Homework 2：判别式神经网络

522030910135 陈元杰

1. 数据集构建

我们从网上下载 CIFAR10 数据集作为测试和训练数据。CIFAR10 数据集是一个彩色三通道的图片数据集，一共有十类。它由 60000 张 $32\times32\times3$ 的图片组成，其中 50000 张为训练集，10000 张为测试集。CIFAR10 数据集示例如下图所示：

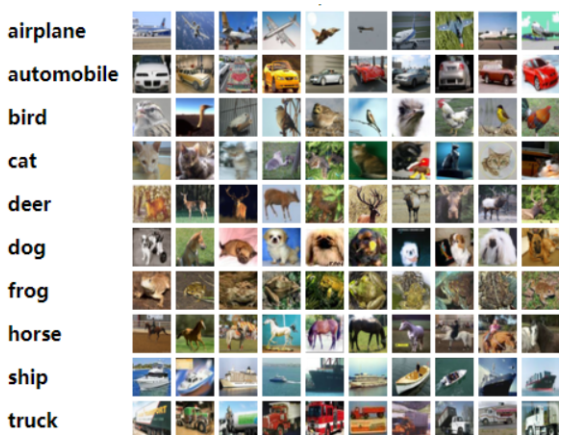

图 1: *CIFAR10* 数据集

我们利用 pickle 模块读取图片和标签的数据，并将图片读入为形状为 $(3, 32, 32)$ 的张量。

2. 神经网络模型

2.1. CNN 卷积神经网络

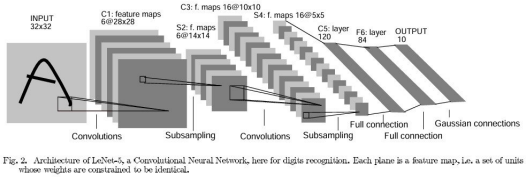

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

图 2: 卷积神经网络

卷积神经网络（Convolutional Neural Network，简称 CNN）是一种深度学习模型，广泛应用于图像识别和分类等任务中。CNN 通过使用卷积层（convolutional layers）、池化层（pooling layers）和全连接层（fully connected layers）来自动提取和学习输入数据中的特征。

卷积层是 CNN 的核心组件，通过卷积运算来提取输入数据中的局部特征。卷积运算通过卷积核（filters 或 kernels）在输入数据上滑动，并计算点积，从而生成特征图（feature map）。卷积层的输出可以表示为：

$$\mathbf{X}_{l+1}^{(k)} = f\left(\sum_{j=1}^{N_l} \mathbf{W}_l^{(k,j)} * \mathbf{X}_l^{(j)} + \mathbf{b}_l^{(k)}\right)$$

其中，$\mathbf{X}_l^{(j)}$ 是第 l 层的第 j 个输入特征图，$\mathbf{W}_l^{(k,j)}$ 是第 l 层的第 k 个卷积核，$\mathbf{b}_l^{(k)}$ 是偏置，f 是激活函数。

对于 CIFAR10 的分类任务，我们通过构建 4 层卷积网络，并以 RELU 作为激活函数并最大池化，然后通过一个线性层作为 CNN 网络结构。具体网络结构如下面的代码所示：

```
        self.layer = nn.Sequential(
                nn.Conv2d(3,32,(5,5),padding=2),
                nn.ReLU(),
                nn.Conv2d(32,16,(3,3),padding=1),
                nn.ReLU(),
                nn.MaxPool2d(2),
                nn.Conv2d(16,64,(5,5),padding=2),
                nn.ReLU(),
                nn.Conv2d(64,16,(3,3),padding=1),
                nn.MaxPool2d(2),
                nn.Flatten(),
                nn.Linear(16*8*8,10)
        )
```

2.2. RNN 循环神经网络

循环神经网络（Recurrent Neural Network，简称 RNN）是一类用于处理序列数据的神经网络，广泛应用于自然语言处理、时间序列预测和语音识别等任务中。RNN 的核心思想是通过隐状态（hidden state）来捕捉序列数据中的依赖关系，从而能够处理变长的输入序列。

RNN 与传统的前馈神经网络不同，其隐状态能够在序列的每个时间步共享参数并传递信息。RNN 的基本单元可以表示为：

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

其中，h_t 是时间步 t 的隐状态，x_t 是时间步 t 的输入，W_{hx} 和 W_{hh} 是权重矩阵，b_h 是偏置，f 是激活函数（如 tanh 或 ReLU）。输出 y_t 可以表示为：

$$y_t = g(W_{hy}h_t + b_y)$$

其中，W_{hy} 是输出权重矩阵，b_y 是输出偏置，g 是输出激活函数（如 softmax）。

RNN 的训练通常使用反向传播算法的扩展版本，称为反向传播通过时间（Backpropagation Through Time，BPTT）。BPTT 通过展开时间维度，将序列数据视为多个时间步的前馈神经网络，然后计算梯度并更新参数。下面是 RNN 训练的伪代码：

Algorithm 1 RNN Training Using BPTT

Require: Training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, learning rate α, iterations num_iters

Ensure: Parameters θ

Initialize θ randomly
for $iter = 1$ to num_iters **do**
  for each (x, y) **do**
    Initialize $h_0 = 0$
    for $t = 1$ to T **do**
      $h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$
      $\hat{y}_t = g(W_{hy}h_t + b_y)$
    end for
    Compute loss $L = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$
    BPTT:
    for $t = T$ to 1 **do**
      Accumulate gradients
    end for
    Update θ using accumulated gradients
  end for
end for
return θ

对于图像分类的任务，我们试图将图像的每一行看成序列向量，然后利用 RNN 来预测得到的结果。我们的 RNN 结构代码如下所示：

```
def __init__(self):
    super(RNN, self).__init__()
    self.layer1 = nn.RNN(96,512,2)
    self.layer2 = nn.Linear(512,10)
```

3. 实验结果与分析

3.1. CNN 实验结果

对于 CNN 网络，我们采用随机梯度下降法，以交叉熵为损失函数，将训练数据分成每个 batch1024 个数据，并进行梯度下降。我们设置学习率为 0.01，训练 100 个 epoch，下图显示了 loss 随训练轮次变化的曲线图。

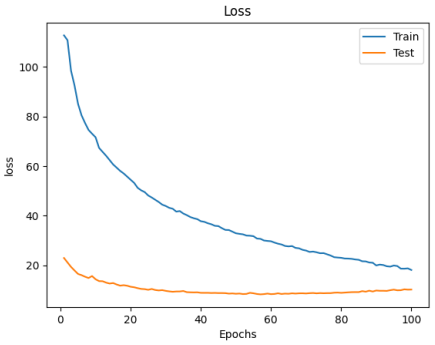

图 3: *Loss* 变化图

以及测试每一轮次我们的模型在训练集和测试集上的准确率的变化，做出如下的曲线：

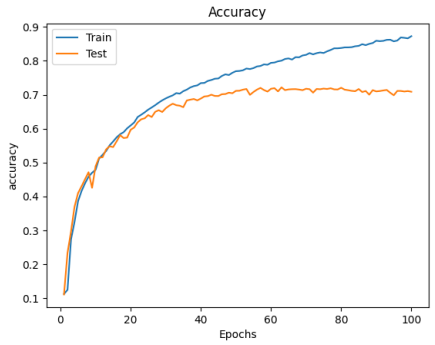

图 4: 准确率变化图

最终在训练集和测试集上的准确率如下表所

示：

表 1: *CNN* 实验结果

	训练集	测试集
准确率	86.4%	71.9%

3.2. RNN 实验结果

对于 RNN 网络，我们同样将训练集分成每个 batch1024 张图片进行随机梯度下降，将交叉熵作为损失函数。我们设置学习率为 0.01，训练 200 个 epoch，做出 loss 随训练轮次变化的曲线图：

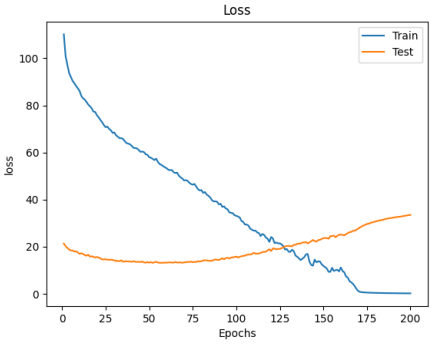

图 5: *loss* 变化图

对于每一轮次，我们测试出模型在训练集和测试集上的准确率变化，并做出如下曲线：

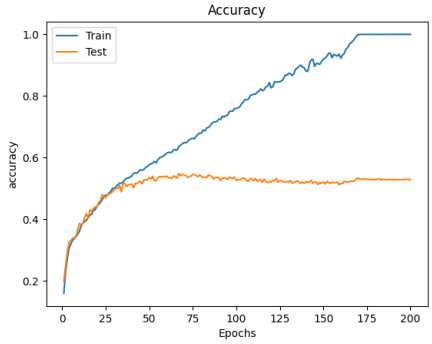

图 6: 准确率变化图

训练结束时，模型在训练集和测试集上的准确率如下表所示：

表 2: *RNN* 实验结果

	训练集	测试集
准确率	100%	52.8%

从上面的表格中可以看出，训练集的准确率可以达到 100%，而测试集的准确率只有 52.8%，产生了过拟合。这是由于 RNN 适用于处理时序的数据，而图片的结构信息无法完全通过图片上下行的时序信息来显示。

4. 长尾数据集

4.1. 实验结果

我们只保留训练集中前五类图片中的 10%，剩下五类图片不变。基于该训练集进行训练，仍然维持学习率为 0.01，每个 batch1024 张图片，以交叉熵为损失函数进行随机梯度下降，训练 200 个 eopch，分别得到 CNN 和 RNN（不改变原来模型的结构）的准确率如下表所示：

表 3: 长尾数据集：准确率

	训练集	测试集
CNN	100%	53.4%
RNN	97.3%	42.0%

从上面的表格中可以发现，相比于之前的数据集，在这个长尾的数据集上训练的 CNN 和 RNN 模型的准确率均有所下降，并且都出现了比原先更为严重的过拟合现象。

这是由于前面五类样本数量显著低于后面五类样本数量，模型在后面五类样本上训练”过度“导致的。模型在前面五类上的准确率显著低于在后面五类中的准确率，使得总体的准确率显著降低。

4.2. 数据增强

对于长尾的数据集来说，由于各类样本数量的不均衡，一般采用对尾部类别过采样和头部类

别欠采样的方法。为了充分利用数据集的数据，我们采用对尾部数据进行数据增强的方式。

我们通过对图片进行水平翻转以及加随机高斯噪声的方式进行数据增强。对于图片矩阵 X

$$X \leftarrow X + \epsilon$$

在上面的式子中，$X \sim N(0, \sigma^2)$ 是服从正态分布的高斯噪声。对于前五类的数据，我们进行水平翻转和加高斯噪声的方式使它们的样本数量与后五类相等。

4.3. 模型改进

为了更好的提高模型的泛化能力，减少过拟合，我们改进模型的结构，引入残差神经网络 Resnet。ResNet 的核心是残差模块（Residual Block）。在传统的神经网络中，随着网络深度的增加，训练误差会趋于饱和甚至上升。残差模块通过引入短连接（skip connections）或跳跃连接，使得网络可以直接学习残差（residual），而不是直接学习输入到输出的映射。这种设计可以显著缓解梯度消失问题。

一个典型的残差模块可以表示为：

$$y = F(x, \{W_i\}) + x$$

其中，x 是输入，y 是输出，$F(x, \{W_i\})$ 表示卷积层、批归一化层和 ReLU 激活函数的组合。短连接直接将输入 x 加到输出 $F(x, \{W_i\})$ 上。

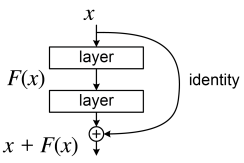

图 7: 残差神经网络

ResNet 可以堆叠多个残差模块形成深层网络，如 ResNet-50、ResNet-101、ResNet-152 等。ResNet-50 表示包含 50 个卷积层的网络。ResNet 网络的总体结构包括一个卷积层、若干残差模块和一个全连接层，具体架构如下：

- **卷积层**：7x7 卷积，64 个滤波器，步幅为 2，后接批归一化和 ReLU 激活。
- **池化层**：3x3 最大池化，步幅为 2。
- **残差模块**：若干残差模块堆叠。以 ResNet-50 为例，包括四个阶段，每个阶段包含不同数量的残差模块。
- **全连接层**：平均池化层后接全连接层，输出类别数。

我们采用 Resnet50 网络并在最后加入线性层，在数据增强之后的训练集上进行训练。以 0.01 为学习率，每个 batch1024 张图片，进行随机梯度下降，训练 10 个 epoch，可以得到下面 loss 随训练轮次变化曲线。

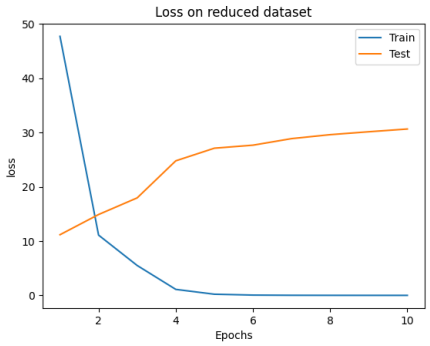

图 8: *loss* 变化曲线

对于 RNN 网络，我们在数据增强后的训练集以 0.01 学习率进行训练，训练 50 个 epoch，得到最终的准确率如下表所示：

表 4: 数据增强后的长尾数据集：准确率

	训练集	**测试集**
CNN(Resnet)	98.2%	67.4%
RNN	94.9%	45.4%

可以发现经过改进后准确率有了明显的提升。