目录

1 摘要

该工程通过构建卷积神经网络 CNN 对 IMDB 影评进行语言情感分析。报告介绍了算法原理，网络结构和代码整体流程，并通过调试参数达到最优输出，并对算法输出结果和性能进行分析。

数据集：IMDB数据集，包含来自互联网电影数据库（IMDB）的50000条严重两极分化的评论。数据集被分为用于训练的25000条评论和用于测试的25000条评论，训练集和测试集中都包括50%的正面评价和50%的负面评价。IMDB数据集内置于Keras库中，它已经过预处理，单词序列的评论已经被转换为整数序列，其中每个整数代表字典中的某个单词。

任务：请在训练数据中选择前10000个最常出现的单词，并且在预训练的网络之上构建分类器。

2 引言

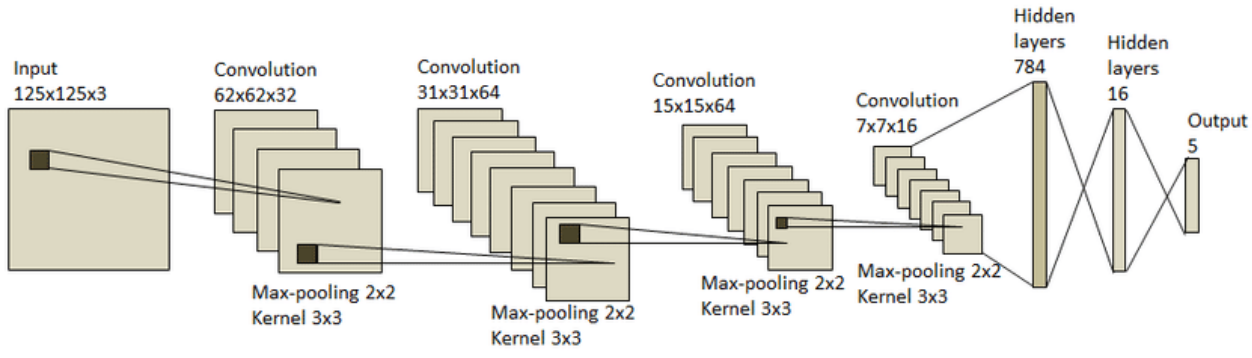
该工程构建了一个简单的卷积神经网络 CNN，在预训练的网络之上构建分类器，对 IMDB 影评进行语言情感分析。

3 算法

3.1 算法原理介绍

卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元。

卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络需要考量的参数更少，使之成为一种颇具吸引力的深度学习结构。



3.2 网络结构说明

3.2.1 常用层

常用层定义了一系列常用的网络层，在本工程中包括全连接层、激活层和 Dropout 层。

- `Dense`就是常用的全连接层，所实现的运算是`output = activation(dot(input, kernel) + bias)`。其中`activation`是逐元素计算的激活函数，`kernel`是本层的权值矩阵，`bias`为偏置向量，只有当`use_bias=True`才会添加。
- 输入

  形如`(batch_size, ..., input_dim)`的 ND 张量，最常见的情况为`(batch_size, input_dim)`的2D张量
- 输出

  形如`(batch_size, ..., units)`的 ND 张量，最常见的情况为`(batch_size, units)`的2D张量
- `Activation`层

  激活层对一个层的输出施加激活函数
- `Dropout`层

  为输入数据施加 Dropout。Dropout将在训练过程中每次更新参数时按一定概率（rate）随机断开输入神经元，Dropout层用于防止过拟合。

3.2.2 一维卷积层

一维卷积即为时域卷积，对一维输入信号进行邻域滤波。其原型为

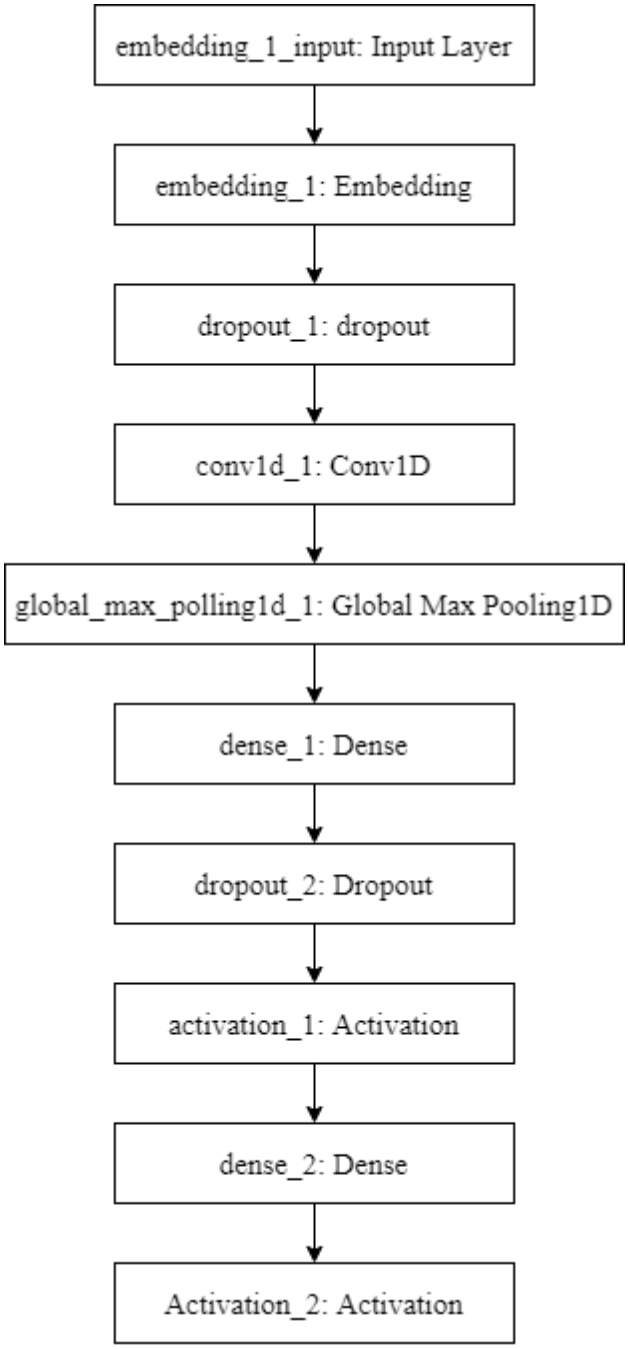
```
keras.layers.convolutional.Conv1D(filters, kernel_size, strides=1,
padding='valid', dilation_rate=1, activation=None, use_bias=True,
kernel_initializer='glorot_uniform', bias_initializer='zeros',
kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None,
bias_constraint=None)
```

该层生成将输入信号与卷积核按照单一的空域（或时域）方向进行卷积。主要参数如下：

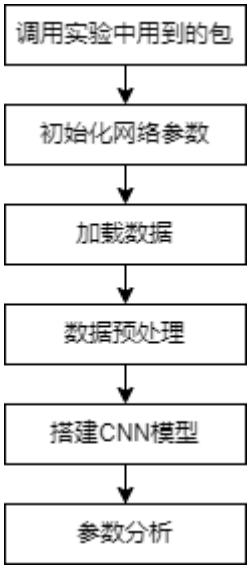
1. `filters`：卷积核的数目（输出维度）
2. `kernel_size`：卷积核的空域或时域窗长度，由整数或由单个整数构成。
3. `strides`：卷积的步长，由整数或由单个整数构成，数据类型为`list/tuple`
4. `padding`：补0策略，为`"valid" "same" "causal"`

5. `activation`：激活函数。

3.2.3 网络结构图



3.3 代码整体流程图



4 实验分析结果

客观指标：必须提供Precision精确率、Recall召回率结果，其他评价指标也可附加。

4.1 超参数调试

对神经网络中 `batch_size，epoch，优化器和损失函数，激活函数` 等超参数进行调试

4.1.1 batch_size

将全连接层神经元个数，`epoch` 设为默认值

```
hidden_dims = 500
epochs = 2
```

```
batch_size = 8  <==

============ Building Model... ============
Epoch 1/2
loss: 0.3691 - accuracy: 0.8266 - val_loss: 0.2655 - val_accuracy: 0.8910
Epoch 2/2
loss: 0.2101 - accuracy: 0.9183 - val_loss: 0.2614 - val_accuracy: 0.8935
```

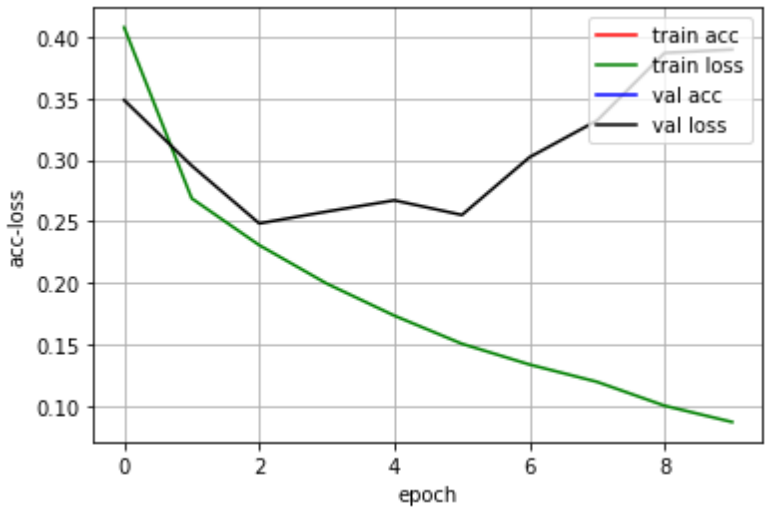
```
batch_size = 16 <==

============ Building Model... ============
Epoch 1/2
loss: 0.3803 - accuracy: 0.8166 - val_loss: 0.2741 - val_accuracy: 0.8848
Epoch 2/2
loss: 0.2077 - accuracy: 0.9172 - val_loss: 0.2591 - val_accuracy: 0.8953
```

batch_size	train loss	test loss	train accuracy	test accuracy
8	0.2101	0.2614	0.9183	0.8935
16	0.2077	0.2591	0.9172	0.8953
32	0.2153	0.2612	0.9166	0.8934
64	0.2271	0.2637	0.9097	0.8909

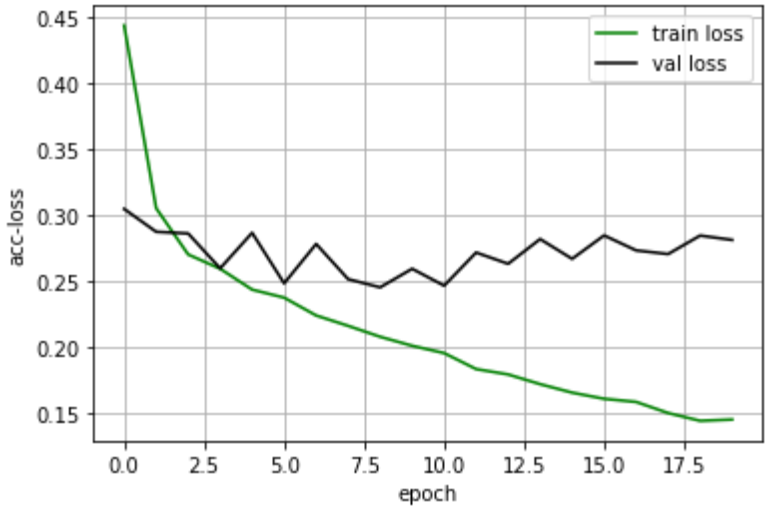
可以看出在`batch_size > 16`时，`train loss`和`test loss`均出现升高趋势，故设置`batch_size = 16`

4.1.2 epoch

```
hidden_dims = 500
batch_size = 16
epoch = 10  <==
```



`train loss`不断下降，而`test loss`随着`epoch`增大而增大，说明神经网络出现过拟合，调整`dropout`来降低过拟合。



改变嵌入层和全连接层的`dropout`神经网络过拟合有所降低，在`epoch=10`处取得最优。

4.1.3 Optimizer

初始选用`Adam`优化器对神经网络进行优化。使用`RMSprop`优化器进行优化后输出结果

```
Epoch 1/3
1563/1563 [==============================] - 88s 56ms/step - loss:
0.4036 - accuracy: 0.8061 - val_loss: 0.4774 - val_accuracy: 0.7798
Epoch 2/3
1563/1563 [==============================] - 86s 55ms/step - loss:
0.2663 - accuracy: 0.8900 - val_loss: 0.2537 - val_accuracy: 0.8966
Epoch 3/3
1563/1563 [==============================] - 86s 55ms/step - loss:
0.2290 - accuracy: 0.9107 - val_loss: 0.2479 - val_accuracy: 0.8981
```

明显优于`Adam`优化方法，故选用`RMSprop`优化器对神经网络进行优化。

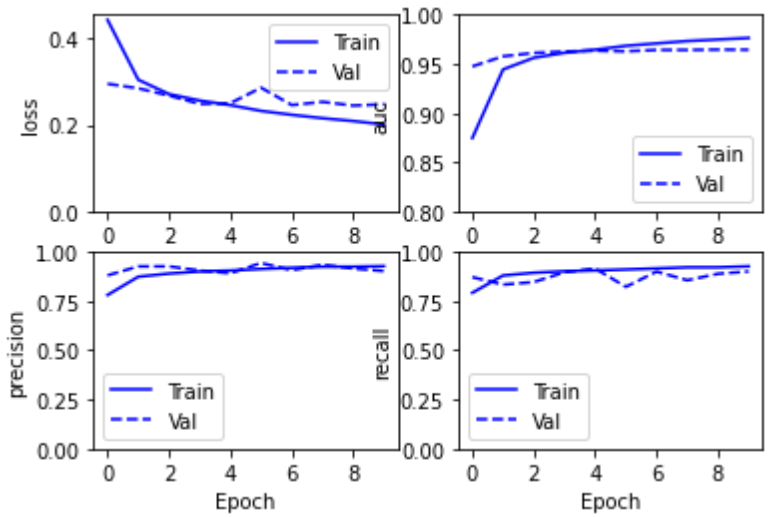
4.2 实验结果对比与算法优劣性分析

4.2.1 实验结果

在调整好的超参数空间中搭建神经网络进行训练

```
# 词向量的维度
embedding_dims = 50
filters = 250
kernel_size = 3
# 全连接层中，神经元的个数，批处理量
batch_size = 16
hidden_dims = 500
epochs = 10
optimizer ='RMSprop'
```

得到 Loss 损失曲线、AUC 曲线、Precision 精确率曲线以及 Recall 召回率曲线



最终的准确率，召回率和F1值为

```
Precision    = 92.55%

Recall       = 92.46%

F1           = 92.50%
```

4.2.2 问题分析

从准确率、召回率可以看出该网络的分类性能较好，但当 `epoch` 增大时，发现 `train loss` 减小 `test loss` 增大，在 `epoch = 5` 时，`precision & recall` 都有所下降，说明网络中出现过拟合。增加 `dropout` 后过拟合有所减小。但由于该网络结构比较简单，若继续减小过拟合，还需采用正则化或多个网络集成来实现。

5 启发与收获

本次实验让我学会了如何搭建神经网络对数据集进行分类，调节神经网络中超参数的方法以及评估神经网络性能的方法，包括 `loss, precision, recall...` 还学到了神经网络中出现过拟合时的解决方法。对 `keras` 中其他结构的使用也进一步熟悉。

6 参考资料或文献

[1] 卷积神经网络 -- Wikipedia

[2] https://www.jiqizhixin.com/articles/2018-05-28-8

[3] https://towardsdatascience.com/how-to-increase-the-accuracy-of-a-neural-network-9f5d1c6f407d

[4] https://keras-cn.readthedocs.io/en/latest/layers/core_layer/