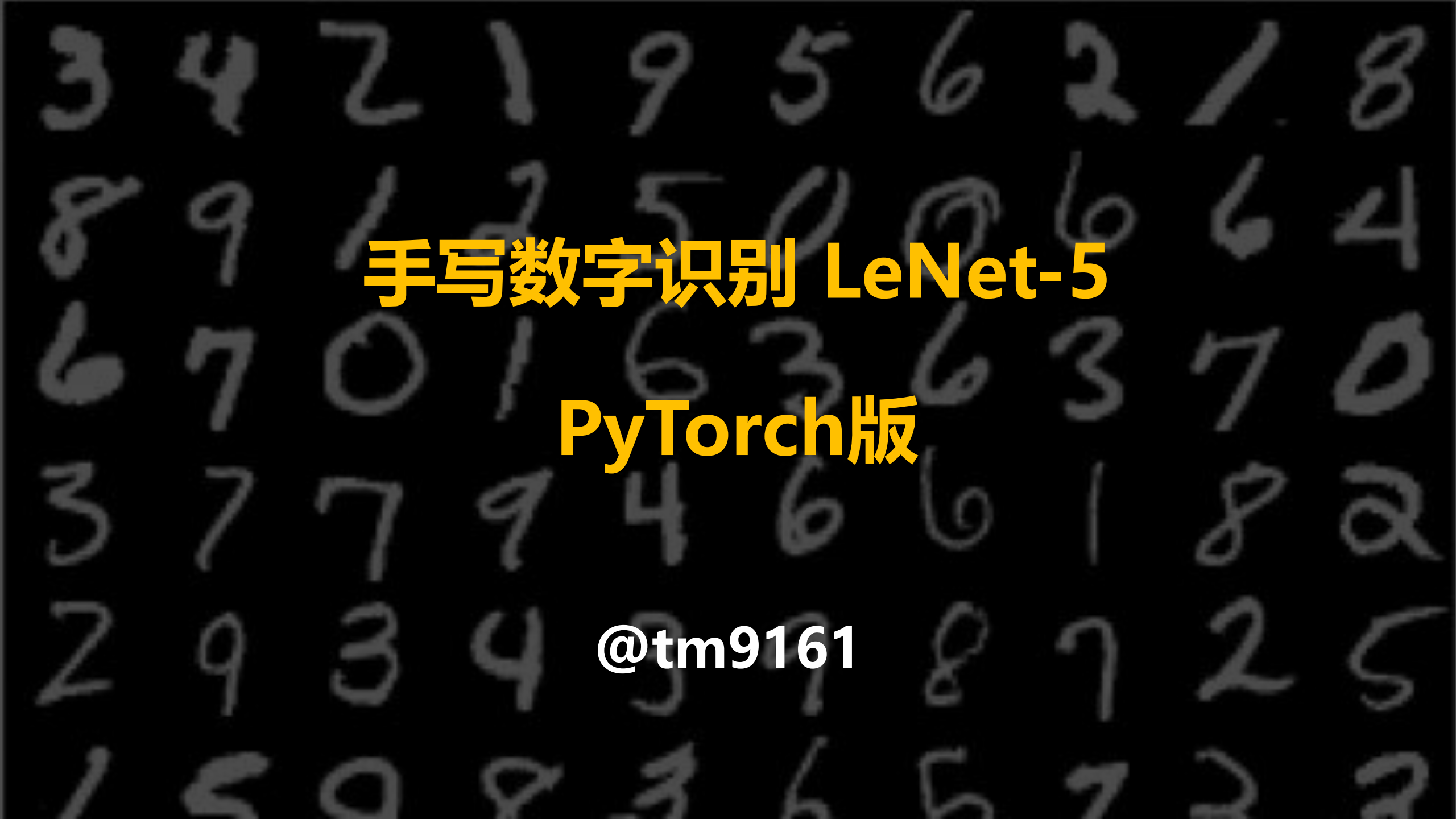


手写数字识别 LeNet-5

PyTorch版

@tm9161

tm9161 LV5 小爱否 15

重启试试

+ 关注 发消息

主页 动态 投稿 24 合集 2 搜索视频、动态

关注数	粉丝数	获赞数	播放数
190	1321	1678	9.4万

合集列表

合集·基于机器学习的目标检测 · 4

▶ 播放全部 查看更多 >

前往创作中心添加

YOLOv3 目标检测任务与数据集

2023-02-08

YOLOv3 模型训练

2024-02-20

YOLOv3 聚类确定锚框尺寸

2024-03-16

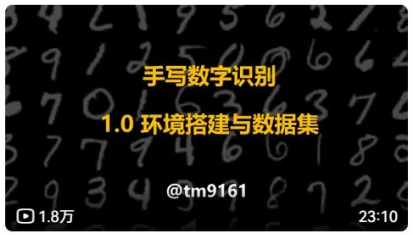
YOLOv3 模型预测

01-30

合集·基于机器学习的图像识别 · 10

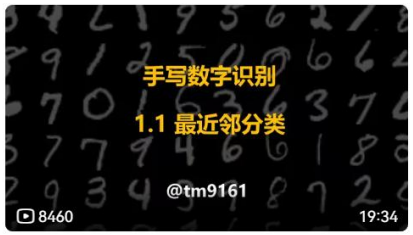
▶ 播放全部 查看更多 >

前往创作中心添加



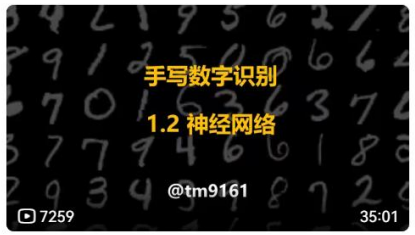
手写数字识别 1.0 环境搭建与数据集

2021-02-22



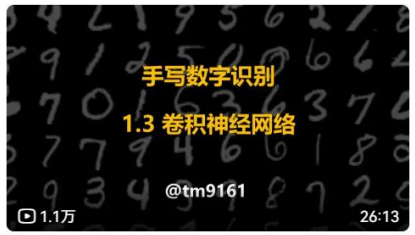
手写数字识别 1.1 最近邻分类

2021-02-26



手写数字识别 1.2 神经网络

2021-03-08



手写数字识别 1.3 卷积神经网络

2021-03-23

手写数字识别 1.4 LeNet-5

2021-03-30

LeNet-5

PyTorch版

1. PyTorch 数据处理
2. PyTorch 网络搭建与训练
3. PyTorch 网络预测

环境配置

1.驱动+CUDA11.8+cuDNN8.6.0

2. Python 3.8.12（conda）

3. PyTorch 2.1.2

v2.1.2

Conda

OSX

```
# conda
conda install pytorch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2 -c pytorch
```

Linux and Windows

```
# CUDA 11.8
conda install pytorch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2 pytorch-cuda=11.8 -c pytorch -c nvidia
# CUDA 12.1
conda install pytorch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2 pytorch-cuda=12.1 -c pytorch -c nvidia
# CPU Only
conda install pytorch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2 cpuonly -c pytorch
```

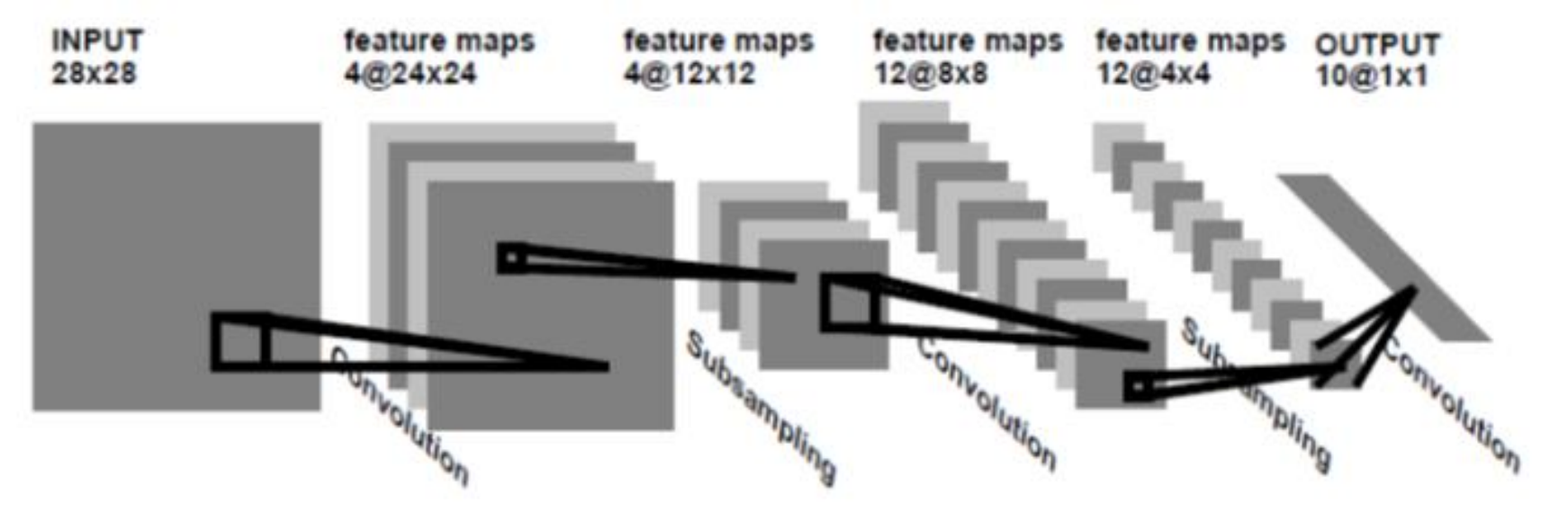
Wheel

OSX

```
pip install torch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2
```

https://pytorch.org/get-started/previous-versions/

LeNet-5 发展历史

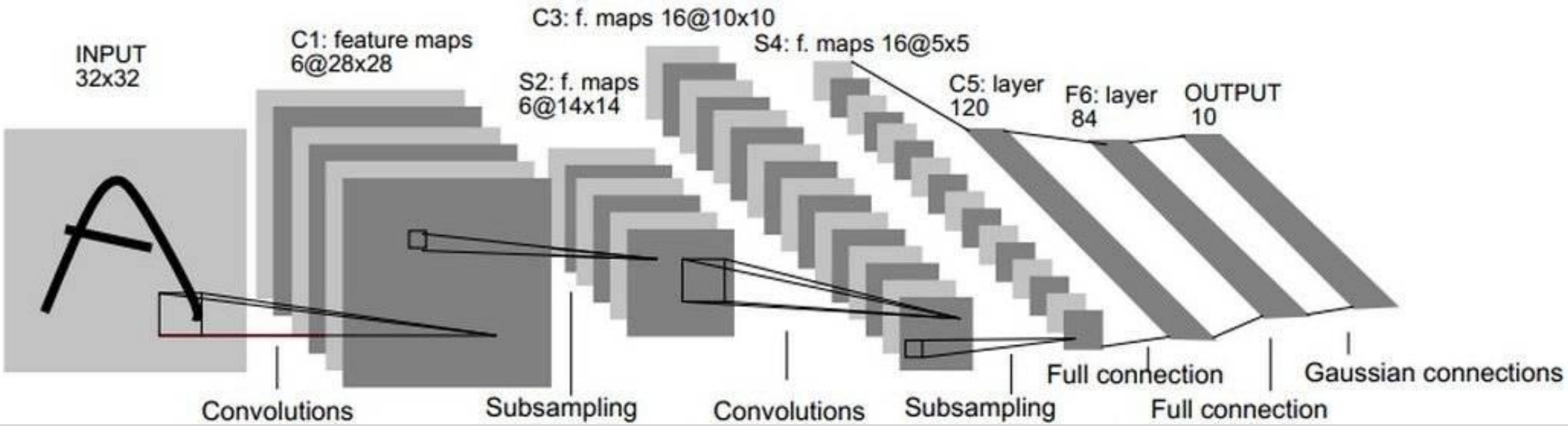


LeNet-1

1989年：Yann LeCun等人，结合反向传播算法的卷积神经网络来识别手写数字，并成功地用于识别手写邮政编码。

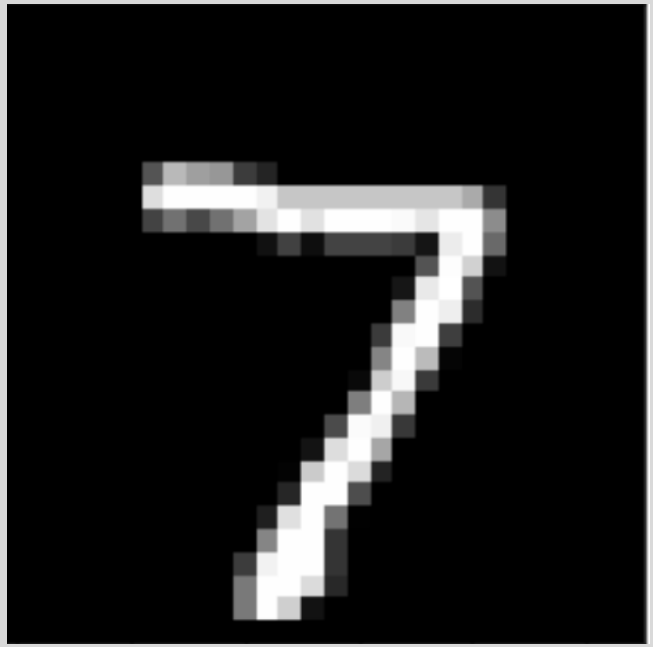
1990年：他们的模型在美国邮政总局提供的邮政编码数字数据的测试结果表明，错误率仅为1%，拒绝率约为9%。

1998年：他们将手写数字识别的各种方法在标准的手写数字识别基准上进行比较，结果表明他们的网络优于所有其他模型，经过多年的研究和迭代，最终发展成为LeNet-5。



	输入	卷积、池化、神经元	输出	训练参数
输入层*			32*32	0
卷积层1	32*32	6个 5*5卷积核 步长为1	6*28*28（32-5+0）/1+1	1*（5*5）*6 +6=156
池化层1	6*28*28	2*2 步长为2	6*14*14	0
卷积层2	6*14*14	16个 5*5卷积核 步长为1	16*10*10 （14-5+0）/1+1	6*（5*5）*16 +16=2416
池化层2	16*10*10	2*2 步长为2	16*5*5	0
全连接层1	16*5*5	120个 5*5卷积核 步长为1	120*1*1（5-5+0）/1+1	16*（5*5）*120+120=48120
全连接层2	120		84	120*84+84=10164
输出层	84		10	84*10+10=850

Pytorch 数据处理



1. 手写数据集：datasets.MNIST

2. 图像调整（尺寸调整、格式转换）：transforms.Compose

3. 数据迭代（批次数量、随机）：DataLoader

PyTorch 网络搭建

1.构建模型结构：nn.Conv2d(1, 6, kernel_size=5, padding=2)
nn.AvgPool2d(kernel_size=2)
nn.Linear(120, 84)

2.设计前向传播过程：x = torch.sigmoid(self.conv1(x))
x = self.pool1(x)
x = torch.sigmoid(self.fc1(x))

3.定义优化器和损失函数：nn.CrossEntropyLoss()
torch.optim.Adam

PyTorch 网络训练

1.设置轮次和切换训练模式：
```
for epoch in range(num_epochs):
    model.train()
```

2.前向传播；计算损失：
```
outputs = model(inputs)
loss = criterion(outputs, labels)
```

3.反向传播；更新参数：
```
loss.backward()
optimizer.step()
```

PyTorch 网络预测

1. 读取模型；切换评估模式：

```
model.load_state_dict(torch.load())
model.eval()
```

2. 读取图像；图像调整：

```
img = Image.open()
transforms.Compose
```

3. 关闭梯度；预测结果：

```
with torch.no_grad():
        output = model(img_tensor)
```

https://www.bilibili.com/opus/1063019665812357126

https://www.bilibili.com/video/BV1Z54y187WW