Lab01
Backpropagation and Basic Pytorch

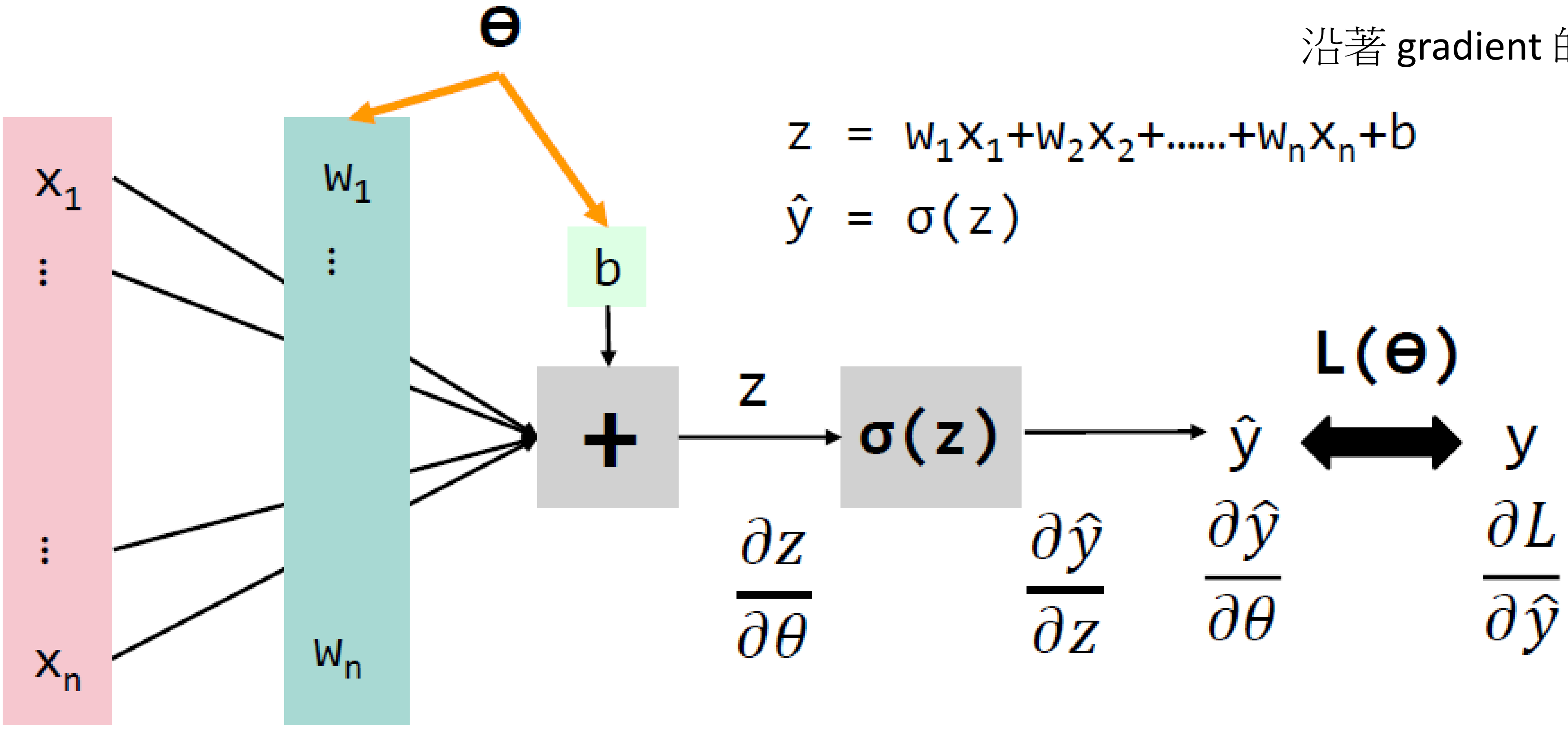
Outline

- Backpropagation
- Dataset
- Task1
- Task2
- Rules

Outline

- Backpropagation
- Dataset
- Task1
- Task2
- Rules

Back Propagation



ϴ
x_1
⋮
⋮
x_n
w_1
⋮
w_n
b
+
z
σ(z)
ŷ
L(ϴ)
y
$\frac{\partial z}{\partial \theta}$
$\frac{\partial \hat{y}}{\partial z}$
$\frac{\partial \hat{y}}{\partial \theta}$
$\frac{\partial L}{\partial \hat{y}}$

$z = w_1x_1+w_2x_2+......+w_nx_n+b$

$\hat{y} = \sigma(z)$

沿著 gradient 的反方向走 Learning rate

$$\theta = \theta - \eta \frac{\partial L}{\partial \theta}$$

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} = \boxed{\frac{\partial L}{\partial \hat{y}}}\boxed{\frac{\partial \hat{y}}{\partial z}}\frac{\partial z}{\partial \theta}$$

Chain rule (又叫 Back propagation 倒傳遞)

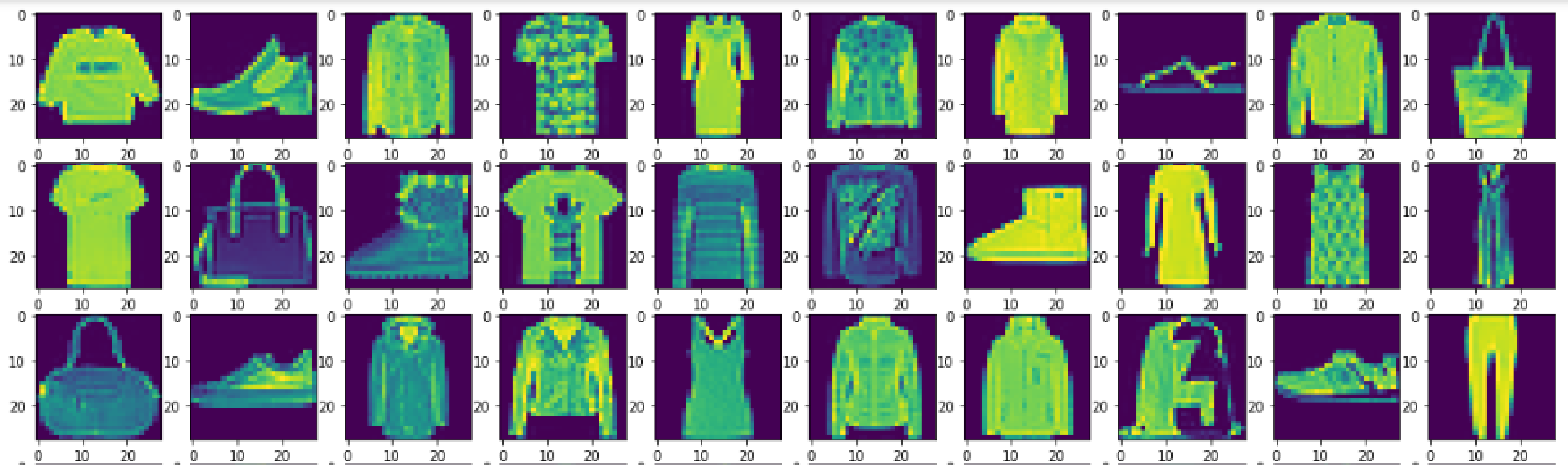
Source: Lecture slide chapter 2

Outline

- Backpropagation
- Dataset
- Task1
- Task2
- Rules

Dataset: Fashion MNIST

- Image Classification
- Classes: 10, image size =28*28 grayscale
- Training: 50,000   Validation: 10,000
- Public Testing: 200   Private Testing: 9,800



Outline

- Backpropagation
- Dataset
- Task1
- Task2
- Rules

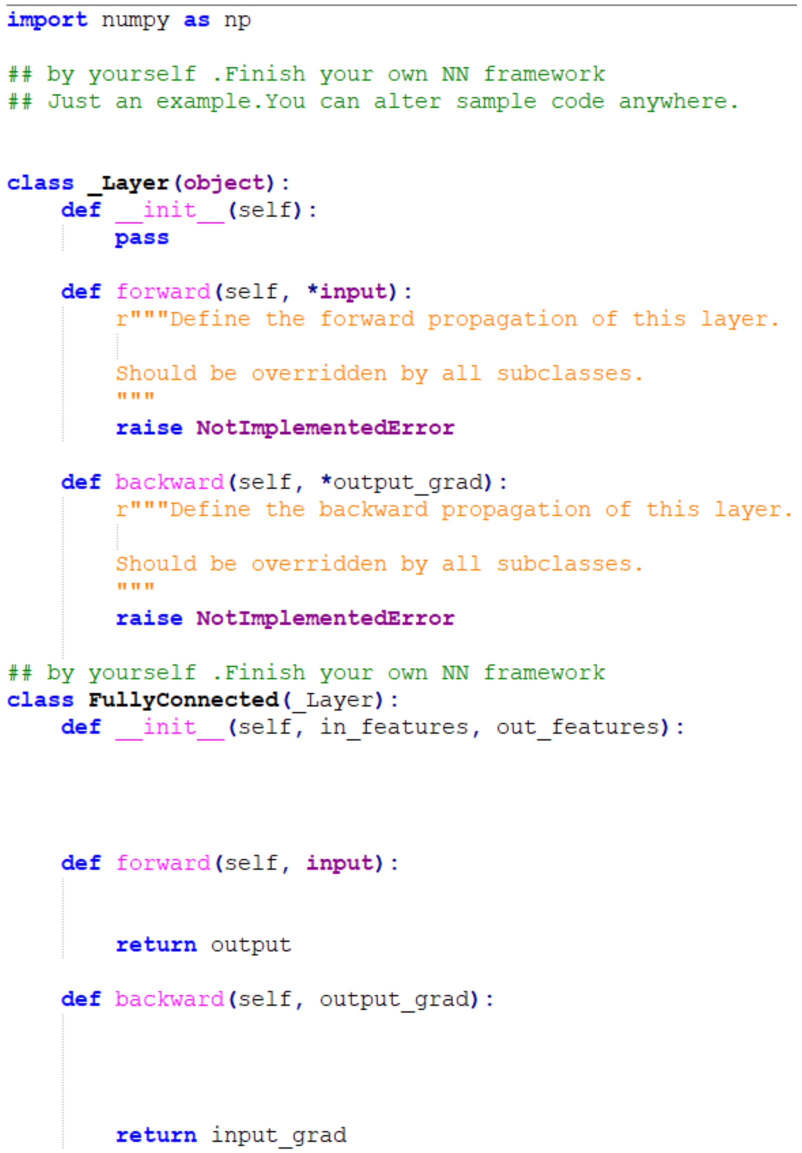
Task 1 – NN from scratch

- You need to build a neural network from scratch
  - layer.py : Define the functions of each layer (include forward & backward)
  - network.py : Build a network with layers you defined in layer.py
  - Lab1_task1.ipynb : Decide the hyperparameters and run your code
    - The settings for training have been written by TA
      - You can change the settings however you like
- For task1, you are not allowed to use deep learning frameworks.

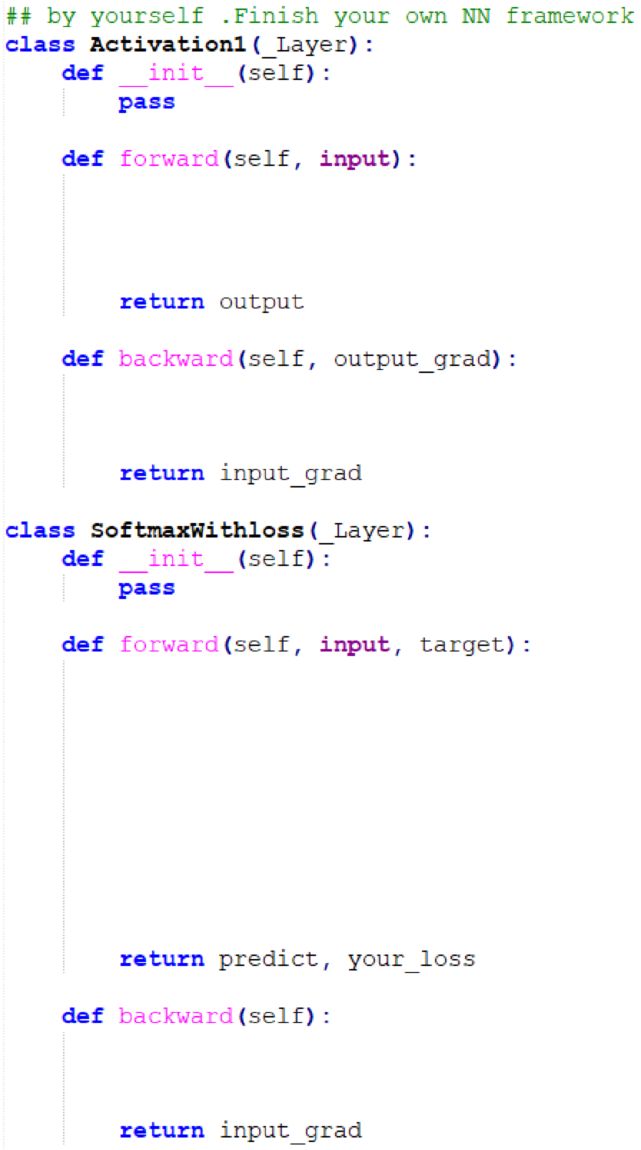
Package available in Task 1

- numpy
- pandas
- ~~Scikit-learn~~
- ~~Pytorch, tensorflow, keras~~
- ~~cupy, scipy~~
- ~~Image processing toolbox~~

model/layer.py

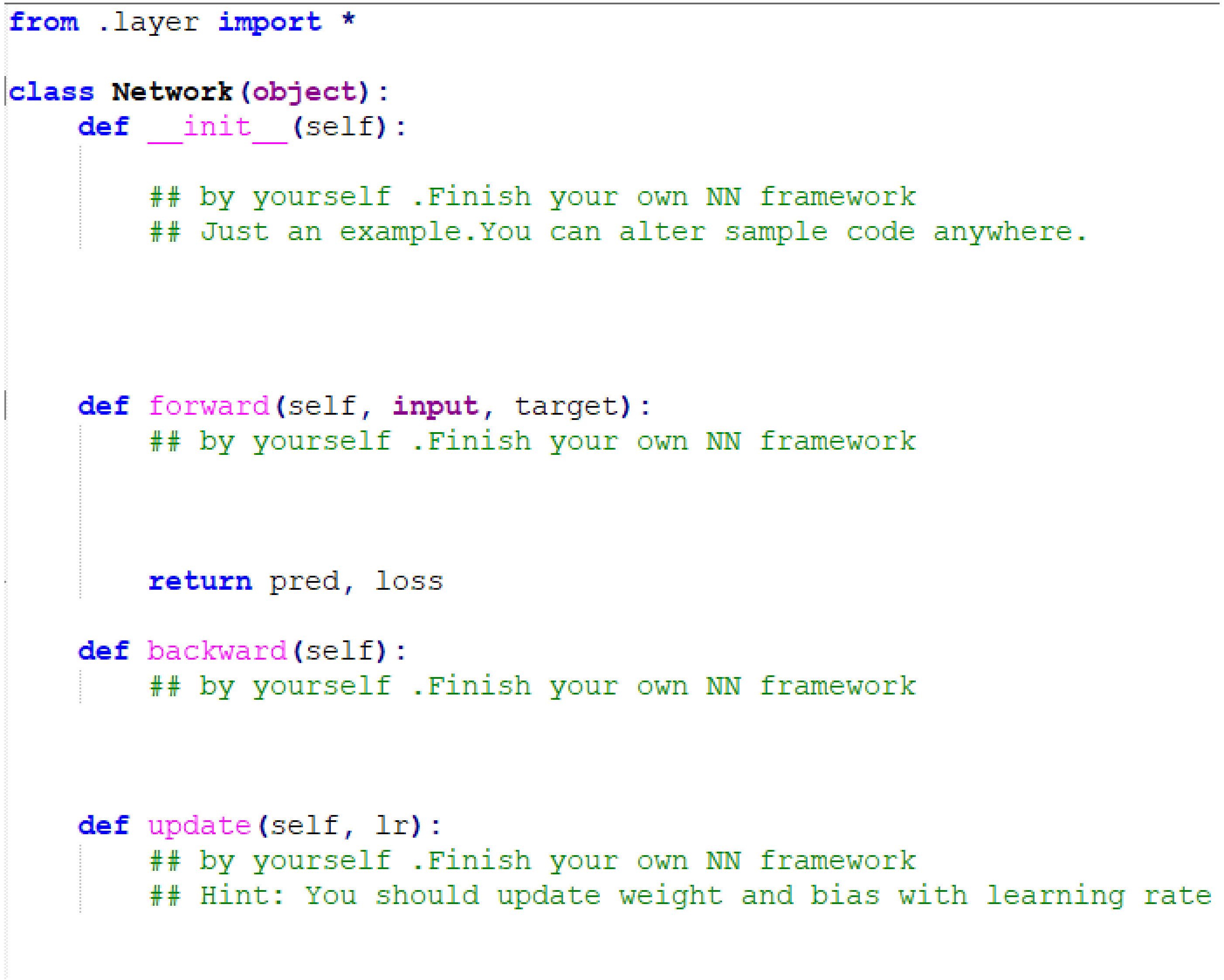


```
import numpy as np

## by yourself .Finish your own NN framework
## Just an example.You can alter sample code anywhere.


class _Layer(object):
    def __init__(self):
        pass

    def forward(self, *input):
        r"""Define the forward propagation of this layer.

        Should be overridden by all subclasses.
        """
        raise NotImplementedError

    def backward(self, *output_grad):
        r"""Define the backward propagation of this layer.

        Should be overridden by all subclasses.
        """
        raise NotImplementedError

## by yourself .Finish your own NN framework
class FullyConnected(_Layer):
    def __init__(self, in_features, out_features):




    def forward(self, input):


        return output

    def backward(self, output_grad):



        return input_grad
```



```
## by yourself .Finish your own NN framework
class Activation1(_Layer):
    def __init__(self):
        pass

    def forward(self, input):




        return output

    def backward(self, output_grad):



        return input_grad

class SoftmaxWithloss(_Layer):
    def __init__(self):
        pass

    def forward(self, input, target):










        return predict, your_loss

    def backward(self):



        return input_grad
```

model/network.py



```
from .layer import *

class Network(object):
    def __init__(self):

        ## by yourself .Finish your own NN framework
        ## Just an example.You can alter sample code anywhere.




    def forward(self, input, target):
        ## by yourself .Finish your own NN framework



        return pred, loss

    def backward(self):
        ## by yourself .Finish your own NN framework



    def update(self, lr):
        ## by yourself .Finish your own NN framework
        ## Hint: You should update weight and bias with learning rate
```

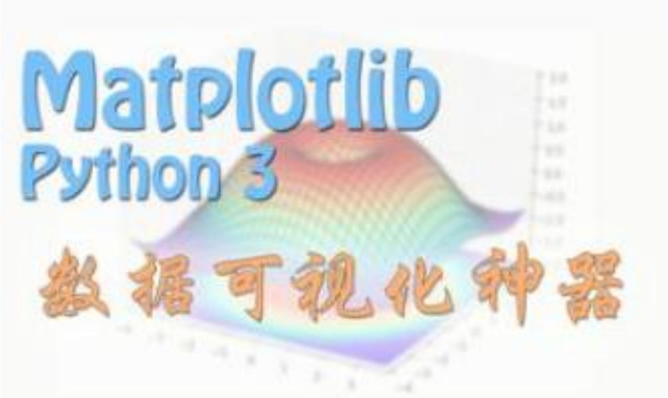
Hint

- We encourage you to implement convolution
  - Convolution would significantly improve performance compared to using only fully connected layers
  - A network with only fully connected layers would also meet the requirements

Task1 Overview

- You need to be familiar with jupyter python and useful library
  - numpy, pandas, matplot, etc…

- If you are not familiar with python, I strongly recommend you to see 莫煩’s python tutorial. (or other online materials)
  - https://mofanpy.com/



Outline

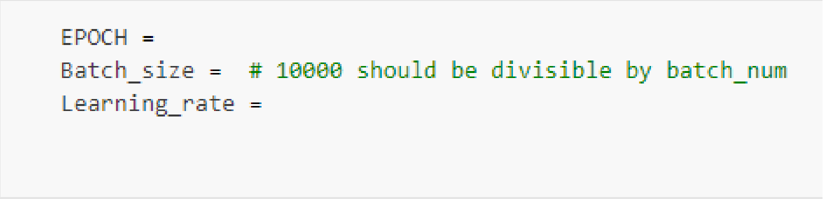
- Backpropagation
- Dataset
- Task1
- Task2
- Rules

Task2 – Build neural network using PyTorch

- In Lab1_task2.ipynb
  - You need to rewrite the network you built in Task 1 using PyTorch
  - The settings for training have been written by TA
    - You can change the settings however you like
  - Network should be the same as Task 1
    - You are free to choose the criterion and optimizer, they don't have to be exactly the same as in Task 1
  - You may only use pytorch in Task 2
    - nn.conv2d, nn.linear, etc…

Task2 Code

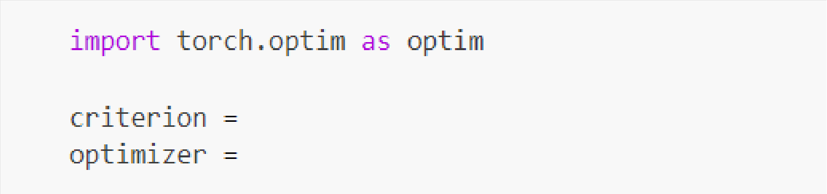
Hyperparameters



```
EPOCH =
Batch_size =  # 10000 should be divisible by batch_num
Learning_rate =
```

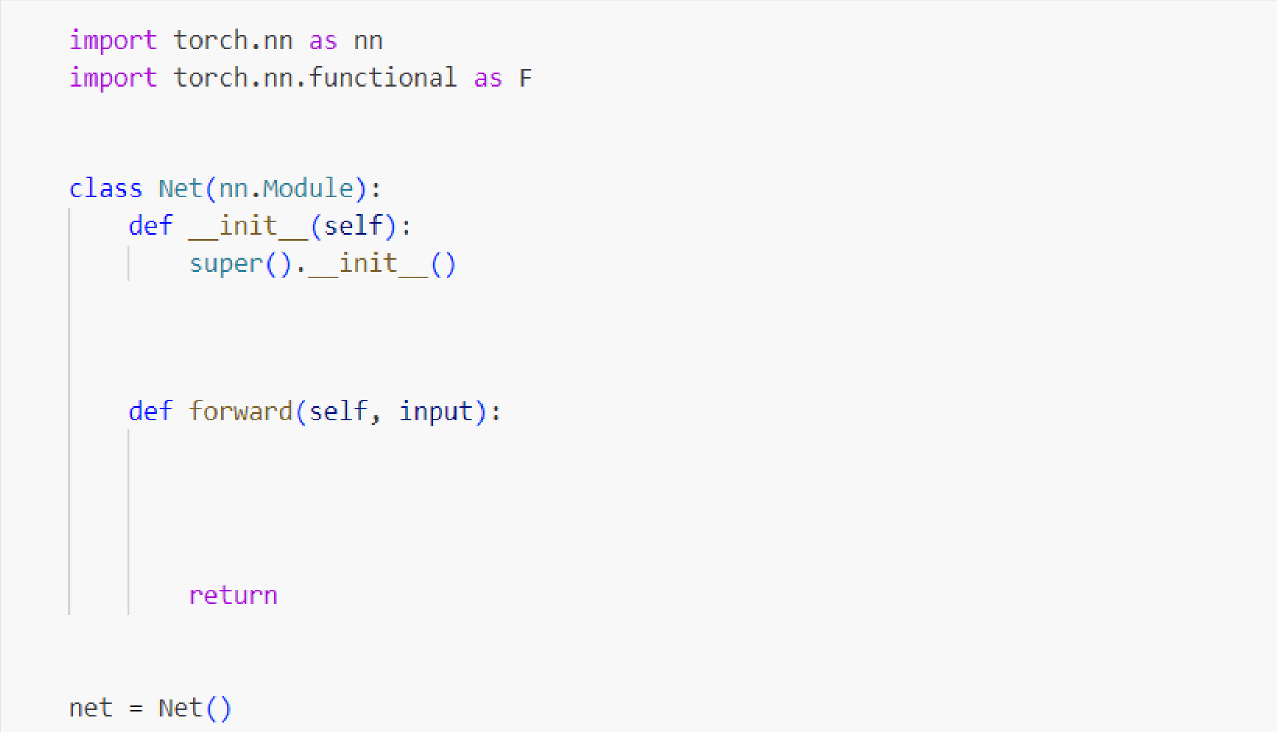
Significantly impact the results

Criterion and Optimizer



```
import torch.optim as optim

criterion =
optimizer =
```

Build the network using pytorch



```
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()



    def forward(self, input):




        return


net = Net()
```

PyTorch tutorial

- Official tutorial
  - https://pytorch.org/tutorials/
- 莫凡
  - https://mofanpy.com/tutorials/machine-learning/torch/
- AssemblyAI - PyTorch Crash Course
  - https://www.youtube.com/watch?v=OIenNRt2bjg

Outline

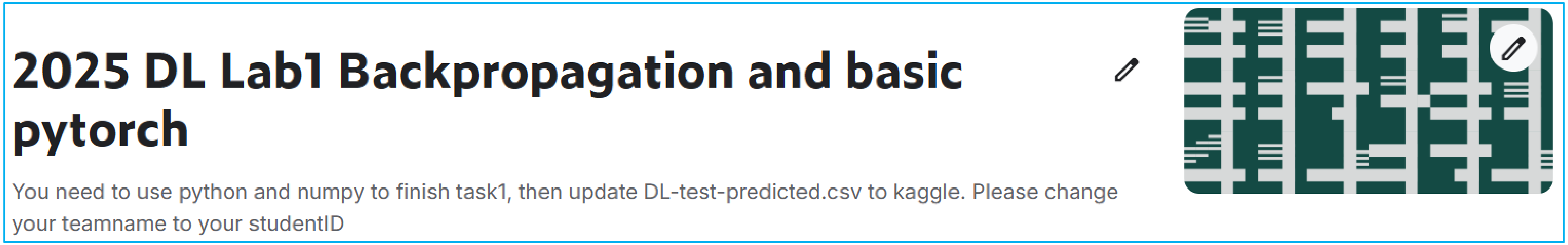
- Backpropagation
- Dataset
- Task1
- Task2
- Rules

Assignment Rules

- Please use Google Colab to finish this lab
- Please refer to “Colab_Tutorial_2025.pptx” for further details
- You don’t need to use GPU in this lab

Join Kaggle competition

- Competition URL:
  - https://www.kaggle.com/t/9ec61d068eca4d1aa56b887b944c22da



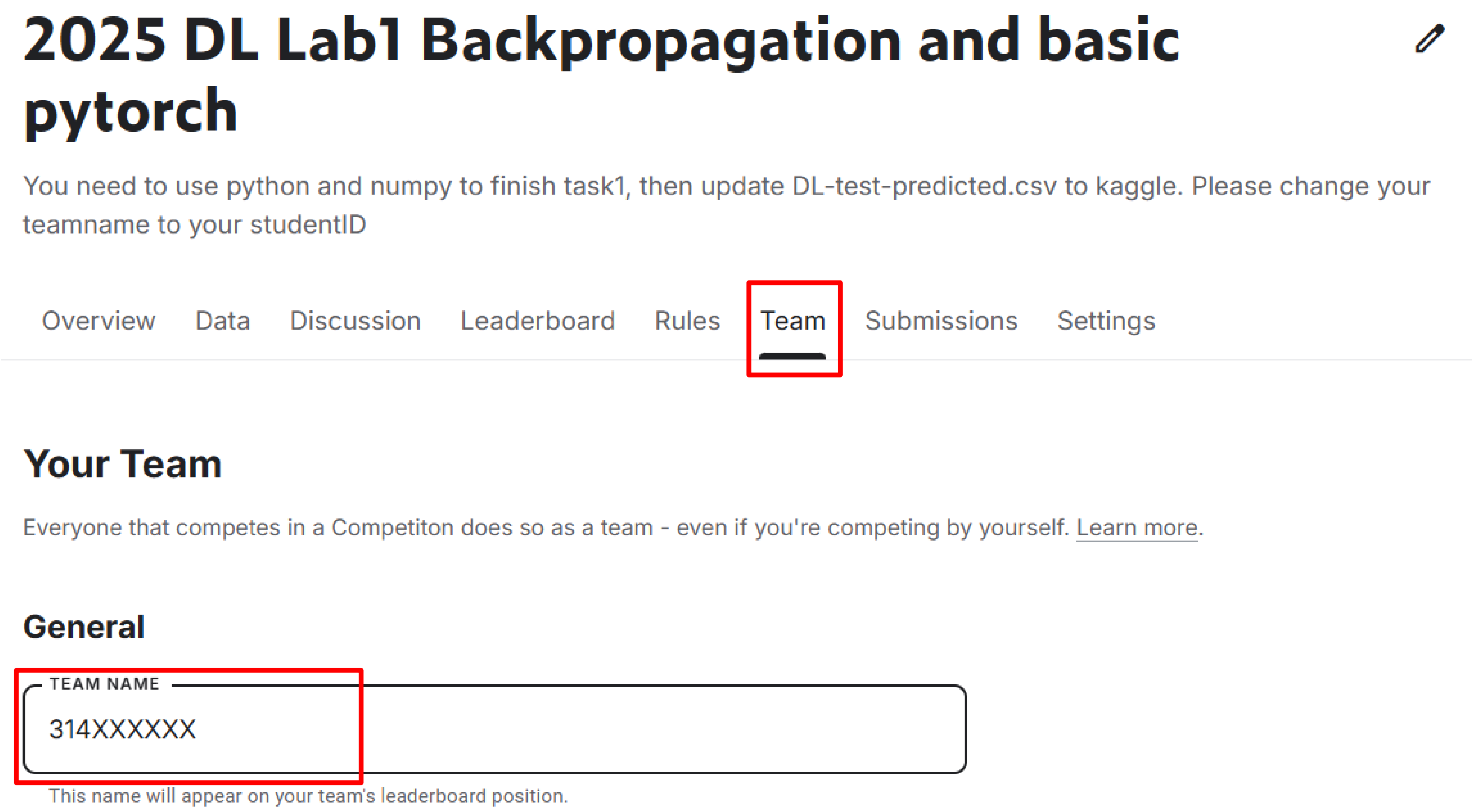
2025 DL Lab1 Backpropagation and basic pytorch

You need to use python and numpy to finish task1, then update DL-test-predicted.csv to kaggle. Please change your teamname to your studentID

- Upload your test prediction (DL-test-predicted.csv)
- Maximum daily submission limit: 10
- Change your team name to your student ID

Change your team name to your student ID

- If TA can not find your ID, you will loss 25% of score for this lab directly



2025 DL Lab1 Backpropagation and basic pytorch

You need to use python and numpy to finish task1, then update DL-test-predicted.csv to kaggle. Please change your teamname to your studentID

Your Team

Everyone that competes in a Competiton does so as a team - even if you're competing by yourself. Learn more.

General

TEAM NAME: 314XXXXXX

This name will appear on your team's leaderboard position.

Kaggle Rules

- The test set will be divided into two parts: public and private
  - Public: 200 images
  - Private: 9800 images
- You will see the public score each time you upload the CSV file
- You cannot see the private score as it will be announced after the deadline
- The final performance score will be based on the private set

Reminder

- Submit Deadline : 2 weeks (2025/9/22 11:59 PM)
- You need to submit your **code and result** to New E3

- Hand in your code and in the following format (5 files)
  - Lab01_task1_studentid.ipynb
  - network_studentid.py
  - layer_studentid.py
  - Lab01_task2_studentid.ipynb
  - Lab01_report_studentid.pdf

Grading policy

- Lab1
  - Submit your homework to E3 (60%)
    - Task1 (40%) (validation accuracy should >= 80%, put the screenshot in your report)
    - Task2 (20%) (validation accuracy should >= 80%, put the screenshot in your report)
  - Performance (25%) (submit to Kaggle)
    - The results you upload to Kaggle should be generated only by task1
    - Your performance score will be calculated by private testing set
  - Report (15%)

- Please do not plagiarize (0 points will be calculated if caught)

Report

- How to improve the accuracy (list your method)
  - Your network?
  - Loss function?
  - Activation function?
  - Hyperparameters?
  - etc…
- What differences do you find between the results of Task1 and Task2?