# 机器也懂感情

本次作业是使用 PyTorch 的 LSTM 模型来实现一个名称国籍识别器。即输入名称的拼写，判断出它的国籍。

## 思路和模型

数据方面我们使用的是18中语言的2万条左右的姓氏文本，训练完毕的模型是可以预测出一个姓氏是属于哪种语言的，并且，我们还可以通过模型的预测结果来分析这些语言之间的一些相似性。

我们首先将这些姓氏文本全部转化成为 ascii，再加上标点就可以当做是我们的字母表了。接下来继续将样本的每个姓氏文字按照字母表的索引转化成为数组，这个数组就是我们最初的输入。

接着，我们再使用嵌入层将不同 size 的输入转化为同样维度的输入。最后我们得到 LSTM 的输出后，取出最后一个时间步的结果，送给全连接层，最后再交给一个 logsoftmax 函数得到预测的结果。

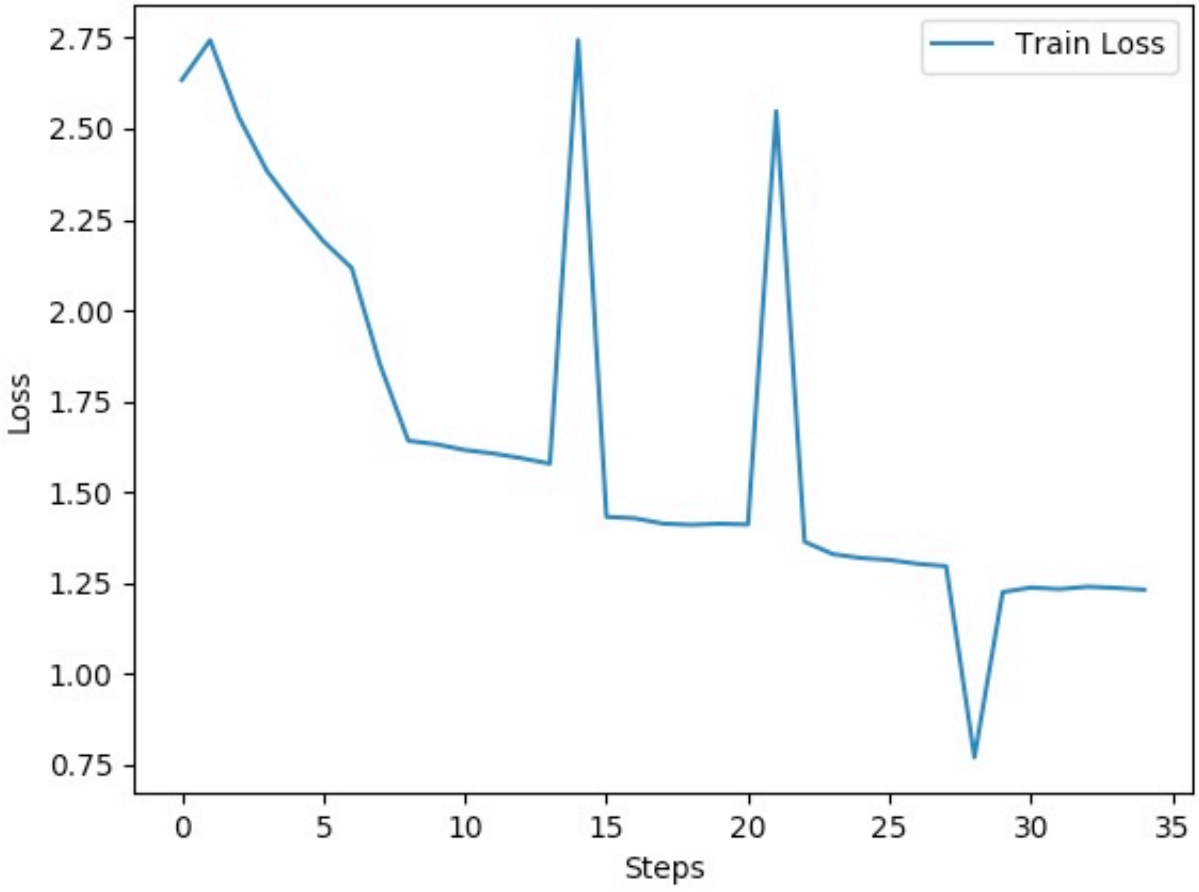
这里在构建 LSTM 实例的时候使用了10个 hidden 单元和2个 layer，我们在实验中也尝试了更多，结果在下面展示。

## 实验结果

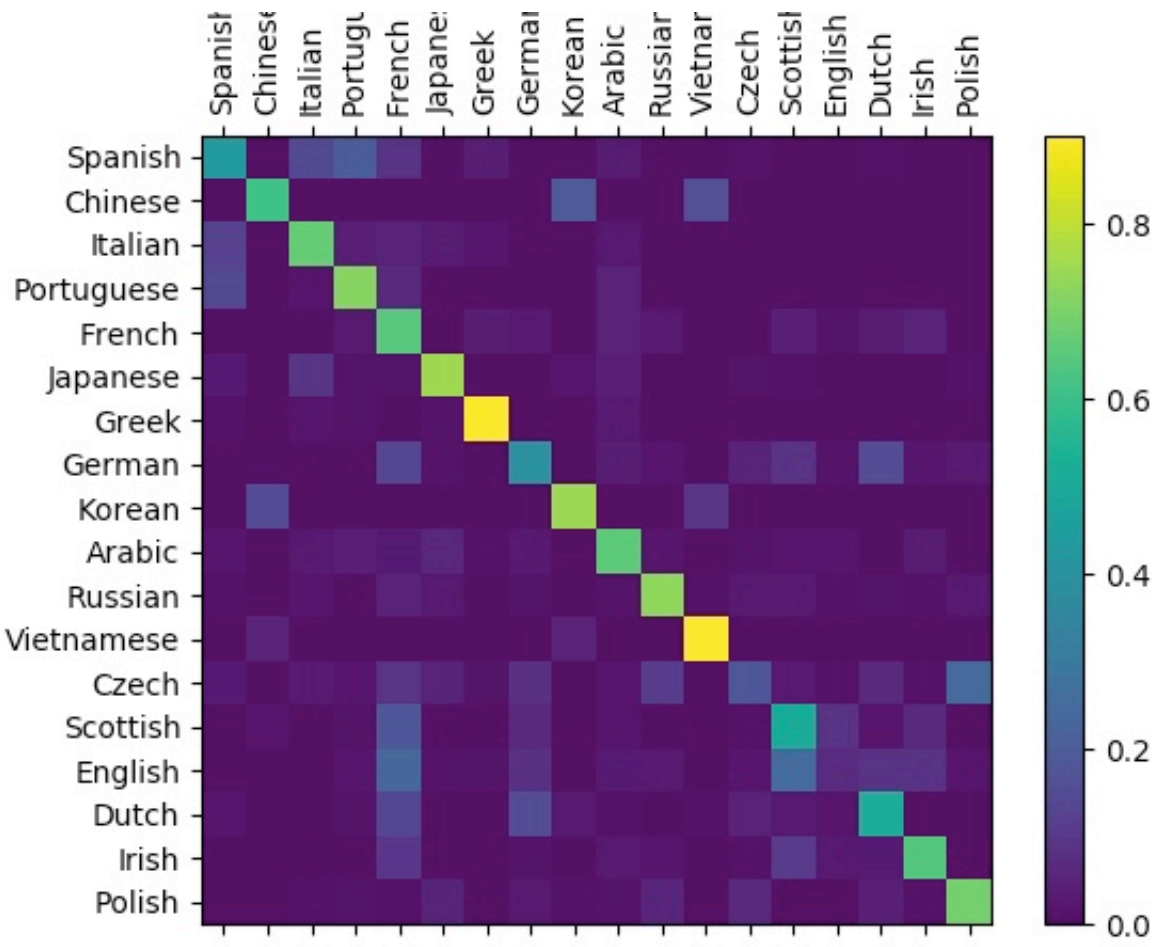
首先是 LSTM 实例中含有10个 hidden 单元和 2 个 layer。

- 第0轮，训练损失：2.63，训练进度：0.0%，（0m 0s），名字：Tahan，预测国家：Arabic，正确？✓
- 第4轮，训练损失：1.23，训练进度：97.93%，（12m 26s），名字：Calpis，预测国家：Greek，正确？✓

我们先来看看损失函数的图：



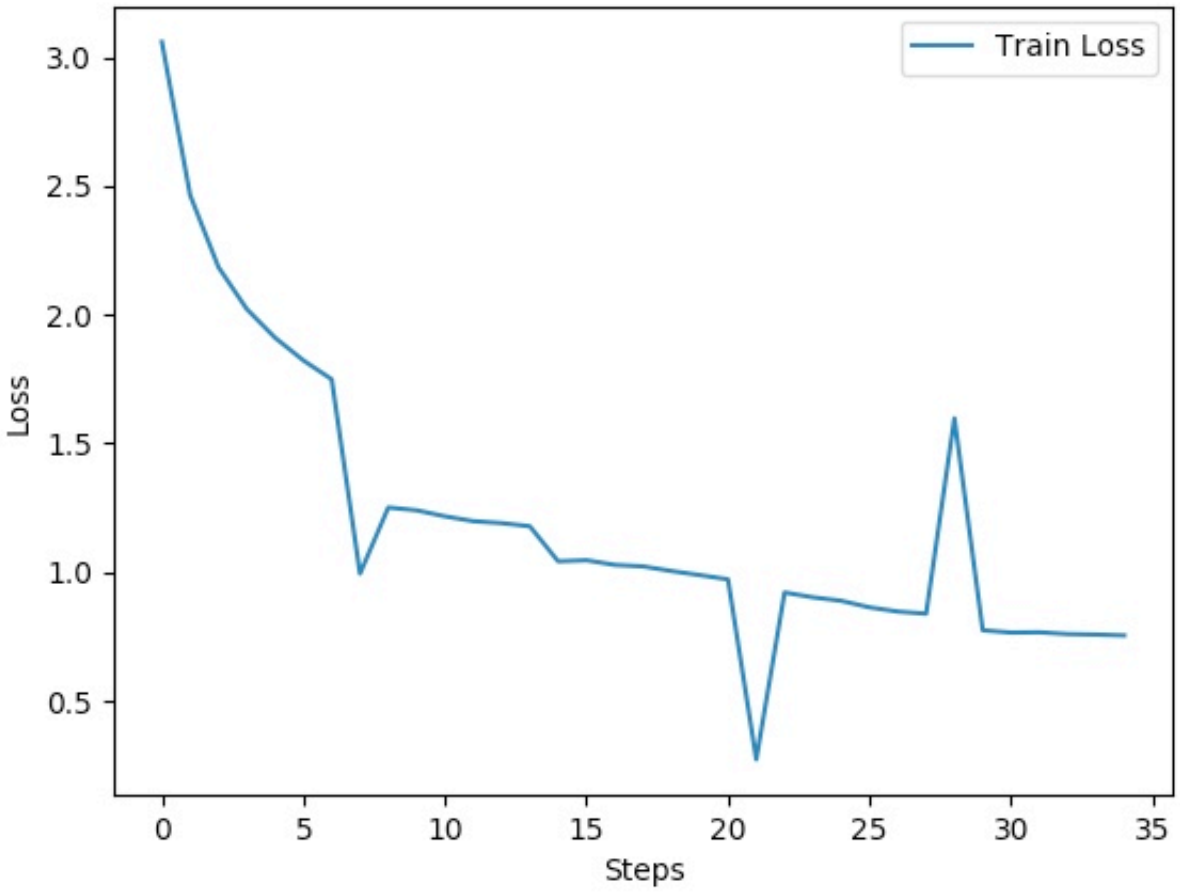

下面是对比语种的测试图：



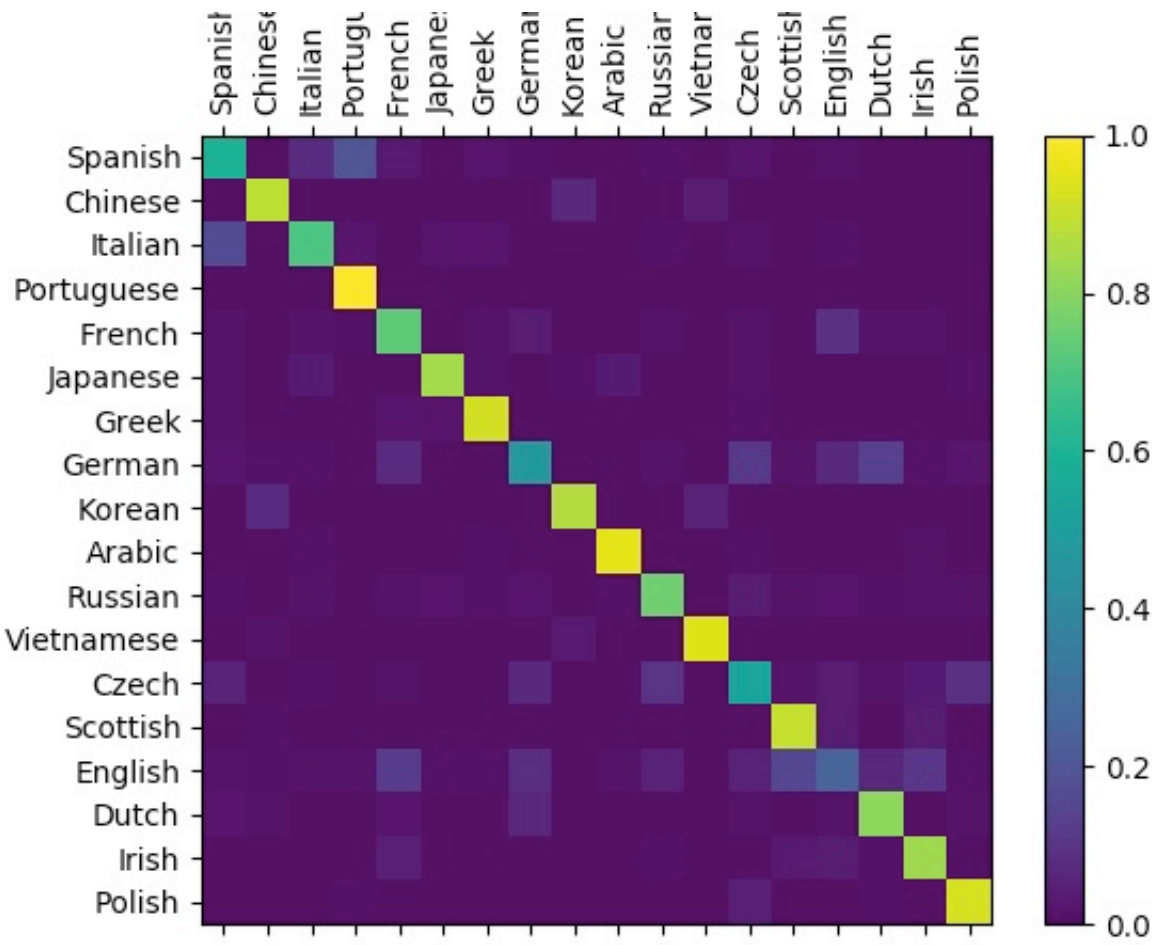

我们看到在在这样的一个测试的结果图中，虽然对角线上很多点已经是绿色和黄色了，但是仍然后个别语言表现很差。具体来看，希腊和越南语因为文字的特殊性比较强，很容易就达到了比较高的正确率，反观英语却不是很好，却很容易被误判成了法语，苏格兰语和德语，一方面是因为现在很多的英语词汇其实都是从这些语言中舶来，另一方面也是因为语系本身的原因。

接下来我们在 hidden 层尝试使用更多的神经元，下面是hidden 含有20个单元的是实验：

- 第0轮，训练损失：3.06，训练进度：0.0%，（0m 0s），名字：Bonnaire，预测国家：Scottish，正确？✗ (French)
- 第4轮，训练损失：0.75，训练进度：97.93%，（13m 13s），名字：Travers，预测国家：French，正确？✓



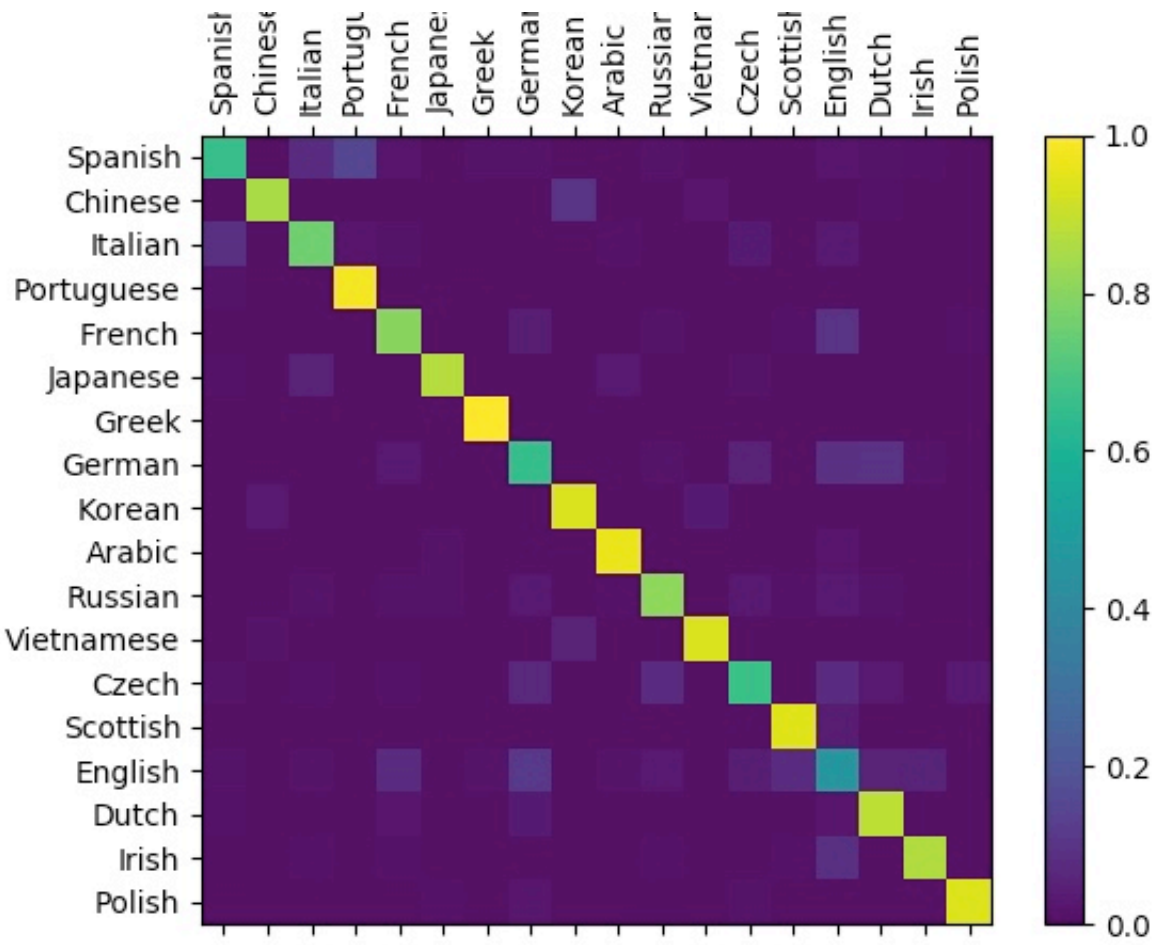

我们发现5轮的训练之后，Loss 相比之前降到了更低，我们继续来看评测图：



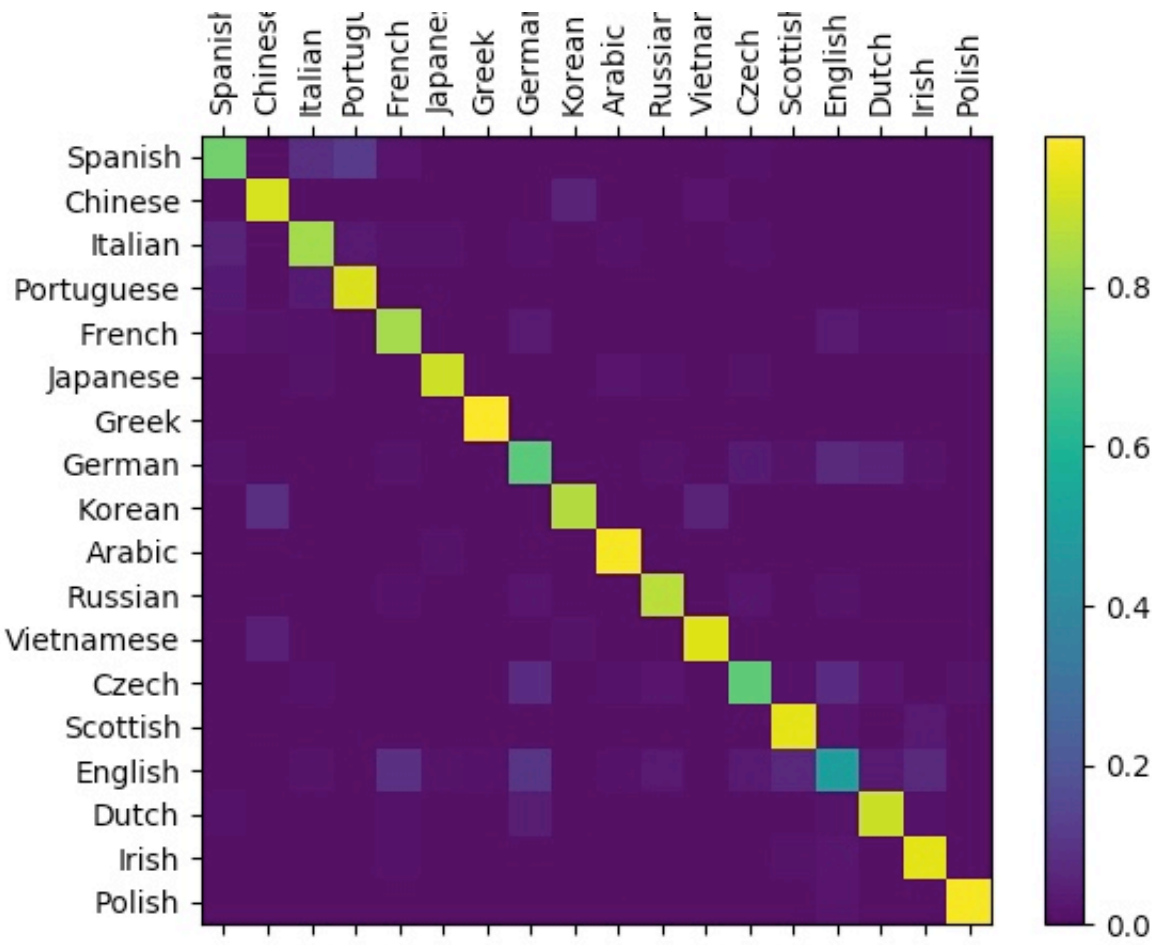

这样的结果图相比之前比来说就更好一些了，除了第一次实验的希腊语和越南语，这次的实验中，葡萄牙语，阿拉伯语和波兰语也得到了比较好的区分，所以说明我们加大 LSTM 结构中隐层神经元单元的数量对分类的效果是有效的，如法炮制，我们不妨继续增多。

下面是分别是30个，40个和50个隐层单元的最终测试：

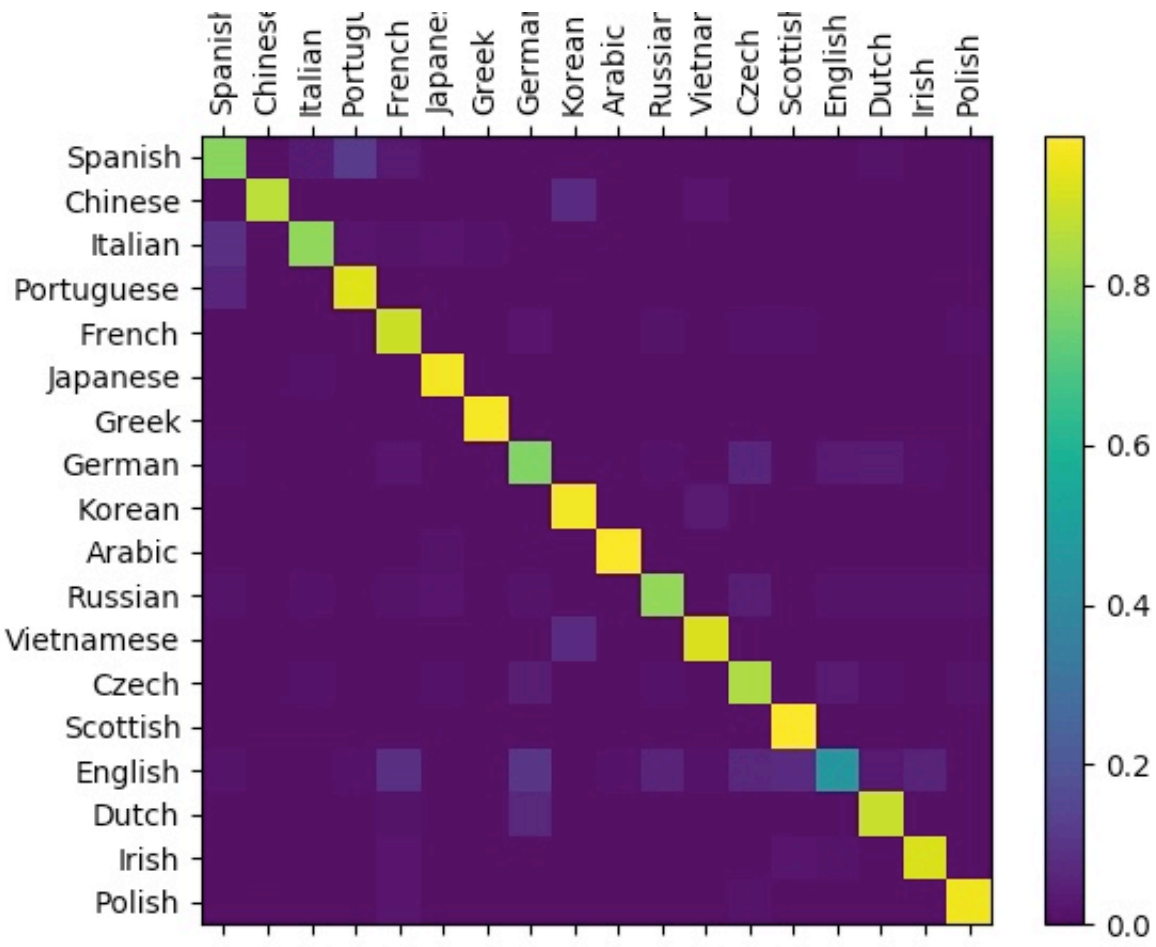
30个隐层单元：


Spanish
Chinese
Italian
Portuguese
French
Japanese
Greek
German
Korean
Arabic
Russian
Vietnamese
Czech
Scottish
English
Dutch
Irish
Polish
1.0
0.8
0.6
0.4
0.2
0.0

40个隐层单元：


Spanish
Chinese
Italian
Portuguese
French
Japanese
Greek
German
Korean
Arabic
Russian
Vietnamese
Czech
Scottish
English
Dutch
Irish
Polish
0.8
0.6
0.4
0.2
0.0

50个隐层单元




我们发现虽然使用更多的隐层单元可以让分类器的性能更好，但是却越来越不明显，实验中使用40个和50个单元已经差别不大了。虽然对比Loss 还是有一些提升。

40个隐层单元的Loss：

- 第0轮，训练损失：3.04，训练进度：0.0%，（0m 0s），名字：Hatoyama，预测国家：Italian，正确？✗ (Japanese)
- 第4轮，训练损失：0.44，训练进度：97.93%，（15m 26s），名字：O'Hannagain，预测国家：Irish，正确？✓

50个隐层单元的 Loss：

- 第0轮，训练损失：2.94，训练进度：0.0%，（0m 0s），名字：Soto，预测国家：Dutch，正确？✗ (Spanish)
- 第4轮，训练损失：0.39，训练进度：97.93%，（16m 59s），名字：Diep，预测国家：Vietnamese，正确？✓

接下来我们在使用50个隐层单元的基础上，继续尝试使用更多的 layer：

3个 layer：

- 第0轮，训练损失：2.91，训练进度：0.0%，（0m 0s），名字：Kim，预测国家：Czech，正确？✗ (Vietnamese)
- 第4轮，训练损失：0.42，训练进度：97.93%，（24m 33s），名字：O'Driscoll，预测国家：Irish，正确？✓

4个 layer：

- 第0轮，训练损失：2.91，训练进度：0.0%，（0m 0s），名字：Haritopoulos，预测国家：Chinese，正确？✗ (Greek)
- 第4轮，训练损失：0.47，训练进度：97.93%，（33m 6s），名字：O'Mooney，预测国家：Irish，正确？✓

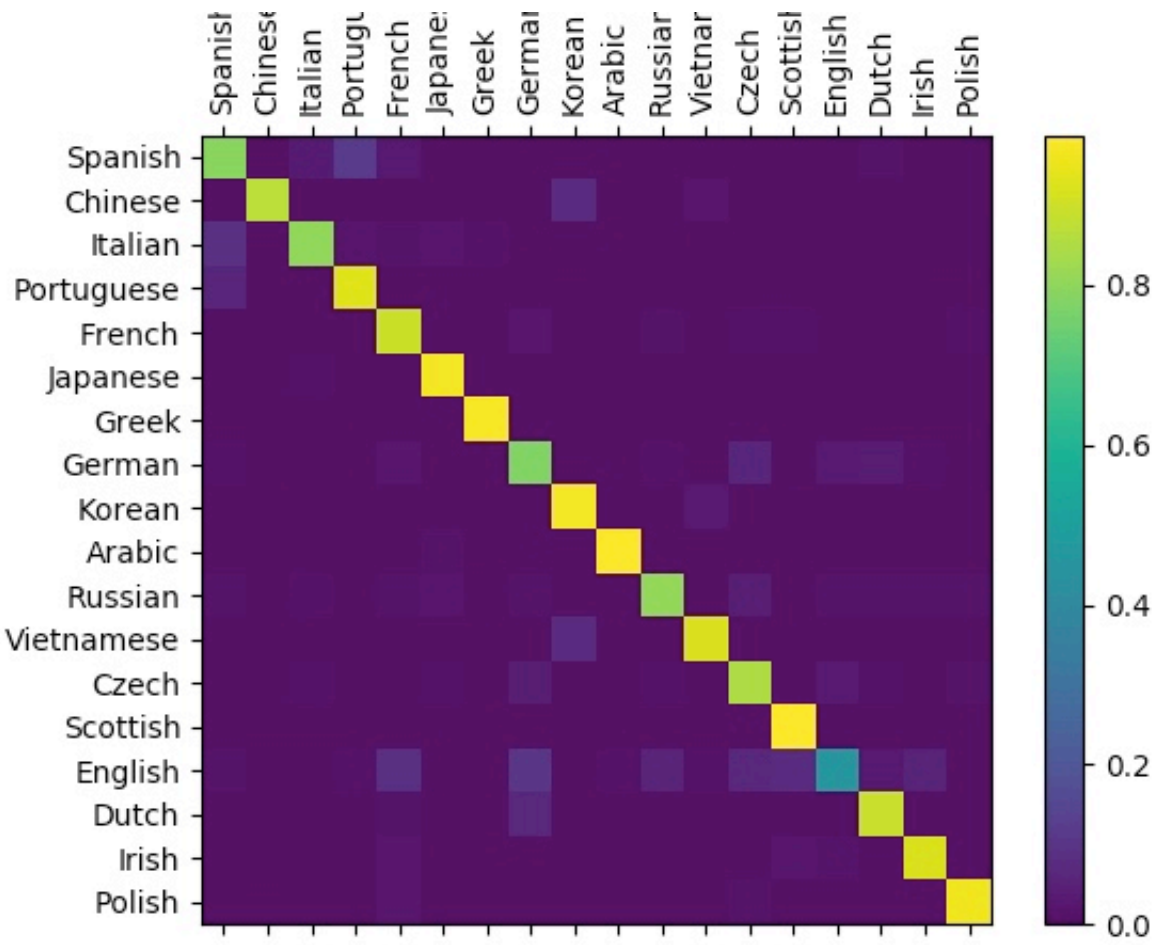
5个 layer：

- 第0轮，训练损失：2.99，训练进度：0.0%，（0m 0s），名字：Laganas，预测国家：Spanish，正确？✗ (Greek)
- 第4轮，训练损失：0.65，训练进度：97.93%，（42m 14s），名字：Davitashvili，预测国家：Italian，正确？✗ (Russian)

从 Loss 的表现看来，单纯增加 layer 并不能提高分类器的性能。

# 结论与分析

下面用hidden size 为50，layer为2的实验测试结果来尝试分析。




从这张图中我们首先可以比较明显的看到有一些语言的分类结果是非常好的，比如阿拉伯语(Arabic)，日语(Japanese)，希腊语(Greek)，爱尔兰语(Irish)和波兰语(Polish)，可能是因为字符规律本身比较特殊导致的。

然后还有一些语言和一两个语言有比较类似的属性导致一些混淆，比如中文和韩语，越南语和韩语，荷兰语和德语和捷克语，可能是因为历史和文化原因导致部分民族语言的共通性。

不出意外的是，英语和很多语言都有类似，比如和苏格兰语，捷克语，俄语，德语，法语，爱尔兰语等等，导致的原因很多，比如民族文化，拉丁语系，以及英美文化的全世界输出等等。