Pie-Lab 2025年暑期培训 Practice1

徐浩
Pie-Lab
北京理工大学计算机学院
北京市海淀区中关村南大街
526358612@qq.com

Abstract

本文是“任务一：图片分类”和“任务二：文本分类”的实验报告。

1 图片分类

1.1 各网络结构的原理解释

1.1.1 残差连接

在传统的神经网络中，当不断加深网络深度，会面临：梯度消失/爆炸、训练误差反而上升、训练难以收敛等问题。

引入残差连接可以解决这一问题，设输入为x，网络为F，则输出为：

$$F(x) + x$$

在反向传播时，对x求导，其导数就不会趋近于0，也就规避了梯度消失问题。

1.1.2 resnet34

resnet34是resnet系列中的一个模型，由多个残差块组成。其网络结构的文字表述见图1。

Layer Name	Output Size	Building Blocks	Stride
Conv1	112x112	7x7 conv, 64, stride 2	2
MaxPool	56x56	3x3 maxpool, stride 2	2
Conv2_x	56x56	[3x3, 64] x 3	1
Conv3_x	28x28	[3x3, 128] x 4	2
Conv4_x	14x14	[3x3, 256] x 6	2
Conv5_x	7x7	[3x3, 512] x 3	2
AvgPool + FC	1x1 → 1000		

Figure 1: resnet34_architecture

Conv1: 1 层

BasicBlock = 2 层卷积

Residual Blocks：

Conv2_x: 3 blocks × 2 = 6

Conv3_x: 4 blocks × 2 = 8

Layer Name	Output Size	Building Blocks	Stride	修改说明
Conv1	32x32	3x3 conv, 64, stride 1	1	替换原来的7x7+stride 2
MaxPool	❌移除			避免过度下采样
Layer1	32x32	[3x3, 64] x 3	1	
Layer2	16x16	[3x3, 128] x 4	2	下采样
Layer3	8x8	[3x3, 256] x 6	2	下采样
Layer4	4x4	[3x3, 512] x 3	2	下采样
AvgPool + FC	1x1 → 10			输出 CIFAR-10 分类

Figure 2: resnet34_architecture_cifar

Conv4_x: 6 blocks × 2 = 12

Conv5_x: 3 blocks × 2 = 6

fc = 1 层

合计：1 + 6 + 8 + 12 + 6 + 1 = 34 层

要想将resnet34用到CIFAR-10，需要对其网络进行微调，描述见图2。

1.2 模型结构设计与损失函数选择

resnet的架构在上文已经给出，这里仅说明本实验使用的模型：

mycnn：自己实现的resnet34去掉所有残差连接。

resnet34：封装的模型。

myresnet34：自己实现的模型。

1.3 实验设置

本实验的设置如下：

BATCH_SIZE = 128

NUM_EPOCHS = 20/30

LEARNING_RATE = 1e-3

NUM_CLASSES = 10

其他网络的参数见上文。

1.4 实验结果与可视化

本实验在租的服务器上训练，显卡为NVIDIA GeForce RTX 3090（做一晚上扣我10块，555）。

这里仅给出myresnet34的图表作为代表，损失见图3，准确率见图4。

本实验的所有输出见github。

1.5 总结与分析

对上述实验结果进行分析如下：

- **整体结果** 经过超参数调整之后，各个模型的结果大差不差，准确率基本都在0.84左右。
- **残差的作用** 实验中发现如果只是深层的cnn，例如上面的mycnn，其收敛速度非常慢；如果加上残差，变成resnet34，则收敛速度明显变快，这与前文提到的残差的作用是吻合的。

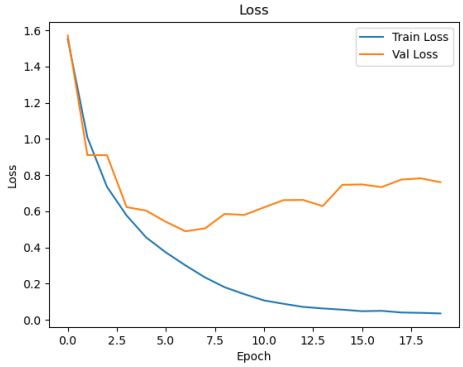

Figure 3: myresnet34_loss

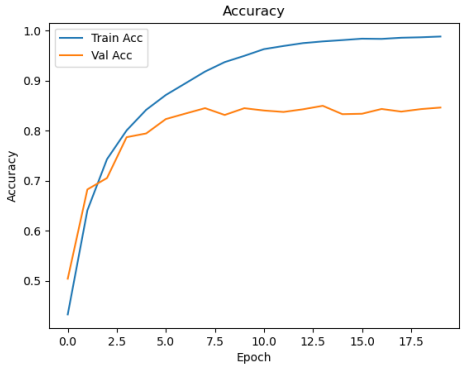

Figure 4: myresnet34_accuracy

- **残差解决网络退化** 残差为什么能让网络变深的同时，效果至少不会退化？残差可以选择性地跳过某些层，变成浅层网络，所以能保障层数越多，至少不会退化。

2 文本分类

2.1 各网络结构的原理解释

2.1.1 循环神经网络

循环神经网络（Recurrent Neural Network, RNN）是一种专门用于处理序列数据的神经网络结构。与前馈神经网络和卷积神经网络不同，RNN在网络中引入了时间维度上的循环连接，使得网络在每个时间步不仅接收当前输入，还能利用之前时刻的信息。这种特性使得RNN在处理时序相关的数据（如自然语言、时间序列信号、音频数据）时具有天然优势。

在标准的RNN结构中，给定一个长度为T的输入序列$(x_1, x_2, \dots, x_T)$，其中x_t是当前时刻t的输入向量。h_t表示t时刻的隐状态向量，它积累了$(x_1, x_2, \dots, x_{t-1})$的信息。则RNN计算$h_t$公式如下：

$$h_t = \sigma\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$

其中W_xh和W_hh表示权重，b_h表示偏置，σ 表示激活函数。由该式子可以看出，h_t来源于之前的隐状态h_{t-1}和当前的输入x_t。随着t的推移，隐状态h_t就会存储历史信息，所以当前时刻的输出会受到之前输入的影响。

而t时刻的输出y_t则取决于隐状态h_t，公式如下：

$$y_t = w_{hy}h_t + b_y$$

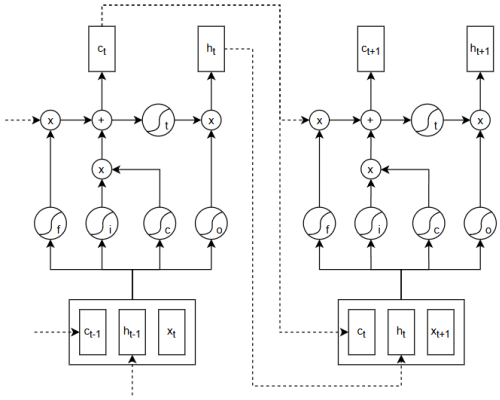

Figure 5: lstm_architecture

上述所有权重和偏置通过学习得到。而在所有时间步中参数是共享的，也就是说，所有时刻使用相同的W_{xh}、W_{hh}、W_{hy}、b_h、b_y，这种设计大大减少了模型参数量，使得模型能够灵活地处理不同长度的输入序列，理论上RNN可以捕捉任意时间跨度的依赖关系。

2.1.2 LSTM

在实际训练过程中，标准RNN面临着严重的梯度消失（vanishing gradient）和梯度爆炸（exploding gradient）问题。具体而言，当进行反向传播时，误差信号在时间步上递归传播：如果权重较小，梯度会逐步衰减，趋近于0，导致早期输入无法有效影响当前输出（梯度消失）；而如果权重较大，则可能出现梯度不断累积、数值发散的现象（梯度爆炸）。这使得标准RNN很难捕捉长距离的依赖关系。

为了克服上述问题，研究者提出了多种改进版本的RNN，其中最为经典的是长短期记忆网络（Long Short-Term Memory, LSTM），它在结构上引入了门控机制，能够有效缓解梯度问题，提高模型在长序列建模中的性能。

LSTM的架构如图5所示。

设t时刻输入为x_t，神经元为c_t，输出为y_t。设W表示权重，b表示偏置，σ和tanh表示激活函数。考虑t时刻到t+1时刻的变化如下。

遗忘门f的公式如下：

$$f_{t+1} = \sigma(W_f[c_t, h_t, x_{t+1}] + b_f)$$

输入门i的公式如下：

$$i_{t+1} = \sigma(W_i[c_t, h_t, x_{t+1}] + b_i)$$

候选神经元状态$\tilde{c}$的公式如下：

$$\tilde{c}_{t+1} = tanh(W_c[c_t, h_t, x_{t+1}] + b_c)$$

神经元c的公式如下：

$$c_{t+1} = f_{t+1} \odot c_t + i_{t+1} \odot \tilde{c}_{t+1}$$

输出门o的公式如下：

$$o_{t+1} = \sigma(W_o[c_t, h_t, x_{t+1}] + b_o)$$

隐状态h（也即输出）的公式如下：

$$h_{t+1} = o_{t+1} \odot tanh(c_{t+1})$$

2.2 模型结构设计与损失函数选择

lstm的架构与计算在上文已经给出，这里仅说明本实验使用的模型：

bilstm2：已封装，两层的双向lstm。

lstm4/8：已封装，4/8层的lstm。

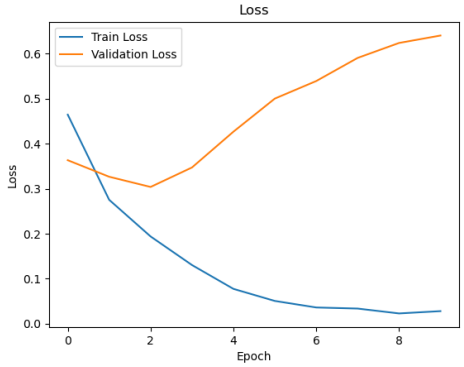

Figure 6: bilstm2_loss

res_lstm8：已封装，8层的lstm，每层lstm之后都带有残差连接。

my_lstm1：自己实现的1层lstm，训练速度极慢无比。

2.3 实验设置

本实验的设置如下：

NUM_LAYERS = ?，LSTM层数，视模型而定

MAX_LEN = 300

BATCH_SIZE = 64

EMBED_DIM = 200，词嵌入维度

HIDDEN_SIZE = 256，lstm隐藏层大小，每个词被表示为EMBED_DIM的向量，经过lstm计算变为HIDDEN_SIZE的向量

NUM_CLASSES = 2

NUM_EPOCHS = 10

LEARNING_RATE = 1e-3

DROPOUT = 0.5

MAX_VOCAB_SIZE = 20000，只留下最常用的词，其他一律当作unk

2.4 实验结果与可视化

本实验在租的服务器上训练，显卡为NVIDIA GeForce RTX 3090。

本实验的图表不如数据直观，这里仅给出BiLSTM2的图表作为代表，损失见图6，准确率见图7。

本实验的所有输出见github。

2.5 总结与分析

对上述实验结果进行分析如下：

- **整体结果** 经过超参数调整之后，各个模型的结果大差不差，准确率基本都在0.88左右。
- **残差的作用** 实验中发现一个特殊现象，lstm设置为4层，还能训得动，结果较为正常；lstm设置为8层，完全训不动，几乎没变化；加入残差连接之后，就能回归正常。这是因为，lstm堆叠较深时存在梯度消失，加入残差连接可以避免这个问题。

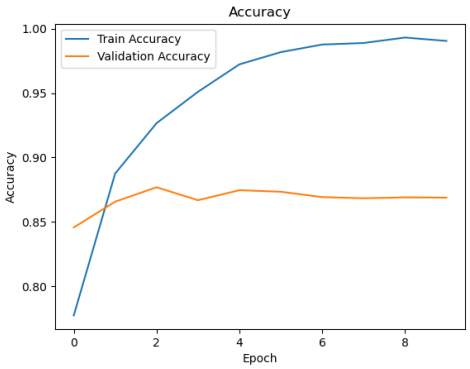

Figure 7: bilstm2_accuracy

- **自定义lstm运行太慢** 实验中发现自己实现的lstm，虽然只有1层，但是训练速度极慢无比，其训练时间是加了残差连接的封装好的8层lstm的两倍多，暂时不清楚原因。