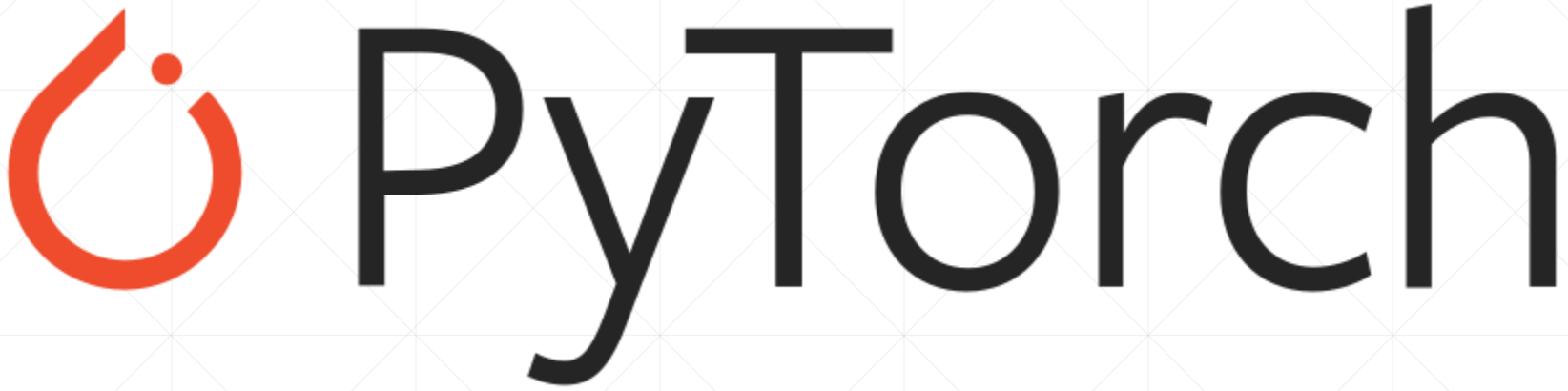


Logistic Regression

主讲人：龙良曲

Recap

- for continuous: $y = xw + b$
- for probability output: $y = \sigma(xw + b)$
  - σ: *sigmoid or logistic*

Binary Classification

- interpret network as $f: x \rightarrow p(y|x;\ \theta)$
- output $\in [0, 1]$
- which is exactly what *logistic function* comes in!

Goal v.s. Approach

- For regression:
  - Goal: $pred = y$
  - Approach: minimize $dist(pred, y)$

- For classification:
  - Goal: maximize benchmark, e.g. *accuracy*
  - Approach1: minimize $dist(p_\theta(y|x), p_r(y|x))$
  - Approach2: minimize $divergence(p_\theta(y|x), p_r(y|x))$

Q1. why not maximize accuracy?

- $acc. = \frac{\sum I(pred_i == y_i)}{len(Y)}$
- issues 1. gradient = 0 if accuracy unchanged but weights changed
- issues 2. *gradient not continuous* since the number of correct is not continuous

Q2. why call logistic regression

- use sigmoid
- Controversial!
  - MSE => regression
  - Cross Entropy => classification

Binary Classification

- $f: x \rightarrow p(y = 1|x)$
  - if $p(y = 1|x) > 0.5$, predict as **1**
  - else predict as **0**

- minimize MSE

- confused?
  - http://www.fharrell.com/post/classification/

Multi-class classification

- $f: x \rightarrow p(y|x)$
  - $[p(y=0|x), p(y=1|x), \ldots, p(y=9|x)]$
- $p(y|x) \in [0,1]$
- $\sum_{i=0}^{9} p(y=i|x) = 1$

Softmax

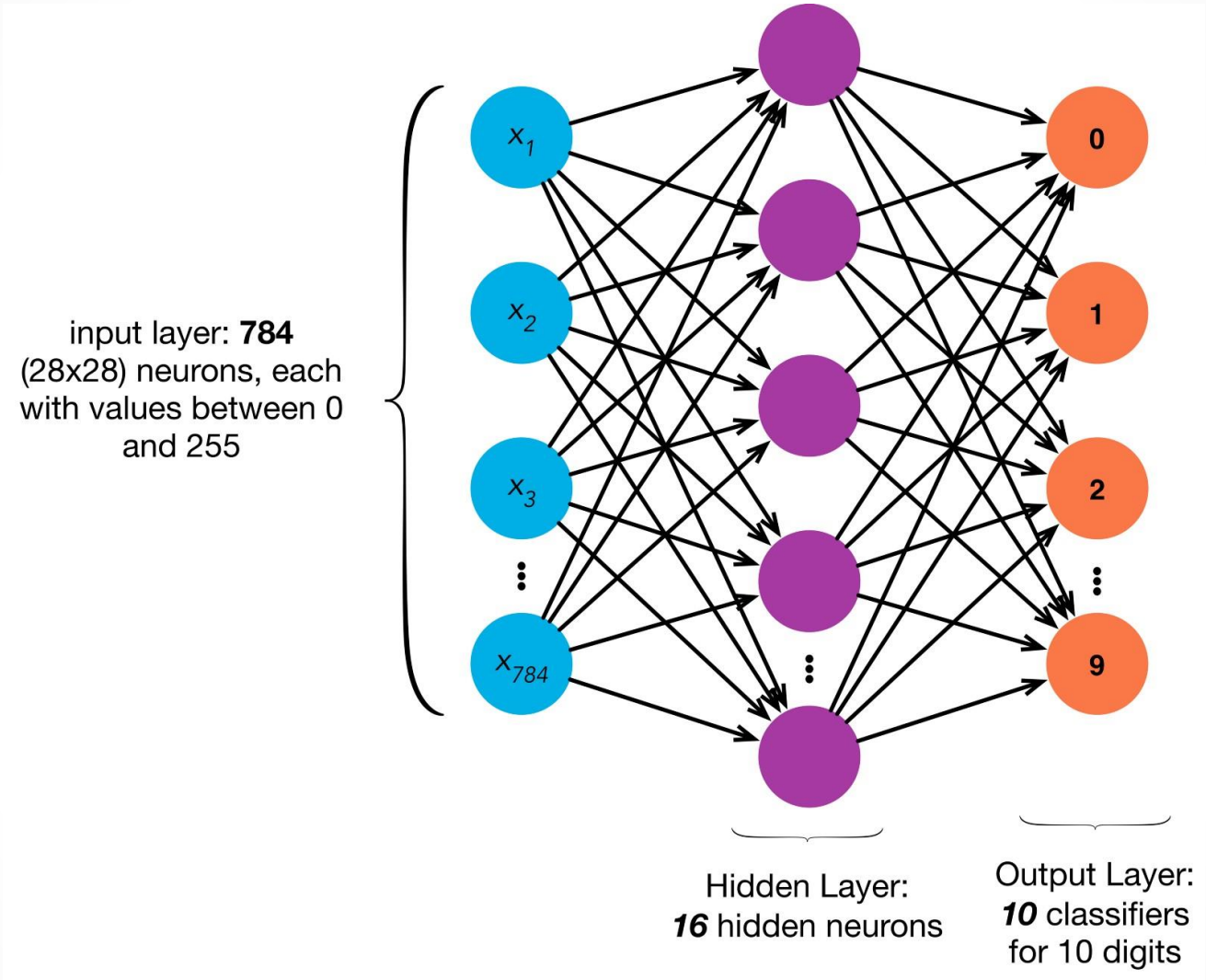
- $\sum_{i=0}^{9} p(y = i|x) = 1$

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

Softmax

- enlarger the larger

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$


input layer: 784 (28x28) neurons, each with values between 0 and 255
x_1
x_2
x_3
x_{784}
0
1
2
9
Hidden Layer: 16 hidden neurons
Output Layer: 10 classifiers for 10 digits

下一课时

交叉熵

Thank You.

PyTorch

交叉熵

主讲人：龙良曲

Why not MSE?

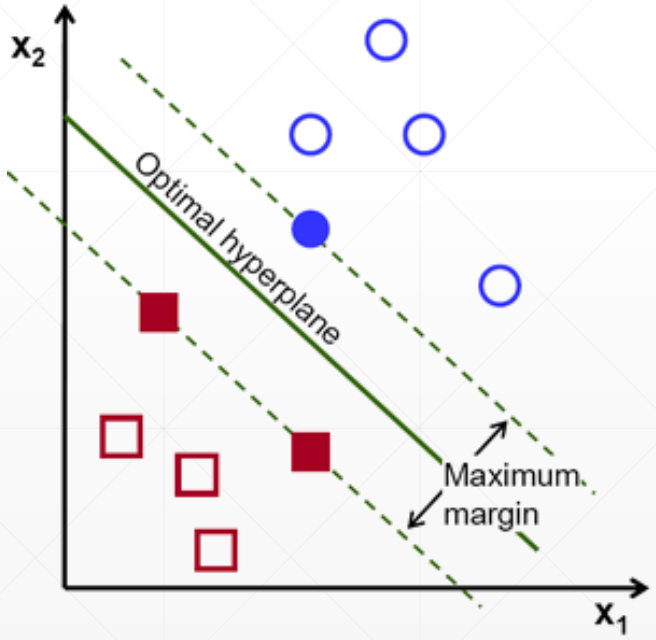
Label	predict	correct
3	[0.3, 0.3, 0.4]	yes
2	[0.3, 0.4, 0.3]	yes
1	[0.1, 0.2, 0.7]	no

Label	predict	correct
3	[0.1, 0.2, 0.7]	yes
2	[0.1, 0.7, 0.2]	yes
1	[0.3, 0.4, 0.3]	no

Loss for classification

- MSE
- Cross Entropy Loss
- Hinge Loss

$$\sum_i max(0, 1 - y_i * h_\theta(x_i))$$



Entropy

- Uncertainty
- measure of surprise

- higher entropy: higher uncertainty.

$$Entropy = -\sum_i P(i) \log P(i)$$

Claude Shannon

https://towardsdatascience.com/demystifying-cross-entropy-e80e3ad54a8

Lottery

```
In [2]: a=torch.full([4],1/4.)
tensor([0.2500, 0.2500, 0.2500, 0.2500])

In [4]: a*torch.log2(a)
Out[4]: tensor([-0.5000, -0.5000, -0.5000, -0.5000])

In [6]: -(a*torch.log2(a)).sum()
Out[6]: tensor(2.)

In [7]: a=torch.tensor([0.1,0.1,0.1,0.7])
In [8]: -(a*torch.log2(a)).sum()
Out[8]: tensor(1.3568)

In [15]: a=torch.tensor([0.001,0.001,0.001,0.999])
In [16]: -(a*torch.log2(a)).sum()
Out[16]: tensor(0.0313)
```

Cross Entropy

$$H(p,q) = -\sum p(x) \log q(x)$$

$$H(p,q) = H(p) + D_{\mathrm{KL}}(p|q).$$

- P=Q
  - cross Entropy = Entropy
- for one-hot encoding,
  - entropy = 1log1=0

Binary Classification

$$H(P,Q) = -P(cat)\log Q(cat) - (1 - P(cat))\log(1 - Q(cat))$$

$$P(dog) = (1 - P(cat))$$

$$H(P,Q) = -\sum_{i=(cat,dog)} P(i)\log Q(i)$$

$$= -P(cat)\log Q(cat) - P(dog)\log Q(dog)$$

$$-(y\log(p) + (1-y)\log(1-p))$$

for example

$$P_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$Q_1 = [0.4 \quad 0.3 \quad 0.05 \quad 0.05 \quad 0.2]$$

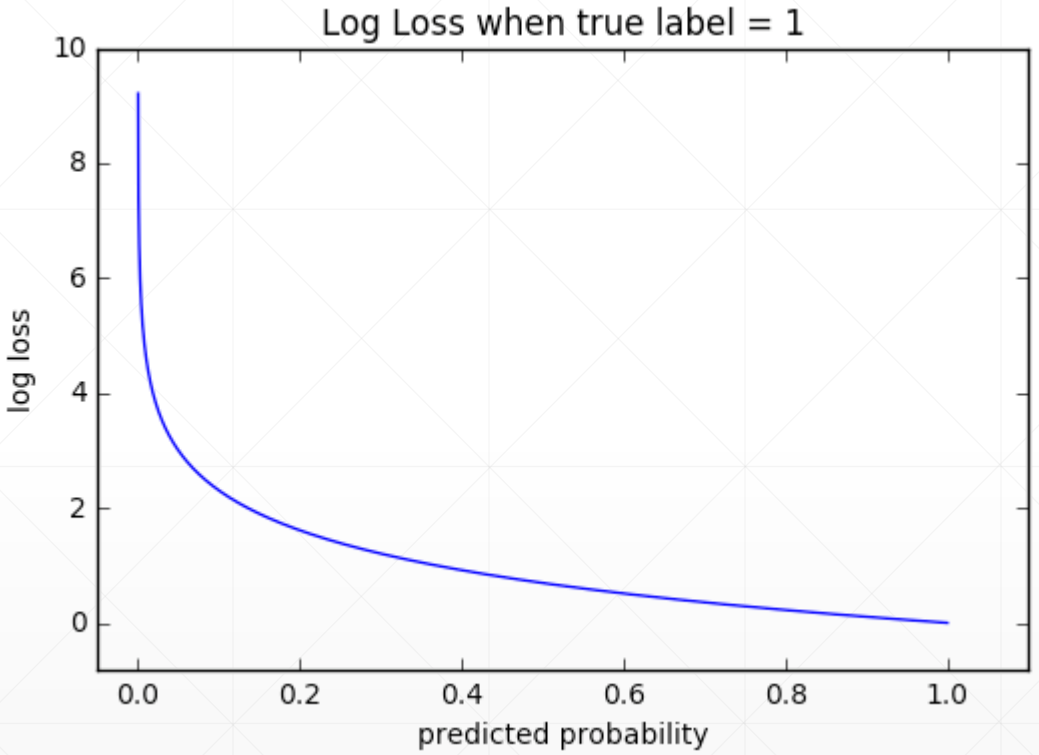
$$\begin{aligned} H(P_1, Q_1) &= -\sum_i P_1(i) \log Q_1(i) \\ &= -(1 \log 0.4 + 0 \log 0.3 + 0 \log 0.05 + 0 \log 0.05 + 0 \log 0.2) \\ &= -\log 0.4 \\ &\approx 0.916 \end{aligned}$$

$$Q_1 = [0.98 \quad 0.01 \quad 0 \quad 0 \quad 0.01]$$

$$\begin{aligned} H(P_1, Q_1) &= -\sum_i P_1(i) \log Q_1(i) \\ &= -(1 \log 0.98 + 0 \log 0.01 + 0 \log 0 + 0 \log 0 + 0 \log 0.01) \\ &= -\log 0.98 \\ &\approx 0.02 \end{aligned}$$

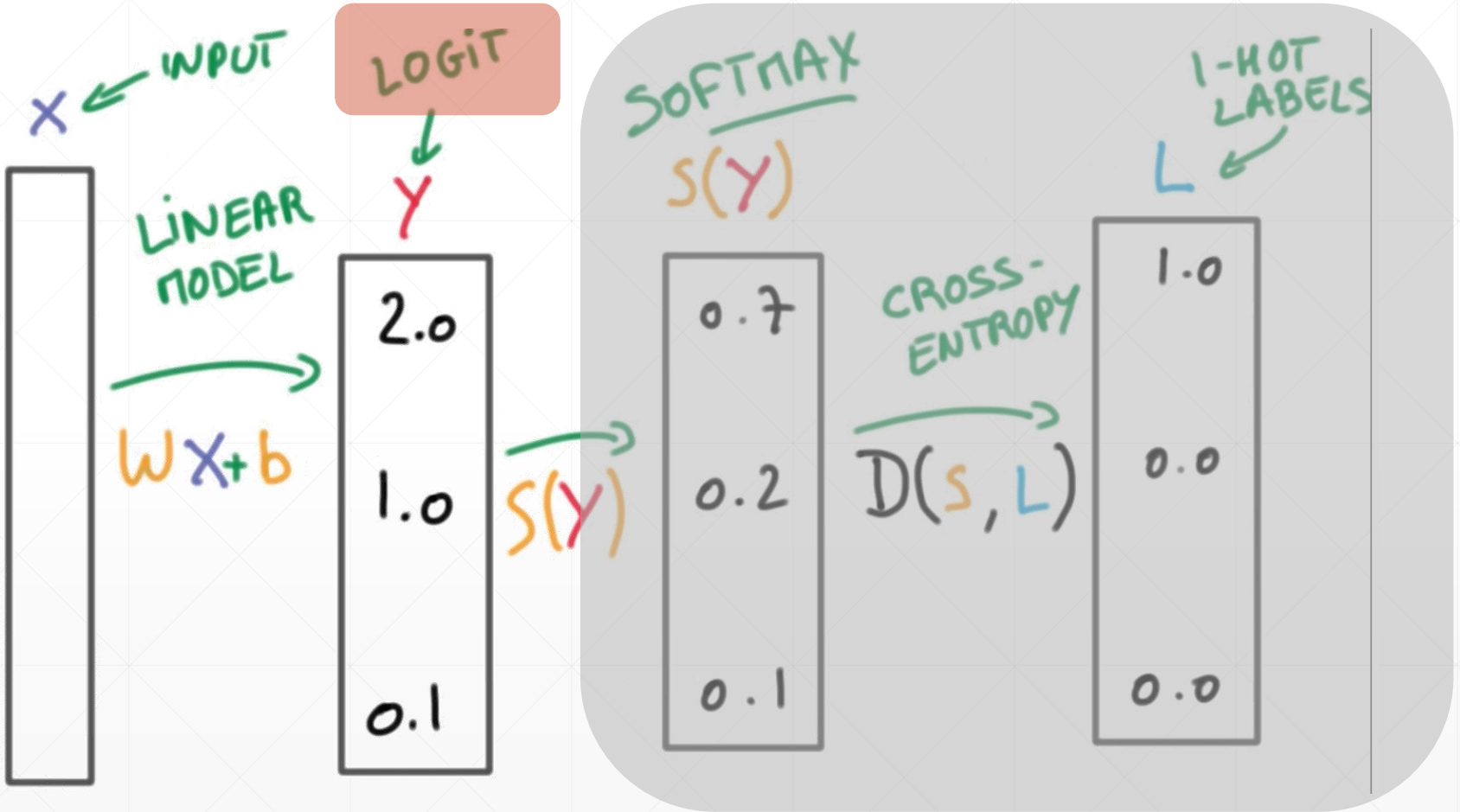
why not use MSE

- sigmoid + MSE
  - gradient vanish
- converge slower
- But, sometimes
  - e.g. meta-learning



https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Therefore


INPUT
X
LOGIT
Y
LINEAR MODEL
WX+b
2.0
1.0
0.1
S(Y)
SOFTMAX
S(Y)
0.7
0.2
0.1
CROSS-ENTROPY
D(S, L)
1-HOT LABELS
L
1.0
0.0
0.0

Numerical Stability

```
In [17]: x=torch.randn(1,784)
In [18]: w=torch.randn(10,784)

In [19]: logits=x@w.t()
Out[20]: torch.Size([1, 10])

In [34]: pred=F.softmax(logits, dim=1)
Out[37]: torch.Size([1, 10])

In [40]: pred_log=torch.log(pred)

In [41]: F.cross_entropy(logits,torch.tensor([3]))
Out[41]: tensor(82.4905)

In [42]: F.nll_loss(pred_log,torch.tensor([3]))
Out[42]: tensor(82.4905)
```

下一课时

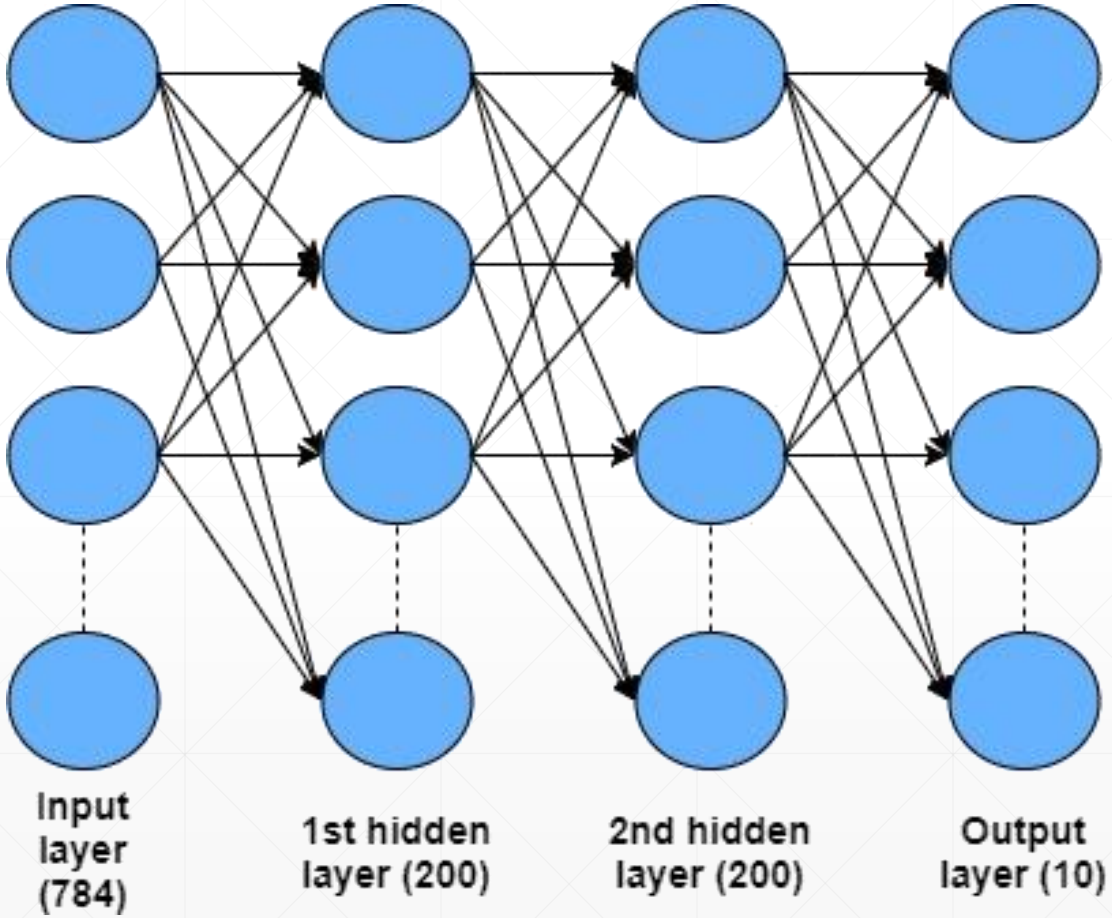
实战多分类问题

Thank You.

PyTorch

多分类问题

主讲人：龙良曲


Input layer (784)
1st hidden layer (200)
2nd hidden layer (200)
Output layer (10)

Network Architecture

```
w1, b1 = torch.randn(200, 784, requires_grad=True),\
         torch.zeros(200, requires_grad=True)
w2, b2 = torch.randn(200, 200, requires_grad=True),\
         torch.zeros(200, requires_grad=True)
w3, b3 = torch.randn(10, 200, requires_grad=True),\
         torch.zeros(10, requires_grad=True)

def forward(x):
    x = x@w1.t() + b1
    x = F.relu(x)
    x = x@w2.t() + b2
    x = F.relu(x)
    x = x@w3.t() + b3
    x = F.relu(x)
    return x
```

Train

```
optimizer = optim.SGD([w1, b1, w2, b2, w3, b3], lr=learning_rate)
criteon = nn.CrossEntropyLoss()

for epoch in range(epochs):

    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.view(-1, 28*28)

        logits = forward(data)
        loss = criteon(logits, target)

        optimizer.zero_grad()
        loss.backward()
        # print(w1.grad.norm(), w2.grad.norm())
        optimizer.step()
```

```
C:\ProgramData\conda\python.exe F:/PytorchTutorial/lesson26-LR多分类实战/main.py
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
```

em....

```
w1, b1 = torch.randn(200, 784, requires_grad=True),\
         torch.zeros(200, requires_grad=True)
w2, b2 = torch.randn(200, 200, requires_grad=True),\
         torch.zeros(200, requires_grad=True)
w3, b3 = torch.randn(10, 200, requires_grad=True),\
         torch.zeros(10, requires_grad=True)

torch.nn.init.kaiming_normal_(w1)
torch.nn.init.kaiming_normal_(w2)
torch.nn.init.kaiming_normal_(w3)
```

下一课时

PyTorch全连接层

Thank You.

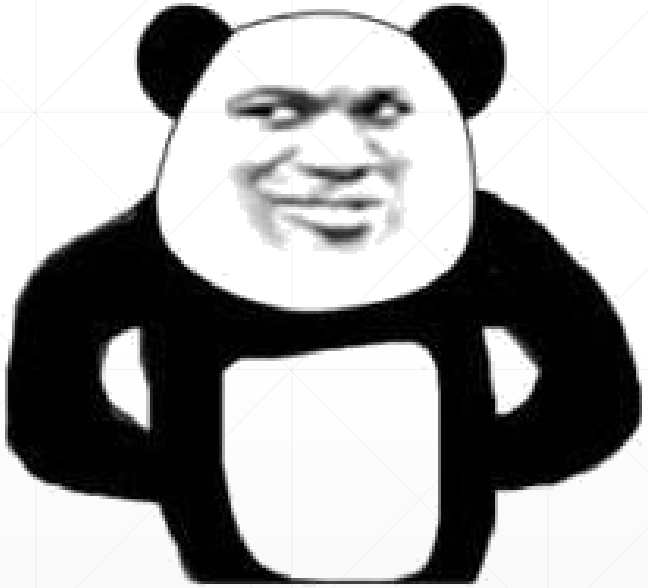
PyTorch

全军出击：全连接层

主讲人：龙良曲

I know nothing

Be practical



在大佬看来说这么多没用
还是直接上去猛干就对了

nn.Linear

```
In [44]: x.shape
Out[44]: torch.Size([1, 784])

In [46]: layer1=nn.Linear(784, 200)
In [47]: layer2=nn.Linear(200,200)
In [48]: layer3=nn.Linear(200,10)

In [49]: x=layer1(x)
In [50]: x.shape
Out[50]: torch.Size([1, 200])

In [52]: x=layer2(x)
In [53]: x.shape
Out[53]: torch.Size([1, 200])

In [54]: x=layer3(x)
In [55]: x.shape
Out[55]: torch.Size([1, 10])
```

relu?

```
In [49]: x=layer1(x)
In [56]: x=F.relu(x, inplace=True)
In [50]: x.shape
Out[50]: torch.Size([1, 200])

In [52]: x=layer2(x)
In [56]: x=F.relu(x, inplace=True)
In [53]: x.shape
Out[53]: torch.Size([1, 200])

In [54]: x=layer3(x)
In [56]: x=F.relu(x, inplace=True)
In [55]: x.shape
Out[55]: torch.Size([1, 10])
```

concisely

- inherit from nn.Module
- init layer in __init__
- implement forward()

Step1.

```
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()
```

Step2.

```
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()

        self.model = nn.Sequential(
            nn.Linear(784, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 10),
            nn.ReLU(inplace=True),
        )
```

Step3.

```
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()

        self.model = nn.Sequential(
            nn.Linear(784, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 10),
            nn.ReLU(inplace=True),
        )

     def forward(self, x):
      x = self.model(x)

      return x
```

nn.ReLU v.s. F.relu()

- class-style API
- function-style API

```
In [55]: x.shape
Out[55]: torch.Size([1, 10])

In [56]: x=F.relu(x, inplace=True)

In [57]: layer=nn.ReLU()

In [58]: x=layer(x)
```

Train

```
net = MLP()
optimizer = optim.SGD(net.parameters(), lr=learning_rate)
criteon = nn.CrossEntropyLoss()

for epoch in range(epochs):

    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.view(-1, 28*28)

        logits = net(data)
        loss = criteon(logits, target)

        optimizer.zero_grad()
        loss.backward()
        # print(w1.grad.norm(), w2.grad.norm())
        optimizer.step()
```

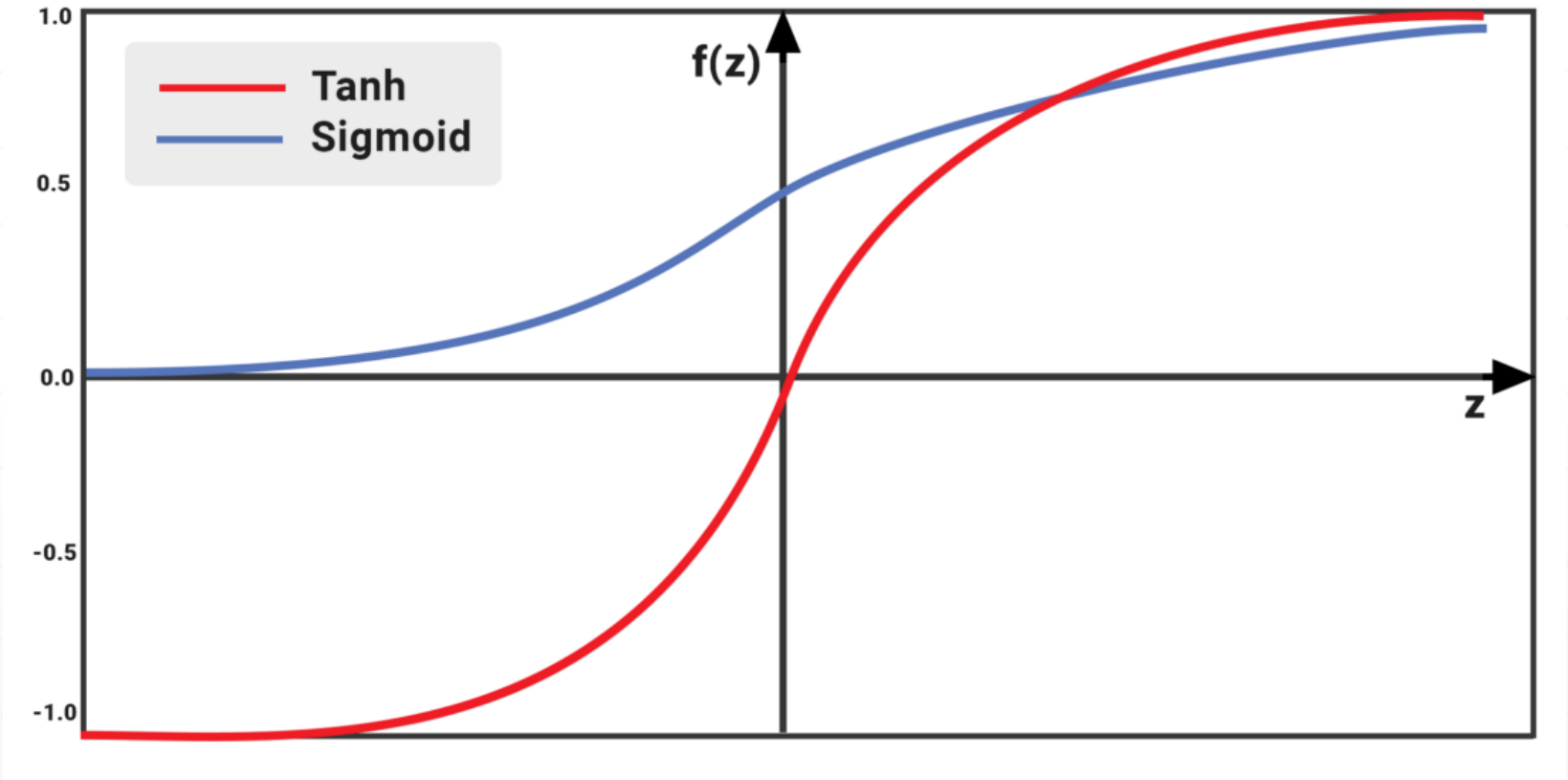
下一课时

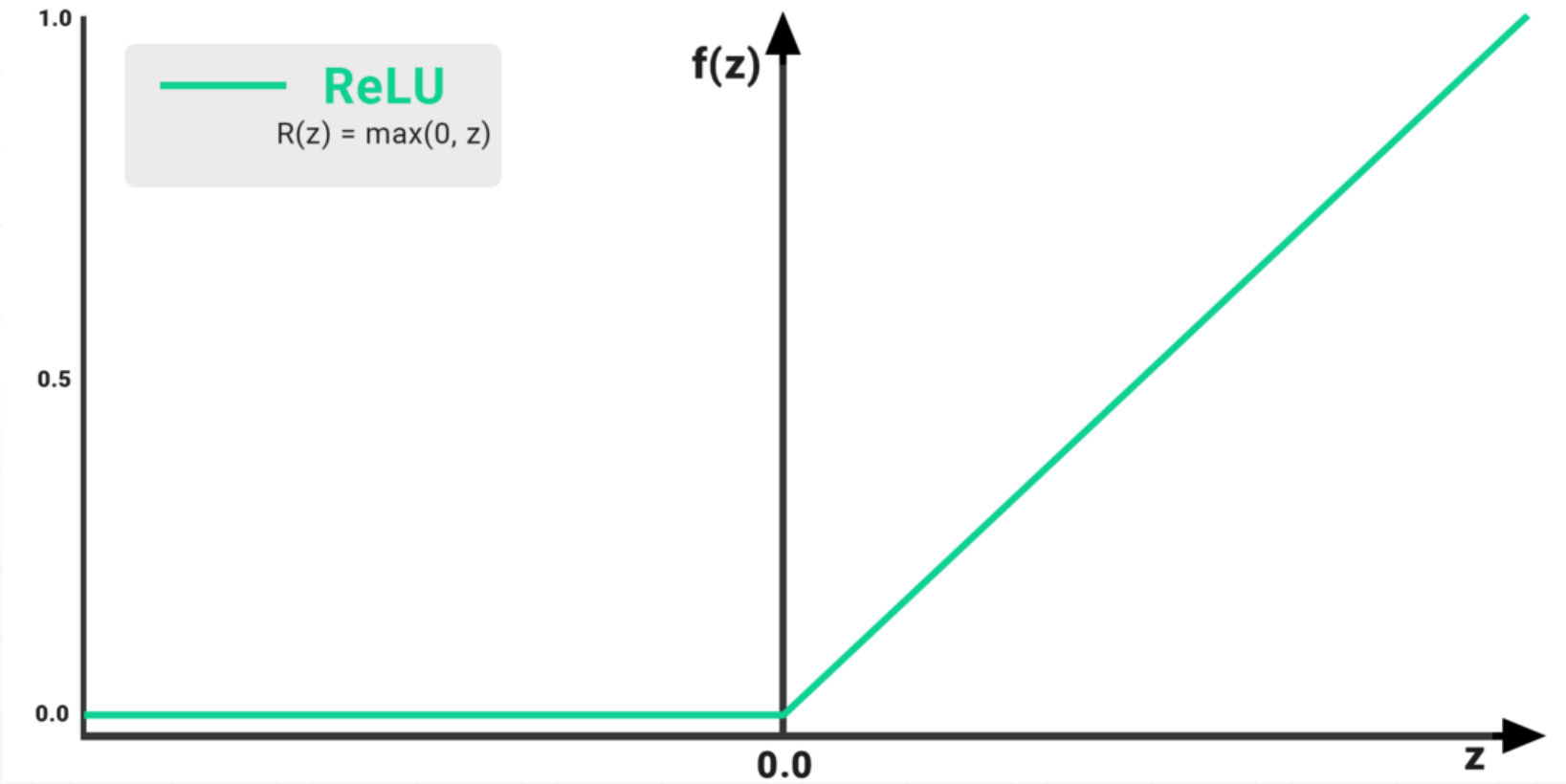
激活函数与GPU

Thank You.

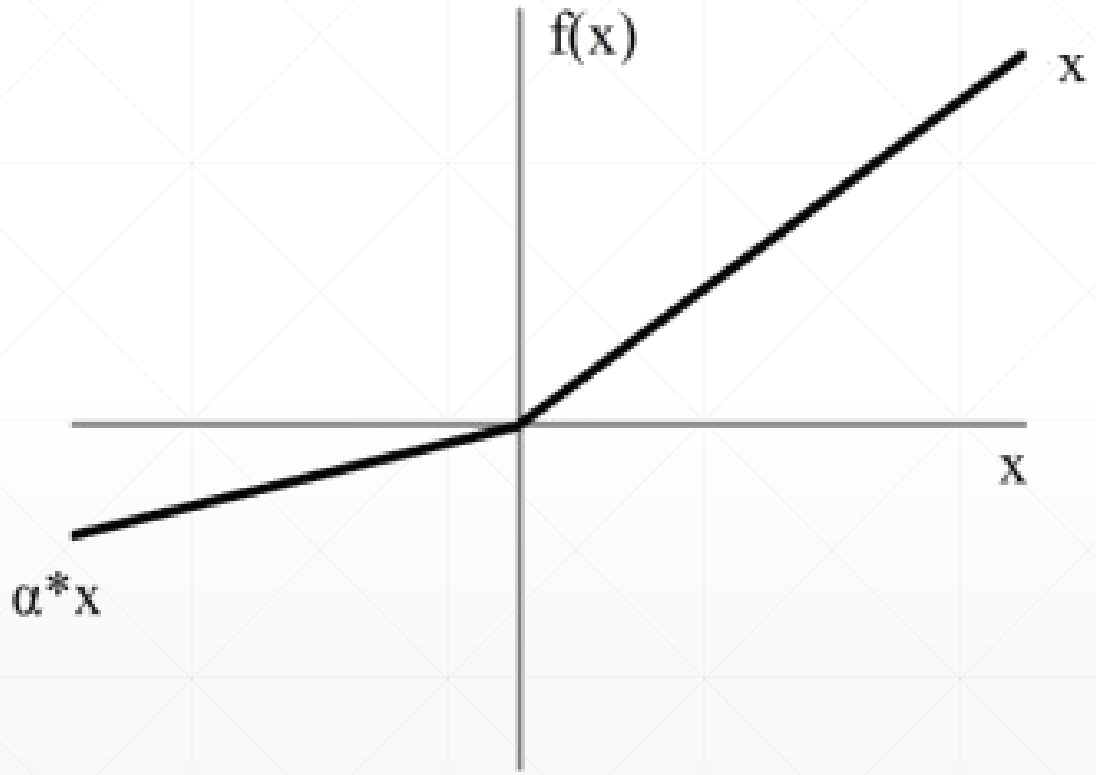
激活函数与GPU加速

主讲人：龙良曲


1.0
0.5
0.0
-0.5
-1.0
f(z)
z
Tanh
Sigmoid


ReLU
R(z) = max(0, z)
1.0
0.5
0.0
f(z)
0.0
z

Leaky ReLU



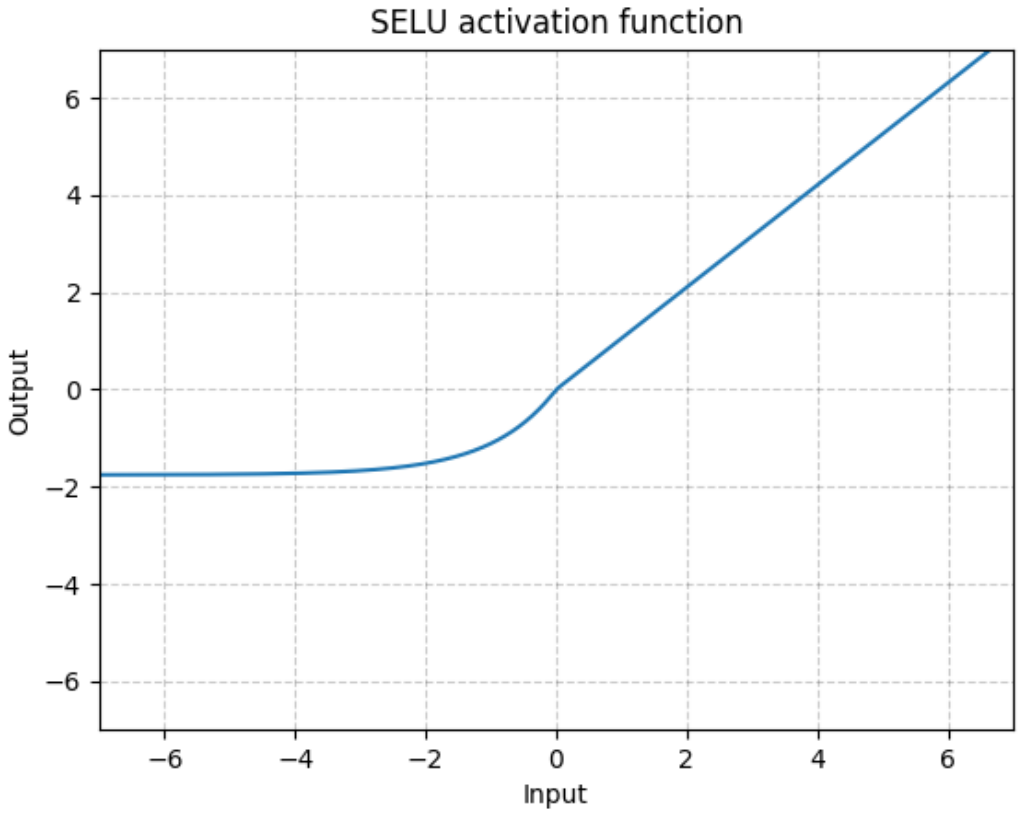
simply

```
self.model = nn.Sequential(
    nn.Linear(784, 200),
    nn.LeakyReLU(inplace=True),
    nn.Linear(200, 200),
    nn.LeakyReLU(inplace=True),
    nn.Linear(200, 10),
    nn.LeakyReLU(inplace=True),
)
```

SELU

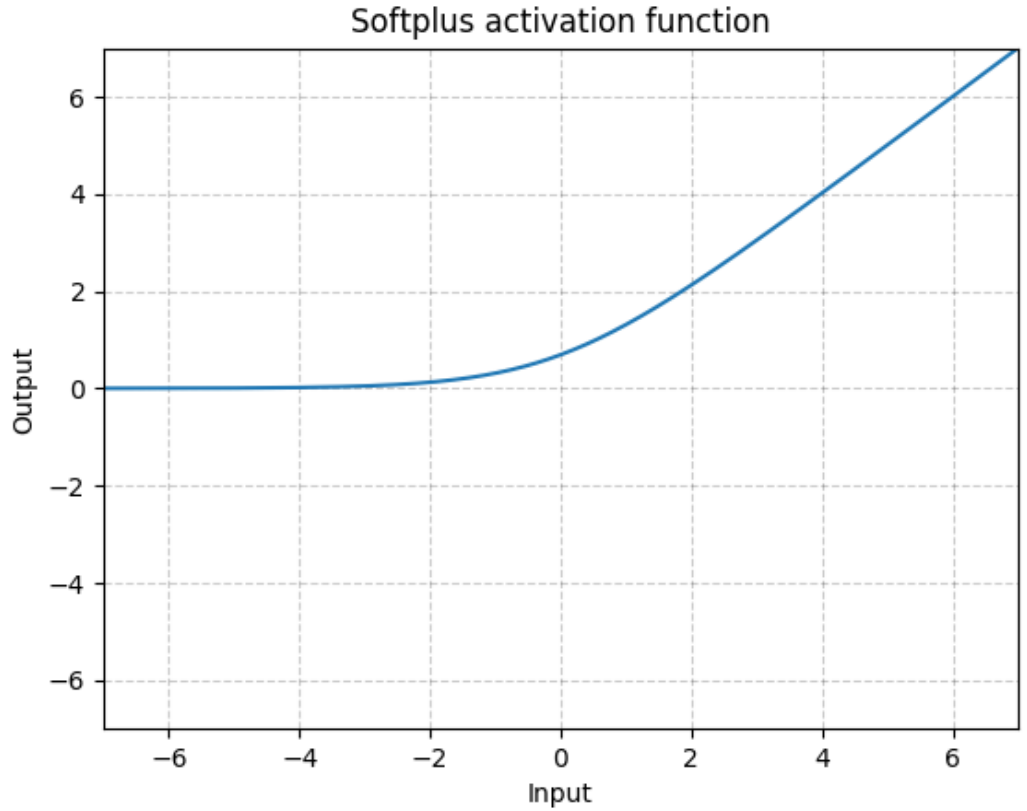
$$\mathrm{SELU}(x) = \mathrm{scale} * (\max(0, x) + \min(0, \alpha * (\exp(x) - 1)))$$

with $\alpha = 1.6732632423543772848170429916717$ and
$\mathrm{scale} = 1.0507009873554804934193349852946.$



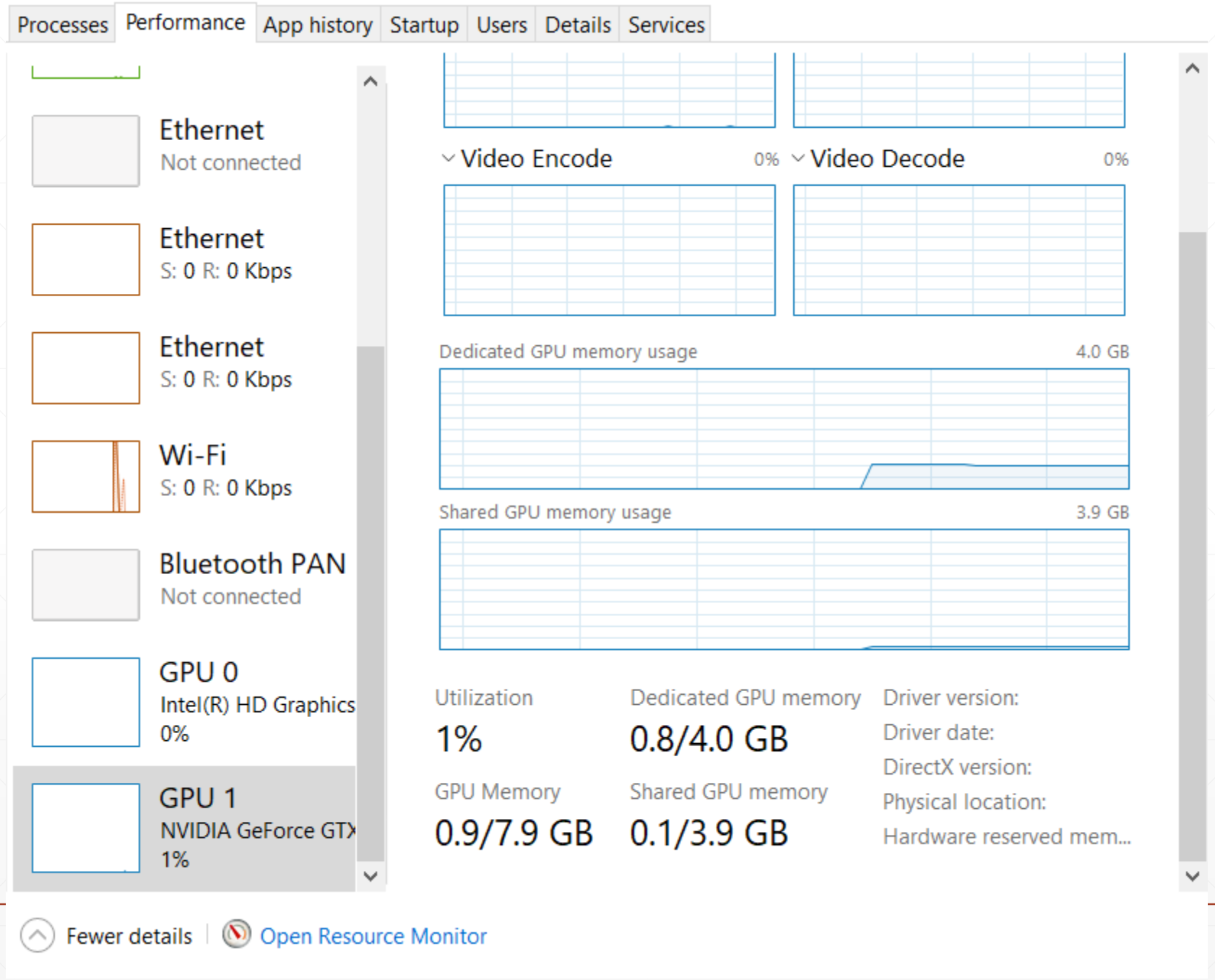
softplus

$$\text{Softplus}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$



GPU accelerated

```
device = torch.device('cuda:0')
net = MLP().to(device)
optimizer = optim.SGD(net.parameters(), lr=learning_rate)
criteon = nn.CrossEntropyLoss().to(device)

for epoch in range(epochs):

    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.view(-1, 28*28)
        data, target = data.to(device), target.cuda()
```


Processes
Performance
App history
Startup
Users
Details
Services
Ethernet
Not connected
Ethernet
S: 0 R: 0 Kbps
Ethernet
S: 0 R: 0 Kbps
Wi-Fi
S: 0 R: 0 Kbps
Bluetooth PAN
Not connected
GPU 0
Intel(R) HD Graphics
0%
GPU 1
NVIDIA GeForce GTX
1%
Video Encode
0%
Video Decode
0%
Dedicated GPU memory usage
4.0 GB
Shared GPU memory usage
3.9 GB
Utilization
1%
Dedicated GPU memory
0.8/4.0 GB
Driver version:
Driver date:
DirectX version:
Physical location:
Hardware reserved mem...
GPU Memory
0.9/7.9 GB
Shared GPU memory
0.1/3.9 GB
Fewer details
Open Resource Monitor

下一课时

测试

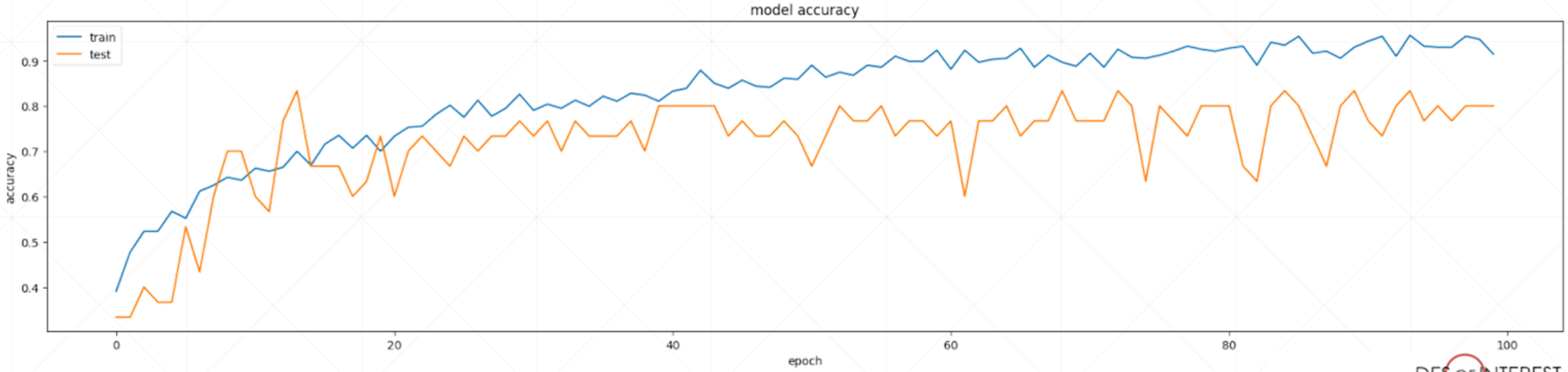
Thank You.

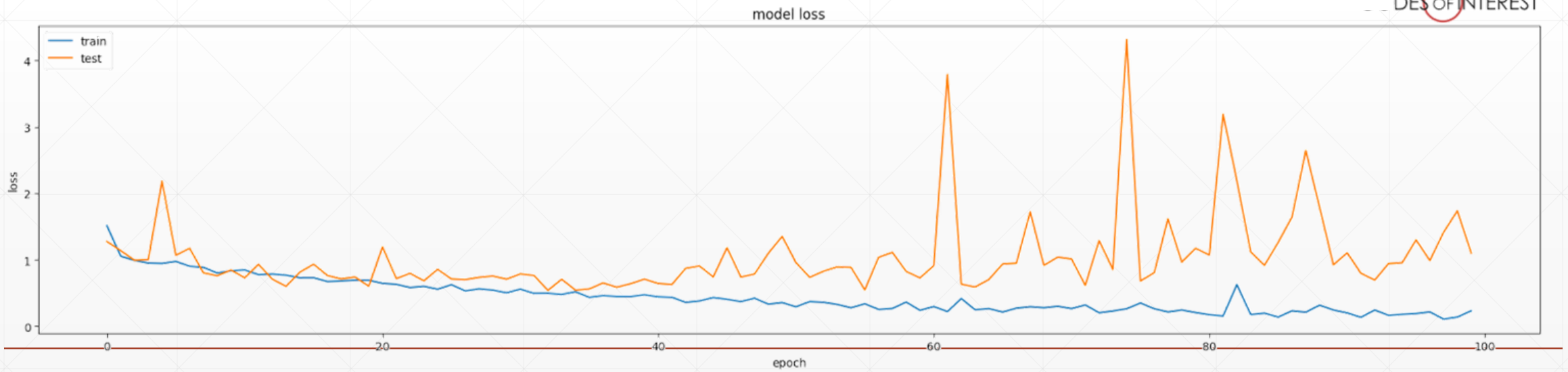
PyTorch

测 试

主讲人：龙良曲

Loss != Accuracy





DES OF INTEREST

argmax

```
In [60]: logits=torch.rand(4, 10)

In [62]: pred=F.softmax(logits, dim=1)
In [63]: pred.shape
Out[63]: torch.Size([4, 10])

In [64]: pred_label=pred.argmax(dim=1)
In [65]: pred_label
Out[65]: tensor([9, 5, 9, 4])

In [66]: logits.argmax(dim=1)
Out[66]: tensor([9, 5, 9, 4])

In [67]: label=torch.tensor([9,3,2,4])
In [68]: correct=torch.eq(pred_label, label)
Out[69]: tensor([1, 0, 0, 1], dtype=torch.uint8)

In [71]: correct.sum().float().item()/4.
Out[71]: 0.5
```

When to test

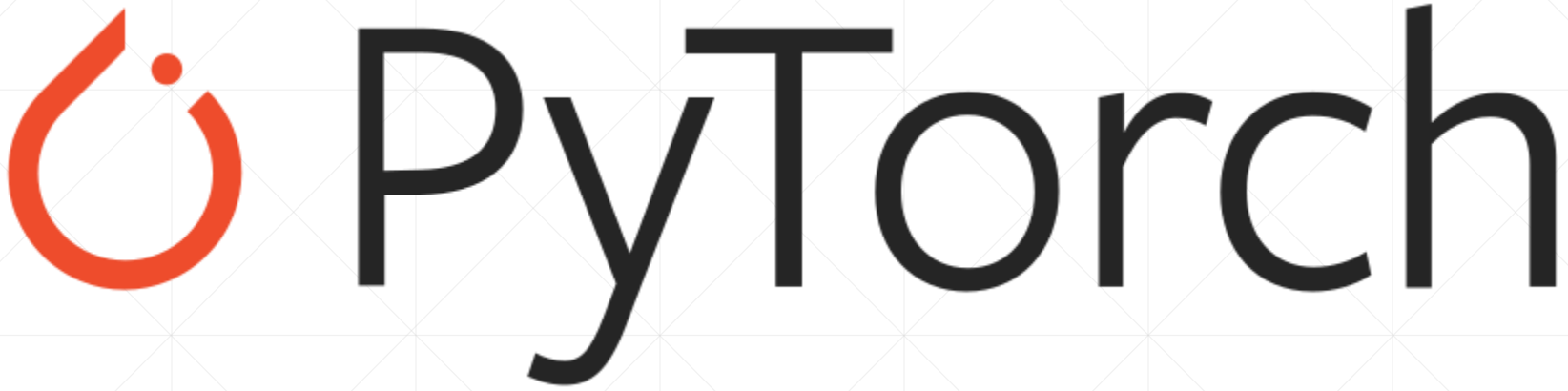
- test once per several batch
- test once per epoch
- epoch V.S. step?

```
    test_loss = 0
    correct = 0
    for data, target in test_loader:
        data = data.view(-1, 28 * 28)
        data, target = data.to(device), target.cuda()
        logits = net(data)
        test_loss += criteon(logits, target).item()

        pred = logits.argmax(dim=1)
        correct += pred.eq(target).float().sum().item()

    test_loss /= len(test_loader.dataset)
    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
```

下一课时

Visdom可视化

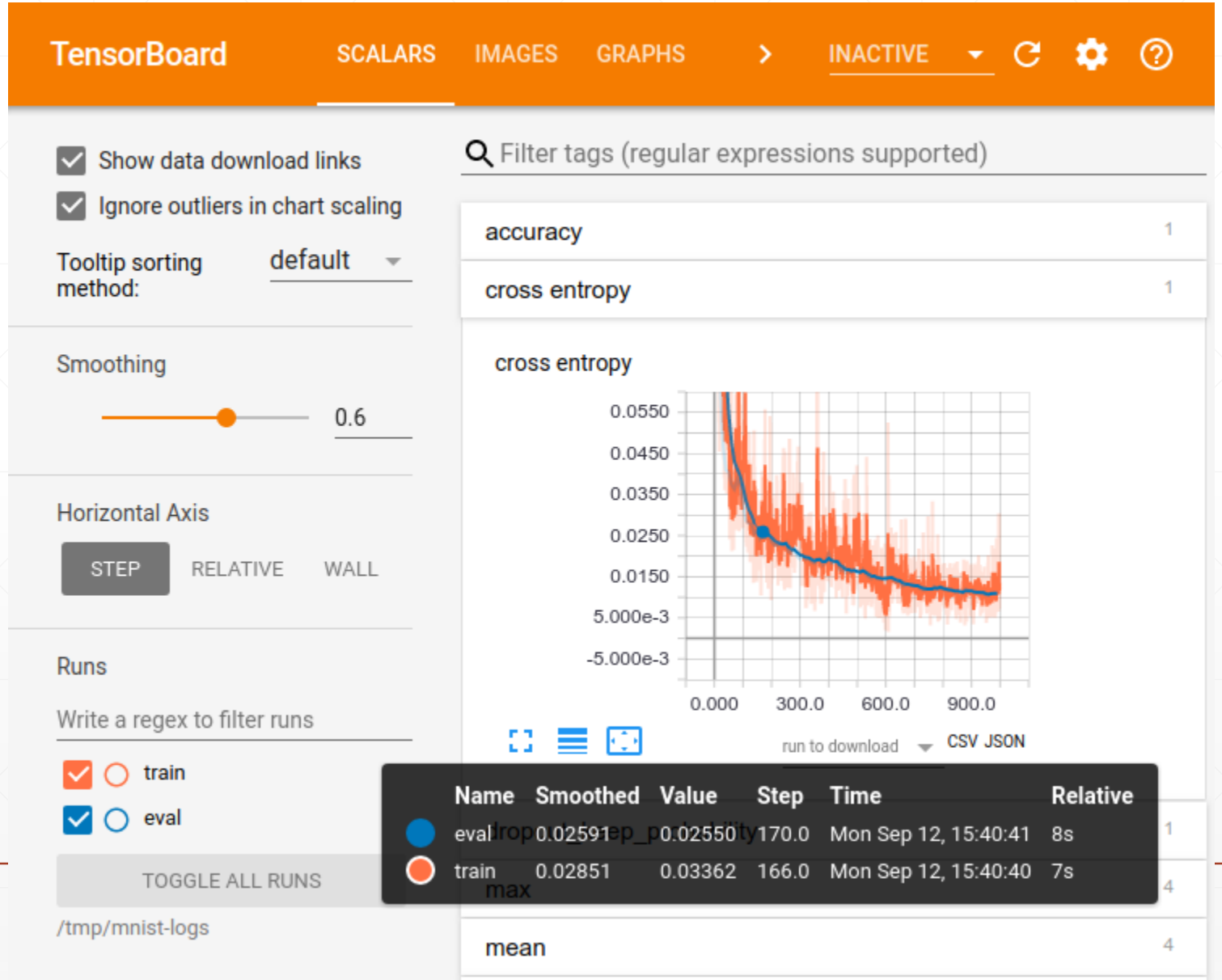
Thank You.



Visdom可视化

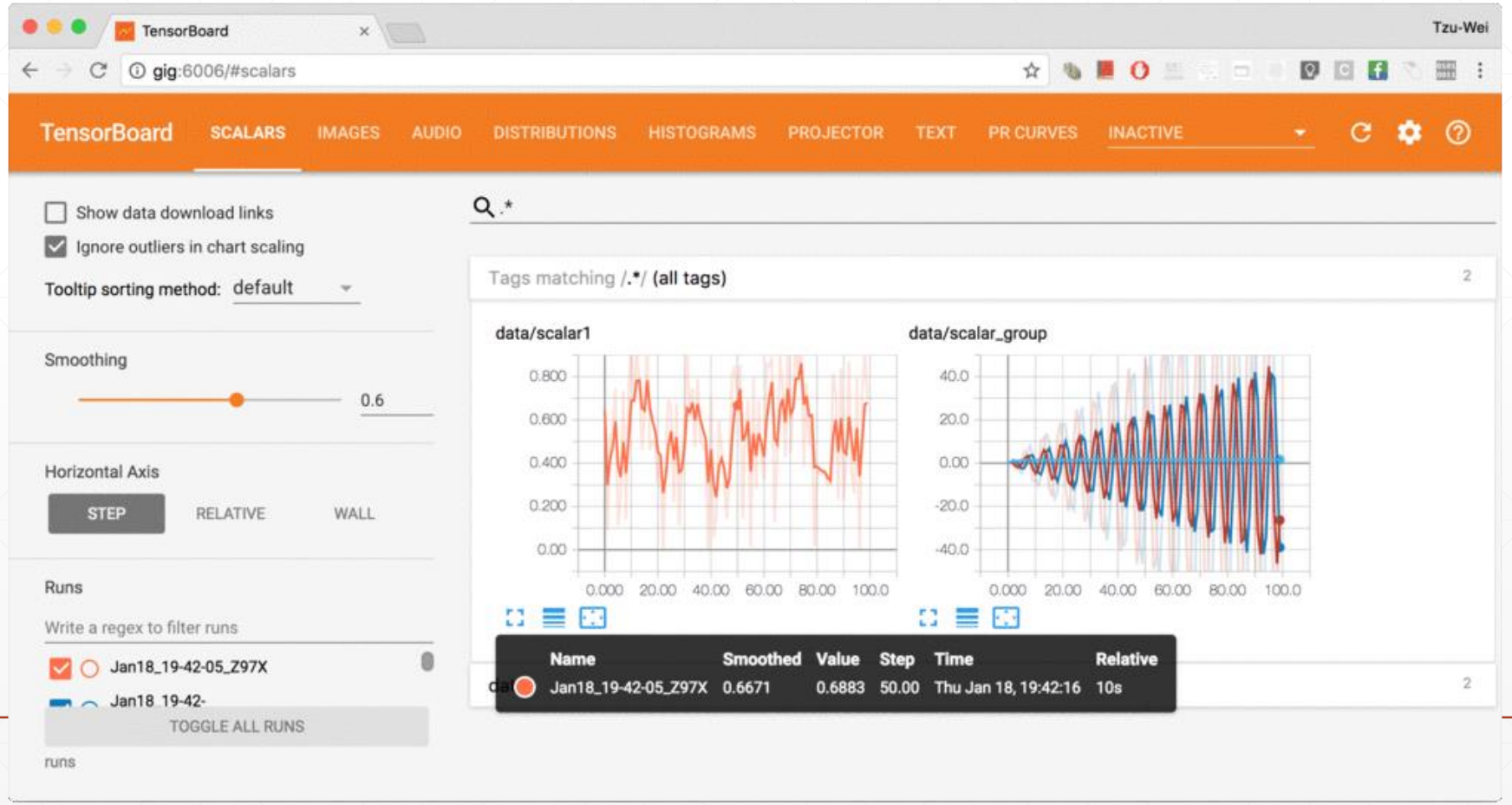
主讲人：龙良曲

TensorBoard?



TensorboardX

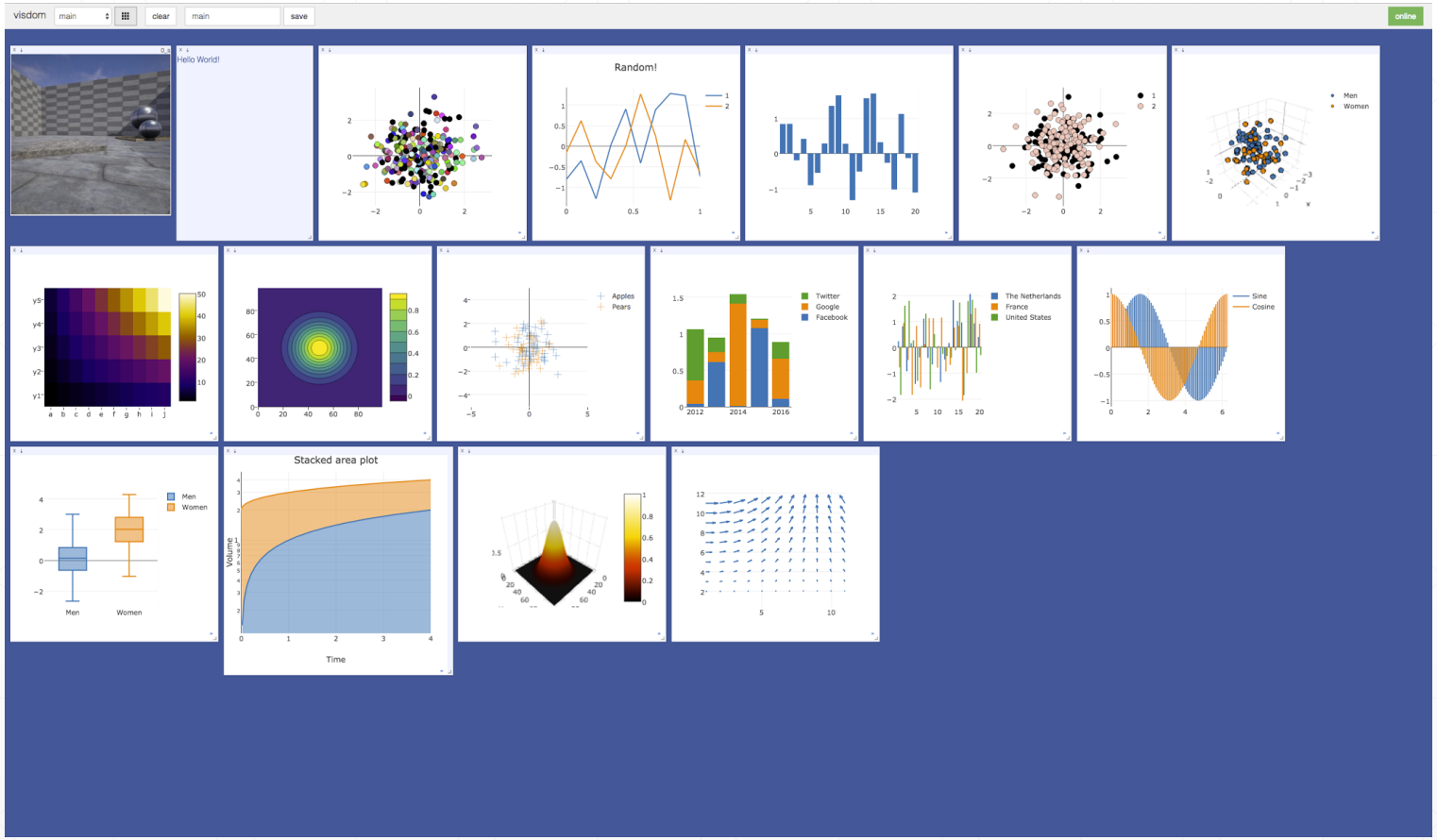
- pip install tensorboardX



TensorboardX

```
from tensorboardX import SummaryWriter

writer = SummaryWriter()
writer.add_scalar('data/scalar1', dummy_s1[0], n_iter)

writer.add_scalars('data/scalar_group', {'xsinx': n_iter * np.sin(n_iter),
                                         'xcosx': n_iter * np.cos(n_iter),
                                         'arctanx': np.arctan(n_iter)}, n_iter)


writer.add_image('Image', x, n_iter)
writer.add_text('Text', 'text logged at step:' + str(n_iter), n_iter)

for name, param in resnet18.named_parameters():
    writer.add_histogram(name, param.clone().cpu().data.numpy(), n_iter)

writer.close()
```

Visdom from Facebook



Step 1. install

```
C:\Users\drage>pip install visdom
Collecting visdom
  Downloading https://files.pythonhosted.org/packages/c1/48/d90e1519768107811fd6e7760bea46fff9e9c9ffb490441684003ae634a9
/visdom-0.1.8.5.tar.gz (248kB)
    100% |████████████████████████████████| 256kB 466kB/s
Requirement already satisfied: numpy>=1.8 in c:\programdata\conda\lib\site-packages (from visdom)
Requirement already satisfied: scipy in c:\programdata\conda\lib\site-packages (from visdom)
Requirement already satisfied: requests in c:\programdata\conda\lib\site-packages (from visdom)
Requirement already satisfied: tornado in c:\programdata\conda\lib\site-packages (from visdom)
Requirement already satisfied: pyzmq in c:\programdata\conda\lib\site-packages (from visdom)
Requirement already satisfied: six in c:\programdata\conda\lib\site-packages (from visdom)
Collecting torchfile (from visdom)
  Downloading https://files.pythonhosted.org/packages/91/af/5b305f86f2d218091af657ddb53f984ecbd9518ca9fe8ef4103a007252c9
/torchfile-0.1.0.tar.gz
Collecting websocket-client (from visdom)
  Downloading https://files.pythonhosted.org/packages/26/2d/f749a5c82f6192d77ed061a38e02001afcba55fe8477336d26a950ab17ce
/websocket_client-0.54.0-py2.py3-none-any.whl (200kB)
    100% |████████████████████████████████| 204kB 5.4MB/s
```

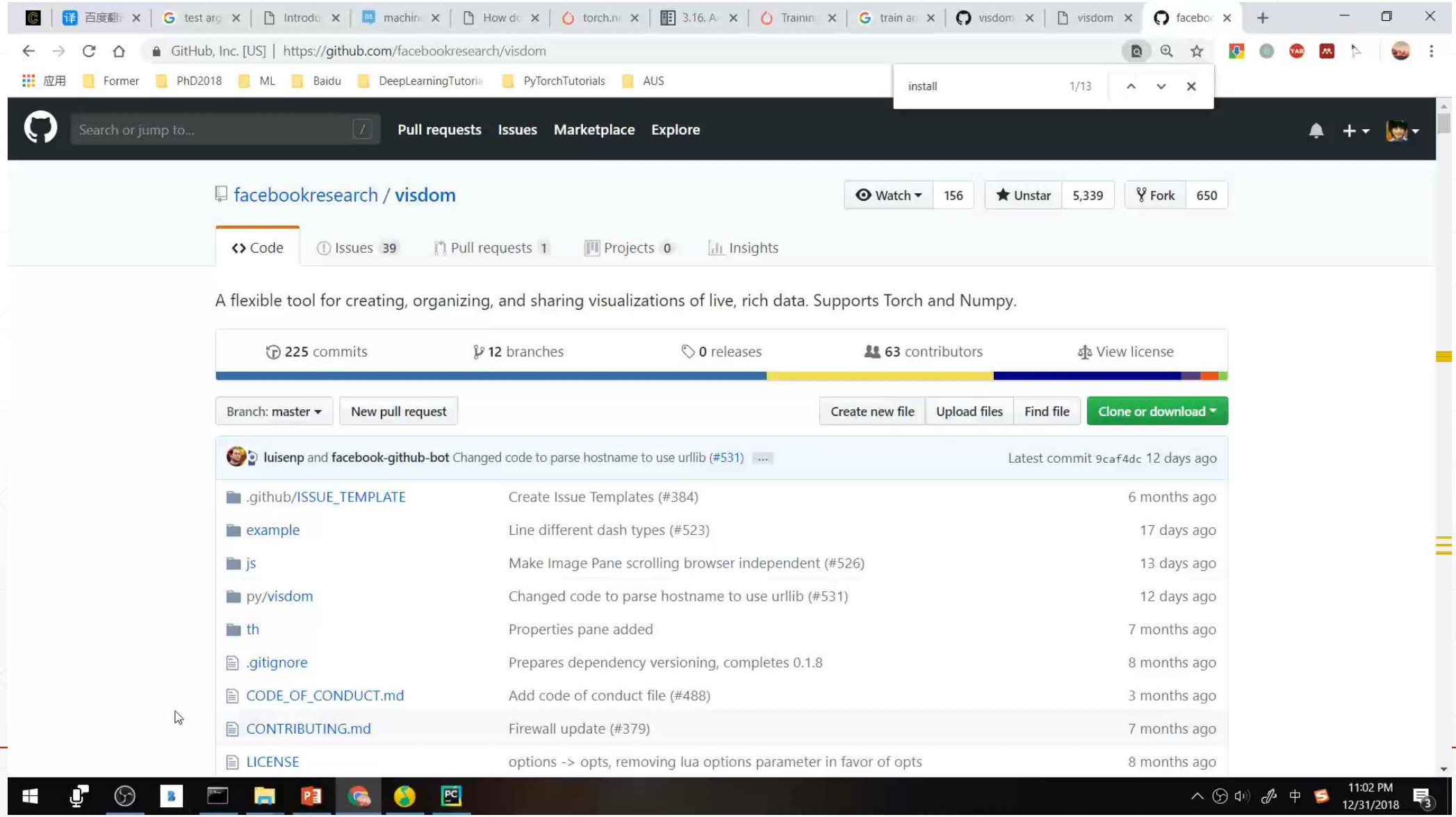
Step2. run server damon

```
C:\Users\drage>python -m visdom.server
Downloading scripts. It might take a while.
ERROR:root:Error 404 while downloading https://unpkg.com/layout-bin-packer@1.4.0
```

Step2. run server damon

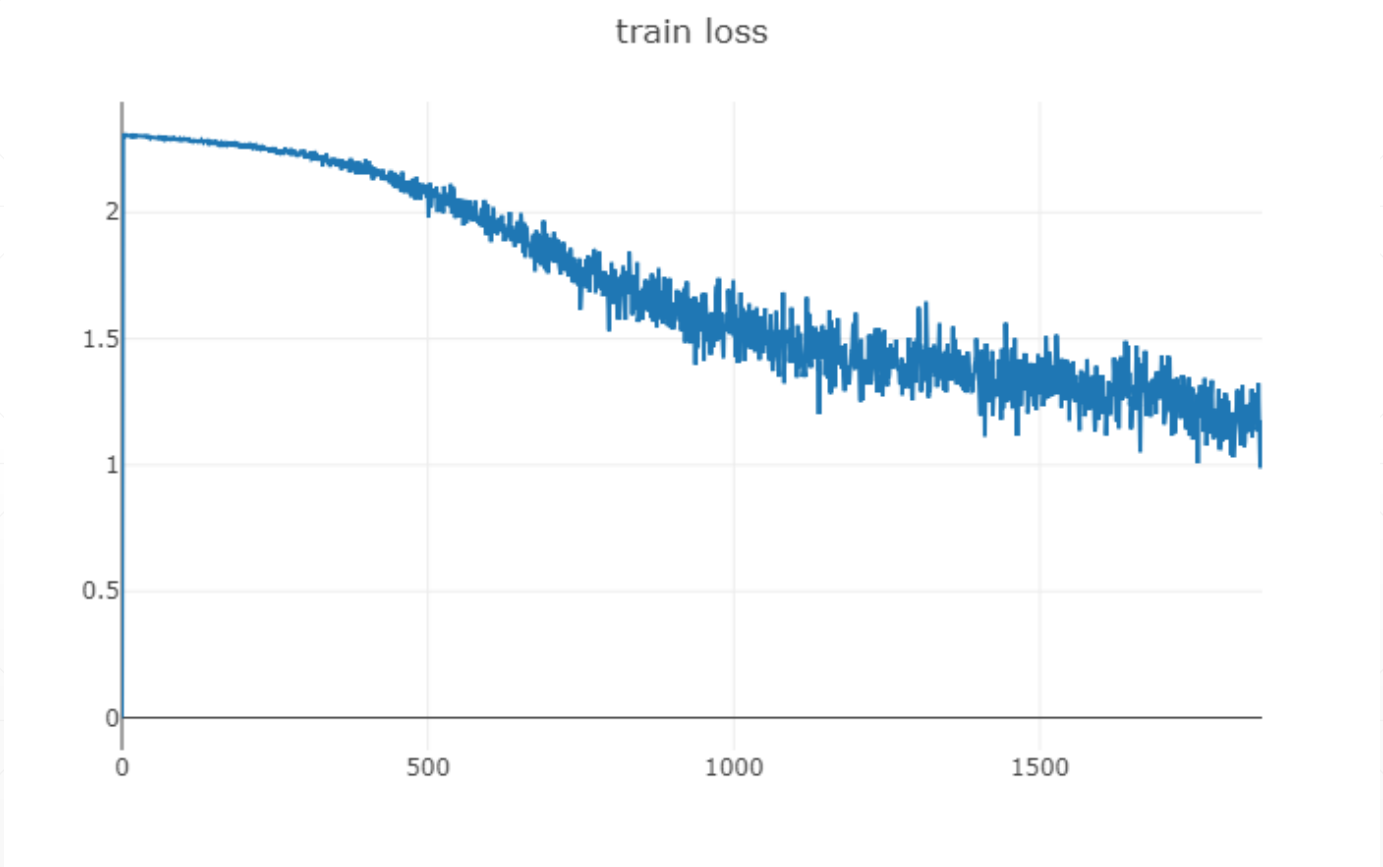
```
C:\Users\drage>python -m visdom.server
Downloading scripts. It might take a while.
ERROR:root:Error 404 while downloading https://unpkg.com/layout-bin-packer@1.4.0
It's Alive!
INFO:root:Application Started
You can navigate to http://localhost:8097
INFO:tornado.access:200 POST /win_exists (::1) 0.00ms
INFO:tornado.access:200 POST /events (::1) 1.00ms
INFO:tornado.access:200 POST /win_exists (::1) 0.00ms
INFO:tornado.access:200 POST /update (::1) 1.00ms
INFO:tornado.access:200 POST /win_exists (::1) 0.00ms
INFO:tornado.access:200 POST /update (::1) 1.03ms
```

install from source



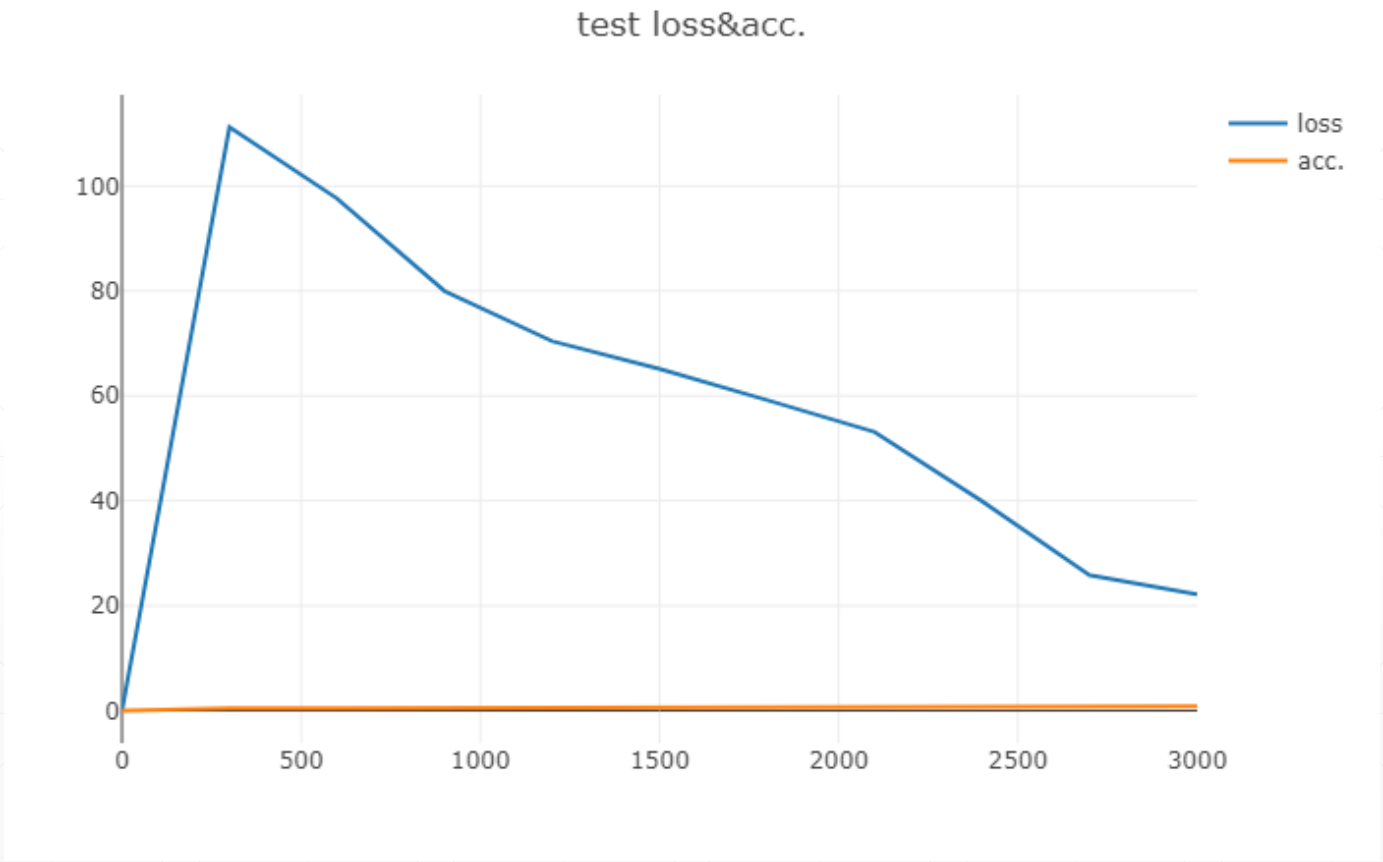
lines: single trace

```
from visdom import Visdom

viz = Visdom()

viz.line([0.], [0.], win='train_loss', opts=dict(title='train loss'))

viz.line([loss.item()], [global_step], win='train_loss', update='append')
```


train loss
2
1.5
1
0.5
0
0
500
1000
1500

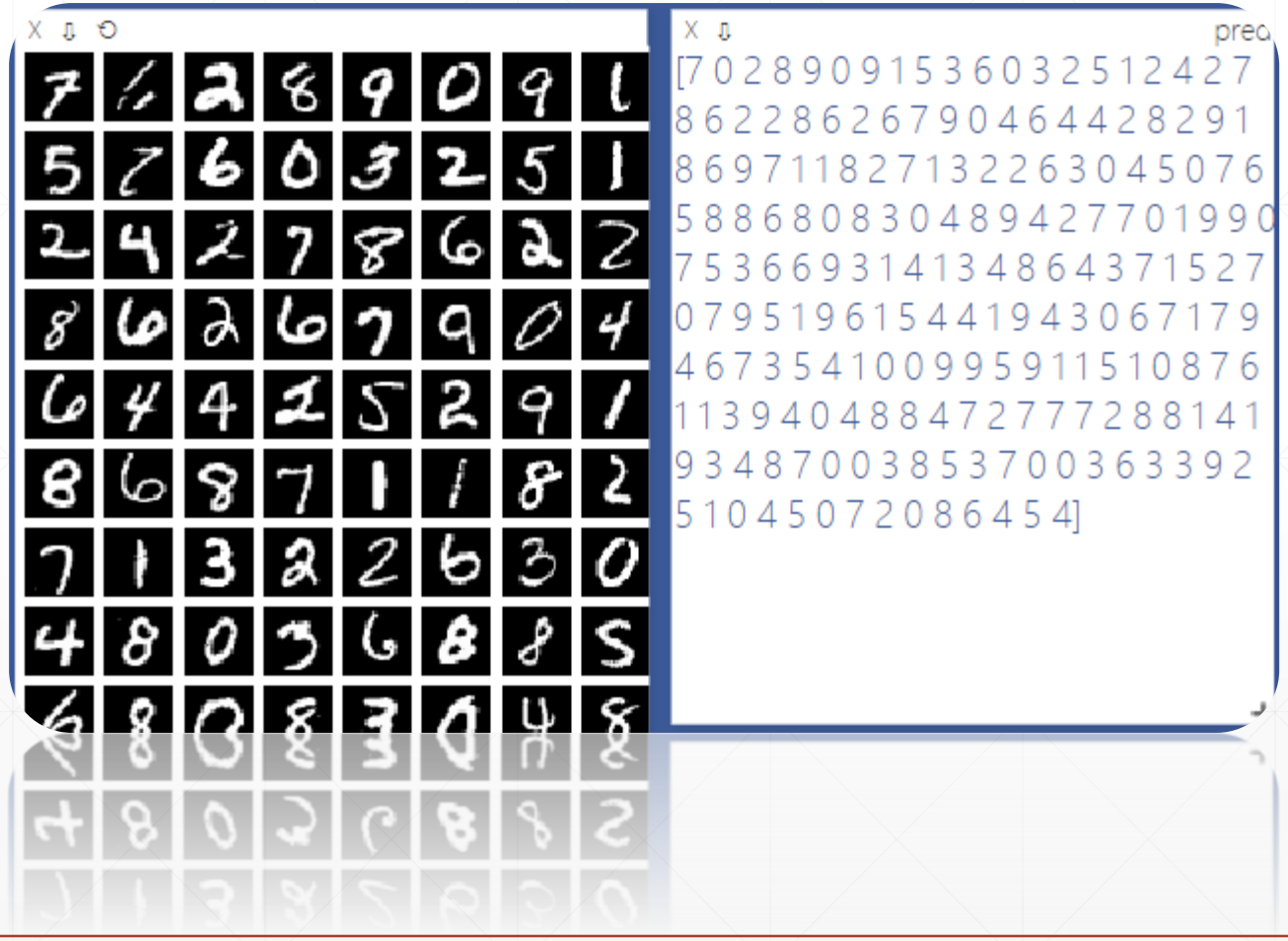
lines: multi-traces

```
from visdom import Visdom

viz = Visdom()
viz.line([[0.0, 0.0]], [0.], win='test', opts=dict(title='test loss&acc.',
                                                   legend=['loss', 'acc.']))

viz.line([[test_loss, correct / len(test_loader.dataset)]],
             [global_step], win='test', update='append')
```


test loss&acc.
loss
acc.
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000

visual X

```
from visdom import Visdom

viz = Visdom()

viz.images(data.view(-1, 1, 28, 28), win='x')
viz.text(str(pred.detach().cpu().numpy()), win='pred',
         opts=dict(title='pred'))
```


X
X
prec
[7 0 2 8 9 0 9 1 5 3 6 0 3 2 5 1 2 4 2 7
8 6 2 2 8 6 2 6 7 9 0 4 6 4 4 2 8 2 9 1
8 6 9 7 1 1 8 2 7 1 3 2 2 6 3 0 4 5 0 7 6
5 8 8 6 8 0 8 3 0 4 8 9 4 2 7 7 0 1 9 9 0
7 5 3 6 6 9 3 1 4 1 3 4 8 6 4 3 7 1 5 2 7
0 7 9 5 1 9 6 1 5 4 4 1 9 4 3 0 6 7 1 7 9
4 6 7 3 5 4 1 0 0 9 9 5 9 1 1 5 1 0 8 7 6
1 1 3 9 4 0 4 8 8 4 7 2 7 7 7 2 8 8 1 4 1
9 3 4 8 7 0 0 3 8 5 3 7 0 0 3 6 3 3 9 2
5 1 0 4 5 0 7 2 0 8 6 4 5 4]

下一课时

train-val-test

Thank You.