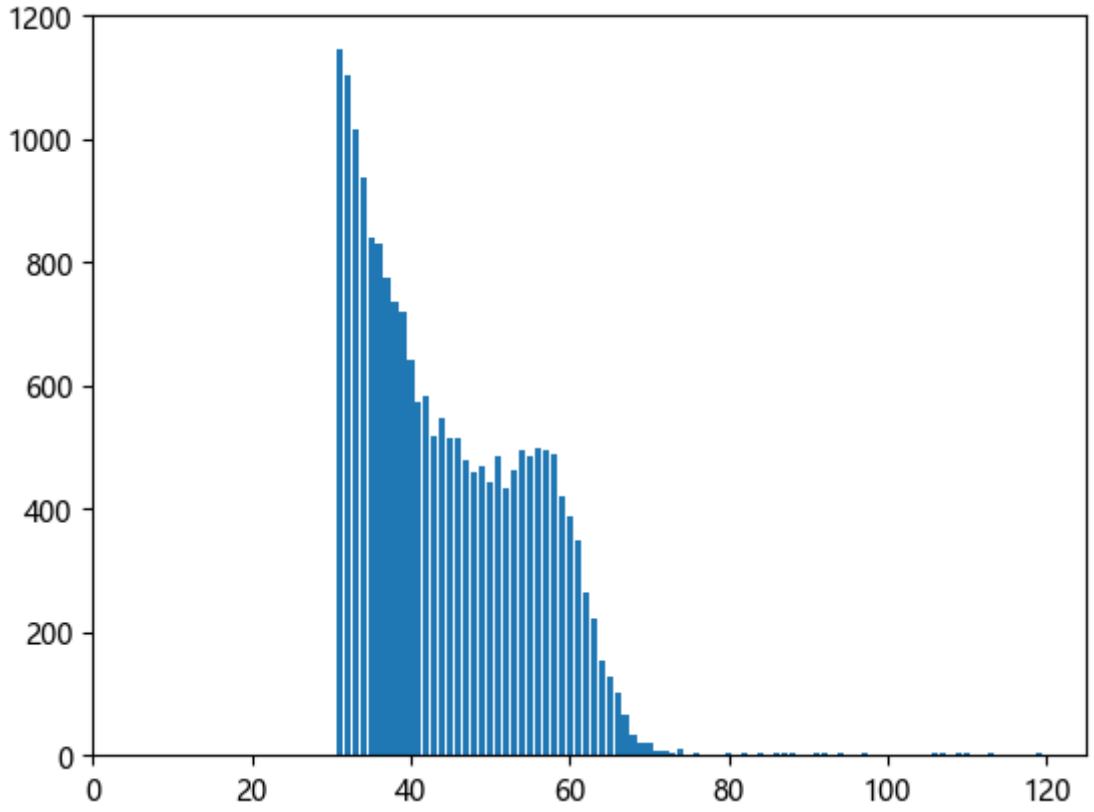
情感分类实验报告

本实验使用 PyTorch 实现了使用 LSTM、GRU 和 CNN 对文本进行情感分类的任务。

数据集

训练集中一共有 53337 个不重复的词汇，共 19998 个句子。验证集中一共 5629 个句子，测试集中一共 369 个句子。

训练集中句子长度分布如下。大多数句子的长度位于于 30 ~ 70 之间，平均句子长度为 45。



实验环境

软件环境：Python 3.8.13，PyTorch 1.13.1。

硬件环境：Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz，Tesla P100-PCIE-16GB

预处理

1. 统计训练集和验证集中的词汇数量。这一步是为了便于对词汇进行筛选，只保留词频较高的词汇的词向量，其他低频词和未知词的词向量同一设置为 `<UNK>` 的词向量。
2. 构建从词汇到序号的映射词典。词典中前 4 个词汇为 `<PAD>`（表示填充较短的句子）、`<UNK>`（表示未知词汇）、`<START>`（表示句子开头）、`<EOP>`（表示句子结尾）。将训练集和验证集中所出现的词频不低于 20 的词汇加入到词典中，其他的词不加入。
3. 构建词嵌入矩阵。首先加载所给的预训练 word2vec 词向量，然后将词典中所出现的词汇对应的词向量挑选出来，拼接成为矩阵；对于词典中出现，但预训练词向量中不包含的词汇（包括预处理时添加的标记性词汇），则分配一个服从 $[-1, 1]$ uniform 分布的词向量。这些词向量在训练时会进行更新。

4. 将句子中不在词典中的词汇替换为 `<UNK>`。
5. 限定句子最大长度 120。对于长度较长的句子，对于训练集和验证集：将其拆分为若干个符合长度限制的句子。例如，一个 200 个词汇的句子，拆分为一个 120 词汇的句子和一个 80 词汇的句子；对于测试集，则直接将句子过长的部分进行裁剪。这一步主要是为了更充分地利用数据集。
6. 在句子首尾分别添加 `<START>` 和 `<EOP>`，因此句子最大长度变为 122。
7. 对于长度不足的句子，在末尾不足 122 的地方使用 `<PAD>` 填充。

经过预处理后，词典大小为 5994，词嵌入为 5994×50 的矩阵，训练集为 20003×122 的矩阵，验证集为 5629×122 的矩阵，测试集为 369×122 的矩阵。

实验设置

RNN 网络结构：词嵌入层 - 双向 LSTM/GRU - 线性层A - ReLU - 线性层B。

其中 LSTM/GRU 的输出维度为 $\text{hidden\_size} \times 2$，线性层 A 的输出维度为 hidden_size，线性层 B 的输出维度为 2。对于 LSTM，$\text{hidden\_size} = 16$；对于 GRU，$\text{hidden\_size} = 32$。

CNN 网络结构：词嵌入层 - 1D卷积层A - ReLU - 1D最大池化层A - 1D卷积层B - ReLU - 1D最大池化层B - 1D卷积层C - ReLU - 1D最大池化层C - 线性层A - ReLU - 线性层B。

其中池化层的窗口均为 2，线性层 A 的输入维度为 1920，输出维度为 32，线性层 B 的输出维度为 2。

	Kernel size	Kernel number	Padding	Stride
卷积层A	3	32	1	1
卷积层B	3	64	1	1
卷积层C	3	128	1	1

超参数设置：

- 初始学习率：0.01
- 优化算法：Adam
- 批量大小：512
- 损失函数：交叉熵损失函数
- 参数初始化：Xavier Uniform

Parameters	LSTM	GRU	CNN
Dropout rate	0.8	0.9	0.5
Epoch	30	30	10

代码说明

文件 `main.py` 用于训练和测试模型。其中核心的函数和类：

- 预处理：
  - `get_word_count`：统计训练集和验证集中的词汇数量
  - `build_vocab`：构建从词汇到序号的映射，并添加四个标记性词汇
  - `build_embedding_matrix`：根据词汇到序号的映射和预训练的词向量，构建词嵌入矩阵
  - `TextDataset`：继承 PyTorch 提供的 Dataset 类，对数据集进行读取和预处理
  - `get_loader`：加载数据集，一遍批量读取
- 模型：

    - `RNN`：实现了文本二分类的循环神经网络，根据 `cell` 参数可以设定为 LSTM 或 GRU 网络
    - `TextCNN`：实现了一个文本二分类的卷积神经网络
- 训练：
    - `train_epoch` ：使用训练集对模型参数进行一轮完整更新，并计算样本平均损失。
    - `train` ：通过调用 `train_epoch` 函数，对模型参数进行 `EPOCH` 轮更新，并保存验证集表现最好的模型参数，绘制训练中的损失曲线。
- 评估：
    - `evaluate`：测试模型表现
    - `count_correct`：统计 TP、TN、FP、FN 四个值
    - `get_metrics`：计算 accuracy、recall、precision、f1-score
    - `plot_figure`：绘制训练曲线
- 辅助功能：
    - `args` ：命令行参数
    - `log` ：日志记录
    - `seed_evetything`：固定随机数种子，确保实验可复现

文件 `config.py` 实现了命令行参数的解析，以及一些常量的配置。

三个模型的权重位于 `models` 文件夹中。

运行代码

安装依赖：

```
pip install torch matplotlib tqdm
```

训练模型：（最佳参数设置）

```
python main.py --model_type LSTM --hidden_size 16 --dropout 0.8 --epoch 30 --
output_path LSTM
python main.py --model_type GRU --hidden_size 32 --dropout 0.9 --epoch 30 --
output_path GRU
python main.py --model_type CNN --dropout 0.5 --epoch 10 --output_path CNN
```

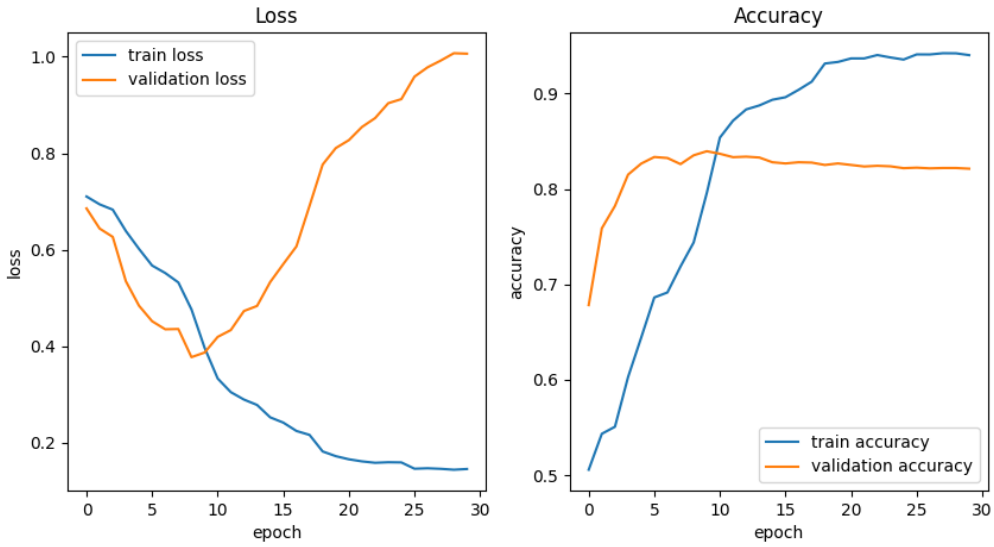
测试模型：（`model_path` 为模型参数文件路径）

```
python main.py --test true --model_path models/LSTM.pth --model_type LSTM
python main.py --test true --model_path models/GRU.pth --model_type GRU
python main.py --test true --model_path models/CNN.pth --model_type CNN
```

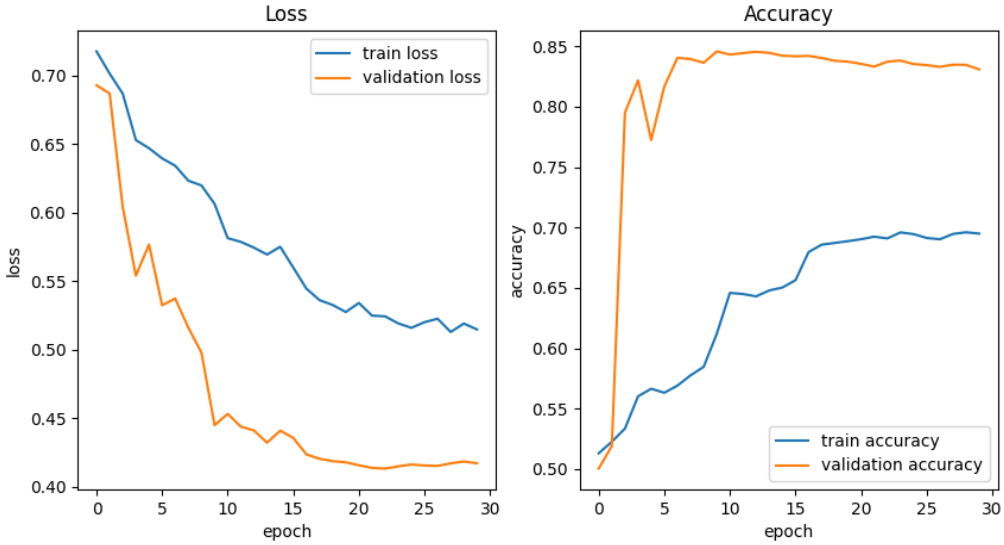
实验结果

训练损失曲线：

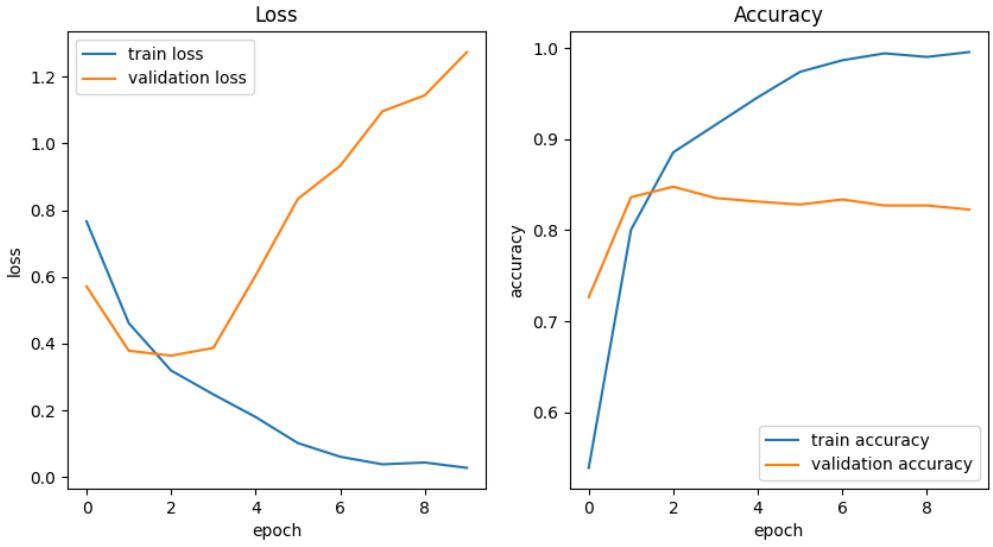
LSTM：



GRU：



CNN：



实验结果：

训练集：

Model	Accuracy	Precision	Recall	F1-score
LSTM	0.7439	0.9299	0.5278	0.6734
GRU	0.6909	0.6195	0.9902	0.7622
CNN	0.8855	0.8901	0.8796	0.8848

验证集：

Model	Accuracy	Precision	Recall	F1-score
LSTM	0.8353	0.8555	0.8065	0.8303
GRU	0.8373	0.8292	0.8492	0.8391
CNN	0.8478	0.8530	0.8400	0.8464

测试集：

Model	Accuracy	Precision	Recall	F1-score
LSTM	0.8537	0.8889	0.8128	0.8492
GRU	**0.8645**	0.8870	**0.8396**	**0.8626**
CNN	0.8509	**0.8976**	0.7968	0.8442

实验分析

预处理

本实验将最大句子长度设为 120，如果训练集的句子过长，则会对其进行拆分。事实上由于本实验使用的数据集句子长度相对较短，该预处理的方法效果并不明显；如果数据集中有较多句子较长的句子，则能够有效提高数据集利用率。

另外，预处理过程还可以做得更加精细。对训练集中的词汇进行统计后发现，词频较高的前十个词汇是：

```
('PS', 268), ('gt', 219), ('lt', 207), ('星给', 152), ('電影', 147), ('a', 143),
('ps', 134), ('s', 122), ('T', 119), ('P', 108)
```

其中 "gt"、"lt" 明显是 HTML 的标签，"T T" 似乎是某个表情，"PS"和"ps" 是同一个含义，“星给”事实上是两个单字，例如“一颗/星/给/剧情”。如果使用性能更好的分词系统，应该能使得分类性能进一步提升。

过拟合

本实验的过拟合现象十分严重。原因是仅词嵌入矩阵有 $5994 \times 50 = 299700$ 个可学习的参数了（并且这还是筛选了词频高于 20 的词汇后的结果），而训练集大小仅约为 20000，这对于一个二分类任务而言，参数量过大了。过拟合在训练过程中表现为随着训练集损失降低，验证集损失却不减反增。因此，本实验采用了若干方法抑制过拟合：

- 将词频低于 20 的词汇设为 `<UNK>`
- 使用较大的 dropout rate
- 使用参数量尽可能小的神经网络

另外，如果神经网络部分如果过于简单（例如将 LSTM 的 hidden_size 设为 8，或将 GRU 的 dropout 设为 0.9），会导致网络直接不收敛。

总结

根据测试集的准确率和 F1-score，可知模型测试的结果是：GRU 略好于 LSTM，LSTM 略好于 CNN 。准确率均明显高于实验的最低要求 0.83。