循环神经网络实验报告

学号：1911433 姓名：林坤

一、 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

二、 实验内容

1. 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
2. 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
3. 解释为什么 LSTM 网络的性能优于 RNN 网络

三、 RNN

对应的rnn网络代码：

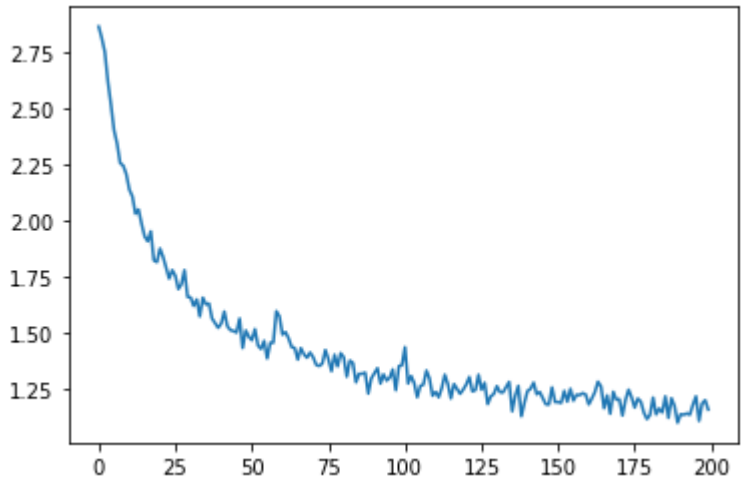
```
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden
```

```
17
18      def initHidden(self):
19          return torch.zeros(1, self.hidden_size)
20
21  n_hidden = 128
22  rnn = RNN(n_letters, n_hidden, n_categories)
```

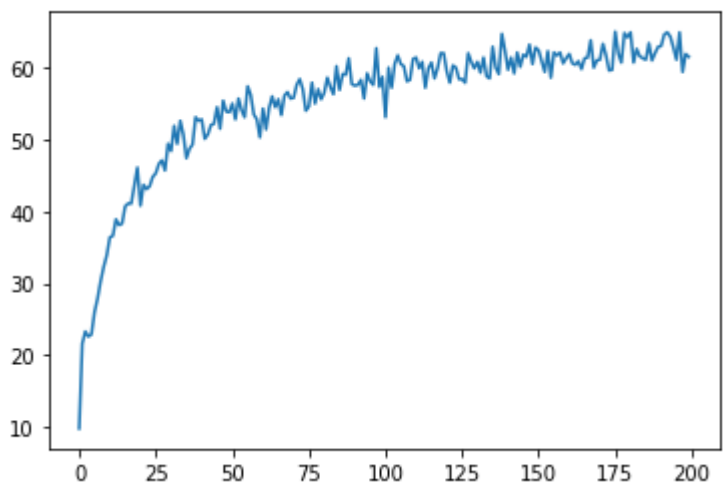
网络结构打印结果如下：

```
1  RNN(
2      (i2h) : Linear(in_features =185, out_features =128, bias=True )
3      (i2o) : Linear(in_features =185, out_features =18, bias=True )
4      (softmax) : LogSoftmax ( dim=1)
5  )
```

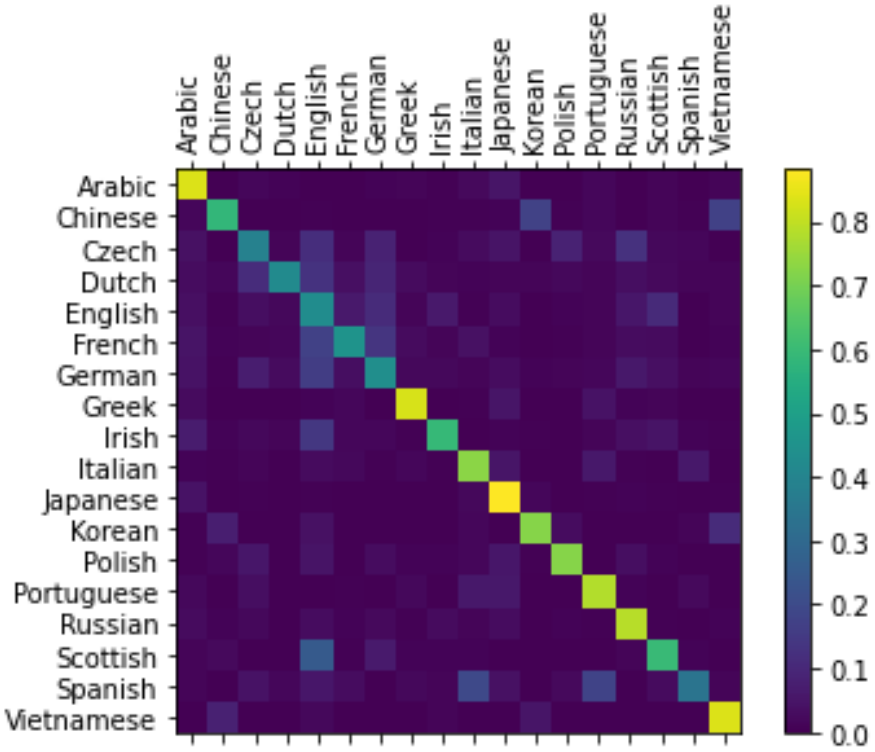
在名字识别验证集上的训练 loss **曲线** (n_iters = 200000): **最终** loss **为** 1.1699



在名字识别验证集上的 accuracy **曲线** (n_iters = 200000): **最终** accuracy **为** 62%



预测矩阵图如下:



四、个人实现 LSTM 网络

其中的 NaiveLSTM Block **网络结构如下：**

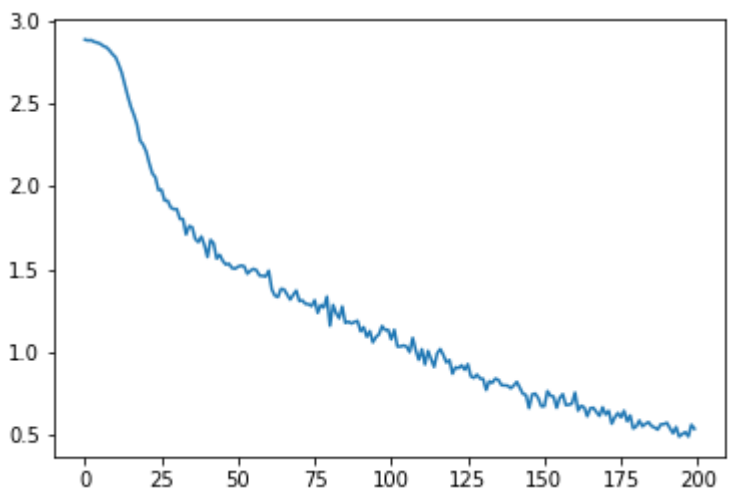
```
class NaiveLSTM(nn.Module):
    def __init__(self, input_sz, hidden_sz):
        super().__init__()
        self.input_sz = input_sz
        self.hidden_size = hidden_sz
        self.W = nn.Parameter(torch.Tensor(input_sz, hidden_sz * 4))
        self.U = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz * 4))
        self.bias = nn.Parameter(torch.Tensor(hidden_sz * 4))
        self.init_weights()

    def init_weights(self):
        stdv = 1.0 / math.sqrt(self.hidden_size)
        for weight in self.parameters():
            weight.data.uniform_(-stdv, stdv)

    def forward(self, x, init_states=None):
        """Assumes x is of shape (batch, sequence, feature)"""
        bs, seq_sz, _ = x.size()
        hidden_seq = []
        if init_states is None:
            h_t, c_t = (torch.zeros(bs, self.hidden_size).to(x.device),
                        torch.zeros(bs, self.hidden_size).to(x.device))
        else:
```

```
            h_t, c_t = init_states

        HS = self.hidden_size
        for t in range(seq_sz):
            x_t = x[:, t, :]
            # batch the computations into a single matrix multiplication
            gates = x_t @ self.W + h_t @ self.U + self.bias
            i_t, f_t, g_t, o_t = (
                torch.sigmoid(gates[:, :HS]), # input
                torch.sigmoid(gates[:, HS:HS*2]), # forget
                torch.tanh(gates[:, HS*2:HS*3]),
                torch.sigmoid(gates[:, HS*3:]), # output
            )
            c_t = f_t * c_t + i_t * g_t
            h_t = o_t * torch.tanh(c_t)
            hidden_seq.append(h_t.unsqueeze(0))
        hidden_seq = torch.cat(hidden_seq, dim=0)
        # reshape from shape (sequence, batch, feature) to (batch, sequence, fea
        hidden_seq = hidden_seq.transpose(0, 1).contiguous()
        return hidden_seq, (h_t, c_t)
```

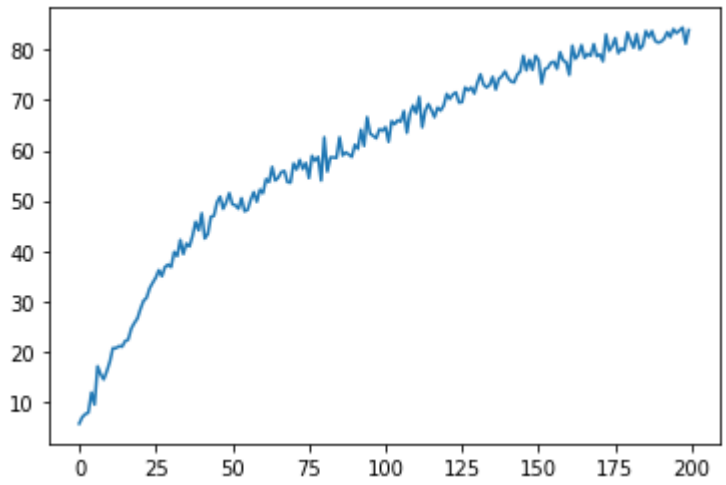
网络结构：

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

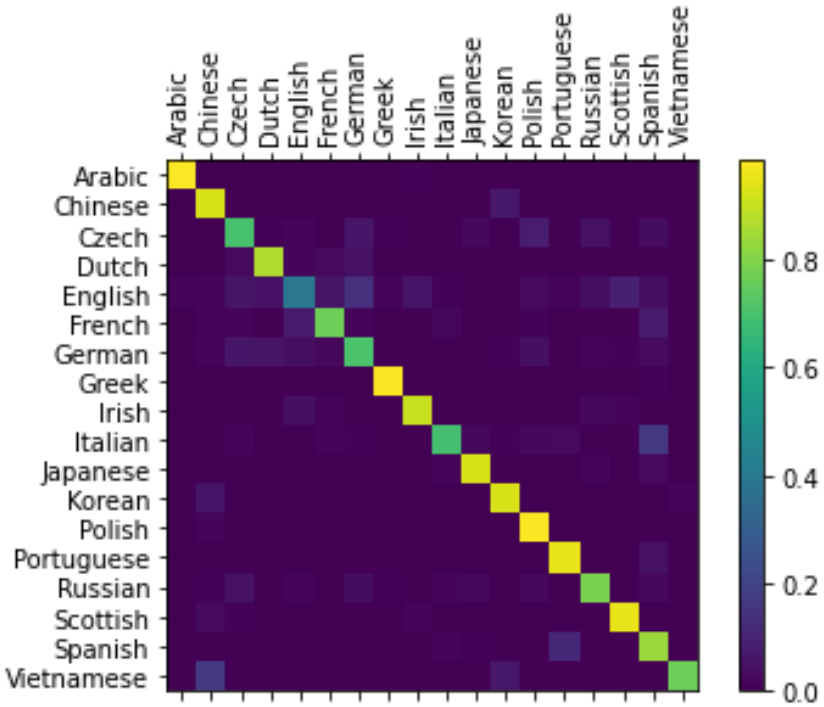
在名字识别验证集上的训练 loss **曲线** (n_iters = 200000): **最终** loss **为** 0.52



在名字识别验证集上的 accuracy **曲线** (n_iters = 200000): **最终** accuracy **为** 82.7%



预测矩阵图如下:



五、 LSTM 与 RNN

解释为什么 LSTM **网络的性能优于** RNN **网络**:

1. 首先分析普通的 RNN 网络的缺点：相较于 MLP/CNN 来说，RNN 中同样的权重在各个时间步共享，即最终的梯度体现为各时间步的梯度和。这会导致在 RNN 中梯度出现越传越弱的现象，最终导致 RNN 的梯度被近距离梯度主导，使得 RNN 模型难以学到远距离的依赖关系。
2. 而反观 LSTM 的结构则天然地克服了上面提到的 RNN 的梯度消失问题。

LSTM 引入了 gate 门机制：

· 当 gate 是关闭的，那么就会阻止对当前信息的改变，这样以前的依赖信息就会被学到

· 而当 gate 是打开的时候，并不是完全替换之前的信息，而是在之前信息和现在信息之间做加权平均。所以，无论网络的深度有多深，输入序列有多长，只要 gate 是打开的，网络都会记住这些信息。

LSTM 除了在结构上克服了梯度消失的问题，更重要的是具有更多的参数来控制模型；通过四倍于 RNN 的参数量，可以更加精细地预测时间序列变量。而在预测过程中，LSTM 通过门机制来控制，哪些历史信息应该记住，哪些历史信息应该遗忘，以及是否每个阶段都应该有有效的信息被输出，从而获得比 RNN 更好的网络性能优势。