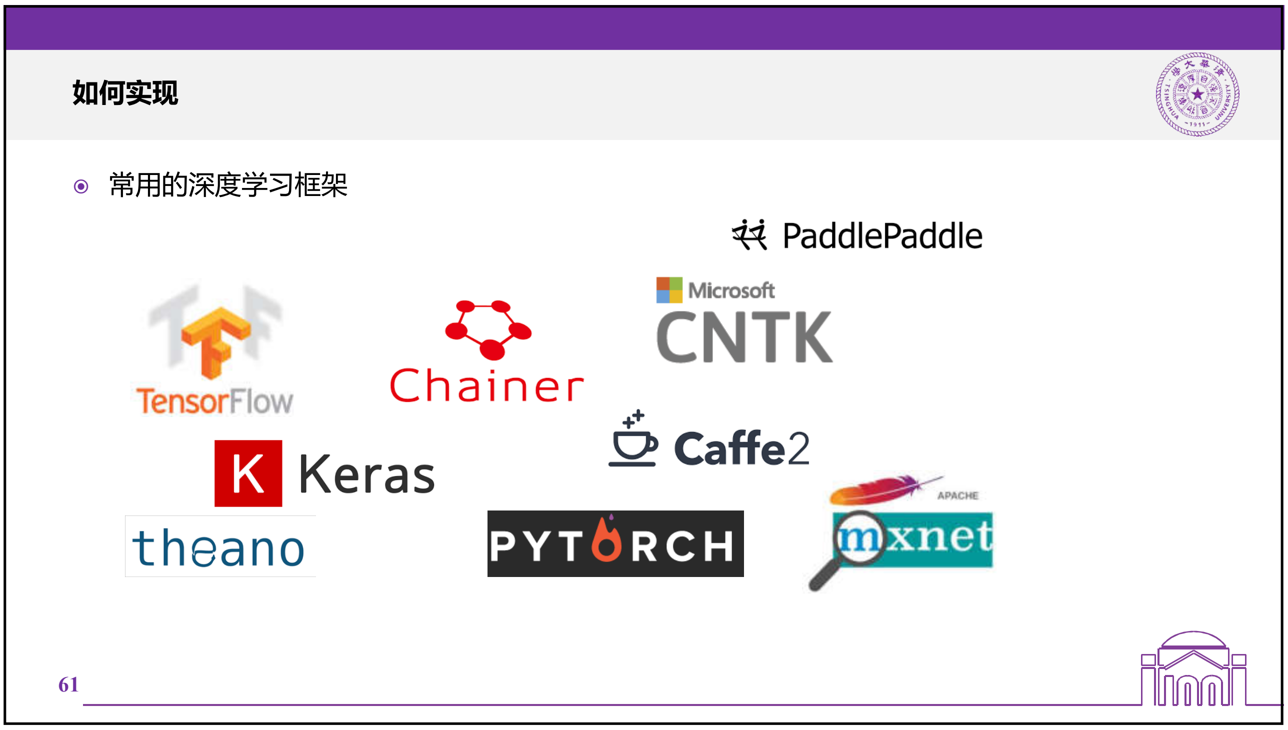


如何实现

- 常用的深度学习框架

PaddlePaddle

Microsoft CNTK

TensorFlow

Chainer

Caffe2

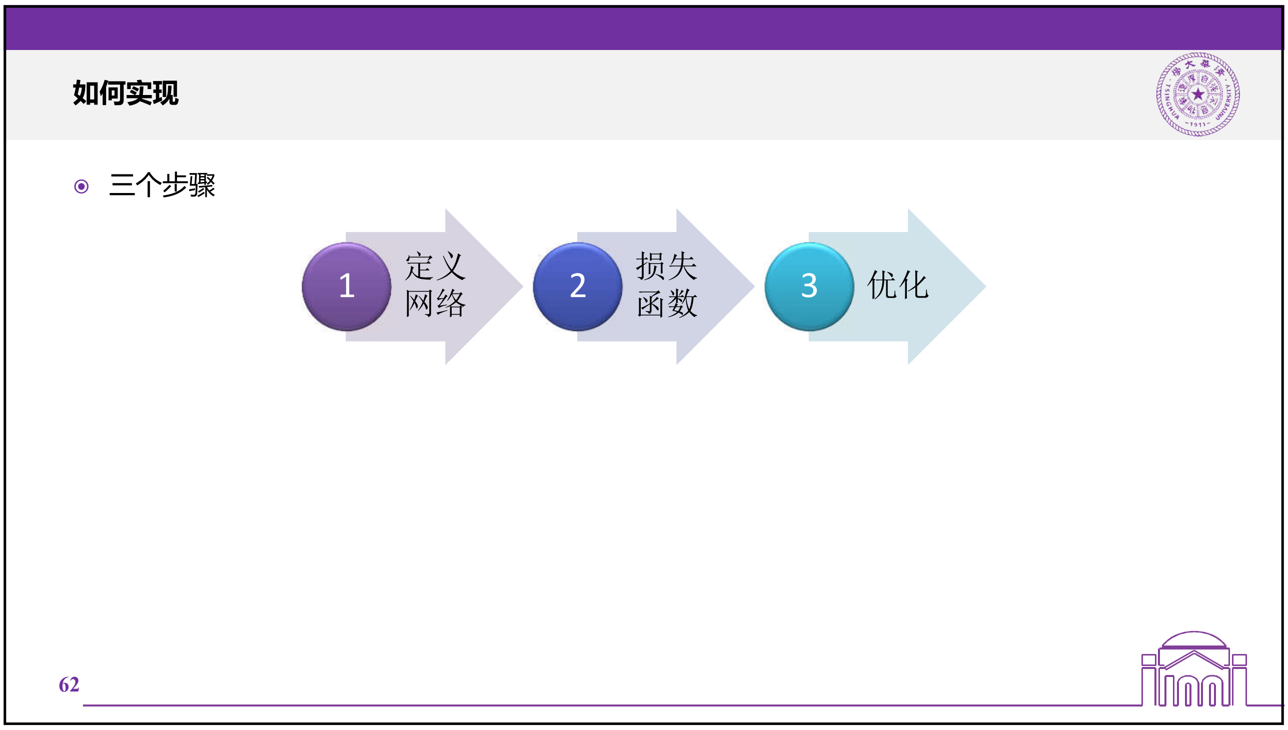
Keras

theano

PYTORCH

APACHE mxnet

61

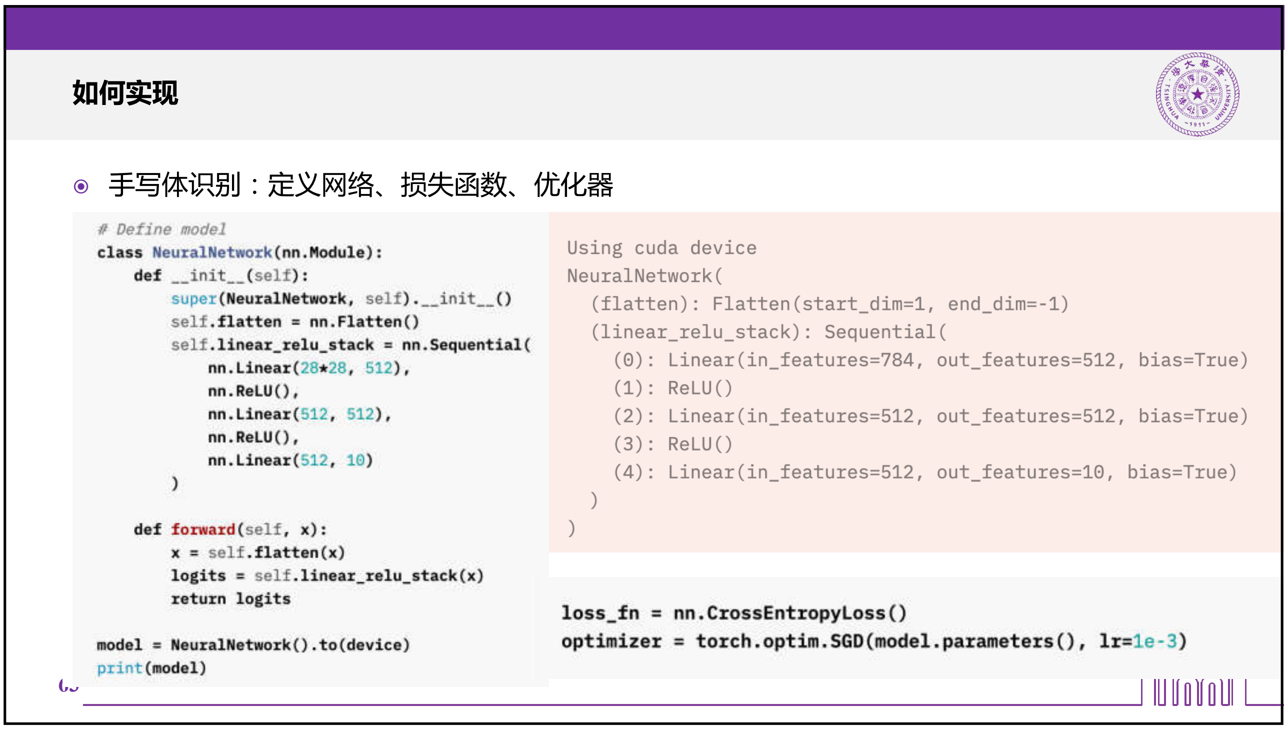


如何实现

- 三个步骤

1 定义网络 → 2 损失函数 → 3 优化

62



如何实现

- 手写体识别：定义网络、损失函数、优化器

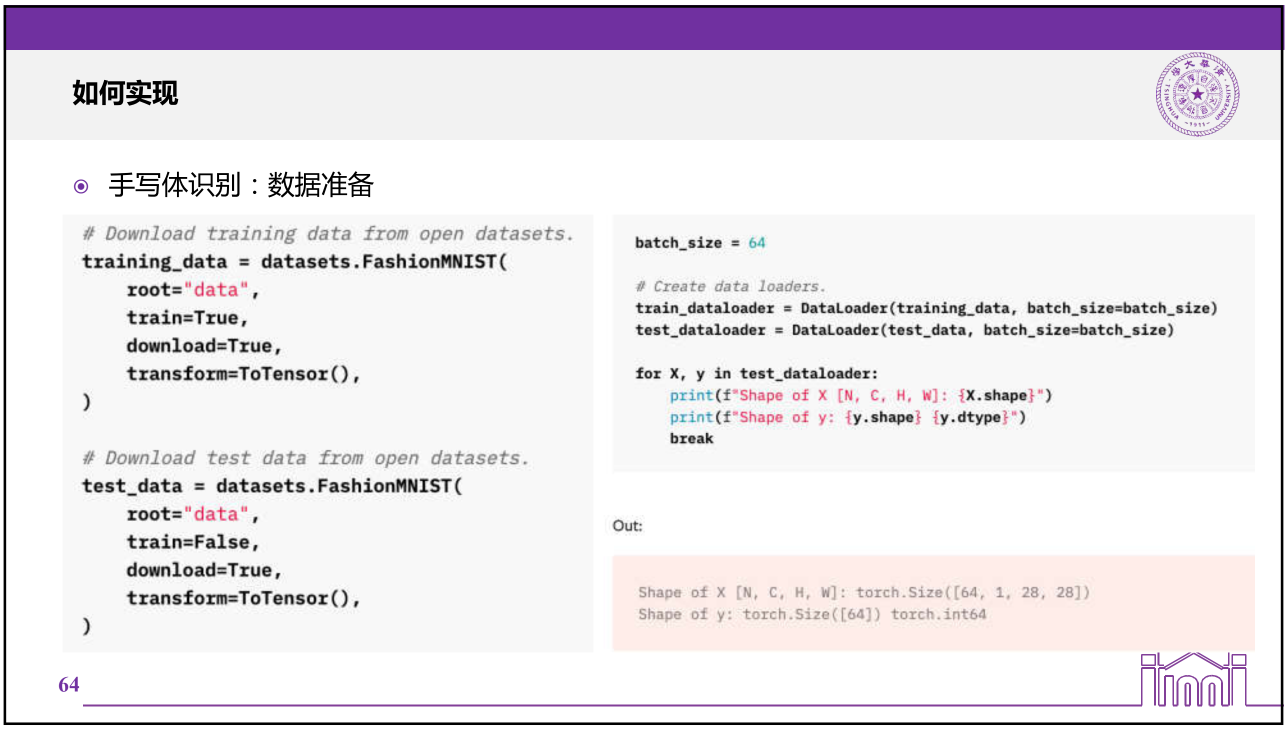
```python
# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
print(model)
```

```
Using cuda device
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

```python
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```



如何实现

- 手写体识别：数据准备

```python
# Download training data from open datasets.
training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor(),
)

# Download test data from open datasets.
test_data = datasets.FashionMNIST(
    root="data",
    train=False,
    download=True,
    transform=ToTensor(),
)
```

```python
batch_size = 64

# Create data loaders.
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

for X, y in test_dataloader:
    print(f"Shape of X [N, C, H, W]: {X.shape}")
    print(f"Shape of y: {y.shape} {y.dtype}")
    break
```

Out:

```
Shape of X [N, C, H, W]: torch.Size([64, 1, 28, 28])
Shape of y: torch.Size([64]) torch.int64
```

64

如何实现

- 手写体识别：模型训练和优化

```
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

```
Epoch 1
-------------------------------
loss: 2.290059  [    0/60000]
loss: 2.278689  [ 6400/60000]
loss: 2.260067  [12800/60000]
loss: 2.260366  [19200/60000]
loss: 2.251141  [25600/60000]
loss: 2.207952  [32000/60000]
loss: 2.221357  [38400/60000]
loss: 2.185180  [44800/60000]
loss: 2.179188  [51200/60000]
loss: 2.156274  [57600/60000]
Test Error:
 Accuracy: 50.4%, Avg loss: 2.1436
```

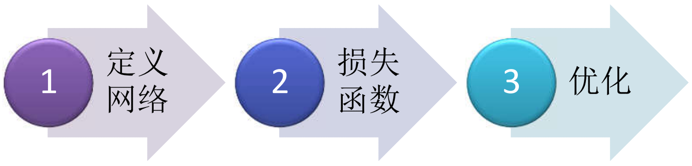
```
Epoch 2
-------------------------------
loss: 2.147471  [    0/60000]
loss: 2.136722  [ 6400/60000]
loss: 2.077863  [12800/60000]
loss: 2.101987  [19200/60000]
loss: 2.059406  [25600/60000]
loss: 1.982178  [32000/60000]
loss: 2.019614  [38400/60000]
loss: 1.936746  [44800/60000]
loss: 1.944287  [51200/60000]
loss: 1.871792  [57600/60000]
Test Error:
 Accuracy: 61.9%, Avg loss: 1.867935
```

```
Epoch 5
-------------------------------
loss: 1.300350  [    0/60000]
loss: 1.288346  [ 6400/60000]
loss: 1.112341  [12800/60000]
loss: 1.230672  [19200/60000]
loss: 1.108364  [25600/60000]
loss: 1.116301  [32000/60000]
loss: 1.159760  [38400/60000]
loss: 1.088508  [44800/60000]
loss: 1.133090  [51200/60000]
loss: 1.046162  [57600/60000]
Test Error:
 Accuracy: 65.1%, Avg loss: 1.066427
```

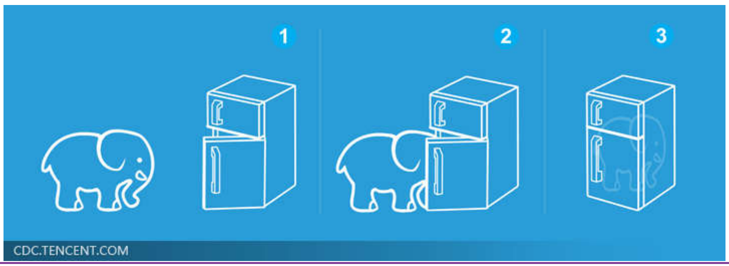
65

如何实现

- 三个步骤



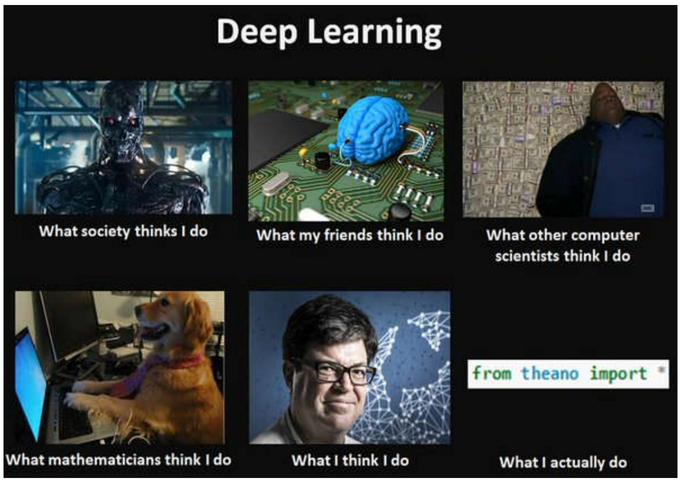
Deep Learning is so simple ……



66

如何实现

- 实现很简单！


Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
from theano import *
What I actually do

67

谢谢！

68