南開大學
Nankai University

网络空间安全学院

深度学习（高阶课）实验报告

循环神经网络实验报告

姓名：王舒瑀

学号：2212777

专业：物联网工程

2025 年 6 月 18 日

目录

1 实验要求

1. 掌握 RNN 原理
2. 学会使用 PyTorch 搭建循环神经网络来训练名字识别
3. 学会使用 PyTorch 搭建 LSTM 网络来训练名字识别

2 实验内容

2.1 RNN 原理

循环神经网络 RNN 是一种用于处理序列数据的神经网络结构，其核心思想是通过隐藏状态将前一时刻的信息传递到当前时刻，从而捕捉序列中的时间依赖关系。RNN 的基本结构是对于一个输入序列 $\{x_1, x_2, \ldots, x_T\}$，RNN 在每一个时间步 t 通过计算完成在时间序列上的传播机制。隐藏状态更新的实现如下：

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

其中：- x_t：当前时间步的输入向量 - h_{t-1}：前一时间步的隐藏状态 - h_t：当前时间步的隐藏状态 - W_{xh}、W_{hh}：权重矩阵 - b_h：偏置项 - σ_h：激活函数（通常为 tanh 或 ReLU）

然后进行输出计算：

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

其中：- y_t：当前时间步的输出 - W_{hy}：隐藏状态到输出的权重矩阵 - b_y：输出偏置 - σ_y：输出激活函数（如 softmax 用于分类）

每个 h_t 都依赖于 h_{t-1}，因此可以将整个序列的依赖关系关联起来。但同样的，由于 RNN 的参数在多个时间步共享，反向传播过程中的梯度会经过多次链式相乘，导致梯度越来越小，无法有效更新早期时间步的参数

2.2 原始版本 RNN 实现名字识别

根据给出的实验文件，运行进行数据获取，数据预处理，基础 RNN 搭建，然后训练模型，最后进行模型评估。整体的 RNN 网络结构如下：

```
RNN(
  (i2h): Linear(in_features=185, out_features=128, bias=True)
  (i2o): Linear(in_features=185, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

图 2.1: Enter Caption

如图所示，这个 RNN 手动实现了最基本的循环神经网络单元，通过拼接输入和前一个隐藏状态来同时计算新的隐藏状态和输出，适合用于序列分类任务（如字符级语言模型、拼写预测等），并在最后输出后接一个 LogSoftmax 层。

对 RNN 模型进行训练后

2.3 LSTM 网络实现名字识别

长短期记忆网络 LSTM 是一种特殊的循环神经网络结构，它通过引入门控机制解决了传统 RNN 在处理长序列时容易出现的梯度消失与爆炸问题。LSTM 特别适用于处理和预测时间序列数据中存在长期依赖的问题，例如自然语言处理中的字符级或词级建模任务。

在本实验中，所使用的 LSTM 模型结构如下：

```
LSTM(
  (rnn): LSTM(57, 128)
  (out): Linear(in_features=128, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=-1)
)
```

图 2.2: Enter Caption

输入维度（$input_size = 57$）：代表每个输入的特征数，这里为 57，对应 one-hot 编码的字符数。

隐藏维度（$hidden_size = 128$）：LSTM 单元中隐藏状态的维度，决定了模型的记忆容量。

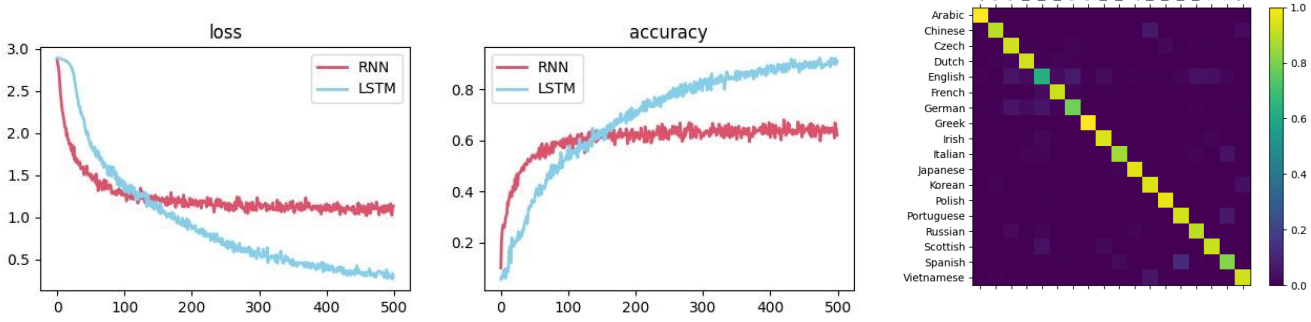
输出类别数（$output_size = 18$）：最终分类类别数量，本实验为 18 类，即 18 种语言。

LogSoftmax：将最后的线性层输出转换为对数概率形式，适配 NLLLoss 作为训练的损失函数。

整个网络的功能是：输入一个字符序列，通过 LSTM 提取时序特征后输出最后一个时间步的隐藏状态，再通过线性层和 softmax 得出分类结果。

2.3.1 与原始 RNN 的对比

LSTM（长短期记忆网络）的性能优于 RNN（循环神经网络）的主要原因在于其结构设计有效解决了 RNN 的长期依赖问题，训练曲线如下：



(a) LSTM 准确率及损失曲线图 (b) LSTM 预测矩阵图

图 2.3: LSTM 训练图

从图中可以看到，LSTM 在训练过程中的表现明显优于 RNN。以下是对 LSTM 训练效果的具体分析。在左图中，LSTM 的损失值随着训练的进行，LSTM 的损失值下降得更快，最终趋于一个更低的稳定值。这表明 LSTM 在训练过程中能够更快地学习到数据的特征，并且能够更好地拟合训练数据。准确率方面 LSTM 的准确率（蓝色曲线）从一开始就高于 RNN（红色曲线），并且随着训练的进行，LSTM 的准确率提升得更快，最终达到一个更高的稳定值。这说明 LSTM 在分类或预测任务中表现更好，能够更准确地识别或预测数据。同时 LSTM 在训练初期的收敛速度明显快于 RNN，损失值和准确率的曲线在前 100 个 epoch 内就有显著的变化。这表明 LSTM 在处理序列数据时，能够更快地

捕捉到长期依赖关系，从而加速模型的收敛。特别的，LSTM 的损失值和准确率曲线在训练后期趋于平稳，波动较小，表明模型在训练后期表现稳定。相比之下，RNN 的曲线波动较大，尤其是在损失值曲线中，显示出 RNN 在处理序列数据时可能存在一定的不稳定性。综上 LSTM 在训练过程中表现出更好的学习能力、更快的收敛速度和更高的稳定性。这些特性使得 LSTM 在处理具有时间序列依赖性的数据时，通常比 RNN 更为有效。图中可以看到，在 Loss 曲线中 LSTM 在 500 步左右时 loss 降至 0.5，且曲线下降趋势稳定，表明模型能有效优化参数，梯度控制良好。而对比而言 RNN 的 loss 下降缓慢，最终高于 1.0，说明由于梯度消失/爆炸问题，难以捕捉长期依赖关系。Accuracy 曲线中 LSTM 准确度在 500 步时达到 0.6，虽仍有提升空间，但趋势平稳上升，未出现剧烈波动。RNN 准确度仅达到 0.6 且波动较大，表明训练不稳定。

2.3.2 性能差异分析

从上述分析和结果我们可以看到在结果上，LSTM 的效果明显优于 RNN。其中的原因是由于 LSTM 具有特别的门控机制，LSTM 通过遗忘门、输入门、输出门动态控制信息流，能够有效缓解 RNN 的梯度问题。同时 LSTM 的细胞状态可长期保留关键信息（如语言语法结构），而 RNN 因梯度衰减难以学习远距离依赖。这也就导致了 LSTM 的 loss 和 accuracy 曲线更平滑，收敛更快；RNN 因梯度问题导致优化困难。

RNN 在处理长序列时，梯度在反向传播中会指数级衰减（梯度消失），导致模型难以学习远距离的依赖关系。从提供的 RNN loss 曲线可见，loss 在 500 步后仍停留在 1.0 左右，下降趋于平缓，表明优化困难；准确度曲线虽从 0.1 逐步提升至 0.7，但增速缓慢且可能未充分收敛，说明模型对复杂模式的学习能力有限。

LSTM 的改进：LSTM 通过门控机制（遗忘门、输入门、输出门）和细胞状态，显式控制信息的保留与遗忘。遗忘门可丢弃无关信息，输入门筛选新信息，输出门决定最终状态。这一机制使得 LSTM 能够长期保持关键信息，避免梯度消失。因此，LSTM 的 loss 曲线通常下降更快（如降至 0.5 以下），准确度更高（如稳定在 0.9 以上）。RNN 的预测矩阵显示 RNN 对长序列任务（如时间序列预测、文本生成）的预测结果存在显著误差（如对角线模糊、非对角线元素较多），表明其对长期模式捕捉不足。LSTM 由于能有效记忆长期信息，预测结果更精准（对角线清晰，错误分散较少），尤其在需要跨时间步推理的任务中优势明显。同时 RNN 的 loss 和准确度曲线常出现剧烈波动（如准确度从 0.1 到 0.7 的不稳定上升），可能是梯度爆炸或消失导致的训练不稳定。LSTM 的门控机制稳定了梯度流动，使其 loss 曲线平滑下降，准确度稳步提升，收敛速度更快且结果更优。

模型类型	能否记忆长期依赖	是否有门控机制	是否能有效控制梯度
RNN	否	否	否
LSTM	是	时（3 个门）	是

表 1: LSTM 与传统 RNN 的对比

2.4 自行实现的 LSTM

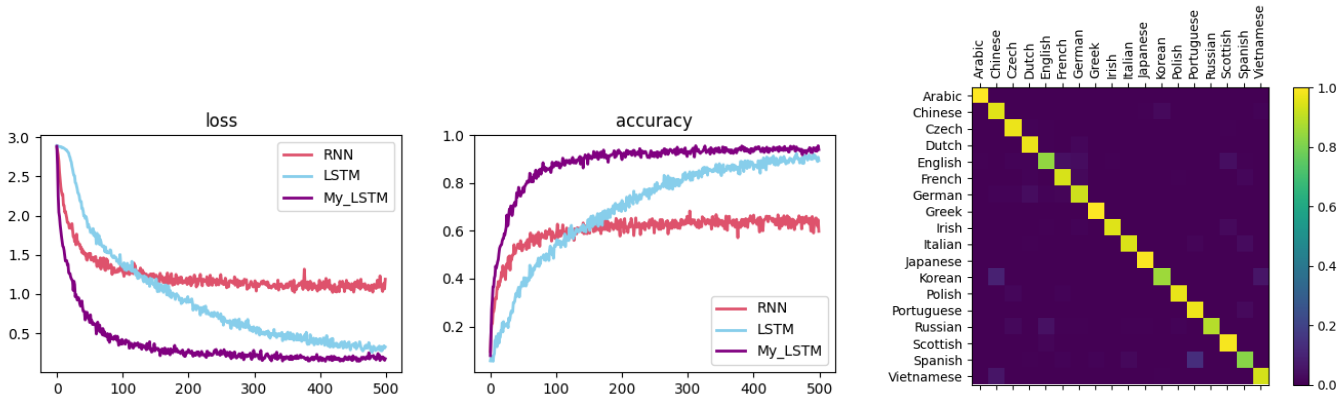
自己手动实现 LSTM 模型，主要包含构造 LSTMBlock 即 LSTM 单元，包含输入门、遗忘门、候选记忆和输出门的计算逻辑。然后在此基础上构建自己的 MYLSTM 为完整的 LSTM 模型，通过循环调用 LSTMBlock 处理序列输入，最后通过线性层和 LogSoftmax 输出分类结果。结构如下

实验设置为输入特征维度 57（one-hot 编码的英文字母），隐藏层大小 128，输出类别数 18，序列长度 6（即每个样本含 6 个字符），最后得到自己手动实现的 LSTM 网络与其他两个方式的训练结果

```
MY_LSTM(
  (lstm): LSTMBlock()
  (classifier): Sequential(
    (0): Linear(in_features=128, out_features=18, bias=True)
    (1): LogSoftmax(dim=1)
  )
)
```

图 2.4: Enter Caption

对比图以及网络的混合矩阵如下：



(a) $MyLSTM$ 准确率及损失曲线图 (b) $MyLSTM$ 预测矩阵图

图 2.5: $MyLSTM$ 训练图

可以看到，自己手动实现的 LSTM 网络在表现上是最优的，在 loss 曲线上下降最快且最后也略低于封装的 LSTM。在准确率方面也同样，上升最陡且最后略高于 LSTM。但同时，自己手动实现的 LSTM 所耗的训练时间也远远高于另外两种方式，在本实验中，将手动实现的 MyLSTM 模型与 PyTorch 内置的 nn.LSTM 以及传统的 RNN 模型在相同训练条件下进行了对比，得出如下训练时长：

模型类型	训练步数	训练时长	最终损失值
MY_LSTM	500,000	95m 3s	0.0033
标准 LSTM	500,000	35m 17s	0.0090
标准 RNN	500,000	29m 31s	2.6639

原因可能是因为手动实现不具备优化加速机制，而 PyTorch 内置的 nn.LSTM 在底层使用了 C++/CUDA 实现，并且自动融合多个线性操作，充分利用了张量并行计算优势。同时手动实现的 LSTMBlock 使用了 Python 的 @ 运算符（即 matmul），在 for 循环中逐步展开，未能充分利用 GPU 的并行计算能力，尤其在处理较长序列或高维数据时，效率大幅下降。同时手动实现必须显式对每个时间步进行循环，而 PyTorch 内置模块是使用 C++ 高效处理整个序列的批次操作。对整个序列并行处理，并对批量输入数据做了优化处理。而手动实现中的 LSTMBlock 无法充分利用这些优化，仅能逐时间步、逐样本地执行运算，导致显著的计算瓶颈。

2.5 RNN 生成名字网络

本小节是基于字符级循环神经网络（RNN）进行人名生成的实验，即通过前面的网络，字符级模型能够捕捉细粒度的拼写和形态特征。通过在不同语言/国家类别条件下训练 RNN，模型能学习并生

成符合各自语言习惯的人名。

RNN 单元 ：进行两次全连接，第一次 $[\text{cat}, x_t, h_{t-1}] \to h_t$ 和 $[\text{cat}, x_t, h_{t-1}] \to o'_t$，第二次全连接：$[o'_t, h_t] \to o_t$，随后 Dropout +LogSoftmax，网络结构如下：

```
RNN(
  (i2h): Linear(in_features=205, out_features=128, bias=True)
  (i2o): Linear(in_features=205, out_features=59, bias=True)
  (o2o): Linear(in_features=187, out_features=59, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (softmax): LogSoftmax(dim=1)
)
```

图 2.6: char rnn

超参数：隐藏层大小 $H = 128$，学习率 $lr = 5 \times 10^{-4}$，Dropout $p = 0.1$。

训练过程 在训练 100 万次迭代后，可视化训练曲线如下：

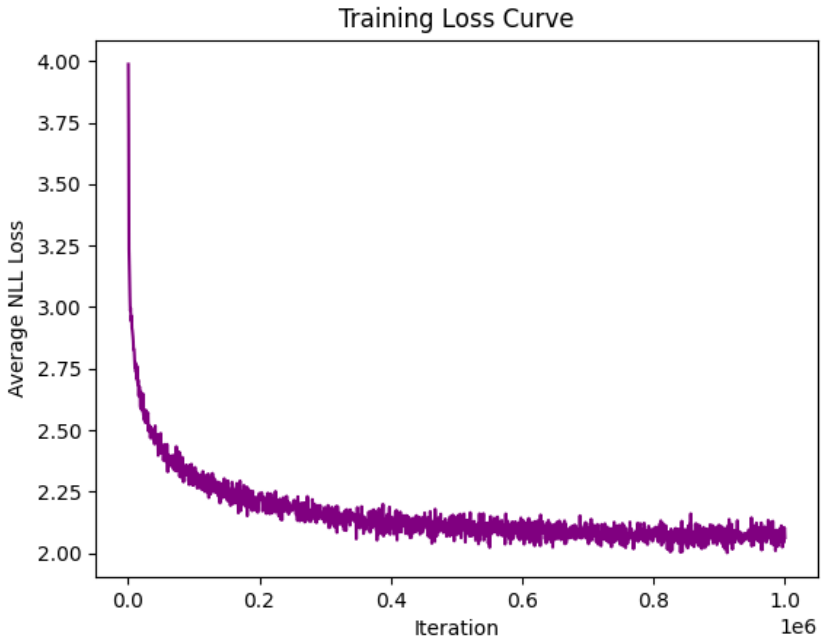


图 2.7: Enter Caption

- 损失曲线观测：最初迭代数千次后，损失下降平缓。通过记录每 `plot_every`=500 次迭代的平均损失，直观地观察学习速率和拟合趋势。
- 学习率调试：一开始使用 $lr = 0.001$，发现损失在较低水平抖动不稳定，改小至 5×10^{-4} 后收敛更为平滑。若进一步减小，收敛速度过慢；增大则易震荡。
- 梯度剪裁尝试：注意到训练中期偶尔出现梯度爆炸，使得某些 batch 损失骤升，遂加入梯度裁剪（clip_grad_norm_）将梯度范数限制在 5 以内，效果显著。
- 采样策略：最初直接使用贪婪采样 (top-1)，生成名字单调且偏好高频后缀；后来改为带温度的随机采样 (temperature=0.8)，生成结果多样度更高。

- 类别融合思考：在当前实现中，类别信号仅作为每一步输入的 one-hot，而未与隐藏状态进行更深层次融合。可考虑在计算 hidden 时加入类别门控（类似于条件 GRU/LSTM）。

实验结果 在训练 100 万次迭代后，对各语言类别进行采样示例：

- Russian：‘Rusyaev‘、‘Rastnikov‘、‘Rubovy‘…
- German：‘Gerlind‘、‘Gerlova‘、‘Gerstner‘…
- Spanish：‘Sarallez‘、‘Sauiro‘、‘Sanquero‘…
- Chinese：‘Chiyuan‘、‘Chijun‘、‘Chixin‘…

大部分生成的名字拼写符合目标语言的声韵规律，但偶有拼写不连贯或过短的案例。

2.5.1 基于手动实现的 LSTM 模型的人名国籍预测

为了验证手动实现的 LSTM 在文本分类任务中的效果，构建了一个根据英文姓名预测国籍的分类器。该模型在前向过程中只需将输入姓名转为字符级张量，经过 LSTM 编码后，通过全连接层输出各国籍的概率分布。

姓名	Top1 (概率)	Top2 (概率)	Top3 (概率)
Dovesky	Russian (0.68)	Czech (0.25)	English (0.03)
Jackson	Scottish (0.70)	English (0.18)	Russian (0.05)
Hou	Chinese (0.49)	Korean (0.34)	Vietnamese (0.10)

表 2: 姓名国籍预测示例

从表2可以看出，模型在常见姓名上的预测较为准确，并能给出合理的候选国籍排序。这证明了手动实现的 LSTM 分类器对短文本（如姓名）特征的有效建模能力，也为我们在其他小样本或字符层面任务上的应用提供了参考。