卫星数据分析和健康监测

——课程大作业报告

张博仕 2022010153 自 24

一、必做部分——卫星故障检测与诊断

数据的预处理过程与助教所提供 demo 相同，步骤如下：

读入数据

↓

按分系统删除无关列 → 离散值处理（处理名义变量） → 删除方差为零的列

↓

按照自然时间做排序

↓

划分训练集和测试集（2:1），针对时间序列采用均匀降采样

经过上述步骤所得的必做部分的**数据的组织形式**如下，注意所有.csv 文件均同时包含正常数据和故障数据，如替换文件夹做测试需要先 follow 我的预处理过程：

```
data/
    ├── 供配电
        ├── all.csv
        ├── atoi.json
        ├── itoa.json
        ├── test.csv
        ├── train.csv
    ├── 姿轨控
        ├── all.csv
        ├── atoi.json
        ├── itoa.json
        ├── test.csv
        ├── train.csv
    ├──激光载荷
        ├── all.csv
        ├── atoi.json
        ├── itoa.json
        ├── test.csv
        ├── train.csv
```

（一）卫星故障检测

本部分实验针对卫星姿轨控、供配电和激光载荷三大子系统，设计了多种故障检测模型，评估了模型在不同分系统上的虚警率和漏警率表现。通过实验比较，不同模型在不同场景下具有各自的优劣性。

在本部分实验中，我尝试了以下方法：

- 随机森林（Random Forest）
- XGBoost（XGB）
- Svm
- LogisticRegression

1、方法基本原理和表现：

method	激光载荷虚警率	激光载荷漏警率	姿轨控虚警率	姿轨控漏警率	供配电虚警率	供配电漏警率
随机森林二分类	0	0.00296	0.0145	0.05080	0.00433	0.07111
XGB 二分类	0	0.00236	0.0144	0.04998	0.00685	0.05466
Svm 二分类	0	0.00118	0	0.82876	0	1.0000
LogisticRegression	0	0.00118	0.03049	0.10005	0.019473	0.54856

随机森林：

随机森林是一种基于决策树的集成学习方法，它通过构建多个决策树并结合投票机制进行分类或回归。这个方法非常暴力，因为是相当于针对每个特征做投票，但是具有好的可解释性。

XGBoost ：

是基于梯度提升框架的增强版本，通过优化分裂节点、加权损失和增量模型的方式实现高效且准确的分类和回归。它额外引入了正则化机制以防止过拟合。

SVM：

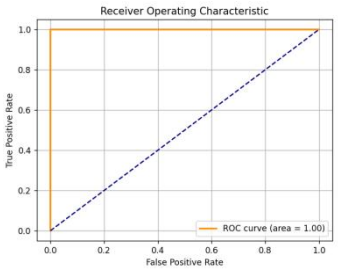
SVM 是一种监督学习方法，通过在高维空间中找到一个超平面将数据分开。SVM 利用核函数（如线性核、RBF 核）将非线性问题转化为线性问题，最大化分类间隔从而提高泛化能力。但 svm 在姿轨控和供配电数据集上的表现**非常灾难**，可能因为姿轨控和供配电数据集的线性可分性质较差，SVM 对此较为敏感，容易受到主导类别的影响。

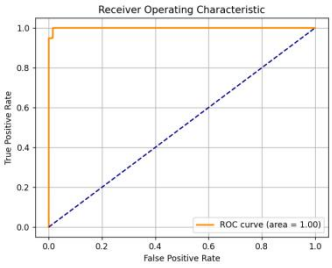
Logistics：

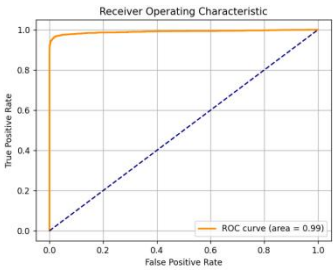
逻辑回归是分类任务中简单、高效的基线方法，适合线性分类问题和小规模数据集。在本任务中其表现效果较差，可能因为姿轨控和供配电数据集的线性度太差导致该模型的效果非常灾难。

2、该数据集最优指标的方案：XGBoost

绘制其 ROC 曲线：







最终实验结果：

分系统	名称	检测次数	报警次数	虚警率/漏警率
姿轨控	正常数据	8330	120	0.014406
	故障数据	2441	2319	0.049980
供配电	正常数据	5546	38	0.006852
	故障数据	2049	1937	0.054661
激光载荷	正常数据	12630	0	0
	故障数据	1692	1688	0.002364

不过虽然基于 xgb 这种学习器的二分类方法在测试结果中展现了较高的准确性，但在实际应用中可能存在一定的局限性。因为实际场景中可能会出现一些全新的故障类型，基于先验知识的二分类方法往往难以应对，无法有效检测这些未知的故障。而基于残差阈值的方法更适合这种情况，因为它不依赖于特定的故障标签，而是基于正常数据的统计特性，通过检测异常行为来发现新的故障类型。因此残差阈值方法可能具有更好的适应性和推广能力。

（二）卫星故障诊断

1、这部分相较于检测部分更困难，因为相当于从二分类器到多分类器这一任务对模型的精细建模能力和数据处理能力提出了更高要求。

在本部分实验中，我尝试了以下方法：

- 贝叶斯决策
- K-近邻法
- Transformer
- 集成学习

2、方法基本原理和表现：

Transformer 就不多介绍了，其实这个实验数据量不太够，我想试试他能不能 work

其他方法都是课上讲过的一些简单的有监督式的方法。

method	激光载荷测试集准确率	姿轨控测试集准确率	供配电测试集准确率
贝叶斯决策	99.912%	83.425%	39.789%

K-近邻法	99.293%	64.561%	75.328%
Transformer	99.763%	88.775%	92.257%
集成学习	99.882%	92.708%	94.729%

分析实验的结果：

激光载荷：所有方法均取得较高的测试集准确率，尤其是 贝叶斯决策 达到了 99.912%，显示了该任务在激光载荷数据上的相对简单性。

姿轨控：不同方法的准确率差异较大，最优方法为 集成学习（92.708%），其次是 Transformer（88.775%），而 K-近邻法（64.561%） 表现较差，可能与数据维度较高，空间距离复杂，产生维度灾难有关。

供配电：贝叶斯决策准确率最低（39.789%），显示其对该数据集的不适应性，而 集成学习（94.729%） 表现最佳。

按照经验，激光载荷数据分布相对简单，大部分方法均能取得接近 99% 的准确率，说明模型对该数据的区分能力较强。其他两个数据集的分布复杂且特征相关性较高，模型需要更强的全局特征建模能力。数据分布极为复杂，类别间的特征差异较小，传统方法（如贝叶斯决策和 KNN）难以有效建模。

3、最优方法--集成学习：

我注意到集成学习在两个任务中都表现出了稳定且良好的 acc 性能。相比单一模型，集成学习能够有效降低过拟合风险，并在综合多个模型预测的基础上实现更高的准确性。

在本实验中，我尝试了以下集成学习策略：

子学习器选择：

AdaBoost：分类性能不够稳定，表现一般。

Gradient Boost 和 XGBoost：效果稳定，泛化能力较强，适应了数据分布的复杂性。

投票器设计：

构建由 SVM（线性核）、Gradient Boost 和 Random Forest 组成的投票器。

软投票：基于子学习器的预测概率平均值。

硬投票：基于子学习器的分类结果进行多数投票。

实践表明，无论软投票还是硬投票，投票器的分类效果均优于单独的子学习器，最终实现了显著的性能提升。

最终取得的实验结果如下：

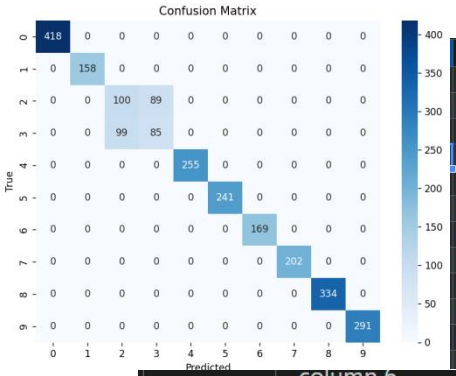
分系统	名称	诊断次数	诊断正确次数	准确率
姿轨控	磁力矩器 X 输出磁矩下降故障	418	418	1.000000
	太阳敏感器 1 故障	158	158	1.000000
	负 Y 翼 A 轴故障	189	115	0.608466
	陀螺 2 输出常零故障	184	80	0.434783
	陀螺 1 输出常值故障	255	255	1.000000
	动量轮 1 输出力矩下降故障	241	241	1.000000
	星敏感器 1 输出噪声变大故障	169	169	1.000000
	动量轮 2 输出力矩下降故障	202	202	1.000000
	动量轮 3 输出力矩下降故障	334	334	1.000000
	动量轮 4 输出力矩下降故障	291	291	1.000000
	Overall	2441	2263	0.927079
供配电	下位机通信异常或死机	229	229	1.000000
	五串电池开路故障	98	94	0.959184
	五串电池性能严重衰降	26	15	0.576923
	五串电池片功率损失故障	147	142	0.965986
	五串电池短路故障	272	272	1.000000

	单串电池片开路	32	24	0.750000
	单串电池片短路	212	204	0.962264
	单体电池开路	11	4	0.363636
	单体电池短路	178	162	0.910112
	第 13 级 S3R 模块只能供电故障	97	90	0.927835
	第 13 级 S3R 模块只能分流故障	67	54	0.805970
	第 17 级 S3R 模块只能供电故障	128	120	0.937500
	第 17 级 S3R 模块只能分流故障	112	102	0.910714
	蓄电池组加热带误断	217	216	0.995392
	蓄电池组加热带误通	223	213	0.955157
	总计	2049	1941	0.947291
激光载荷	俯仰轴解锁异常	87	87	1.000000
	转台俯仰轴精度下降	673	672	0.998514
	转台方位轴精度下降	662	662	1.000000
	章动跟踪快反镜执行体故障	214	214	1.000000
	转台俯仰轴电机故障	2	1	0.500000
	转台俯仰轴驱动器故障	1	1	1.000000
	转台方位轴电机故障	1	1	1.000000
	转台方位轴码盘故障	1	1	1.000000
	转台方位轴驱动器故障	2	2	1.000000
	EDFA 光输出中断	7	7	1.000000
	信号激光器故障	6	6	1.000000
	SWIR 图像探测器有坏点	12	12	1.000000
	精跟踪快反镜 1 执行体故障	24	24	1.000000

	俯仰轴解锁异常	87	87	1.000000
	总计	1692	1690	0.998818

4、数据集本身的一些问题：

姿轨控的负 Y 翼 A 轴故障和陀螺 2 输出常零故障，用**任何 model 都无法良好地区分**。陀螺 2 输出常零故障的产生根据名字，主要由于“光纤陀螺 2 角速率”这一特征的值始终为 0。然而，在 负 Y 翼 A 轴故障 这一类别的异常数据中，也存在部分点“光纤陀螺 2 角速率”持续为 0 的情况。这种特征值的重复与重叠使得模型在判断这些异常时容易混淆，进而导致所有模型在该异常的分类上出现较大的误判问题。



column 2	column 3	column 4	column 5
Description	Total	Correct	Accuracy
磁力矩器X输出磁矩下降故障	418	418	1.000000
太阳敏感器1故障	158	158	1.000000
负Y翼A轴故障	189	100	0.529101
陀螺2输出常零故障	184	85	0.461957
陀螺1输出常值故障	255	255	1.000000
动量轮1输出力矩下降故障	241	241	1.000000
星敏感器1输出噪声变大故障	169	169	1.000000
动量轮2输出力矩下降故障	202	202	1.000000
动量轮3输出力矩下降故障	334	334	1.000000
动量轮4输出力矩下降故障	291	291	1.000000
Overall	2441	2253	0.922982

	column 6	column 7	column 8	column
4904	0.0	0.0	0.0	-0.03974
4905	0.0	0.0	0.03975	-0.03973
4906	0.0	0.0	0.0	-0.03974
4907	0.0	1e-05	0.03974	-0.03974
4908	0.0	0.0	0.0	-0.03975
4909	0.0	1e-05	0.03974	-0.03974
4910	0.0	0.0	0.0	-0.03975
4911	0.0	0.0	0.0	-0.03974
4912	0.0	-1e-05	0.0	-0.03975
4913	0.0	0.0	0.0	-0.03975
4914	0.0	0.0	0.0	-0.03975
4915	0.0	0.0	0.03974	-0.03976

4910、4914 为陀螺 2 输出常零故障、4911~4913 为负 Y 翼 A 轴故障

而在供配电数据集中，类别样本数量分布极为不均衡。少数类数据不足可能导致模型倾向于主导类别，进而降低对少数类的区分能力。

5、针对数据集本身特点的改进：

为了解决姿轨控的两类 fault 无法区分的问题，我曾经尝试单独针对这两类单独做子学习器，但对于指标的提升于事无补，我单独查询了一下 label3 和 label4，发现二者的数据在**几乎所有指标上是一模一样**的，即使有个别 col 有差异也局限在小的范围内，我实在是无能为力（蚌埠住了）。

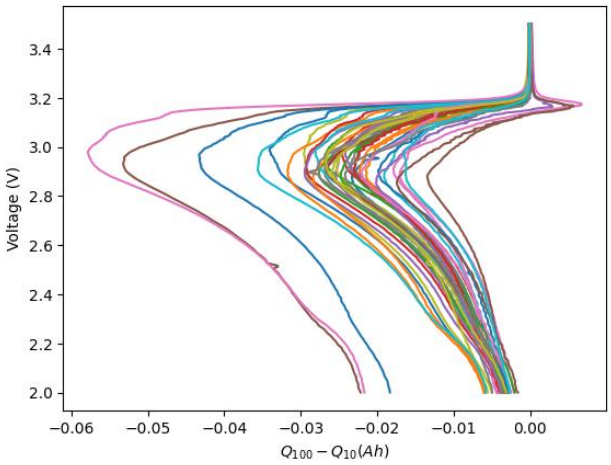
对于供配电数据集，我尝试了构造二阶统计量、构造组合特征，对现有特征进行交叉计算或非线性变换（如特征乘积、对数变换等），以增加模型对细粒度差异的感知能力，但是涨点非常有限，所以我认为即使有些微的涨点，也大概率是对数据分布的一种拟合，因此弃用了这种特征工程。

数据集/方法	整体诊断正确率
姿轨控(改进前)	92.189%
姿轨控(自学习器)	92.708%

二、选做部分——电池剩余寿命的故障检测

1、非神经网络方法——学习器与回归模型

在本部分实验中，我尝试通过非神经网络方法对电池的性能进行回归预测。使用了助教提供的 Demo 中论文构造好的统计量，并基于统计量与电池电压的相关性进行了特征选择，最终用于回归模型的训练和预测。然而，由于数据分布上的差异，这些方法在验证集和测试集上的表现存在较大偏差。



论文发现：统计量 Q100-Q10 与电池电压有较强的相关性

- $\Delta Q_{100-10}(V)$ 对数方差 [DeltaQ_var]
- $\Delta Q_{100-10}(V)$ 对数最小值 [DeltaQ_min]
- 容量衰减曲线2-100循环的线性拟合斜率 [CapFadeCycle2Slope]
- 容量衰减曲线2-100循环的线性拟合截距 [CapFadeCycle2Intercept]
- 第2个循环的放电容量 [Qd2]
- 前5个循环的平均充电时间 [AvgChargeTime]
- 2-100循环的最小内阻 [MinIR]
- 第2个和第100个循环的内阻差 [IRDiff2And100]

选择统计量如上做回归预测

基于原有的 Demo 我总共尝试了 4 种方法：

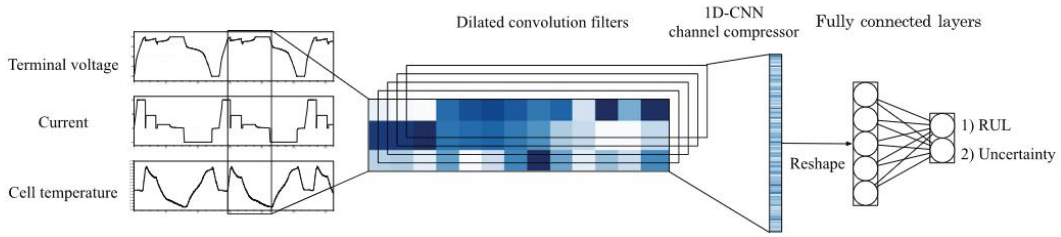
Method	测试集/MSE	测试集/MPE	验证集/MSE	验证集/MPE
随机森林	30490.1183	0.1400	20899.5331	0.1255
XGB	31673.6817	0.1793	32058.8043	0.1608
Lasso	28808.4410	0.1903	24483.7815	0.1370
Ridge	29493.9930	0.1884	23288.2790	0.1355

随机森林和 XGB 不再赘述，Lasso 是一种带 L1 正则化的线性回归方法，通过引入惩罚项，将一些不重要的特征权重压缩为 0，从而实现特征选择。Ridge 是一种带 L2 正则化的线性回归方法，能够通过对特征权重的平方惩罚项来防止过拟合。

原本按照助教的 demo 跑出来结果非常不均匀，因为我 shuffle 了一下；在该分布下，随机森林在测试集和验证集的指标上相对有优越性。

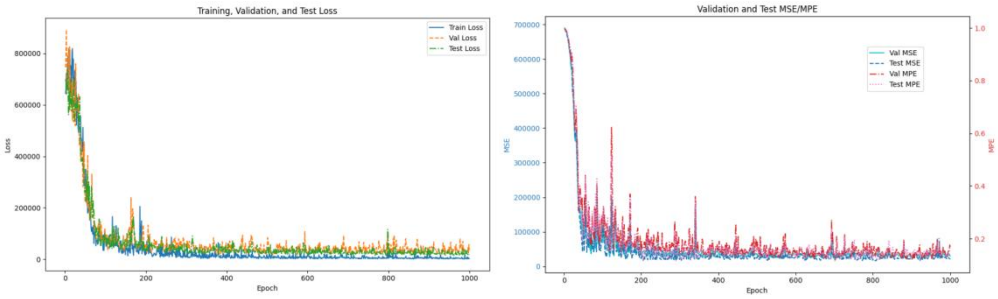
2、E2E 神经网络方法--1DCNN

在江老师和助教的提醒下，我参考课件论文里的方法，我构建了一个一维的卷积神经网络来提取每一个 lifecycle 里的时序特征：

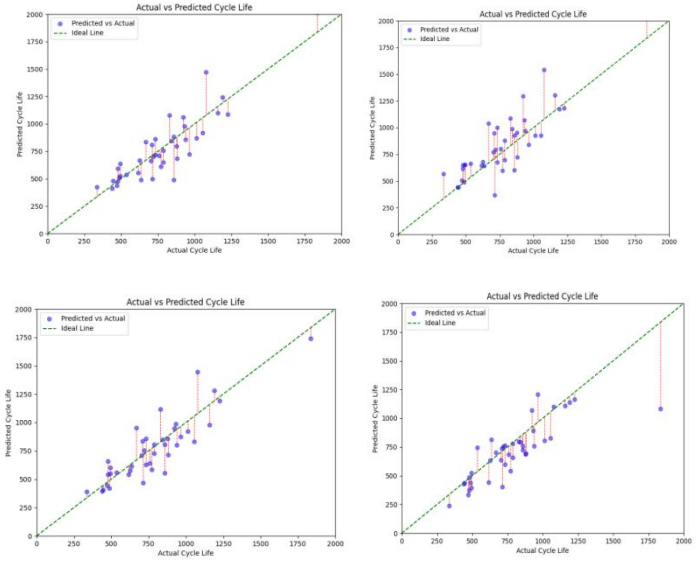


```python
class BatteryRULCNN(nn.Module):
    def __init__(self, input_channels, num_channels, kernel_size=3, dilations=[1, 3, 9, 27]):
        super(BatteryRULCNN, self).__init__()
        self.layers = nn.ModuleList()

        # 多层卷积，每一层有不同的膨胀率
        for dilation in dilations:
            self.layers.append(
                Conv1DBlock(
                    in_channels=input_channels if dilation == 1 else num_channels,
                    out_channels=num_channels,
                    kernel_size=kernel_size,
                    dilation=dilation
                )
            )
```



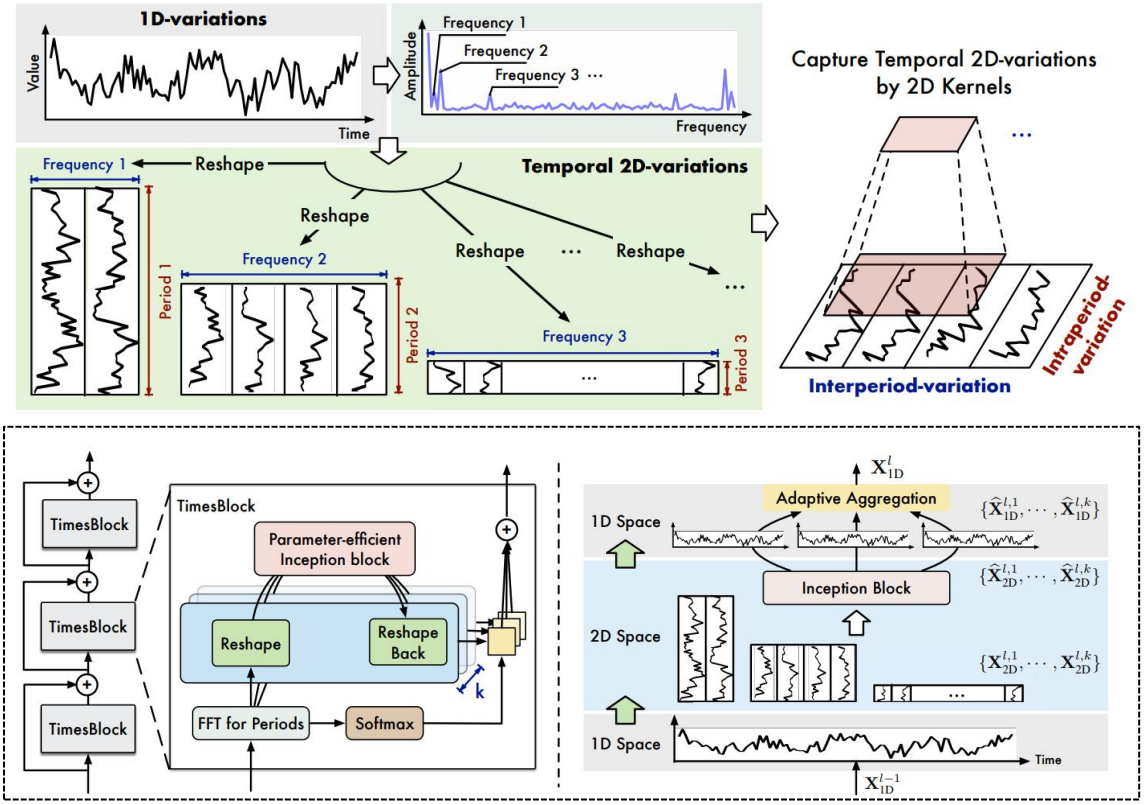
1DCNN 的训练与预测过程随时函数与 val-acc 变化曲线图



使用 1DCNN 的实验结果的可视化

1、使用频域 DCT 采样--受到 TMN 启发的构想：

我校软件学院龙老师组的工作 TMN 给了我一些 insight：根据原 cycle 里的数据在频域里的信息，更好捕捉到电池剩余寿命的有关信息，而且我们认为网络依然是端到端的。



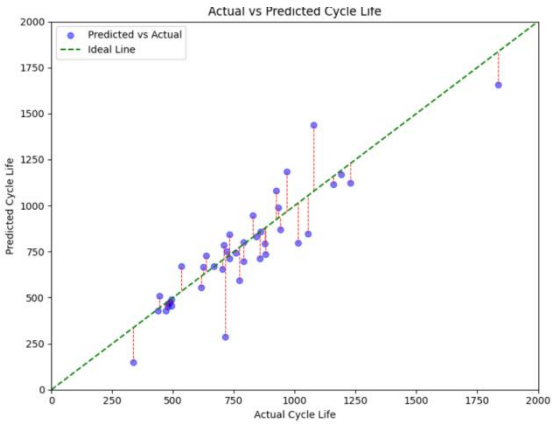
原论文以多周期视角，使用 FFT 来捕捉其周期性的频域特征并提取二维时序变化表征做训练。

由于原论文的开源代码数据和我们的电池数据结构完全不一样，我尝试修改我们的数据流和他的 loader 来对齐他提供的接口，但由于原文数据形式周期尺度的多样性工作量实在巨大，我选择自己搭建相关的网络来测试能否提升在这个任务上的性能。

但是遇到了采样点一多就爆显存的困难，于是我想到了 DCT，因为 DCT 利用信号对称性，将信号能量压缩到前几个低频分量，**减少了不必要的高频成分和负频率部分**。而且 DCT 比 FFT 更简单，适合大规模数据处理，占用显存的数量**只有 FFT 的一半，不涉及复数运算**，因此计算过程更简单，也减少了临时变量的显存消耗。

```python
def dct(x, norm='ortho'):
    """
    实现一维 DCT (Discrete Cosine Transform)
    :param x: 输入数据, [B, H, W]
    :param norm: 正交归一化
    :return: DCT 结果
    """
    N = x.shape[-1]
    v = torch.arange(N, device=x.device).float()
    coeff = torch.cos(torch.pi * (v + 0.5).view(1, -1) * v.view(-1, 1) / N)
    if norm == 'ortho':
        coeff[0] *= 1 / torch.sqrt(torch.tensor(2.0, device=x.device))
        coeff *= torch.sqrt(torch.tensor(2.0 / N, device=x.device))
    return torch.matmul(x, coeff)
```

选择 DCT 替代 FFT 做频域信息提取



MyTMN 取得的预测效果图

经过反复地实验，我认为这个网络相较于我先前的方法具有优越性，因为：

1、**最稳定的效果：**在 20 轮随机抽样分布下，平均 MSE 小于 13000、平均 MPE 小于 0.120。这种稳定性证明模型可以减少因输入数据分布波动而带来的误判率。

2、**出色的显存性能：**因为 DCT 占用显存仅为 FFT 的一半，且无需处理复数运算，显著减少了临时变量带来的显存消耗，适合大规模数据处理。

3、使用的采样点少（每 Cycle 只需要 10 个点）：这种稀疏采样方式极大降低了数据采集与计算成本。

最终实验结果汇总如下：

Method	测试集/MSE	测试集/MPE
随机森林	20899.5	0.1255
XGB	32058.8	0.1608
Lasso	24483.8	0.1370
Ridge	23288.3	0.1355
e2e	18975.9	0.1224
tmn	12945.3	0.1066

三、总结与感想

在本次卫星数据分析和健康监测大作业中，我系统性地完成了故障检测和故障诊断的必做任务，同时使用多种方式完成了对于我具有一定挑战性的电池寿命预测附加任务。

通过多次迭代改进与实验，我在任务上取得了较高的准确率和稳健性结果。回顾整个任务，我经历了从理论理解到算法实现，再到结果分析和优化的完整过程，尤其在最后一周投入了大量精力。为确保模型性能达到预期，我不仅反复完善数据处理流程，还不断剖析实验

结果中存在的问题并尝试改进方法。

面对论文开源代码与我们的数据结构完全不一致的问题，我选择自己搭建网络。在多次试验失败后，我通过引入离散余弦变换（DCT），显著降低了显存消耗，从而使模型可以在低采样密度下运行并取得更优结果。这种在实验中不断创新、尝试的过程不仅提升了我的问题解决能力，也让我更加意识到理论与实践结合的重要性。

尽管整个过程充满了挑战，但我深感受益良多，尤其最让我感到兴奋地是我在选做中成功验证了自己的猜想 work。

在完成任务的过程中，我深深感受到课程团队的支持与帮助。从课程设计、数据准备到课堂讲授和课后答疑，杨帆老师、江奔奔老师、陈鹏宇助教始终给予了我极大的耐心与指导。这让我更加认识到优秀的课程设计对于这种实践类型课程学习过程的重要性，也深深感谢他们在课程中的辛勤付出。

参考文献

[1] Wu, Haixu, et al. "Timesnet: Temporal 2d-variation modeling for general time series analysis." arXiv preprint arXiv:2210.02186 (2022).

[2] Joonki Hong, Dongheon Lee, Eui-Rim Jeong, Yung Yi,Towards the swift prediction of the remaining useful life of lithium-ion batteries with end-to-end deep learning,Applied Energy,Volume 278,2020,115646,ISSN 0306-2619,