循环神经网络实验报告

姓名：靳乐卿　　学号：2012159

实验要求：

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

报告内容：

- 老师提供的原始版本 RNN 网络结构（可用 print(net)打印，复制文字或截图皆可）、在名字识别验证集上的训练 loss 曲线、准确度曲线图以及预测矩阵图
- 个人实现的 LSTM 网络结构在上述验证集上的训练 loss 曲线、准确度曲线图以及预测矩阵图
- 解释为什么 LSTM 网络的性能优于 RNN 网络（重点部分）
- 格式不限

作业提交：

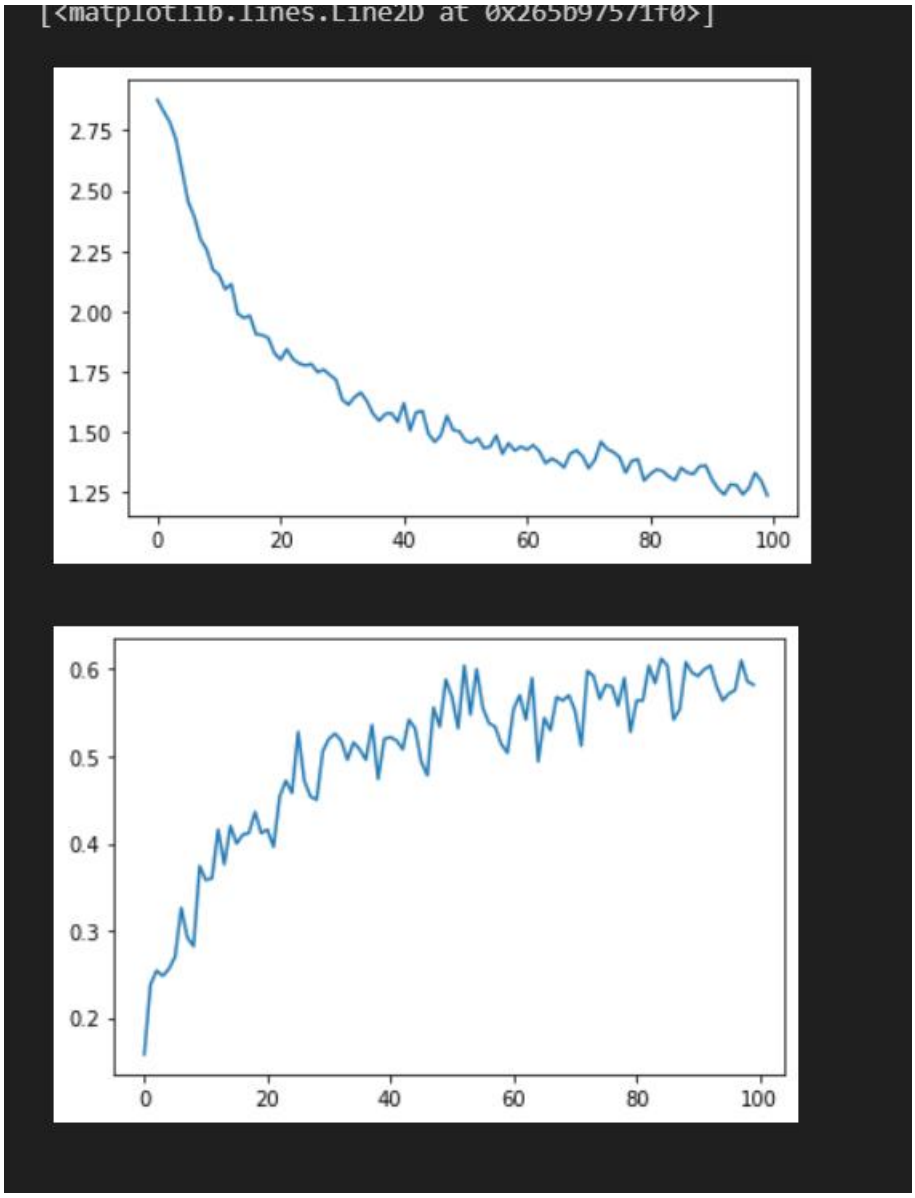
- 期末前将报告和代码（可将 jupyter notebook 里代码复制到一个 xxx.py 文件中）打包（学号+姓名.zip），提交方式另行通知
- 实验报告内容应工整
- 加分项：自己实现 LSTM 或者 GRU

原始版本 RNN

网络结构

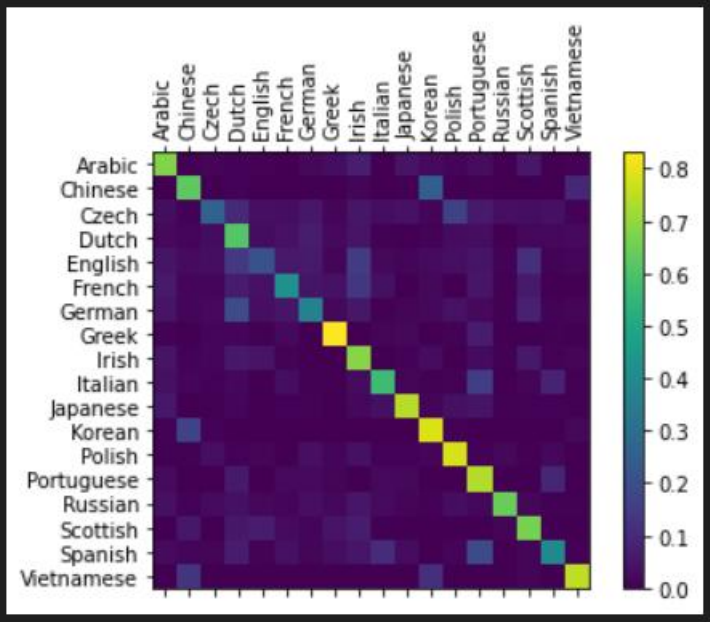
```
[21]  ✓ 0.0s

...  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)
     )
```

loss 曲线、准确度曲线图以及预测矩阵图



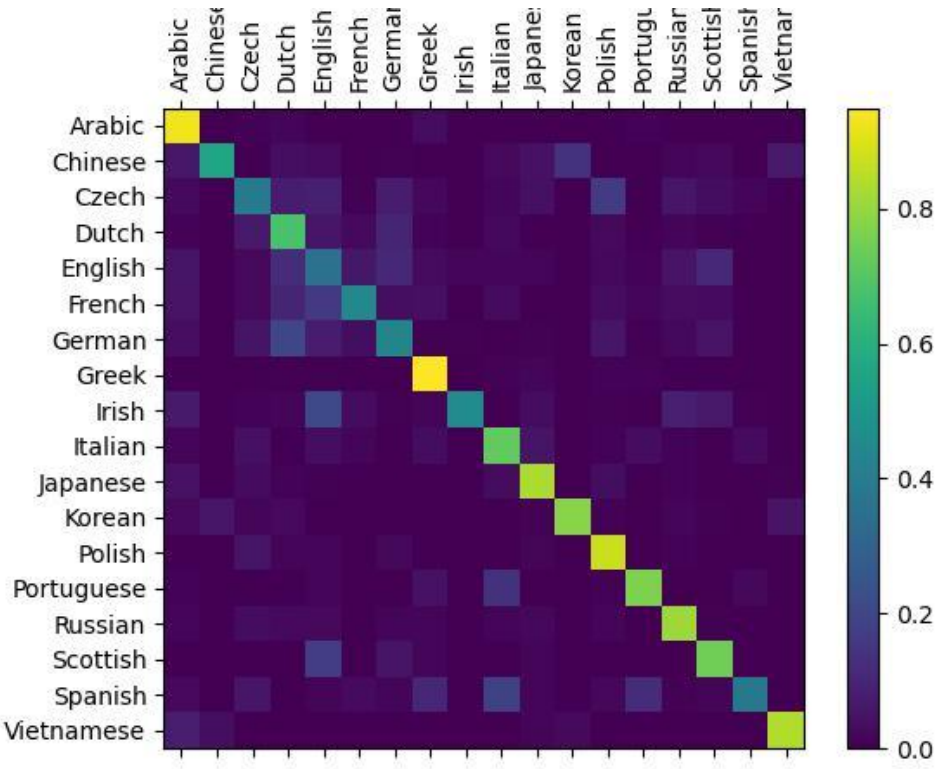
```
ax.set_yticklabels([''] + all_categories)
```

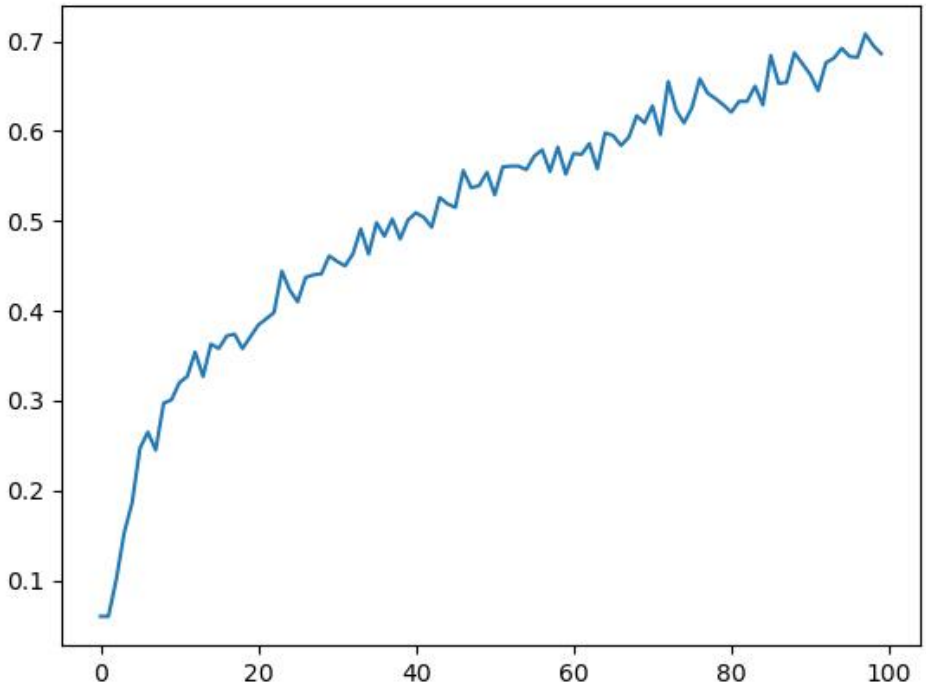
</>

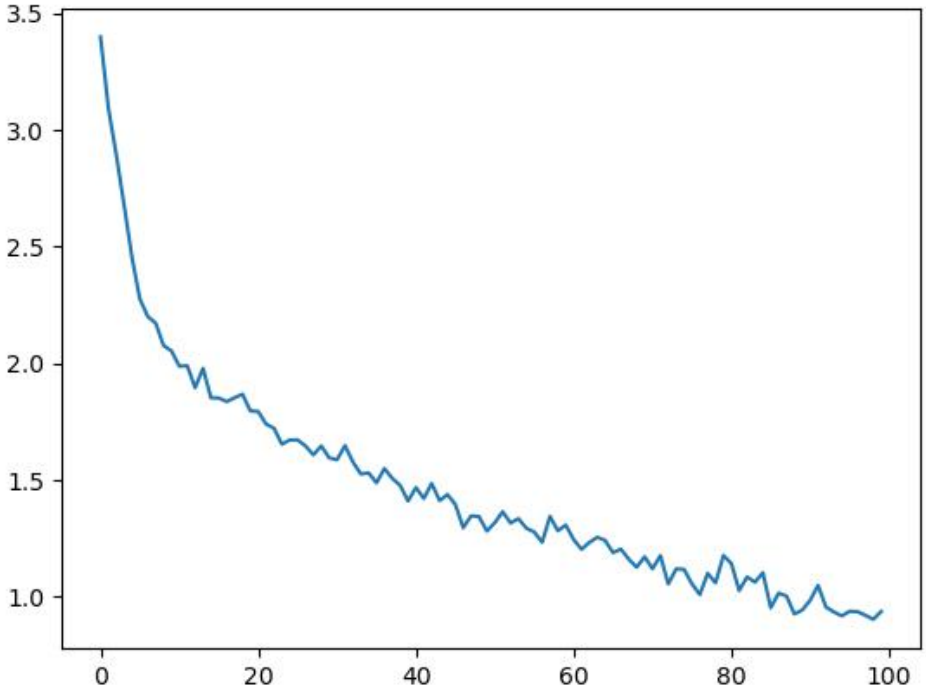


个人实现的 LSTM

loss 曲线、准确度曲线图以及预测矩阵图







可以看到准确率是有所提升的

网络结构

```
LSTM(
  (lstm): LSTM(
    (forget_gate): Linear(in_features=185, out_features=128, bias=True)
    (input_gate): Linear(in_features=185, out_features=128, bias=True)
    (cell_update): Linear(in_features=185, out_features=128, bias=True)
    (output_gate): Linear(in_features=185, out_features=128, bias=True)
    (classify): Linear(in_features=128, out_features=18, bias=True)
    (softmax): LogSoftmax(dim=1)
  )
```

```python
class LSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim, block_num = 1, output_class = 10):
        super(LSTM, self).__init__()
        self.hidden_dim = hidden_dim
        self.block_num = block_num
        self.forget_gate = nn.Linear(in_features=input_dim + hidden_dim, out_features=hidden_dim)
        self.input_gate = nn.Linear(in_features=input_dim + hidden_dim, out_features=hidden_dim)
        self.cell_update = nn.Linear(in_features=input_dim + hidden_dim, out_features=hidden_dim)
        self.output_gate = nn.Linear(in_features=input_dim + hidden_dim, out_features=hidden_dim)
        self.classify = nn.Linear(in_features=hidden_dim, out_features=output_class)
        self.softmax = nn.LogSoftmax(dim=1)
        # self.device = device

    def forward(self, input):
        input_length = input.size()[0]
        hidden = torch.zeros(1, self.hidden_dim).to(input.device)
        cell = torch.zeros(1, self.hidden_dim).to(input.device)
        output = None
        for i in range(input_length):
            x = input[i]
            state = torch.concat((x, hidden), dim=-1)
            f = torch.sigmoid(self.forget_gate(state))
            i = torch.sigmoid(self.input_gate(state))
            c = torch.tanh(self.cell_update(state))
            cell = f * cell + i * c
            output = torch.sigmoid(self.output_gate(state))
            hidden = output * torch.tanh(cell)
        output = self.softmax(self.classify(output))
        return output
```

为什么 LSTM 网络的性能优于 RNN 网络

LSTM 优于 RNN，主要有以下几个原因：

1. 处理长期依赖关系：RNN 网络在处理长序列时，会面临梯度消失或梯度爆炸的问题，导致无法有效捕捉到长期依赖关系。而 LSTM 网络通过引入门控机制，特别是遗忘门和输入门，能够更好地管理和控制信息的流动，从而更好地处理长期依赖关系。通过门控机制，LSTM 可以选择性地保留或遗忘过去的信息，有效地解决了梯度消失和梯度爆炸的问题。
2. 学习多个时间尺度的特征：LSTM 网络能够学习和利用多个时间尺度的特征，这对于处理时间序列数据非常重要。通过 LSTM 的记忆单元和各种门控机制，网络可以灵活地选择性地保留和利用不同时间尺度上的信息。这种能力使得 LSTM 网络能够更好地捕捉序列数据中的长期和短期模式。

3. 避免梯度消失和梯度爆炸：RNN 网络在反向传播过程中容易出现梯度消失或梯度爆炸的问题，导致模型难以收敛。LSTM 网络通过引入门控机制，有效地控制梯度的流动，避免了梯度的不稳定性问题。这使得 LSTM 网络更容易训练和优化。

LSTM 网络通过引入门控机制和记忆单元，能够更好地处理长期依赖关系，学习多个时间尺度的特征，并且避免了梯度消失和梯度爆炸的问题。这些特性使得 LSTM 网络在许多序列建模任务中表现出更好的性能，相对于基本的 RNN 网络而言更具优势。