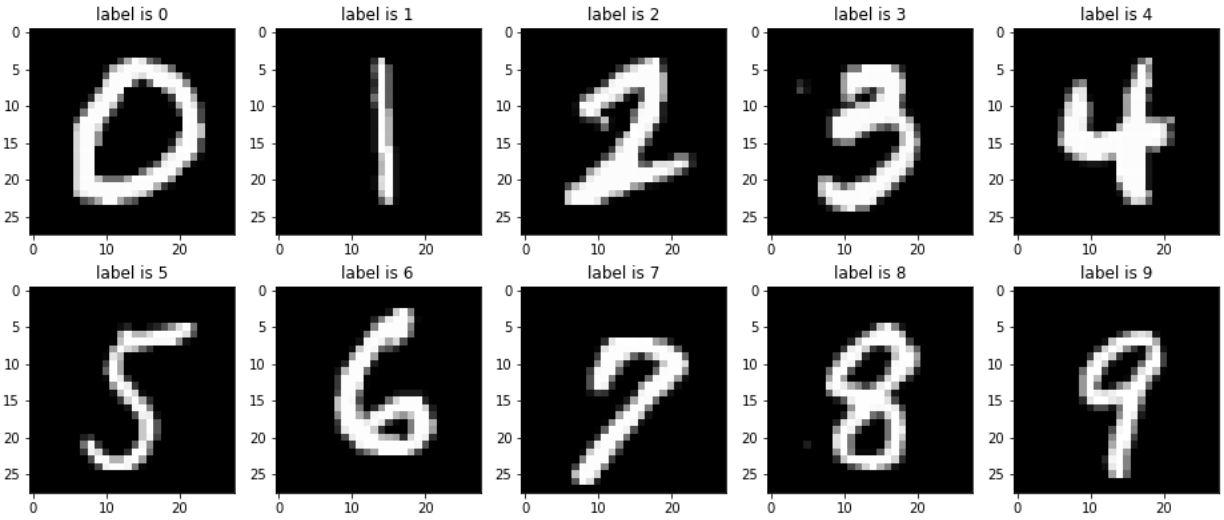
COMP9517 – Deep Learning
Deep Learning Homework
Week 9, T3 2020

The goal of this homework is to become familiar with the training/testing process for deep learning. You need to implement a CNN network using a small high resolution MNIST dataset.

Note: This is a homework for self-learning only, and you do not need to submit your implementations.

MNIST dataset

Our dataset contains 2000 MNIST images as training data and 500 MNIST images as testing data. All the images are high resolution (28 x 28) with corresponding labels ranging from 0 to 9, and were designed to test classification algorithms.



Sample of 10 different digits images and their corresponding labels.

Lab Task – Build a CNN network for image classification

Develop a program to perform digit recognition by using a deep learning method. To classify the digits using your own CNN network, you also need to complete the main function for training and testing processes. The program has two versions: PyTorch and Keras. You may complete any one of them. The steps as follows:

Set Up

1. Load MNIST data set

```
# Load data(do not change)
data = pd.read_csv("src/mnist_train.csv")
train_data = data[:2000]
test_data = data[2000:2500]
```

Preprocessing data

2. Data Normalization: which can make your network more efficient
3. Transform data into PyTorch tensor(if you use PyTorch)

Build your own CNN network

4. Complete your CNN network
5. Define your hyperparameters, e.g. learning rate
6. Define your optimizer and loss function(criterion).

Complete PlotLearningCurve function

7. You need to complete PlotLearningCurve function, the template result sees below.

Main Process for training and testing

8. Define the number of iterations (50 ~ 100 is enough for PyTorch, 20 ~ 50 is enough for Keras)
9. Fit training data to your model
10. Model evaluation by using testing data
11. Plot learning curve

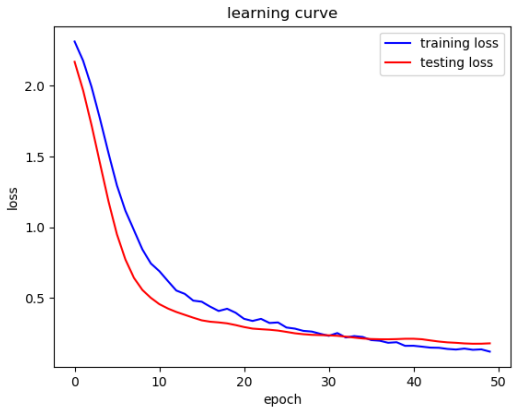
If you use PyTorch, you need complete below part before you do model evaluation:

1. Calculate your loss
2. Calculate backpropagation
3. Update network parameters
4. Reset your optimizer
5. During training, you need to store your loss of training error and testing error

You are encouraged to experiment with different hyperparameters, e.g. different learning rates can show different results, a suitable learning rate can make your model learning more efficiently and effectively.

Task one:

Plot the learning curve, the number of epochs is not important. In the learning curve, you can see whether your model is underfitting or overfitting. The template result is below:



Task two:

1. For PyTorch version:

Print the testing accuracy every **10 epochs**. **Your final test accuracy needs to be great than 90%**.

An example can be seen here:

```
Epoch: 10 test Accuracy: 86.4
Epoch: 20 test Accuracy: 94.2
Epoch: 30 test Accuracy: 94.2
Epoch: 40 test Accuracy: 96.0
Epoch: 50 test Accuracy: 95.8
Epoch: 60 test Accuracy: 96.0
Epoch: 70 test Accuracy: 96.0
Epoch: 80 test Accuracy: 97.4
Epoch: 90 test Accuracy: 97.2
Epoch: 100 test Accuracy: 97.2
```

2. For Keras version:

Use "model.evaluate" to obtain your final testing accuracy. **Your final test accuracy needs to be great than 90%**.

An example can be seen here:

```
Epoch 46/50
2000/2000 [==============================] - 2s 1ms/step - loss: 0.1338 - accuracy: 0.9620 - val_loss: 0.1815 - val_accuracy: 0.9520
Epoch 47/50
2000/2000 [==============================] - 2s 1ms/step - loss: 0.1406 - accuracy: 0.9560 - val_loss: 0.1771 - val_accuracy: 0.9560
Epoch 48/50
2000/2000 [==============================] - 2s 1ms/step - loss: 0.1320 - accuracy: 0.9600 - val_loss: 0.1748 - val_accuracy: 0.9560
Epoch 49/50
2000/2000 [==============================] - 2s 1ms/step - loss: 0.1350 - accuracy: 0.9570 - val_loss: 0.1751 - val_accuracy: 0.9580
Epoch 50/50
2000/2000 [==============================] - 2s 1ms/step - loss: 0.1191 - accuracy: 0.9655 - val_loss: 0.1776 - val_accuracy: 0.9560
final test accuracy [0.1776074873805046, 0.9559999704360962]
```

Reference

Keras: https://www.datacamp.com/community/tutorials/deep-learning-python

Pytorch: https://nextjournal.com/gkoehler/pytorch-mnist