NKU 深度学习（高阶课）实验报告

实验名称：循环神经网络

学　　院：网络空间安全学院

姓　　名：邢清画

专　　业：物联网工程

二〇二五年五月

1 实验要求

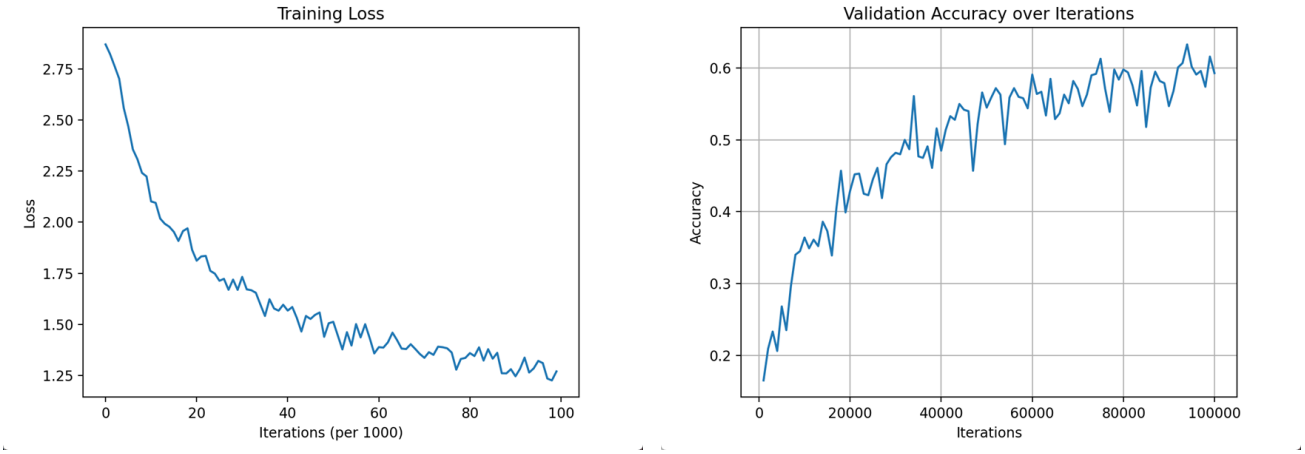
1.1 原始版本 RNN

1.1.1 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）

```
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 1: 原始 RNN 网络结构

1.1.2 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



(a) 原始 RNN loss 曲线　　(b) 原始 RNN acc 曲线

图 2: 原始 RNN 模型的实验结果展示

结果说明

图 2(a) 展示了原始 RNN 模型在训练过程中的损失函数下降趋势。可以看到，模型在初始阶段损失值较高（约为 2.8），随着迭代的进行，损失快速下降，并在后期趋于平稳，最终稳定在 1.2 左右。这表明模型具备一定的学习能力，能够在训练数据上有效拟合，且没有明显的过拟合迹象（未出现训练损失回升）。

图 2(b) 展示了模型在验证集上的准确率变化情况。初始准确率约为 18%，随着训练推进，准确率迅速提升，最终在约 100,000 次迭代时达到 60% 左右，并在该水平波动。这表明模型对验

证数据具有一定的泛化能力，但提升趋势在中后期有所减缓，存在模型表达能力不足的问题，可能由于 RNN 在长序列建模中面临梯度消失、上下文依赖弱等限制。

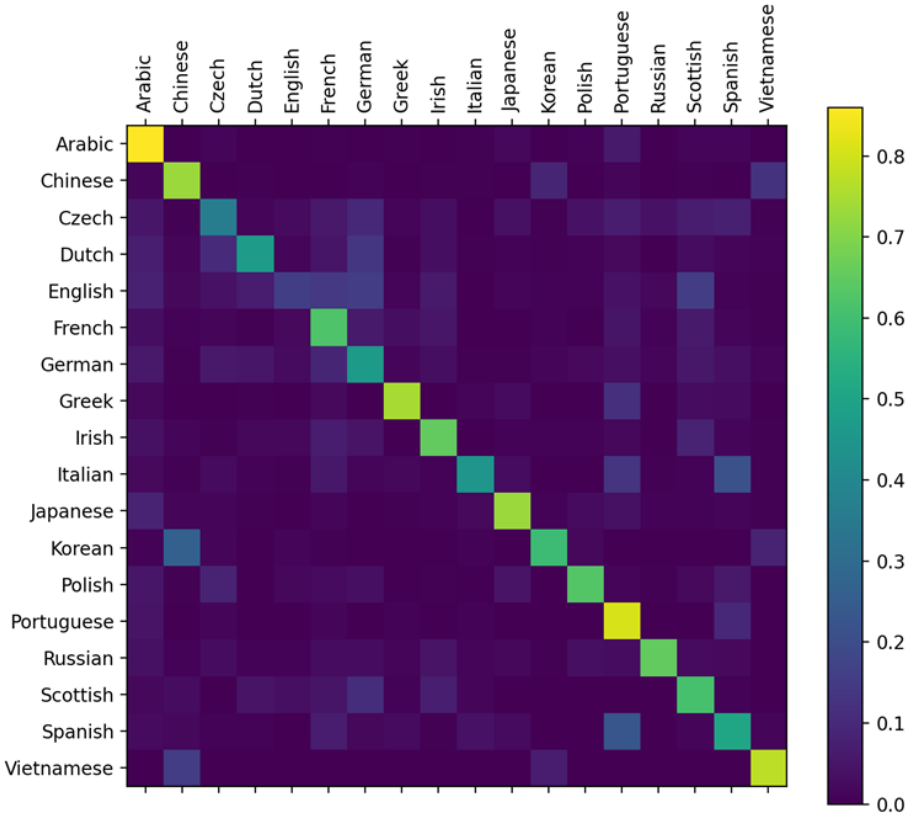
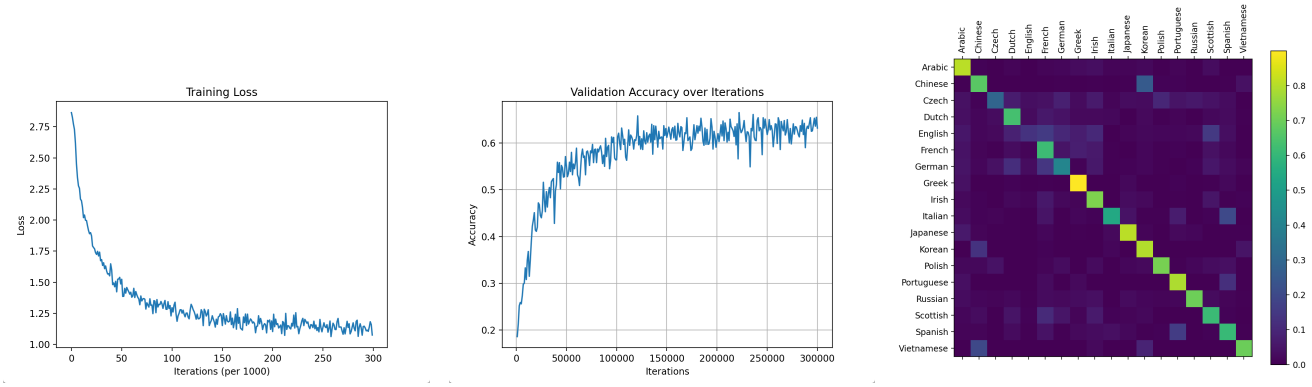


图 3: 原始 RNN 预测结果矩阵

混淆矩阵说明

图 3 展示了原始 RNN 模型在多语言分类任务中的归一化混淆矩阵。从图中可以看出，矩阵主对角线的颜色普遍较亮，说明模型能够较准确地识别大多数语言类别，尤其是 Arabic、Chinese、Irish、Portuguese 和 Vietnamese 的预测准确率较高（接近 90%）。这表明模型在处理这些语言时表现出较强的判别能力。

然而，Czech、Dutch、English、French、German 等语言之间存在较为明显的混淆，推测是由于它们属于相近语系，词汇与语法结构存在相似性，使得模型在特征提取阶段难以有效区分。此外，Korean 与 Chinese、Vietnamese 与 Japanese 也存在一定程度的误判，说明模型在东亚语言上的区分能力仍有待提升。



(a) 迭代 30 万次的 RNN Loss 曲线　(b) 迭代 30 万次的 RNN Accuracy 曲线　(c) 迭代 30 万次的 RNN 混淆矩阵

图 4: RNN 模型在迭代 30 万次后的训练损失、验证准确率与混淆矩阵结果（对齐 LSTM 迭代次数）

RNN 模型通过长时间训练已实现基本收敛，在验证集上取得了较为稳健的性能（准确率约 63 %），但由于结构上缺乏长期依赖建模能力，分类精度和语言区分能力存在局限，特别在多类别复杂任务中易混淆。接下来使用 LSTM 改进结构进行提升。

1.2 自主实现 LSTM 网络结构

1.2.1 网络结构

```python
class LSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTM, self).__init__()
        self.hidden_size = hidden_size
        self.i2f = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2i = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2c = nn.Linear(input_size + hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden, cell):
        combined = torch.cat((input, hidden), 1)
        forget_gate = torch.sigmoid(self.i2f(combined))
        input_gate = torch.sigmoid(self.i2i(combined))
        output_gate = torch.sigmoid(self.i2o(combined))
        cell_candidate = torch.tanh(self.i2c(combined))

        cell = forget_gate * cell + input_gate * cell_candidate
        hidden = output_gate * torch.tanh(cell)
```

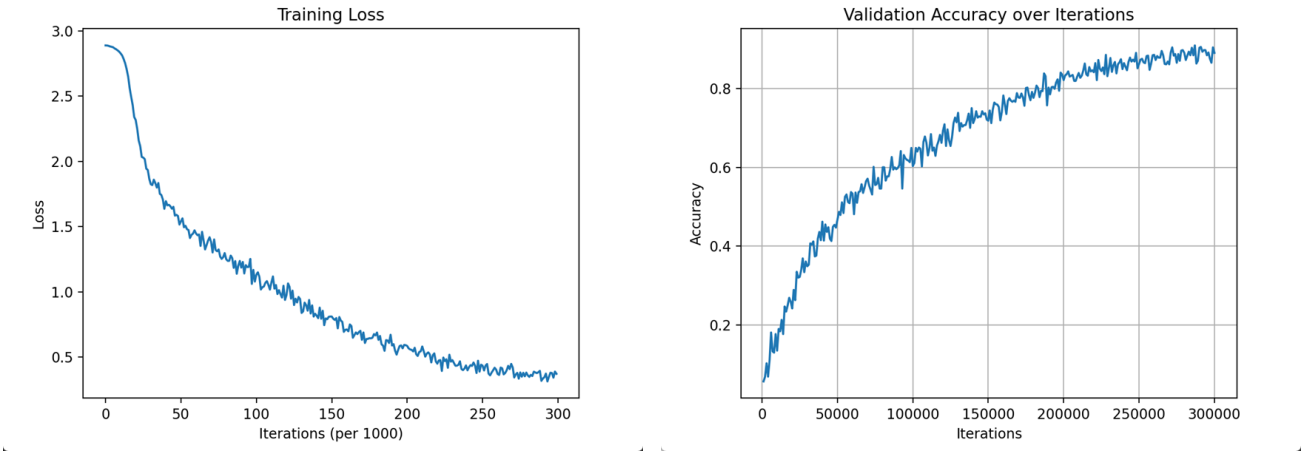
```
        output = self.out(hidden)
        output = self.softmax(output)
        return output, hidden, cell

    def initHidden(self): return torch.zeros(1, self.hidden_size).to(device)
    def initCell(self): return torch.zeros(1, self.hidden_size).to(device)
```

Listing 1: LSTM 模型定义

```
LSTM(
  (i2f): Linear(in_features=185, out_features=128, bias=True)
  (i2i): Linear(in_features=185, out_features=128, bias=True)
  (i2o): Linear(in_features=185, out_features=128, bias=True)
  (i2c): Linear(in_features=185, out_features=128, bias=True)
  (out): Linear(in_features=128, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

图 5: LSTM 网络结构

1.2.2 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



(a) LSTM loss 曲线

(b) LSTM acc 曲线

图 6: 手动实现的 LSTM 结果曲线

结果曲线分析

图 6(a) 和图 6(b) 分别展示了 LSTM 模型在训练过程中的损失下降曲线与验证集准确率提升曲线。从图中可以看出，训练损失从初始的 2.9 左右迅速下降，并在迭代约 30 万次时降至 0.4 以下，且波动较小，说明模型收敛良好、训练稳定。

与此同时，模型在验证集上的准确率持续提升，从初始的 10% 快速上升至接近 90%，表现出强泛化能力。相较于迭代相同次数（30w）的 RNN 模型的验证准确率（约 60%），LSTM 模型的分类性能有了显著提升，表明其更适合建模语言序列中的长距离依赖关系。

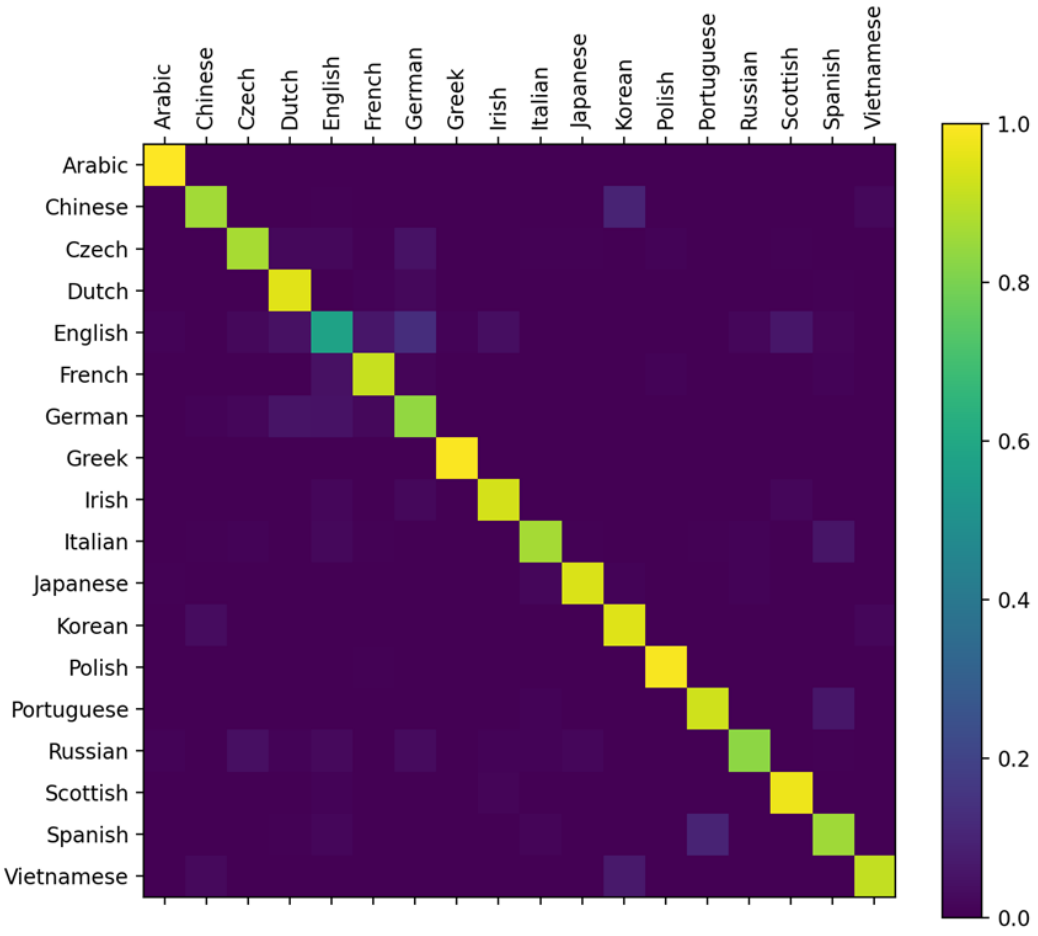


图 7: LSTM 混淆矩阵图

图 5 展示了 LSTM 模型在多语言分类任务中的归一化混淆矩阵。从图中可以看出，矩阵的主对角线呈现出明显的亮黄色，说明模型在绝大多数语言类别上的预测准确率极高，具有良好的分类性能。

与原始 RNN 模型相比，LSTM 显著减少了语言之间的误分类现象。特别是在 Arabic、Chinese、Portuguese、Vietnamese 等语言上的预测几乎无误。此外，尽管 English 与 Dutch、German 等语言之间仍存在一定混淆，但误判比例已显著降低，显示出 LSTM 在处理长依赖语言序列时的优势。

1.2.3 解释为什么 LSTM 网络的性能优于 RNN 网络

1. **缓解长期依赖问题**:

RNN 在处理长序列时，由于信息在时间步之间逐步传递，模型对早期输入的记忆往往会随时间衰减，导致难以捕捉长距离的依赖关系。而 LSTM 通过引入**记忆单元（Cell State）**，结合**遗忘门、输入门、输出门**等门控机制，使得模型可以选择性地保留、更新或丢弃信息，从而**有效延长了记忆的时间跨度**。

在实际任务中，如语言建模与机器翻译，LSTM 能准确建模句首对句尾的影响，而 RNN 则容易丢失早期上下文信息。

2. **解决梯度消失问题**：

   在 RNN 中，由于反向传播跨越多个时间步，梯度容易发生指数级衰减或爆炸，特别是在序列较长时，这将严重影响模型训练效果。而 LSTM 的**线性记忆路径（恒等传递路径）**允许梯度长距离流动，从而**缓解了梯度消失问题**，提高了模型在训练过程中的稳定性。

3. **增强非线性建模能力**：

   RNN 单元结构相对简单，每一步的状态更新仅依赖单一的非线性变换，表达能力有限。而 LSTM 包含多个非线性组合，例如 sigmoid 控制门、tanh 生成候选状态等，**增强了模型的非线性表达能力**，使其能够拟合更复杂的时间动态模式。

1.3 自主实现的 GRU 网络结构

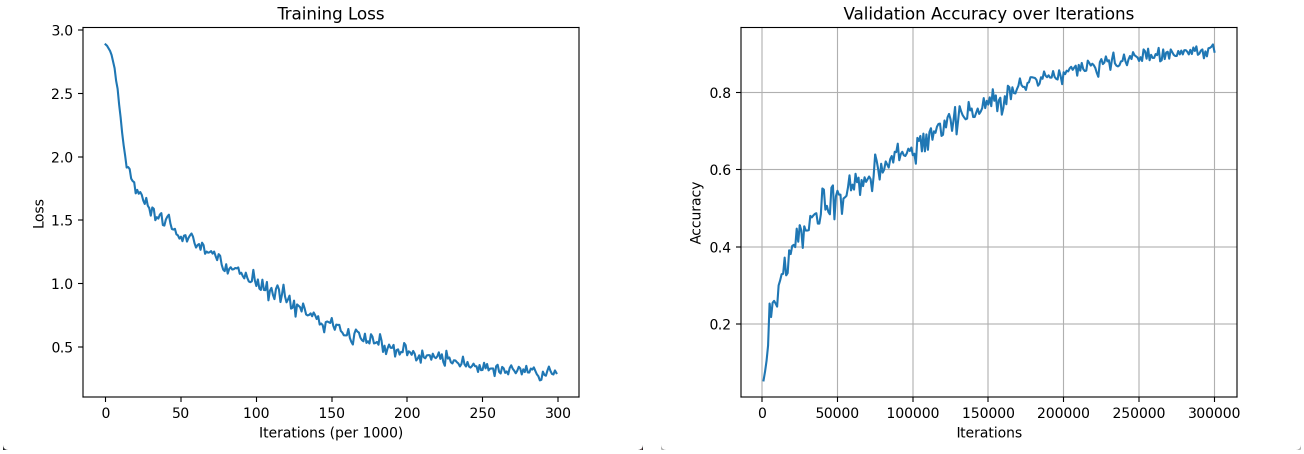
1.3.1 GRU 网络结构

```
GRU(
  (i2z): Linear(in_features=185, out_features=128, bias=True)
  (i2r): Linear(in_features=185, out_features=128, bias=True)
  (i2n): Linear(in_features=185, out_features=128, bias=True)
  (out): Linear(in_features=128, out_features=18, bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

图 8: GRU 网络结构

与 LSTM 相比，GRU 结构更简单，只有两个门控单元（而 LSTM 有三个），没有单独的记忆单元，计算效率更高。

1.3.2 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



(a) GRU 模型的训练 Loss 曲线

(b) GRU 模型的验证 Accuracy 曲线

图 9: GRU 模型在训练过程中的损失与准确率变化

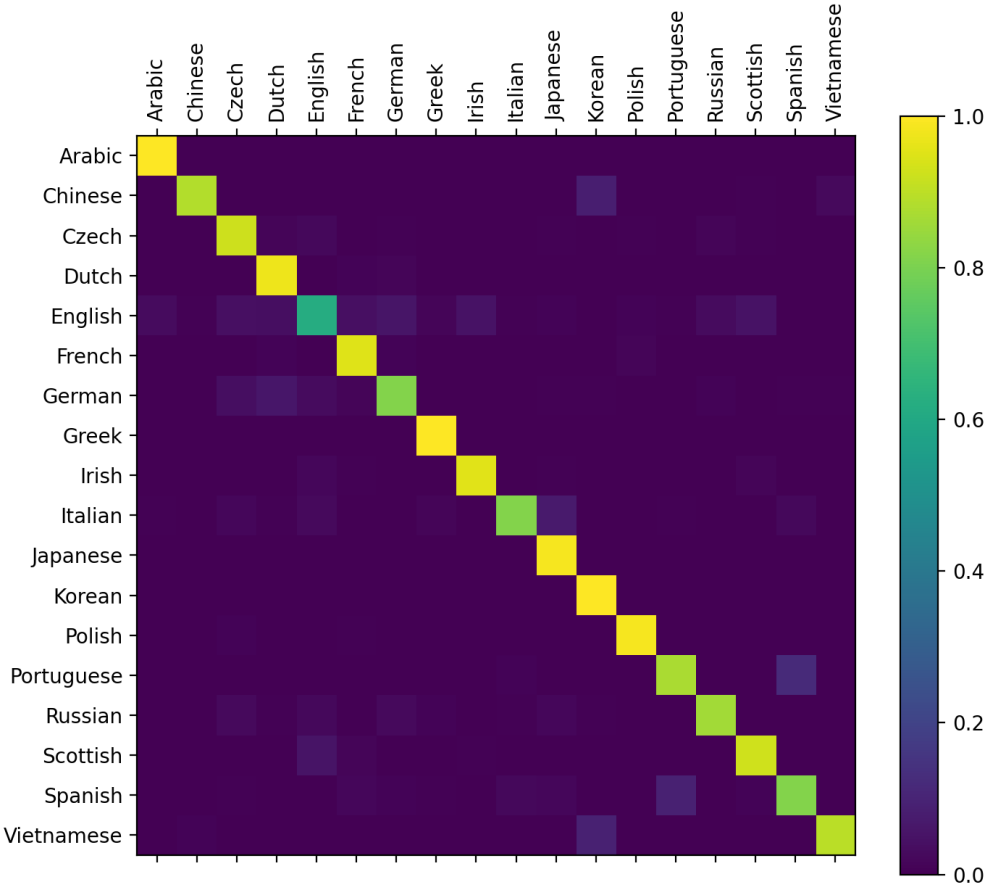


图 10: GRU 混淆矩阵

GRU 模型实验结果分析

图 9(a) 和图 9(b) 展示了 GRU 模型在训练过程中的损失与准确率变化趋势。从图中可以看出，GRU 模型初始训练损失接近 3.0，随着训练迭代的进行，损失快速下降，并最终稳定在 0.4 左右，表明模型训练过程稳定且收敛良好。

同时，验证集上的准确率从初始的约 20% 稳步上升，在迭代约 30 万次后达到约 90%，相比 RNN 的 60% 和 LSTM 的 88%，GRU 模型的泛化能力更强，能够更有效地学习序列中的长期依赖信息。

图 10 为 GRU 模型的归一化混淆矩阵。主对角线颜色亮度高，说明模型在多数语言类别上的识别准确率较高。尤其在 Arabic、Chinese、Portuguese 和 Vietnamese 等语言类别上几乎没有误判。同时，English 与 Dutch、French、German 等语言之间仍存在少量混淆，但混淆程度相较于 RNN 和 LSTM 模型已明显减轻，说明 GRU 对相似语言具备更强的区分能力。

综上所述，GRU 模型在本实验中的表现优异，不仅训练稳定性和准确率高，而且具有较强的泛化能力。相较于 LSTM 结构，GRU 在保持结构相对简单的前提下实现了相当甚至略优的性能，是一种在序列建模中值得关注和推广的结构选择。

1.4 实验总结

通过本次序列建模实验，我深入理解了 RNN 与 LSTM 两类循环神经网络的结构差异与性能表现。在动手实现与训练过程中，我明显感受到传统 RNN 在面对长序列任务时存在梯度消失、信息遗忘等问题，模型性能受限。而 LSTM 通过引入门控机制与记忆单元，有效缓解了这些问题，不仅训练过程更稳定，最终准确率也显著更高。

此外，在绘制 Loss 和 Accuracy 曲线、分析混淆矩阵的过程中，我更直观地理解了模型在训练与泛化方面的差异，这让我意识到理论结构与实际效果之间的紧密联系。实践中也锻炼了我对模型调试、可视化分析以及结果解读的能力。

本实验让我深刻体会到结构设计对于序列学习任务的重要性，也激发了我进一步探索更高效结构（如 GRU、Transformer）的兴趣。