

# **INFO1111: Computing 1A Professionalism**

## **2021 Semester 1**

### **Self-Learning Assignment**

**Tool: Pytorch**

**Domain: Data science**

**Domain of Application: Deep learning**

**Level 1**

## Table of Contents

<b>1. Introduction.....</b>	<b>3</b>
<b>2. Learnings.....</b>	<b>4</b>
2.1. Outcomes for the level.....	4
2.2. What I Learned .....	4
2.3. Application of your Learning .....	5
2.4. Artifact .....	6
<b>3. Resources.....</b>	<b>7</b>
<b>4. Process .....</b>	<b>8</b>
<b>5. References .....</b>	<b>9</b>

## 1. Introduction

PyTorch is an open source machine learning library developed by Facebook . It has a powerful tensor computing capability and covers many of the algorithms and mathematical functions needed in neural networks. PyTorch is a deep learning framework based on torch, and its underlying framework is identical to that of torch. However, it has been rewritten in python to make it more flexible, supporting not only dynamic graphs but also GPU acceleration. Compared to other frameworks PyTorch is a fairly clean and efficient framework, it uses less packaging and is designed in python to be more human friendly.

## 2. Learnings

### 2.1. Outcomes for the level

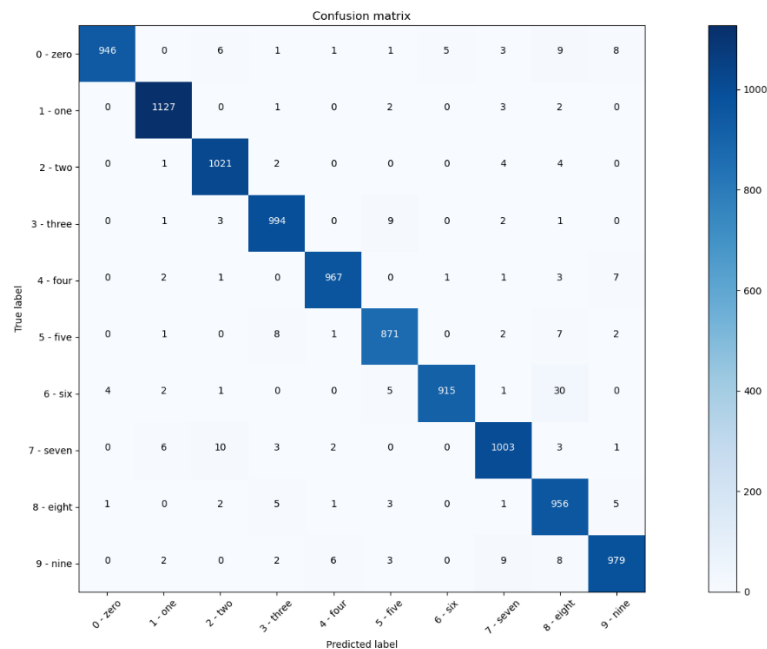
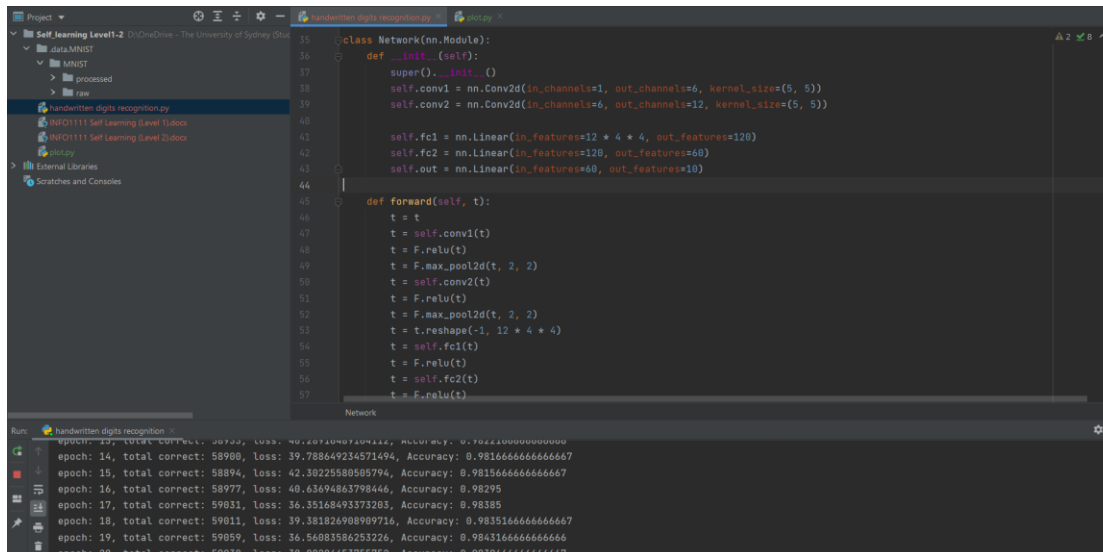
	Outcome
Level 1 - Basic Application	Be able to create an Implementation of handwritten digits recognition.

### 2.2. What I Learned

At this level, I learned the basic understanding of neural network (including what is perceptron, activation function and multidimensional array operation etc.) and basic use of Pytorch, which contains comprehending the concept of tensor and some fundamental syntax of torch etc. By learning the basic concept of the neural network, I learned to build the basic structure of a neural network including an input layer, output layer and hidden layer. In addition, the sigmoid activation function and ReLU activation function are widely used in the field of deep learning. The multidimensional matrix operations also play a crucial role in neural networks, so understanding multidimensional array operations are the basis for learning neural networks.

Tensor is an important concept in learning PyTorch, it is essentially a multidimensional array, the purpose of which is to create higher dimensional matrices and vectors. For example, in a convolutional neural network (CNN) we usually use a four-dimensional tensor, and each dimension is represented as batches, channels, heights, widths. Besides, I learned the basic syntax of Pytorch, including how to implement tensors operation, create a tensor, call datasets by using the "torchvision" package, and build a basic neural network.

## 2.3. Application of your Learning

```

class Network(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6, kernel_size=(5, 5))
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=12, kernel_size=(5, 5))

        self.fc1 = nn.Linear(in_features=12 * 4 * 4, out_features=120)
        self.fc2 = nn.Linear(in_features=120, out_features=60)
        self.out = nn.Linear(in_features=60, out_features=10)

    def forward(self, t):
        t = t
        t = self.conv1(t)
        t = F.relu(t)
        t = F.max_pool2d(t, 2, 2)
        t = self.conv2(t)
        t = F.relu(t)
        t = F.max_pool2d(t, 2, 2)
        t = t.reshape(-1, 12 * 4 * 4)
        t = self.fc1(t)
        t = F.relu(t)
        t = self.fc2(t)
        t = F.relu(t)

```

Run: handwritten digits recognition

epoch: 14, total correct: 58908, loss: 39.788649234571494, Accuracy: 0.9816666666666667

epoch: 15, total correct: 58894, loss: 42.3022588059794, Accuracy: 0.9815666666666667

epoch: 16, total correct: 58977, loss: 40.63494863798446, Accuracy: 0.98295

epoch: 17, total correct: 59031, loss: 36.35168493373203, Accuracy: 0.98385

epoch: 18, total correct: 59011, loss: 39.381826908909716, Accuracy: 0.9835166666666667

epoch: 19, total correct: 59059, loss: 36.56083586253226, Accuracy: 0.9843166666666666

epoch: 20, total correct: 59035, loss: 38.08204653755752, Accuracy: 0.9839666666666667

I used Pytorch to create a standard convolutional neural network for handwritten digit recognition, and I improved the recognition accuracy by training the neural network. In this second screenshot, you can see that by using the training model built with PyTorch, the recognition accuracy and loss values continue to decrease, while the number of corrects increase progressively. The first screenshot shows that the confusion matrix has the shape of a diagonal matrix, which is a sign of high accuracy.

## 2.4. Artifact

This artifact is a basic implementation of handwritten digits recognition. The implementation of this artifact is divided into four major parts, prepare the data, building the model (neural network), training the neural network and testing the training results (visualization). The first step is preparing the data, I will need to extract the data which getting the MNIST data from the database by using `<torchvision.datasets.MNIST>`. Then I should transform the data as a tensor format, and finally, I load the data into an object by using `<torch.utils.data.DataLoader>`. This is called ETL which is a type of data integration. The second step is building the neural network. In this model, I use 5 hidden layers which including two convolution layers and three full-connected layers. In addition, in the forward propagation, I use the Rectified Linear Unit function (ReLU) as an activation function and I execute twice pooling to reduce the size of the feature map (both pooling and ReLU are the functions in `<torch.nn.functional>`). The next step is training the neural network. In this step, I use the forward propagation method that just constructed in the neural network to obtain the predicted value, and then I use the cross-entropy error loss function to get the loss between the predicted value and the correct value. The gradient is then obtained by backward propagation and the weights of the model are updated using the Adam optimization algorithm (which is in `<torch.optim>` package). Finally, I plotted the confusion matrix to visualize the accuracy of the neural network after being trained.

### 3. Resources

- **Resource:**
  - <https://www.youtube.com/channel/UC4UJ26WkceqONNF5S26OiVw>
  - Book: Deep Learning from Scratch: Building with Python from First Principles.
  - <https://pytorch.org/tutorials/>
  - <https://deeplizard.com/>
- **Overview of resource:**
  - Youtube: basic method of using PyTorch
  - Printed book: Theory and implementation of deep learning
  - Website: searching syntax and the exercise of PyTorch
- **Use of Resource:**
  - Learn how to implement some underlying theories of neural networks based on python through printed books. Then learn how to use PyTorch through YouTube, and deepen the understanding of PyTorch with some web search.

## 4. Process

1. Using the printed book to learn the underlying theories of deep learning (including neural network, backward propagation, mathematical theory, implementation of layers, and updating algorithm etc.)
2. Learn how to use PyTorch through YouTube tutorials (some basic syntax)
3. Deepen my understanding of PyTorch syntax and neural networks by using Google or looking up the official PyTorch website



## 5. References

PyTorch Prerequisites - Syllabus for Neural Network Programming Course - YouTube. (n.d.). Retrieved May 6, 2021, from [https://www.youtube.com/watch?v=v5cngxo4mlg&list=PLZbbT5o\\_s2xrfNyHZsM6ufI0iZENK9xgG&index=1&t=285s](https://www.youtube.com/watch?v=v5cngxo4mlg&list=PLZbbT5o_s2xrfNyHZsM6ufI0iZENK9xgG&index=1&t=285s)

deeplizard - Building Collective Intelligence. (n.d.). Retrieved May 6, 2021, from <https://deeplizard.com/>

Welcome to PyTorch Tutorials — PyTorch Tutorials 1.8.1+cu102 documentation. (n.d.). Retrieved May 6, 2021, from <https://pytorch.org/tutorials/>

Weidman, S. (2019). *Deep Learning from Scratch: Building with Python from First Principles*. O'Reilly Media