 & 3 & 947 & 12 & 26 \\ 2 & 14 & 36 & 17 & 9 & 30 & 2 & 2 & 864 & 7 \\ 0 & 0 & 6 & 8 & 37 & 7 & 0 & 33 & 8 & 908 \end{bmatrix}.$$

Here, we can see that one of the mostly confused digits are 4 and 9. Digit 9 is misclassified as 4 for 37 times, where the inverse confusion happened 31 times.

III. TASK 2: A DENSE NN MODEL

In this section, we train NNs with at least 1 hidden layer. First, our model is built a single hidden layer. We train different models for 100 epochs with different number of units in the hidden layer. The change of test accuracy vs the number of units is shown in Figure 2. For 1 and 2 hidden layer models, ADADELTA optimizer is used. However, for 3 hidden layers it didn't work well. Instead, SGD is used.

For the single hidden layer structure, training accuracy can reach 100% (with higher than 98% test accuracies) when the number of hidden layer units are higher than 80. Different networks with different number of hidden layers and units are

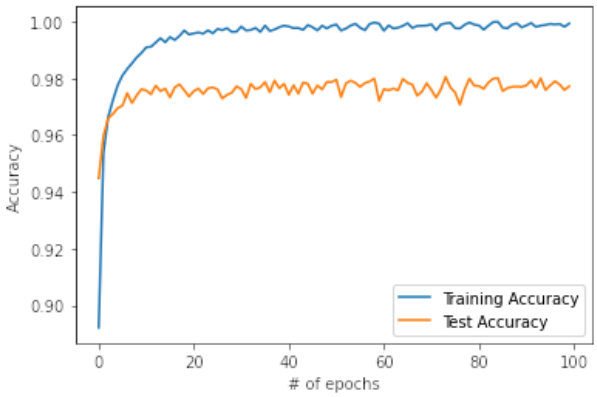

Fig. 3. Training and test accuracies vs. Epoch numbers of the NN model with $80-60-40$ units in hidden layers (Task 2)

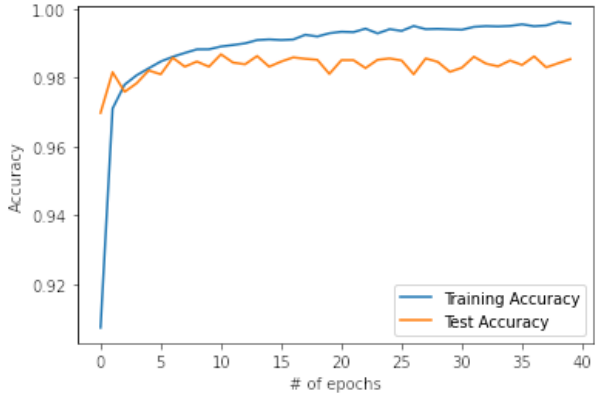

Fig. 4. Training and test accuracies vs. Epoch numbers of the CNN model given in the last row of Table II (Task 3)

experimented. The results are given in Table I where each element of the first column represents the number of hidden layers from left to right.

TABLE I
TEST ACCURACY AND PARAMETER NUMBERS FOR DIFFERENT MODELS

Unit numbers	Test Accuracy	Parameter number
1	0.39	805
5	0.9	3985
10	0.93	7960
1,10	0.43	915
10,5	0.93	7965
20,15,10	0.96	16175
40,30,20	0.973	33460
80,60,40	0.980	70510

For a good accuracy (e.g., 98.0%), a model with 3 hidden layers and $80-60-40$ units can be used. This model's accuracy change is shown in Figure 3. Around epoch 40, model seems to overfit. Also, there are some important interpretations from Table I. Such as:

- If the first hidden layer is too narrow, it acts like a bottleneck and lots of information get lost. Having relatively wider layers afterwards does not improve the model any more.
- From first hidden layer to the output layer, decreasing the number of units systematically is a good idea.
- Increasing the number of model parameters has a diminishing marginal return.

IV. Task 3: Adding Convolutional Layers

In this section, we experiment networks models with different convolutional layer structures. We, train the model with ADAM optimizer. The optimizer learning rate is chosen as 0.01 and $\beta_1 = 0.9, \beta_2 = 0.999$. We train the models with 40 epochs. Accuracies and total number of parameters are given in Table II. For more than 98% accuracy, we may choose the model in the last row of Table II such that there will be a convolutional layer with 5 4×4 kernels, 2×2 pooling, and 2 dense layers with $40-30$ units. When this model is trained for 30 epochs, we get the learning curve as shown in Figure 4. In 10 epochs, test accuracy becomes slightly higher than 98% where further training causes more overfitting.

TABLE II
TEST ACCURACY AND PARAMETER NUMBERS FOR DIFFERENT MODELS

Ker. #	Ker. size	P. size	Hid. Units	Accur.	Param. #
1	**2**	6	5	0.791	150
1	**4 (best)**	**6**	5	0.833	162
1	**8**	6	5	0.792	175
1	4	**2 (best)**	5	0.871	802
1	4	**4**	5	0.852	262
3	4	2	5	0.944	2276
5 (best)	4	2	5	0.965	3750
5	4	2	10	0.976	7405
5	4	2	20,20	0.981	15135
5	4	2	40,30	0.985	24724

V. Task 4: Visualization of Kernels of First Convolutional Layers

In the final task, we visualize the evolution of the kernels. Since they are initialized randomly, they don't give any information without training. However, each kernel converges a significant pattern by every epoch. In Figures 5 and 6, evolution of 2 kernels are shown. From left to right, they shows the kernels after epochs $0-10-100-1000-10000$.

Fig. 5. Evolution of Kernel 3 (Task 4).

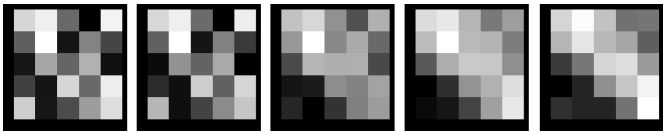

Fig. 6. Evolution of Kernel 4 (Task 4).

Here, we can see that Kernel 3 is evolved in such a way that it detects horizontal edges. Likewise, Kernel 4, filters the changes from the lower left corner to the diagonal. We observe from the Figures 5 and 6 that kernels converge to the more specific filters as we train the model further.