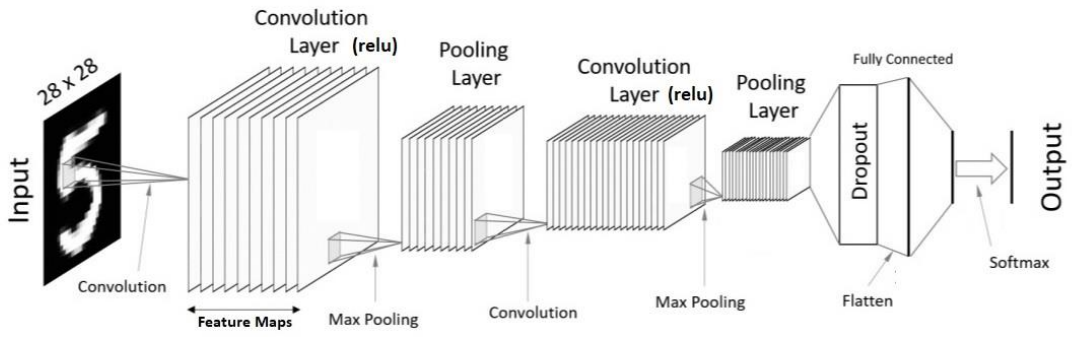
OASIS ML Group TRAINING 06

◆ CNN using PyTorch

In this practice, you are going to build an CNN model by using a useful framework called PyTorch. Please refer to the website which in "Reference" part for some useful trick. MNIST datasets is a common dataset for CNN. The original dataset is in a format that is difficult for beginners to use, so which I provided is used the work of Joseph Redmon to provide the MNIST dataset in a CSV format. **Shout out to Joseph Redmon!** Follow hints below to finish it.



▫ Practice：

■ Analysis datasets：

The dataset consists of two files：mnist_train.csv, mnist_test.csv

The mnist_train.csv file contains the 60,000 training examples and labels. The mnist_test.csv contains 10,000 test examples and labels. Each row consists of 785 values: the first value is the label (a number from 0 to 9) and the remaining 784 values are the pixel values (a number from 0 to 255).

■ PyTorch：

- Please try to answer what's PyTorch?
- Why we want to use it instead of using Python, Numpy …only.
- After finish the practice, what you feel compare with training 05?

Hint 01：Import library you needed.

For example： *(Below are **frequently-used** libraries)*

```
import os
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.autograd import Variable
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Hint 02：Use pandas to read data. After that, I suggest to do datasets information analysis which show below.

```
Info of training dataset :
# of training samples: 60000, Shape: (60000, 785)
# of training pixels : 60000, Shape: (60000, 784)
# of classes : 10
shape of labels : 60000, Shape: (60000,)

Info of validation dataset :
# of training samples: 10000, Shape: (10000, 785)
# of training pixels : 10000, Shape: (10000, 784)
shape of label : 10000, Shape: (10000,)
```

For hint 3 and hint 4, you are asked to visualize the datasets.

Hint 03：Display some images in training datasets through code provided below. In the meantime, try to understand the code.

```
from torchvision.utils import make_grid

random_sel = np.random.randint(len(train), size=8)
print('random_sel index =',random_sel)

grid = make_grid(torch.Tensor((train.iloc[random_sel, 1:].values/255.).reshape((-1, 28, 28))).unsqueeze(1), nrow=8)
plt.rcParams['figure.figsize'] = (16, 2)
plt.imshow(grid.numpy().transpose((1,2,0)))
plt.axis('off')
print(*list(train.iloc[random_sel, 0].values), sep = ', ')
```

Hint 04：Show the **histogram** of the classes through code provided below.

```
plt.rcParams['figure.figsize'] = (8, 5)
plt.bar(train['label'].value_counts().index, train['label'].value_counts())
plt.xticks(np.arange(len(set(train["label"]))))
plt.xlabel('Class', fontsize=16)
plt.ylabel('Count', fontsize=16)
plt.grid('on', axis='y')
```

Hint 05：Doing **normalization** through code provided below.

- Discuss why we need to do normalize?
- Discuss why we need to do ".astype('float32')"?
- Discuss why divide 255?

```
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
```

Hint 06：Split train datasets to "training" and "validation" by using train_test_split in sklearn. Remember to define test_size and random_state.

(Recommend test_size = 0.2~0.3)

Hint 07： Use code below to convert datasets to Tensor and package into TensorDataset. Discuss why and what the code means~

(Remind：You will have three kinds of datasets after this step.)

For example：

```
ImgTrain = torch.from_numpy(img_train.values)
TargetTrain = torch.from_numpy(target_train.values).type(torch.LongTensor)

train = torch.utils.data.TensorDataset(ImgTrain, TargetTrain)
```

Hint 08： Define Hyper parameters by yourself.

Recommend initial parameters：

learning_rate = 0.01

batch_size = 100

n_iters = 10000

```
# Hyper Parameters
learning_rate = ????
batch_size = ????
n_iters = ????

num_epochs = int(n_iters / (len(img_train) / batch_size) )  <--  ????
```

(You can tune by yoursels.)

Hint 09： Construct Dataloader through code below.

- Discuss what's dataloader?

(Remind：You will have three dataloaders after this step.)

For example：

```
train_loader = torch.utils.data.DataLoader(train, batch_size = batch_size, shuffle = True)
```

Hint 10：Construct CNN model.

You can refer to code which I provided in reference, please understand every step.

Hint 11：Set the optimizer and loss_func .

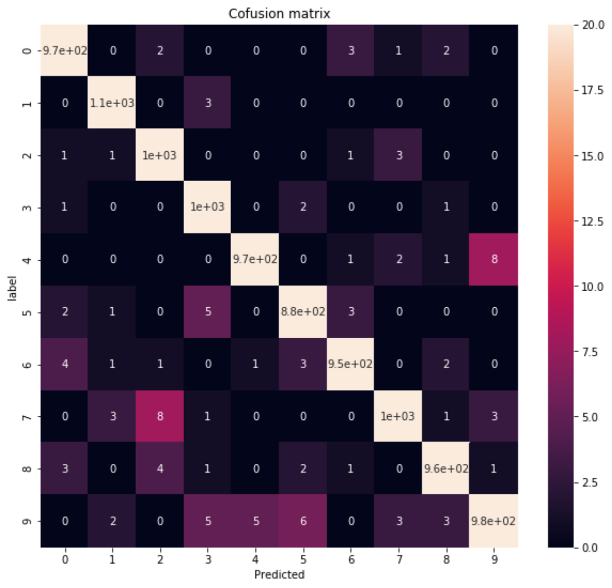
```
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
loss_func = nn.CrossEntropyLoss()
input_shape = (-1, 1, 28, 28)
```

Hint 12：Define train, evaluation, fit_model,…functions.

You can refer to code which I provided in reference, please understand every step.

Hint 13： After training, please show the testing result by using testing dataset. At the same time, please plot "Training & Validation loss curve" and "Training & Validation accuracy curve".

Hint 14： Plot the confusion matrix based on testing data and also print the accuracy of each digits.



Reference：

- PyTorch document [Link]
- PyTorch document *(Chinese)* [Link]
- The MNIST Database of handwritten digits [Original datasets]
- MNIST in CSV from Joseph Redmon [Link]
- Convolutional Neural Network (CNN) Tutorial – Kaggle [Link]
- Sample code [Link]

Reference paper： *[Link]*

- S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186

- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

- Wang, Zijie J., et al. "CNN explainer: Learning convolutional neural networks with interactive visualization." *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020): 1396-1406.