# 人工智能程序设计


python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for 1 in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)

# 人工智能程序设计

# 12.4 深度学习实践案例

北京石油化工学院 人工智能研究院

刘 强

# 本节概述

理论学习需要通过实践来巩固和深化

- 本节通过两个经典的深度学习项目进行实践
- 完成从数据准备到模型部署的完整流程
- 体验深度学习解决实际问题的全过程

# 12.4.1 图像分类实践：手写数字识别

**学习内容：**

- 数据加载与预处理
- 全连接神经网络模型设计
- 训练过程：损失函数、优化器、反向传播

# 示例 12.4.1：手写数字识别系统

**MNIST数据集**是深度学习领域最著名的入门数据集：

**项目目标**：构建一个能够准确识别手写数字的分类器（10分类问题）

| 特性 | 说明 |
| --- | --- |
| 图像数量 | 70,000张（60,000训练 + 10,000测试） |
| 图像尺寸 | 28×28像素 |
| 图像类型 | 手写数字（0-9） |
| 特点 | 图像简单、标注准确、数据量适中 |

# 数据加载与预处理

**数据预处理和加载**是项目的第一步：

**核心步骤**：

- **数据预处理**：将图像转换为张量格式，并进行标准化
- **数据加载器**：批量加载数据，支持随机打乱和并行处理
- **数据可视化**：查看样本数据，确保数据加载正确

```
## 数据预处理和加载
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))])
train_loader = DataLoader(datasets.MNIST('./data', train=True, transform=transform), batch_size=64)
```

# 模型设计：全连接神经网络

**全连接神经网络**采用简单的两层结构处理MNIST任务：

**结构特点**：

- **展平处理**：直接将2D图像转换为1D向量，丢弃空间信息
- **两层设计**：一个隐藏层(512维) + 一个输出层(10维)
- **参数规模**：约40万参数，相比CNN更加轻量

```
## 全连接网络结构
x = x.view(-1, 784)        # 图像展平：28×28 → 784
x = F.relu(fc1(x))         # 隐藏层：784 → 512
output = fc2(x)            # 输出层：512 → 10
```

# 训练过程：损失函数

**损失函数：误差计算**

损失函数衡量模型预测与真实标签之间的差异。

**CrossEntropyLoss特点**：

- 结合了Softmax和负对数似然损失
- 特别适合多分类任务
- 能够输出概率分布并计算分类误差

```
criterion = nn.CrossEntropyLoss()  # 多分类损失函数
loss = criterion(output, target)   # 计算预测误差
```

# 训练过程：优化器

**优化器：参数更新**

优化器根据梯度信息更新网络参数。

**Adam优化器特点**：

- 结合了动量和自适应学习率
- 自动调整每个参数的更新幅度
- 训练稳定且收敛快速

```
optimizer = optim.Adam(model.parameters(), lr=0.001)  # Adam优化器
optimizer.zero_grad()  # 清零梯度
optimizer.step()       # 更新参数
```

# 训练过程：反向传播

**反向传播：梯度计算**

反向传播通过链式法则计算损失函数对每个参数的梯度。

**完整训练流程**：

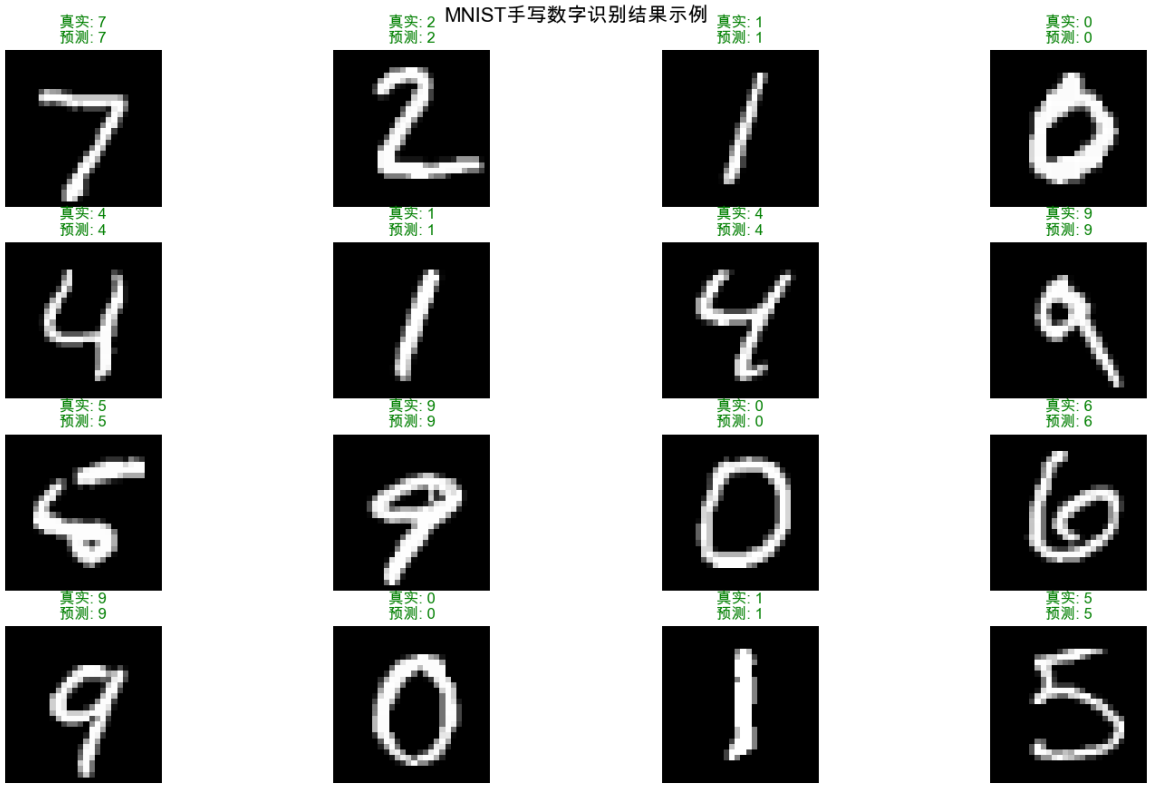
前向传播计算输出 → 损失函数计算误差 → 反向传播计算梯度 → 优化器更新参数

这个循环不断重复，使模型逐步学习数据中的模式。

```
loss.backward()  # 计算梯度
```

# MNIST手写数字识别效果

## 图 12.4.1 MNIST手写数字识别



# 模型评估与分析

**模型评估**验证训练好的模型在测试数据上的性能：

**评估要点**：

- **eval()模式**：关闭Dropout等训练特性，确保结果一致性
- **no_grad()**：节省内存和计算时间
- **准确率计算**：统计预测正确的样本比例

```
## 模型评估核心代码
model.eval()                                # 切换评估模式
with torch.no_grad():                       # 禁用梯度计算
    output = model(data)
    pred = output.argmax(dim=1)             # 获取预测类别
    accuracy = pred.eq(target).sum() / len(dataset)  # 计算准确率
```

# 12.4.2 图像分类进阶：AlexNet实现CIFAR-10分类

**学习内容：**

- CIFAR-10数据集介绍与MNIST对比
- 数据增强技术的应用
- AlexNet架构的PyTorch实现

# 项目升级与挑战

**CIFAR-10数据集**相比MNIST更具挑战性：

| 特性 | MNIST | CIFAR-10 |
| --- | --- | --- |
| 图像数量 | 70,000张 | 60,000张 |
| 图像尺寸 | 28×28 | 32×32 |
| 颜色 | 灰度 | 彩色 |
| 类别 | 数字0-9 | 飞机、汽车、鸟等10类 |
| 难度 | 入门级 | 进阶级 |

# 数据增强技术

**数据增强**通过对训练图像应用随机变换来人工扩充数据集：

**数据增强的作用**：

- 有效防止模型过拟合
- 提升在新数据上的泛化能力

```
transform_train = transforms.Compose([
    transforms.Resize(224),                    # 调整到AlexNet输入尺寸
    transforms.RandomCrop(224, padding=4),     # 随机裁剪
    transforms.RandomHorizontalFlip(),         # 随机水平翻转
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
])
```

# AlexNet架构实现：卷积层（一）

**第一、二卷积块**：多尺度特征提取

```
## 第一卷积块：大卷积核捕获粗粒度特征
conv1 = nn.Conv2d(3, 96, kernel_size=11, stride=4, padding=2)   # 224→55
pool1 = nn.MaxPool2d(kernel_size=3, stride=2)                   # 55→27

## 第二卷积块：中等卷积核细化特征
conv2 = nn.Conv2d(96, 256, kernel_size=5, padding=2)            # 27→27
pool2 = nn.MaxPool2d(kernel_size=3, stride=2)                   # 27→13
```

# AlexNet架构实现：卷积层（二）

**第三、四、五卷积层**：小卷积核提取细节特征

**设计理念**：按照多尺度设计，从11×11大卷积核到3×3小卷积核，实现层次化特征提取。

```
## 第三、四、五卷积层：小卷积核提取细节特征
conv3 = nn.Conv2d(256, 384, kernel_size=3, padding=1)          # 13→13
conv4 = nn.Conv2d(384, 384, kernel_size=3, padding=1)          # 13→13
conv5 = nn.Conv2d(384, 256, kernel_size=3, padding=1)          # 13→13
pool3 = nn.MaxPool2d(kernel_size=3, stride=2)                  # 13→6
```

# AlexNet架构实现：全连接层

**全连接层**采用三层递进式设计完成分类任务：

**说明**：只将最后一层从1000类ImageNet输出调整为10类CIFAR-10输出。

```
## 特征映射：从空间特征到抽象表示
fc1 = nn.Linear(256 * 6 * 6, 4096)    # 9216→4096，压缩空间信息

## 高级特征组合：学习复杂特征关系
fc2 = nn.Linear(4096, 4096)           # 4096→4096，保持特征维度

## CIFAR-10分类输出：适配10个类别
fc3 = nn.Linear(4096, 10)             # 4096→10，CIFAR-10类别数
```

# AlexNet关键技术总结

**关键技术**：

- **多尺度卷积**：11×11 → 5×5 → 3×3 渐进式特征提取
- **Dropout正则化**：防止全连接层过拟合
- **ReLU激活**：解决梯度消失问题

**完整流程**：

32×32图像 → 调整到224×224 → AlexNet多层卷积特征提取 → 全连接层分类 → 10类输出

# 12.4.3 Ask AI：深度学习前沿

**学习内容：**

- 顶级学术会议：NeurIPS、CVPR、ACL等
- 提升实践能力的方法
- 建立个人项目作品集

# 跟踪前沿研究

**跟踪前沿研究**的最佳方式是关注顶级会议论文：

这些会议的论文代表了该领域的最新进展。

| 领域 | 顶级会议 |
| --- | --- |
| 机器学习 | NeurIPS、ICML、ICLR |
| 计算机视觉 | CVPR、ICCV、ECCV |
| 自然语言处理 | ACL、EMNLP |

# 提升实践能力的方法

**参与开源项目**

- GitHub上有大量高质量的深度学习项目
- 通过阅读、修改、贡献代码，深入理解算法细节

**实践经典论文复现**

- 从零开始实现论文算法
- 加深对算法原理的理解

**建立个人项目作品集**

- 整理学习过程中完成的项目
- 包括项目描述、技术方案、实验结果

# 实践练习

**练习 12.4.1：Ask AI补充完整MNIST项目**

1. 补充完整的训练循环（包含进度显示和损失记录）
2. 添加数据可视化代码（显示样本图像和预测结果）
3. 实现混淆矩阵和分类报告分析
4. 添加模型保存和加载功能

**提示**：向AI描述："请帮我补充完整的MNIST训练循环，添加进度显示、损失记录、数据可视化和模型保存功能。"

# 实践练习

**练习 12.4.2：Ask AI实现CNN图像分类**

1. 使用Ask AI实现CIFAR-10图像分类项目

2. 设计CNN模型并应用数据增强技术

3. 实现训练过程监控和可视化

4. 完成模型性能评估和分析

**提示**：向AI请求："请帮我用PyTorch实现CIFAR-10图像分类，使用卷积神经网络，包含数据增强和完整的训练评估流程。"