实验四：循环神经网络实现

120L021716 蔡泽栋

一、实验目的：

利用 Pytorch 自己实现 RNN、GRU、LSTM 和 Bi-LSTM。利用上述四种结构进行文本多分类（60%）计算测试结果的准确率、召回率和 F1 值；对比分析四种结构的实验结果。

任选上述一种结构进行温度预测（40%）使用五天的温度值预测出未来两天的温度值；给出与真实值的平均误差和中位误差。

二、实验环境：

Python 3.9.13; torch =2.0.0+cu117; torchvision=0.15.0+cu117; GPU

```
python                    3.9.13          h6244533_2
python-dateutil           2.8.2              pypi_0
pytorch-mutex             1.0                  cpu

pytz                      2023.3             pypi_0
regex                     2023.3.23          pypi_0
requests                  2.28.1        py39haa95532_1
scikit-learn              1.2.2              pypi_0
scipy                     1.10.1             pypi_0
setuptools                65.6.3        py39haa95532_0
six                       1.16.0          pyhd3eb1b0_1
sqlite                    3.41.1          h2bbff1b_0
sympy                     1.11.1        py39haa95532_0
threadpoolctl             3.1.0              pypi_0
tk                        8.6.12          h2bbff1b_0
torch                     2.0.0+cu117        pypi_0
torchvision               0.15.0+cu117       pypi_0
tqdm                      4.65.0             pypi_0
```

三、文本多分类实验内容：

1. 网络结构实现：

用于文本多分类的网络结构如下：（以 RNN 为例）

```python
class RNN(nn.Module):
    def __init__(self, embeddings, embedding_dim, hidden_dim, output_dim):
        super().__init__()
        self.embedding = nn.Embedding.from_pretrained(embeddings=embeddings, freeze=False, padding_idx=0)
        self.rnn = RNN_series.RNN(embedding_dim, hidden_dim)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, text):
        # text = [sent len, batch size]

        embedded = self.embedding(text)  # embedded = [batch size, sent len, emb dim]

        output, hidden = self.rnn(embedded)  # output = [sent len, batch size, hid dim]

        # hidden = [1, batch size, hid dim]

        assert torch.equal(output[:, -1, :], hidden.squeeze(0))

        return self.fc(hidden.squeeze(0))
```

包含三层：embedding 层，RNN 层以及全连接层。embedding 层使用了预训练权重，RNN 层是自行实现的循环神经网络结构，全连接层用于归一化结果。

2. 数据读取与预处理：

2.1 build_words.py

首先读入 online_shopping 数据集并将 label 映射为 number。对于 review 使用 jieba 库进行分词，按照要求按序号划分训练集、测试集和验证集。最后保存到本地.word.txt 文件中

2.2 build_vocab.py

按照词的顺序构建词典，词表保存至 vocab.words.txt

2.3 build_embeddings.py

为了给 RNN 模型的 embedding 层提供初始化权重,使用开源词向量 Chinese-Word-Vectors 所提供的 sgns.merge.bigram。其中 embedding 矩阵的大小为[vocab_size+1,300]：vocab 表中出现的所有词都有对应的词向量，多出的 1 表示<UNK>和<PAD>。

对于词表中的词，将其词向量置为在开源词向量 Chinese-Word-Vectors 中对应的 300 维向量，若其未出现在开源词向量 Chinese-Word-Vectors 中，则将其对应的 300 维词向量全置为 0。初始化的词向量权重保存在 w2v.npz 中。

2.4 load_online_shopping.py

按照 max_len 切分每一段 review，并将文本转化为文本在 embedding 矩阵中对应的行号。最后将数据装入 dataloader 以供之后训练。

3. 定义优化器并训练

使用交叉熵作为损失函数，Adam 作为优化器，训练模型的主要模块与前几次实验相同。增加了 sklearn.classification_report 方法，以便直接计算准确率，召回率与 F1 值。

对于每个模型，进行 50 个 epoch 训练，学习率为 0.0001。

四、文本多分类实验结果分析：

测试过程不需要 train 训练，而是将之前保存好的最优模型 load 并在 test 集上测试即可

RNN 测试结果：

```
Epoch:49/50   Train Loss: 0.1799   Val Loss: 0.5980   Acc:0.8343
Epoch:50/50   Train Loss: 0.1717   Val Loss: 0.6978   Acc:0.8281
Finished 50 epoch
Test Acc:0.8349   Test Loss:0.6608
Online shopping evaluation results:
              precision    recall  f1-score   support

          书籍       0.88      0.90      0.89       770
          平板       0.77      0.71      0.74      1999
          手机       0.68      0.74      0.71       464
          水果       0.88      0.90      0.89      1996
         洗发水       0.76      0.83      0.79      1997
         热水器       0.47      0.06      0.11       115
          蒙牛       0.87      0.85      0.86       407
          衣服       0.84      0.85      0.84      1999
         计算机       0.75      0.74      0.75       798
          酒店       0.98      0.96      0.97      1999

    accuracy                           0.84     12544
   macro avg       0.79      0.75      0.76     12544
weighted avg       0.83      0.84      0.83     12544
```

GRU 测试结果：

```
Test Acc:0.8767   Test Loss:0.4219
              precision    recall  f1-score   support

          书籍       0.99      0.92      0.95       769
          平板       0.75      0.82      0.79      1999
          手机       0.81      0.80      0.81       464
          水果       0.94      0.88      0.91      2000
         洗发水       0.79      0.86      0.82      1997
         热水器       0.50      0.03      0.05       115
          蒙牛       0.99      0.98      0.99       407
          衣服       0.87      0.90      0.88      1998
         计算机       0.94      0.85      0.89       797
          酒店       0.99      0.97      0.98      1998

    accuracy                           0.88     12544
   macro avg       0.86      0.80      0.81     12544
weighted avg       0.88      0.88      0.88     12544
```

LSTM 结果：

```
Test Acc:0.8797   Test Loss:0.4474
              precision    recall  f1-score   support

          书籍       0.94      0.94      0.94       768
          平板       0.79      0.81      0.80      2000
          手机       0.82      0.83      0.83       463
          水果       0.92      0.89      0.90      1999
         洗发水       0.81      0.83      0.82      1999
         热水器       0.62      0.35      0.45       115
          蒙牛       0.99      0.95      0.97       406
          衣服       0.88      0.89      0.89      1997
         计算机       0.91      0.90      0.91       798
          酒店       0.98      0.97      0.97      1999

    accuracy                           0.88     12544
   macro avg       0.87      0.84      0.85     12544
weighted avg       0.88      0.88      0.88     12544
```

Bi-LSTM 结果：

```
Test Acc:0.8823   Test Loss:0.4200
              precision    recall  f1-score   support

          书籍       0.96      0.94      0.95       769
          平板       0.81      0.78      0.80      1996
          手机       0.84      0.80      0.82       464
          水果       0.90      0.91      0.90      2000
         洗发水       0.81      0.84      0.82      1997
         热水器       0.62      0.51      0.56       114
          蒙牛       1.00      0.96      0.98       407
          衣服       0.87      0.92      0.89      1999
         计算机       0.90      0.88      0.89       798
          酒店       0.98      0.97      0.98      2000

    accuracy                           0.88     12544
   macro avg       0.87      0.85      0.86     12544
weighted avg       0.88      0.88      0.88     12544
```

	precision	recall	F1
RNN	0.84	0.84	0.84
GRU	0.88	0.88	0.88
LSTM	0.88	0.88	0.88
Bi-LSTM	0.88	0.88	0.88

对比实验结果可以得出：

1.GRU，LSTM，Bi-LSTM 的结果几乎没有差距，RNN 的训练效果略差

2.RNN 的收敛速度也慢于另外三个网络，GRU，LSTM，Bi-LSTM 在 epoch=20 以后结果基本稳定，而 RNN 用了约 40 轮。

五、温度预测实验内容：

1. 网络结构实现：

用于温度预测的网络结构如下：

```
class JenaNet(nn.Module):
    def __init__(self, embedding_dim, hidden_dim, output_dim):
        super().__init__()
        self.rnn = RNN_series.GRU(embedding_dim, hidden_dim)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, data):
        output, hidden = self.rnn(data)

        assert torch.equal(output[:, -1, :], hidden.squeeze(0))

        return self.fc(hidden.squeeze(0))
```

不涉及词与词向量的映射，相较于之前去掉了 embedding 层。

2. 数据读取与预处理：

首先处理日期数据。通过 datetime.strptime()将原本的字符串转化成 datetime 类型的数据，便于后续处理。将日期数据转化为正余弦函数的数值形式，使得模型在训练时能够感知到时间周期。按要求前 5 年为一段，后 2 年为一段划分数据集。根据原始数据数据集得到模型的输入与输出，将数据装入 dataloader。

3. 定义优化器并训练

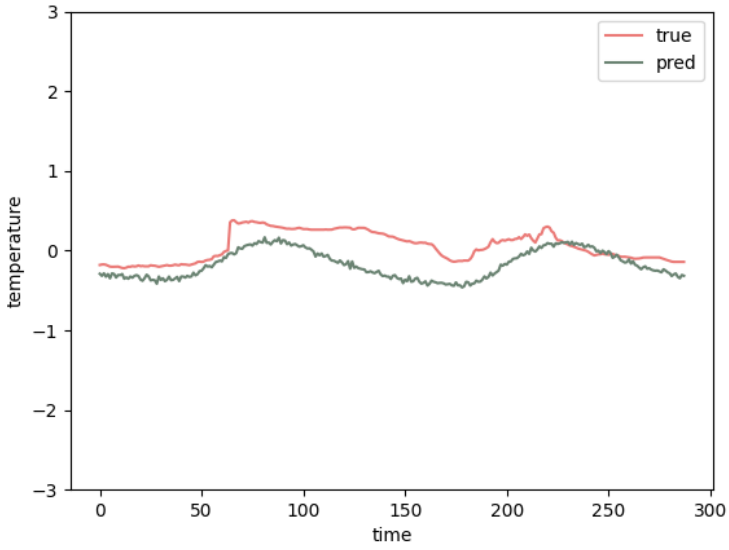
使用交叉熵作为损失函数，Adam 作为优化器，训练模型的主要模块与前几次实验相同。

对于每个模型，进行 50 个 epoch 训练，学习率为 0.0001。

六、文本多分类实验结果分析：

训练结果：

```
Test Loss:0.1375
Avg error: 0.2303
Median error: 0.1966
```



附：相关文件说明
JenaNet.py：用于温度预测的网络
OnlineShoppingNet.py：用于文本多分类的网络
RNN_series.py：循环神经网络结构
RNNcell_series.py：循环神经网络单个单元的结构
model 文件夹：训练好的模型
runs 文件夹：文本多分类的 tensorboard 日志
runs_jena 文件夹：温度预测的 tensorboard 日志