

代码文件和深度学习小知识

目录

1 代码的正确打开方式	3
1.1 jupyter 格式	3
1.2 代码编辑器	3
1.3 Python 环境问题	5
2 代码报错指南	5
2.1 常见错误集合	5
2.2 其他代码运行问题	10
3 必看深度学习知识点	10
3.1 什么是 batch_size ?	10
3.2 什么是过拟合、欠拟合，怎么解决对应问题	11
3.3 什么是中间数据文件	11
3.4 什么是权重文件 .pt	11
3.5 模型混淆矩阵问题	12
3.6 样本数据做归一化?	12
3.7 样本数据集长度选择问题 119808	13
4 模型调参	13
4.1 CNN 模型（包含 CNN 模块的）	13
4.2 LSTM （GRU）模型调参	14
4.3 Transformer 相关模型调参	14
4.4 其它模型调参	15

1 代码的正确打开方式

注意：代码严格按照我们提供的方法或者教程先去跑通，然后再自己进行转换，.ipynb 代码不能直接 粘贴到.py 文件里面运行，需要分文件编写，因为一些训练、测试等耗时操作不能放在一个文件里面运行！（不要直接 拿着上来就用 pycharm 跑， 或者尝试修改数据集和代码，咱们最开始应该先跑通 原始数据集和代码，再做替换或者修改！）

先按照以下方法和教程跑通代码（关于转为.py 文件可以参考我们后期推出的视频）

1.1 jupyter 格式

 FFT-CNN-Tranformer故障分类.ipynb

使用 Jupyter 风格的代码有以下几个优势：

(1) 交互性：Jupyter 风格的代码可以逐个单元格地执行，可以实时查看中间结果并进行调试。这种交互性使得代码的开发和调试过程更加方便和高效。

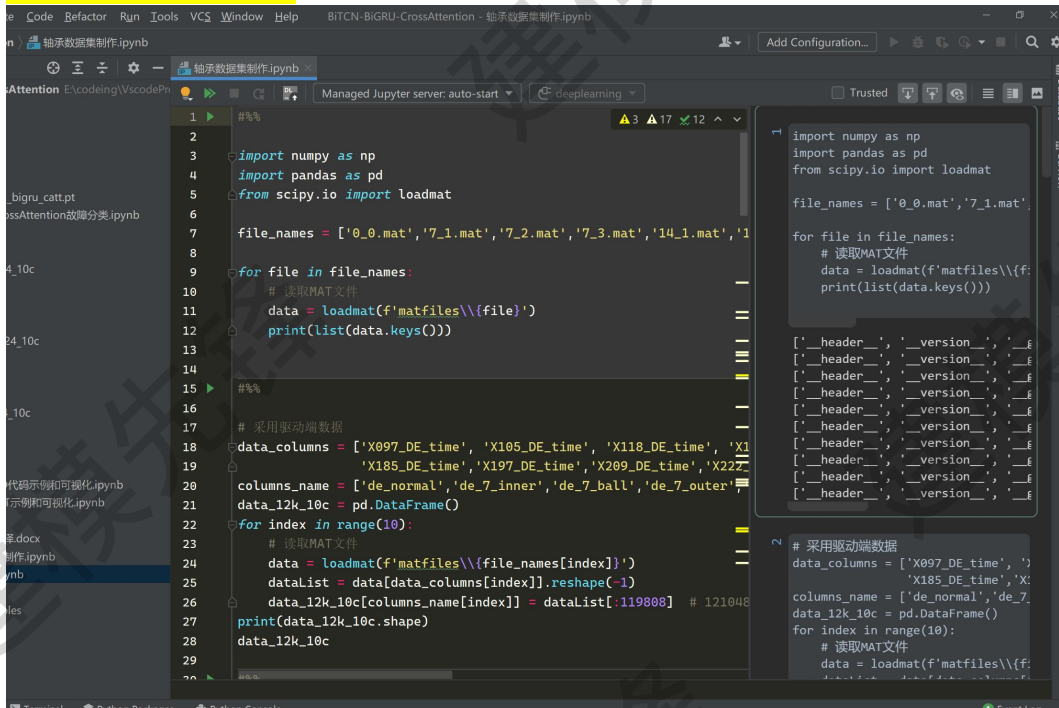
(2) 可读性：Jupyter 风格的代码以单元格的形式展示，每个单元格可以包含一段完整的代码或文档，使得代码更具可读性和可理解性。这对于分享和协作非常有用。

(3) 数据可视化：Jupyter 支持在代码环境中直接生成图表、图像和其他形式的数据可视化。这使得数据分析和探索更加直观和方便。

1.2 代码编辑器

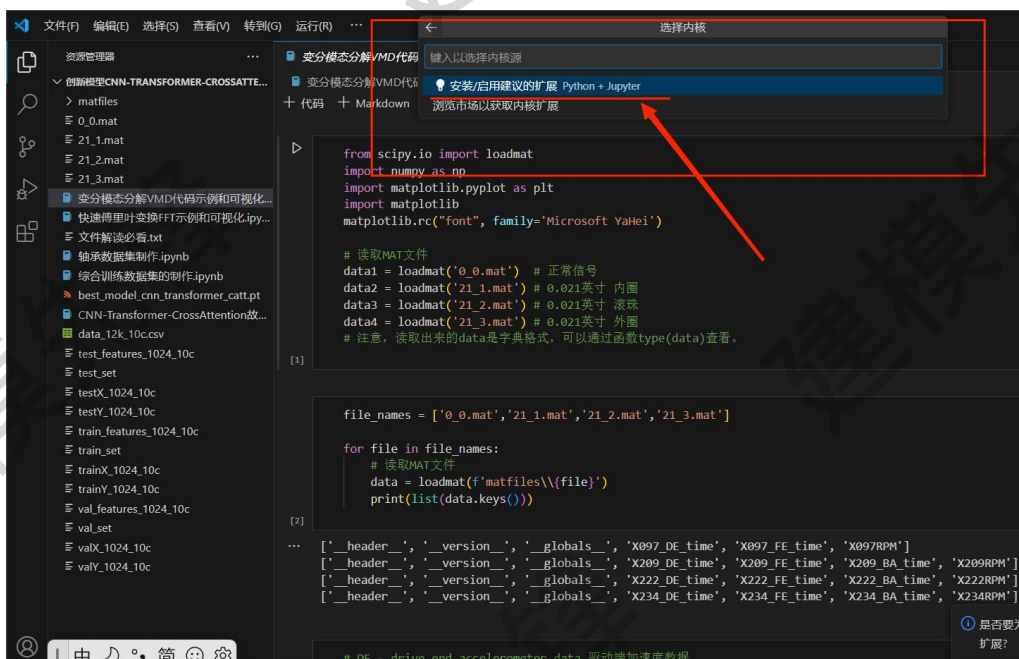
推荐两种运行方式：

(1) Pycharm 运行 jupyter 代码，需要自己环境： `pip install jupyter` 来运行代码!!!



(2) VScode （推荐使用！）我们也提供详细的教程

[《Python 入门教程》从环境安装，到深度学习实战（三）VSCode 编辑器 \(qq.com\)](#)




(3) 不推荐 网页端 使用 jupyter

路径没设置正确的话，容易出错，读取不到同目录下的其他文件！


1.3 Python 环境问题


- (1) 要求：python 3.8 ， pytorch 1.8 版本及其以上
- (2) 如果你是初学者，或者没有用过 Anaconda 来管理 python 环境，我们强烈建议按照我们提供的详细教程 来重新配置环境：

[《Python 入门教程》从环境安装，到深度学习实战\(一\)Anaconda 环境管理\(上\)\(qq.com\)](#)

 Anaconda3-2024.02-1-Windows-x86_64.exe

 requirements.txt

 VSCodeUserSetup-x64-1.87.0.exe

 pip.ini

- (3) 如果你已经配置好了 Anaconda 的 python 环境，但是安装的相关 python 包比较少， 或者和我们的版本不太一样，我们强烈建议，按照我们的教程 **批量安装 我们代码模型所需要的包**

[《Python 入门教程》从环境安装，到深度学习实战\(二\)Anaconda 环境管理\(下\)\(qq.com\)](#)

我们提供我们模型代码所需要的各种 python 包，和批量按照教程！

2 代码报错指南

2.1 常见错误集合

- (1) 路径报错

```

In [6]: from scipy.io import loadmat
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font', family='Microsoft YaHei']

# 读取MAT文件
data1 = loadmat('0_0.mat') # 正常信号
data2 = loadmat('21_1.mat') # 0.021英寸 内圈
data3 = loadmat('21_2.mat') # 0.021英寸 滚珠
data4 = loadmat('21_3.mat') # 0.021英寸 外圈
# 注意, 读取出来的data是字典格式, 可以通过函数type(data)查看.

FileNotFoundError                                Traceback (most recent call last)
D:\Anaconda\lib\site-packages\scipy\io\matlab\_mio.py in _open_file(file_like, appendmat, mode)
    38     try:
--> 39         return open(file_like, mode), True
    40     except OSError as e:

FileNotFoundError: [Errno 2] No such file or directory: '0_0.mat'

During handling of the above exception, another exception occurred:

FileNotFoundError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_3920\1138907418.py in <module>
      6
      7 # 读取MAT文件
--> 8 data1 = loadmat('0_0.mat') # 正常信号
      9 data2 = loadmat('21_1.mat') # 0.021英寸 内圈
     10 data3 = loadmat('21_2.mat') # 0.021英寸 滚珠

D:\Anaconda\lib\site-packages\scipy\io\matlab\_mio.py in loadmat(file_name, mdict, appendmat, **kwargs)
    222
    223     variable_names = kwargs.pop('variable_names', None)
--> 224     with _open_file_context(file_name, appendmat) as f:

```

问题 1: 网页端 使用 jupyter, 路径没设置正确的话, 容易出错, 读取不到同目录下的其他文件!

解决方法: 百度设置正确的路径, 或者使用 pycharm 或者 VScode

问题 2: 没有下载 完整的代码数据文件

解决方法: 仔细检查 购买界面中 付费页面 的百度网盘链接 (有的话) 和附件!

(2) 混淆矩阵画不出来, 或者 其他绘图不出现

```

97     print('Best accuracy: ', best_accuracy)
98
99
100
101 batch_size = 64
102 epochs = 50
103 # 模型训练
104 model_train(batch_size, epochs, model, optimizer, loss_function, train_loader, val_loader)

Epoch: 46 val_Loss:0.12353009, validate_Acc:0.9682
Epoch: 47 train_Loss: 0.11167810 train_Accuracy:0.9688
Epoch: 47 val_Loss:0.24160907, validate_Acc:0.9310
Epoch: 48 train_Loss: 0.08718583 train_Accuracy:0.9708
Epoch: 48 val_Loss:0.07830544, validate_Acc:0.9790
Epoch: 49 train_Loss: 0.04738315 train_Accuracy:0.9858
Epoch: 49 val_Loss:0.07980966, validate_Acc:0.9779
Epoch: 50 train_Loss: 0.03632217 train_Accuracy:0.9893
Epoch: 50 val_Loss:0.07813856, validate_Acc:0.9795

Duration: 262 seconds

```

如这种情况: 损失和准确率的图就是不会画出来, 后面的混淆矩阵也是没有参考: [使用 Seaborn 画热力图, 只有第一行单元格显示数值 python 热力图只显示第一行的数字-CSDN 博客](#)

解决办法: 把画图包版本, 退后几个版本, 如下:

```

scipy 1.10.1
seaborn 0.12.2

```


3.7.1

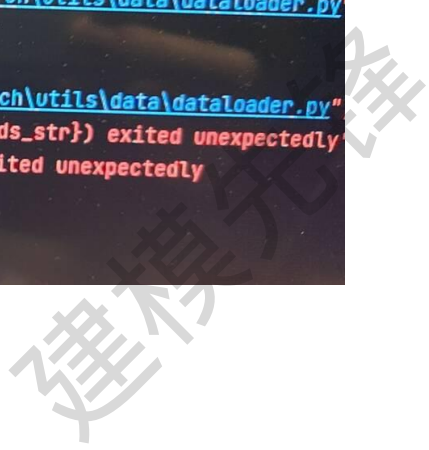
以了

建模先锋

```
1 # 下载
2 pip install EMD-signal
3
4 # 导入
5 from PyEMD import EMD
```

```
1 # 下载
2 pip install EM
3
4 # 导入
5 from PyEMD imp
```

ch\Lib\site-packages\torch



```

torch.manual_seed(100) # 设置随机种子，以使实验结果
device = torch.device("cuda" if torch.cuda.is_ava
# 加载数据集
def dataloader(batch_size, workers=2):
    # 训练集
    train_xdata = load('trainX_1024_10c')
    train_ylabel = load('trainY_1024_10c')
    # 验证集
    val_xdata = load('valX_1024_10c')
    val_ylabel = load('valY_1024_10c')
    # 测试集
    test_xdata = load('testX_1024_10c')
    test_ylabel = load('testY_1024_10c')

    # 加载数据
    train_loader = Data.DataLoader(dataset=Data.T
                                batch_size=batch_size,

```

解决办法：把 workers 设置 为 0， 把 batch_size 设置小一点。

(5) 其它报错问题

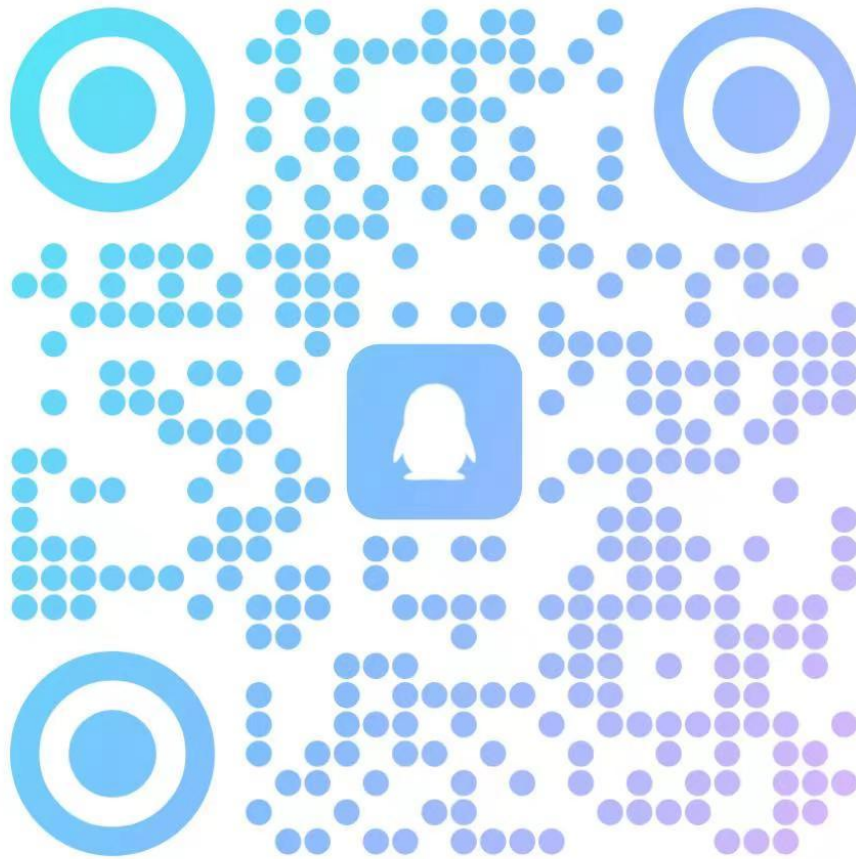
第一步，先按照 我们提供的环境包进行安装，然后按照我们的教程进行运行代码，前面都提供了的！

第二步，**在第一步的情况下，如果还有错误！**，直接添加开发人员 qq, 留言即可！



建模先锋

504965502



扫一扫
加我为好友



2.2 其他代码运行问题

(1) 代码运行时长问题

跟自己的设备性能相关，需要更大的内存和 GPU！

(2) 图片数据运行 GPU 相关问题

为什么 GPU 没用起来，反倒 CPU 利用率那么高？

情况 1：看训练过程，自己设备的 GPU 用上没有，没用上的话，那就是自己设备的 cuda、torch 和驱动版本没安装好，这个百度一下，网上教程很多的！

情况 2：涉及到大量的数据传输操作，例如大规模的模型权重加载或大规模的数据集传输，这些操作可能会导致 CPU 占用率升高，而 GPU 占用率较低；某些模型的结构和计算特性可能导致在 GPU 上无法充分并行化执行，从而导致 GPU 的占用率较低；CPU 占用率较高而 GPU 占用率较低可能是正常的，因为 PyCharm 本身也在使用 CPU 资源

(3) 每次运行结果不一样？

这个深度学习模型，参数初始化是随机种子固定了，但是每次运行 因为参数更新梯度下降的过程中，下降步长和方向具有随机性，导致参数更新的过程 是没办法一致的，结果都不尽相同，所以我们需要保留实验结果最好的那次，注意备份训练好的权重文件和网络结构（主要是参数记录好就行），一般论文 大家都是写的最好的结果的！

3 必看深度学习知识点

3.1 什么是 batch_size ？

(1) 制作完的数据集 和 送入 网络中的数据集长度 会不一样，比如说，测试集有 100 个样本，你的 batch_size 设置为 16 那么送入网络中的数据集就是 $100/16 = 6 \dots 4$ 就是有 6 个 batch，共 96 个样本，剩下的 4 个会被舍弃，就是保留的样本量一定会被 batch_size 整除，余量会被舍弃，这里不理解的话，百度再查！！

(2) batch_size 选择，一般为 2 的个数，比如 8，16，32，64，128，256；样本量多 batch_size 就大，样本量特别少，batch_size 就大就少，这个后面可以通过对比实验进行测试，再选择；

3.2 什么是过拟合、欠拟合，怎么解决对应问题

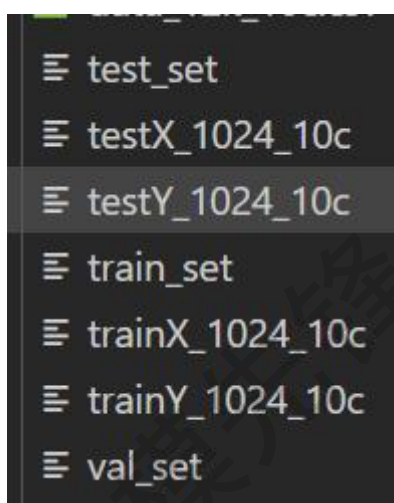
(1)过拟合：就是训练集准确率特别高，和测试集准确率差 10 个点左右及其以上这样的情况；

解决方法：减小模型参数量，减小模型层数和维度数；

(2)欠拟合：训练集、测试集效果都不好，准确率都偏低

解决方法：增大模型参数量，增加模型层数，每层维度数，或者融入其他模块！

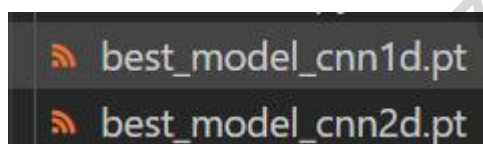
3.3 什么是中间数据文件



这种数据格式都是中间二进制文件的，不用管，读取使用就行。

不要在意，它保留是什么格式，要想它保留前是什么类型的数据，读取后就是什么类型的数据。保存，就好比：一个黑箱子，你数据本来是什么格式，放进去，拿出来还是什么格式的，但是不要关心这个黑箱子是什么形状。而且也打不开！

3.4 什么是权重文件 .pt



.pt 文件是一种二进制格式，用于将 PyTorch 模型保存到磁盘。你可以使用 PyTorch 提供的 `torch.save()` 函数将模型保存为 .pt 文件，然后使用 `torch.load()` 函数加载模型以便在其他地方使用。是 PyTorch 框架中用于保存和加载模型权重和结构的一种格式。

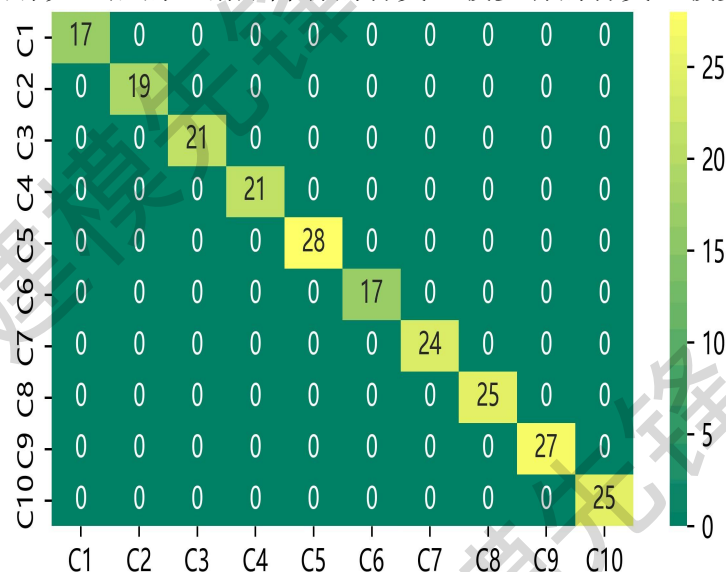
.pt 是网络训练好的权重文件，后面你要是自己训练，遇到好的结果，就把

这个权重文件 备份一下，同时参数的设置也要备份，因为每次运行训练的代码，结果都会覆盖这个文件，而且深度学习每次运行结果都不尽相同，好的结果需要备份的！

不备份，下次运行都会直接覆盖的！

3.5 模型混淆矩阵问题

训练完之后的混淆矩阵有的种类比较多有的种类比较少？



虽然最开始 每个 样本的数据集 长度是一样的，但是数据制作过程中。是随机的，然后 数据集加载 也是随机的，样本都是随机分配的，所以测试集的样本量 并不是均匀分布的。

如果要均匀分布，得进行样本分层 抽样；每种样本取同样的数据量。

3.6 样本数据做归一化？

不一定，做归一化 的作用是加速模型收敛，节省运行时间，可能 对结果更好，也可能对最终结果 变换不大

3.7 样本数据集长度选择问题 119808

```
data = loadmat(f'matfiles\\{file}')
print(list(data.keys()))

['_header_', '_version_', '_globals_', 'X097_DE_time', 'X097_FE_time', 'X097RPM']
['_header_', '_version_', '_globals_', 'X105_DE_time', 'X105_FE_time', 'X105_BA_time', 'X105RPM']
['_header_', '_version_', '_globals_', 'X118_DE_time', 'X118_FE_time', 'X118_BA_time', 'X118RPM']
['_header_', '_version_', '_globals_', 'X130_DE_time', 'X130_FE_time', 'X130_BA_time', 'X130RPM']
['_header_', '_version_', '_globals_', 'X169_DE_time', 'X169_FE_time', 'X169_BA_time', 'X169RPM']
['_header_', '_version_', '_globals_', 'X185_DE_time', 'X185_FE_time', 'X185_BA_time', 'X185RPM']
['_header_', '_version_', '_globals_', 'X197_DE_time', 'X197_FE_time', 'X197_BA_time', 'X197RPM']
['_header_', '_version_', '_globals_', 'X209_DE_time', 'X209_FE_time', 'X209_BA_time', 'X209RPM']
['_header_', '_version_', '_globals_', 'X222_DE_time', 'X222_FE_time', 'X222_BA_time', 'X222RPM']
['_header_', '_version_', '_globals_', 'X234_DE_time', 'X234_FE_time', 'X234_BA_time', 'X234RPM']

# 采用驱动端数据
data_columns = ['X097_DE_time', 'X105_DE_time', 'X118_DE_time', 'X130_DE_time', 'X169_DE_time',
                'X185_DE_time', 'X197_DE_time', 'X209_DE_time', 'X222_DE_time', 'X234_DE_time']
columns_name = ['de_normal', 'de_7_inner', 'de_7_ball', 'de_7_outer', 'de_14_inner', 'de_14_ball', 'de_14_outer', 'de_14_ball', 'de_14_outer', 'de_14_ball', 'de_14_outer']
data_12k_10c = pd.DataFrame()
for index in range(10):
    # 读取MAT文件
    data = loadmat(f'matfiles\\{file_names[index]}')
    print(data[data_columns[index]].shape)

(243938, 1)
(121265, 1)
(122571, 1)
(121991, 1)
(121846, 1)
(121846, 1)
(121846, 1)
(122136, 1)
(121991, 1)
(122426, 1)
```

因为每个文件序列长度不一致，为了让每个文件 长度一致，才取这个值，木桶效应，你得顾及 信号长度最短的那个；这个 119808 取值比较灵活，可以变动；

4 模型调参

4.1 CNN 模型（包含 CNN 模块的）

我们参考经典网络模型 VGG，进行我们 CNN 相关模块的设计和实现，主要是卷积核等相关参数的 参考设置，二是利用其卷积池化的特点，没经过一次卷积池化层，通道数翻倍，序列尺寸减半的特征！

VGG优点

- VGGNet的结构非常简洁，整个网络都使用了同样大小的卷积核尺寸（3x3）和最大池化尺寸（2x2）。
- 几个小滤波器（3x3）卷积层的组合比一个大滤波器（5x5或7x7）卷积层好：
- 验证了通过不断加深网络结构可以提升性能。

所以，要调整 和 CNN 相关的参数，主要是层数 和每层的通道数，VGG 必学，不然看不懂卷积池化，也不懂怎么调参！

4.2 LSTM （GRU）模型调参

LSTM 这种循环神经网络的特点就是，不改变输入序列的长度，只改变维度数，这是核心！

所以调参的话，就是改变其 层数和每层的神经元个数

具体怎么进行调参呢，首先 lstm 模型参数设置的时候 [32, 64] 代表有两层，每层分别为 32 个神经元，64 个神经元，这个是最主要的参数，也就是调节层数 和每层的神经元个数，如果过拟合 就要适当 减小层数 和每层的神经元个数；如果欠拟合（效果不好），就要适当增加参数量（增加层数和每层神经元个数）；然后再调节学习率，可以更小的学习率，和更多的 epocho（迭代次数）

4.3 Transformer 相关模型调参

为什么只用编码层？信号序列不适合进行编码，一是没有年月日这种时间分辨率特性！

二是只用编码器层效果反而最好，而且本来起作用的就是编码器层的结构和多头注意力！建议去看看论文 《How Much Attention Do You Need? A Granular Analysis of Neural Machine Translation Architectures》看看是哪些结构起了作用！

还有一个特性：编码层结构的运用 既不改变序列长度，也不改变维度数！

一般多头注意力的头数设置为 2 或者 4，编码器层数可以灵活设置！

这个 transformer 参数设置和优化问题：trasformer 模型中超参数多头注意力机制的头数（N-head）必须能被 input_dim 整除


当出现不能整除的情况，我们遇到奇数特征的输入，就得把序列进行堆叠，比如 [32, 7, 1024] 可以重新改变形状为 [32, 7*8, 128]，这样输入特征就从 7 奇数变成偶数了，这是一种比较基础的做法；我们后续推出的创新模型中关于 transformer 的话，采用比较好的做法是进行输入特征的上采样，即通过 1d 卷积把特征映射到一个偶数的高维特征，然后再送入 transformer，事实证明，这种做法比较合理而且效果也比较好的！


4.4 其它模型调参


模型调参没有捷径可走，如果对对应的网络模型不了解的话，肯定就看不懂参数的调整，比如 CNN，如果连卷积池化都看不懂，那么参数也就理解不了；大家一定要先看对应的论文，建议不要直接看所谓的网上的解说，这些都是二手资料，直接看一手的论文会更加清晰！！

推荐一个入门经典学习教程，b 站吴恩达深度学习 入门课程

[\[双语字幕\] 吴恩达深度学习 deeplearning.ai 哔哩哔哩 bilibili](#)

 215篇神经网络核心论文仓库思维导图.xmind

 DeepLearningBook.pdf


 Python零基础知识手册.pdf


 ResNet.pdf


 RNN.pdf


 Swin Transformer.pdf


 TCN.pdf


 Transformer.pdf

 VGG16.pdf

 动手深度学习.pdf

 利用Python进行数据分析.pdf

 数字图像处理-第4版-冈萨雷斯.pdf

 统计学习方法-李航-第2版.pdf

资料免费下载链接：

<https://mbd.pub/o/bread/mbd-ZpWXk5pq>

如果查阅上述文档后还有问题，请直接添加 微信公众号[建模先锋]，点击联系博主，留言即可！

