# COMP9517 – Deep Learning
Lab 5, T3 2019

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

Code and results should be submitted via WebCMS3 for marking (1 mark).

Submission is due at 23:59:59 on November 1st, 2019. Submission instructions will be posted prior to the lab.

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.
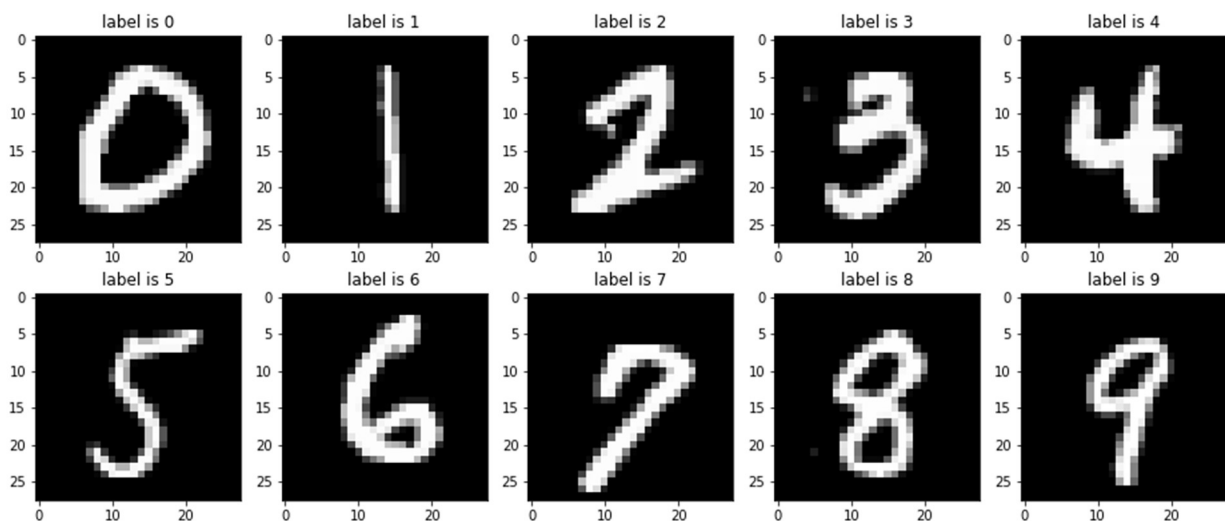
Sample of 10 different digits images and their corresponding labels.

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 only need to complete one of them for final submission. 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 evalutation:
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.

You only need to submit **one** report for either PyTorch or Keras. You need to tell us which version you used, in your report. The report needs to include two tasks, see below:

Task one:
You need to **plot learning curve**, the number of epochs is not important, in learning curve, you can see your model whether is underfitting or overfitting. You need to submit a screenshot for marking. The template result sees below:
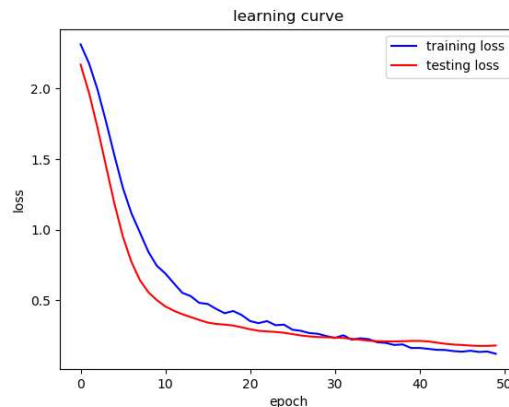
Task two:
1. For PyTorch version:
Print the testing accuracy every **10 epochs**, and submit a screenshot for marking. **Your final test accuracy needs to be great than 90%**.
An example can be seen here:

2. For Keras version:

Please use "model.evaluate" to get your final testing accuracy, and submit a screenshot for marking. **Your final test accuracy needs to be great than 90%**.
An example can be seen here:

Reference
Keras: https://keras.io/examples/mnist_cnn/
Pytorch: https://nextjournal.com/gkoehler/pytorch-mnist