

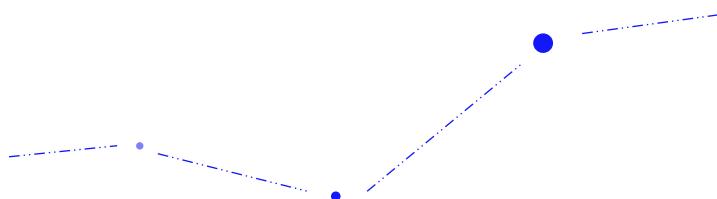


2024年(第17届)中国大学生计算机设计大赛
人工智能实践赛



基于深度学习算法的 非侵入式电力负载分类与预测系统

Non-invasive Power Load Classification and Prediction System Based On Deep Learning Algorithm



齐鲁工业大学(山东省科学院)
计算机科学与技术学部
软件工程(软件开发)21-1&2班
杜宇、姜川、李晓语、李庆隆、张一雯

指导教师：贾瑞祥老师、陈静老师



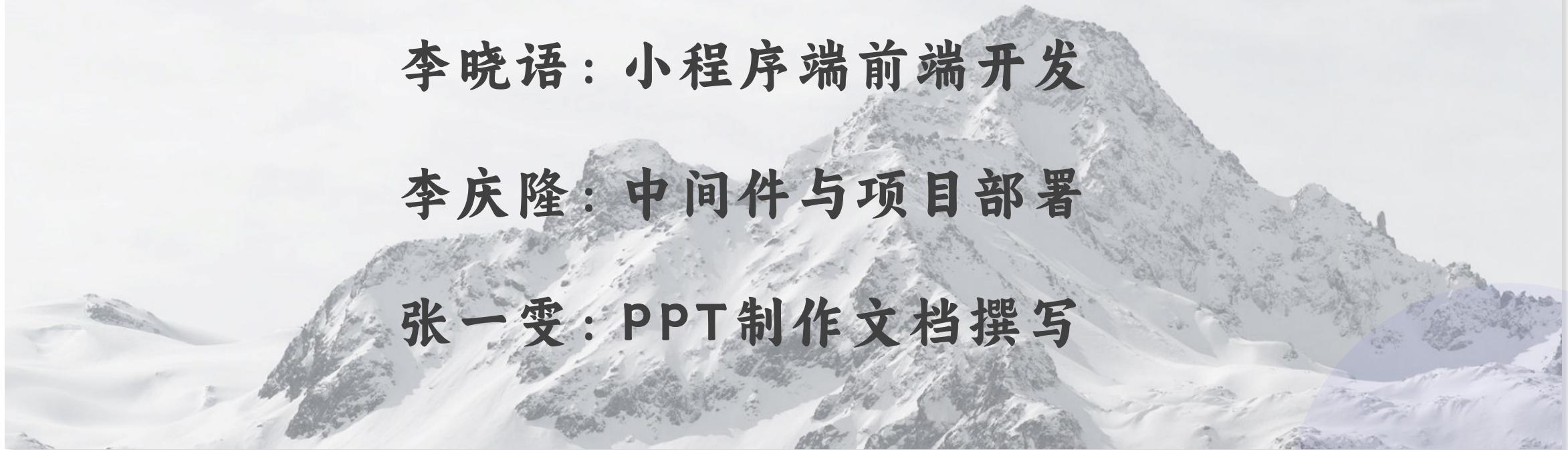
杜宇：算法设计与架构设计

姜川：WebServer后端开发

李晓语：小程序端前端开发

李庆隆：中间件与项目部署

张一雯：PPT制作文档撰写



01

设计动机

PART ONE

02

算法设计

PART TWO

03

项目部署

PART THREE

04

项目总结

PART FOUR



第一部分” 设计动机

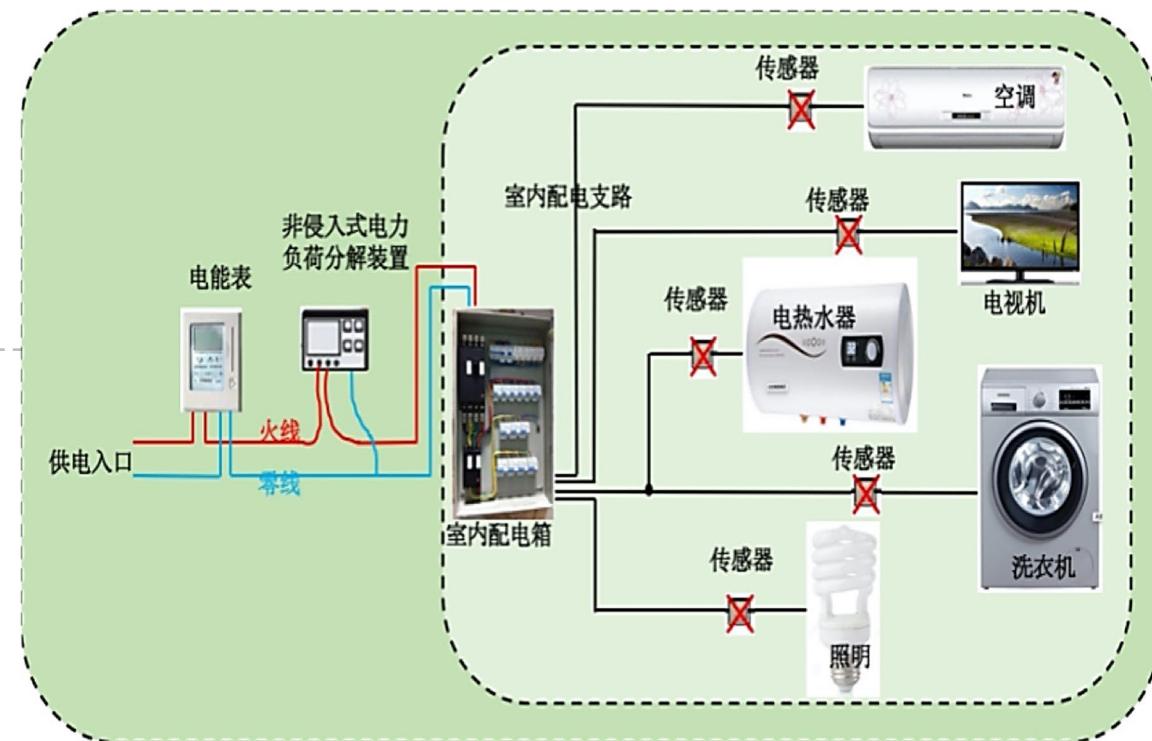
Motivation Of NILM Algorithm

何为非侵入式电力负载分类与预测方法？

What is NILM Algorithm?

非侵入式电力负载分类与预测方法是一种基于统计数学的算法、人工智能算法与电学理论的方法。在不监测具体电器设备的情况下，通过分析整个电力系统的总用电数据来推断分解各个电器设备用电情况的技术。这种方法不依赖于在每个电器设备上安装传感器或监测装置，而是通过分析电网供给的电信号，来识别和分类不同的电器设备及其工作状态。

非侵入检测装置可应用在住户进户线总开关处，对于工业负荷的总线，可以在它上面安装一个传感器，通过采集和分析电力用户端电压和用电总电流来辨识总负荷内部每个或者每类用电设备的用电功率和工作状态，以达到知晓每个或者每类用电设备的用电情况，此过程被称作**非侵入式电力负荷检测** (Non-Intrusive Load Monitoring，简记作**NILM**)。





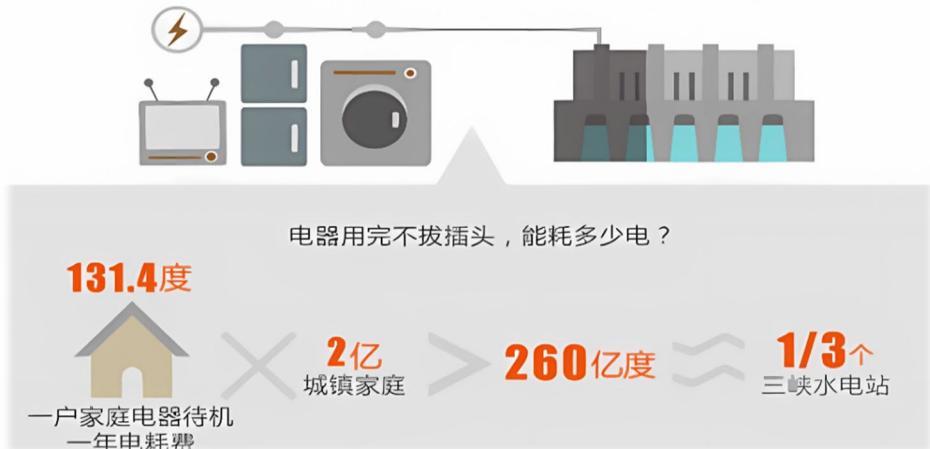
01. 节能减排

我国节能产品认证中心的调查发现，一个普通城市家庭平均每天的待机能耗为0.36千瓦时，相当于开着一只30瓦的长明灯。仅以彩电一项待机耗电计算，如果每天待机2小时耗电0.02度，全国4亿台彩电一年的待机耗电量就高达29.2亿度，相当于大亚湾核全年1/3的发电量。

02. 技术需求

随着智能电网的迅速发展，针对智能电网的规划、运行与控制的方法逐渐深入，并取得极大的研究成果，其中大多数方法的实现离不开数据的支撑。高级量测体系是智能电网的关键技术，而面对海量数据需求，侵入式检测方法已经很难满足需求，因此，经济高效的非侵入检测方法获得越来越高的关注，电力市场迫切需要研究非侵入式方法，解决传统方法监测设备投入过多、管理复杂或部分数据无法检测识别的问题。

全国年待机能耗相当于近**1/3**个三峡水电站



03.低成本，高可靠

工程安装实施比较容易，减少了硬件部署安装的费用；所需传感器数量较少，大大降低了硬件的经济成本；按照系统组成和结构越简单则可靠性相对越高的原则，非侵入式系统可靠性较高。

04.容易推广

不用入户安装，大大提高了用户的接受程度，因而便于推广使用；并且由于只需检测入口总数据，因此可以覆盖到的监测的范围更广，一台机器可以监测较大规模的用电。





- **能效监测与管理:** 通过准确识别和监测家庭或商业建筑中的各个电器设备的能耗，用户可以更好地理解和管理他们的电力使用，从而采取措施减少不必要的能耗，提高能效。
- **需求响应:** 电力公司可以利用这些数据来实施需求响应计划，鼓励用户在电力高峰期减少用电，以减轻电网压力，优化电力资源的分配和利用。
- **故障检测与预防:** 通过分析电器设备的能耗模式，可以及时发现异常情况，如设备性能下降或故障，从而进行预防性维护，避免设备损坏或电力系统故障。
- **用户行为分析:** 通过对用户用电行为的长期监测和分析，可以更好地理解决用户的用电习惯，为电力公司提供用户画像，帮助他们设计更符合用户需求的电力服务和定价策略。
- **电网规划与优化:** 电力公司可以使用这些数据来优化电网规划，确保电力供应的稳定性和可靠性。
- **节能减排:** 通过提高能效和优化电力使用，非侵入式负载监测有助于减少能源消耗和温室气体排放，支持可持续发展目标。



响应政府号召

1月22日上午，山东省第十四届人民代表大会第二次会议在山东会堂隆重开幕，省长周乃翔代表省政府向大会作政府工作报告。报告指出，2024年，山东将抓实抓好数字经济高质量发展，以数字变革新赛道引领形成经济发展新动能。加力夯实数字经济底座。

让算力像电力一样走进千家万户、赋能千行百业，这一20年前提出的设想如今正成为现实。我们的电力负载分类与预测系统便是这一事实的一个重要的事例与体现。



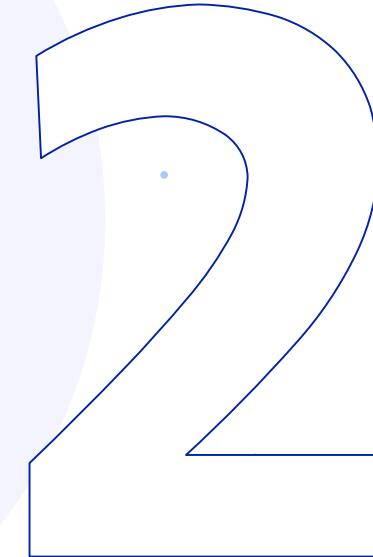


无论是普通家庭还是企业工厂，都亟需一套智能算法和系统来监控电器的实时运行状态，识别不必要的电力消耗。这不仅有助于节约能源，还能降低用电成本。系统还应具备预测短期电能消耗的能力，使企业能够更有效地规划电力使用，特别是在实行阶梯电价机制的情况下。

为此，我们研发了NILM电力负载分类与预测系统。其中，“分类”功能可实时告知用户各电器设备的运作状态；“预测”功能则能预估用户未来短时间内的电能消耗，助力节能减排。

第二部分” 算法设计

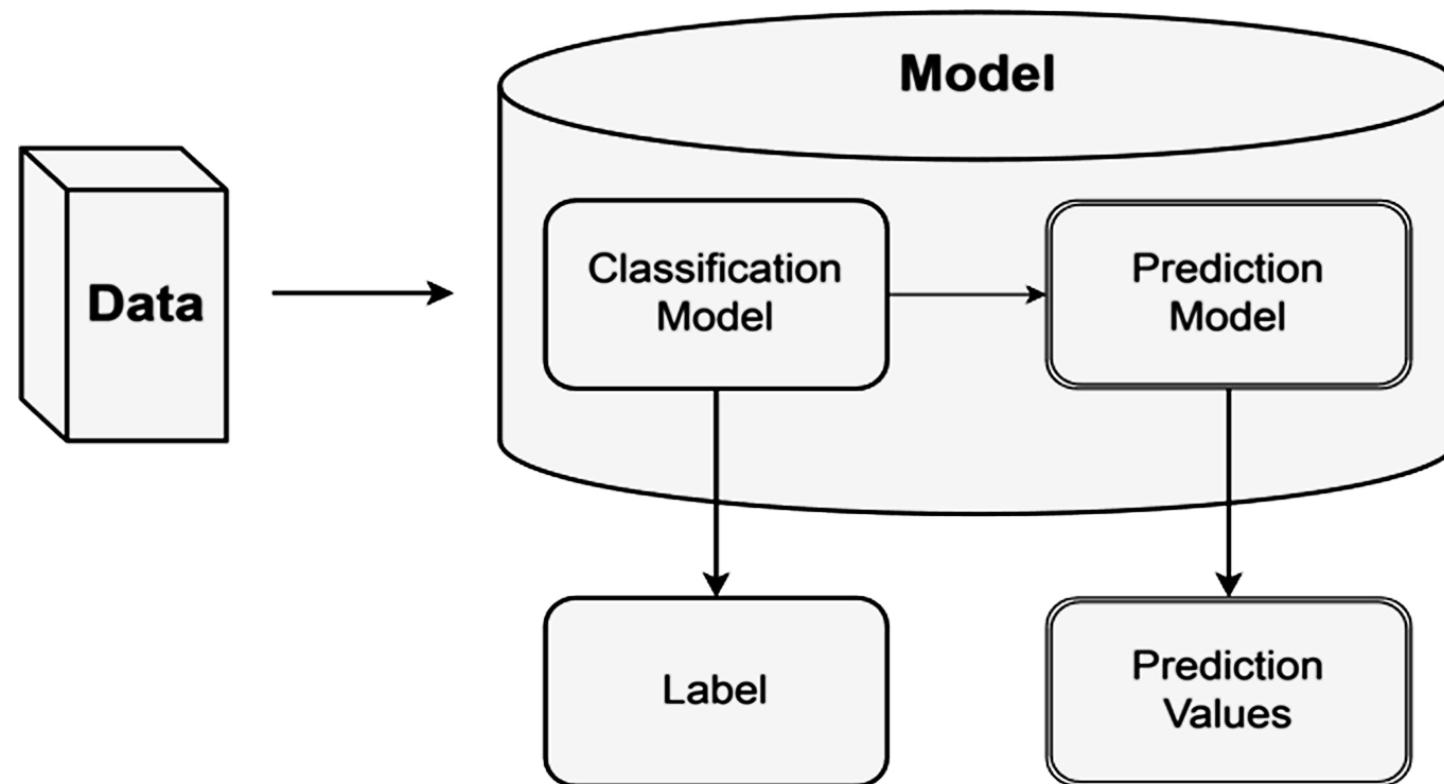
Design of Classification and
Prediction Algorithm



电力负载分类分解与预测模型工作流

Workflow of Classification, Decompose and Prediction Models

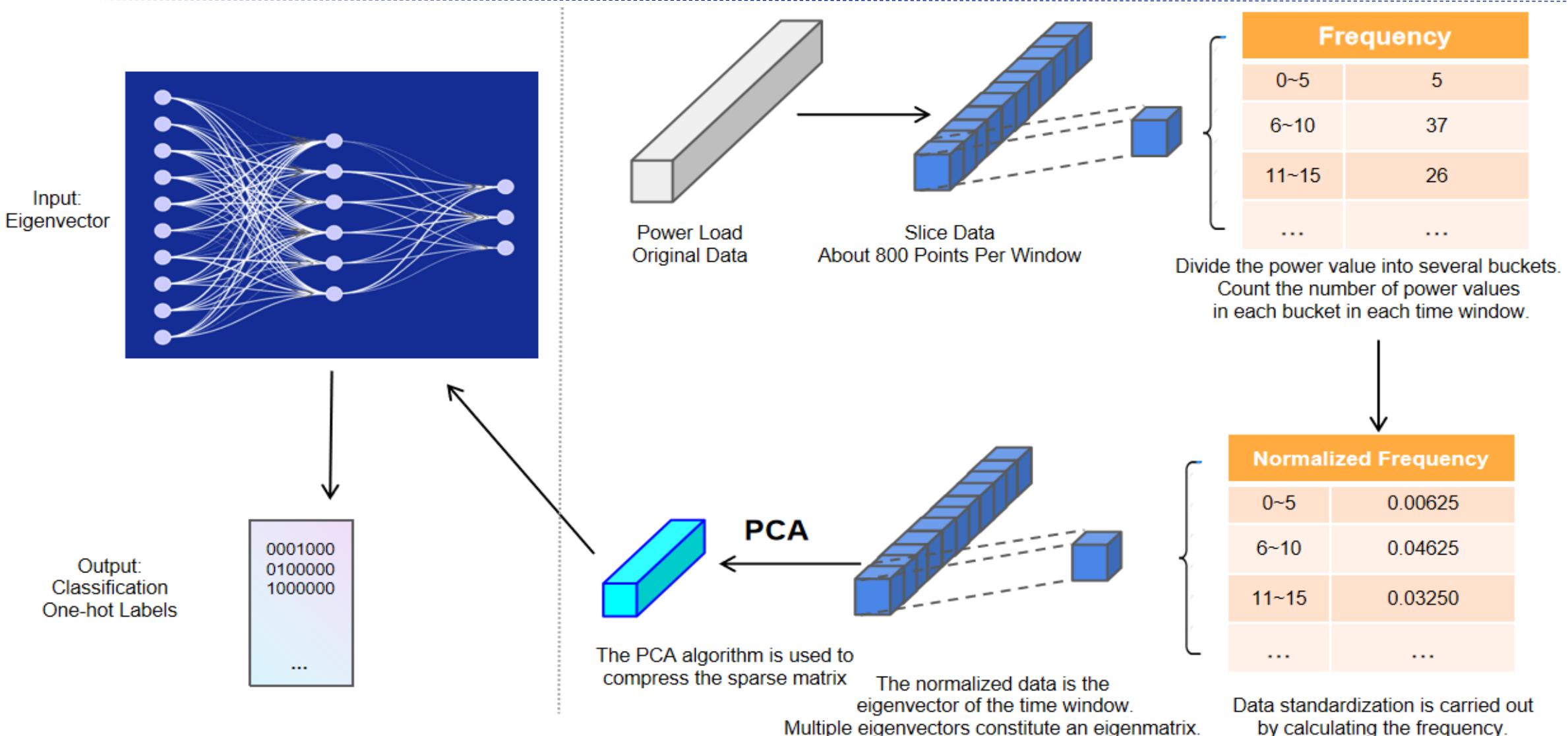
模型分为两部分 - 分类分解模型与数值预测模型。首先数据会被前者进行分类，然后数据以及通过分类得到的标签会再被送入数值预测模型中，输出预测的功率数值。





电力负载分类分解算法设计

Classification and Decompose NILM Algorithm

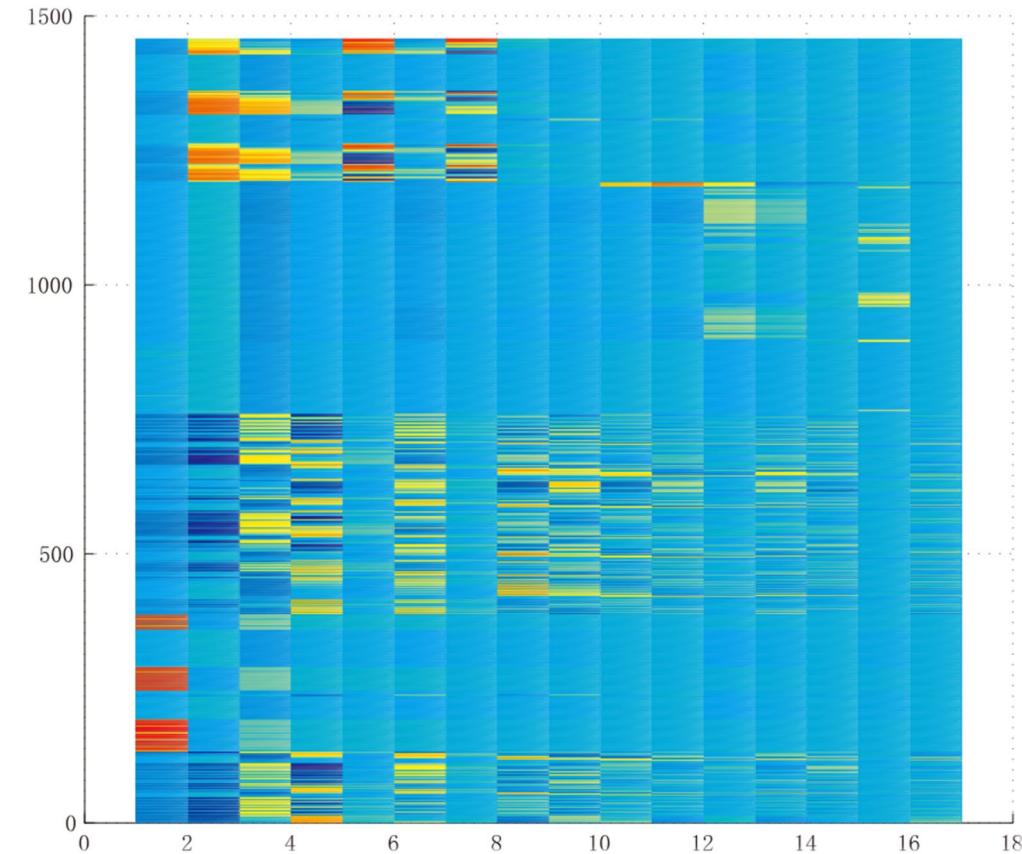




电力负载分类分解算法设计

Classification and Decompose NILM Algorithm

特征工程方面，首先对**数据分帧**，就是把每个标签对应的非常长的时序数组切分成长度相等的若干小段，每帧的长度(设为 $L=800$)作为参数；**进一步**
我们对样本值的大小再作分组，比如：功率大小
(以下均以瓦特为单位)从0开始，到3300(设为
 $M=3300$ ，作为参数)是最大值，中间每隔5个单位
(设为 $I=5$ ，作为参数)分一组，则功率大小可以被
分为 $0 \sim 5, 6 \sim 10, \dots, 3295 \sim 3300$ ，共
 $[M/I]=660$ 组。**然后分别统计每一帧中的样本值分**
别落在这些组里的个数，这些数可以构成一个
 $[M/I]$ 维的向量V。最后做归一化处理，将V除以帧
长L，可得该帧的特征向量 $V/L=F$ 。处理后的特征
矩阵是一个非常稀疏的矩阵，不利于模型收敛，故
使用PCA(主成分分析)算法“压缩”特征。

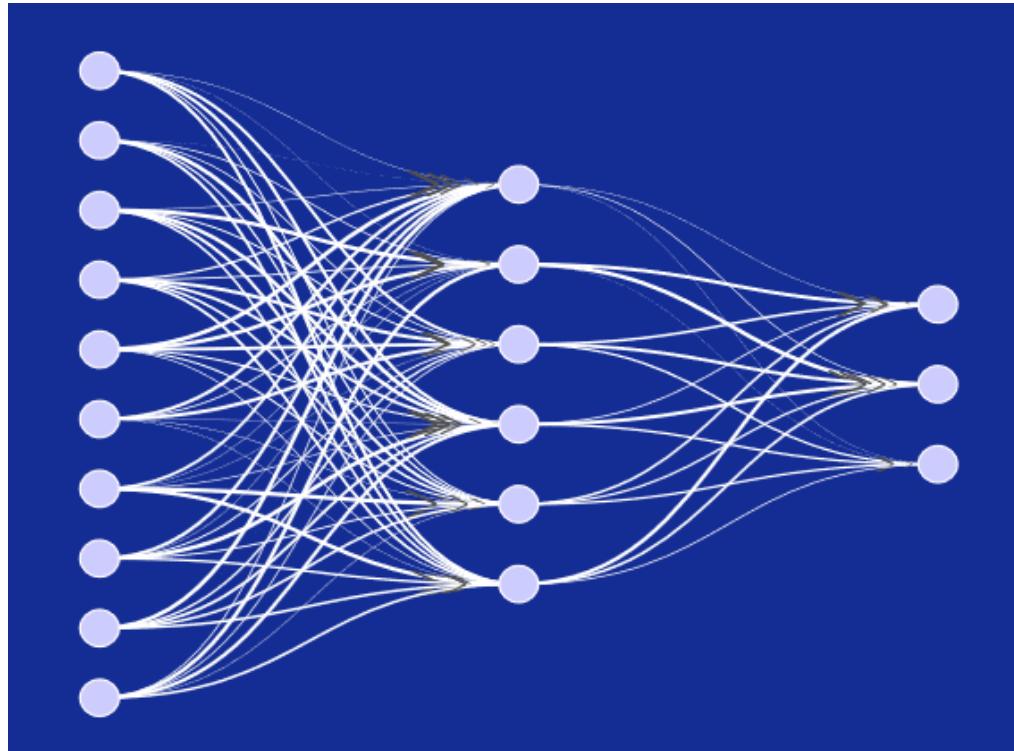


经过PCA算法压缩后的特征矩阵热图(方差保留96%)



电力负载分类分解算法设计

Classification and Decompose NILM Algorithm



Input Layer $\in \mathbb{R}^{28}$ Hidden Layer $\in \mathbb{R}^{14}$ Output Layer $\in \mathbb{R}^7$

$$FNN(x) = \text{ReLU}(\text{ReLU}(xW_1 + b_1)W_2 + b_2)W_3 + b_3$$

GRU

FNN

我们使用3层全连接神经网络模型(28, 14, 7, 参数量5K)进行数据分类任务；使用3层全连接神经网络模型(84, 42, 21, 参数量20K)进行功率标签分解任务。

经过实验发现，使用全连接神经网络与相同网络结构的门控循环单元(GRU)效果相差无几，但是推理性能却为前者的数倍。另一方面，在我们假定每个时间窗之间是互相时序不相关的，这样模型可以专注的处理每一个时间窗中的特征而做出判断，并将标签数据传给预测模型的嵌入层进一步处理，预测模型再利用时序特性进行预测，各司其职。综上两种原因而选择全连接网络。

```
Windows PowerShell

(base) PS C:\User
Pred:
91.046%
(base) PS C:\User
Pred:
90.852%
(base) PS C:\User
```

电力负载数据预测算法设计

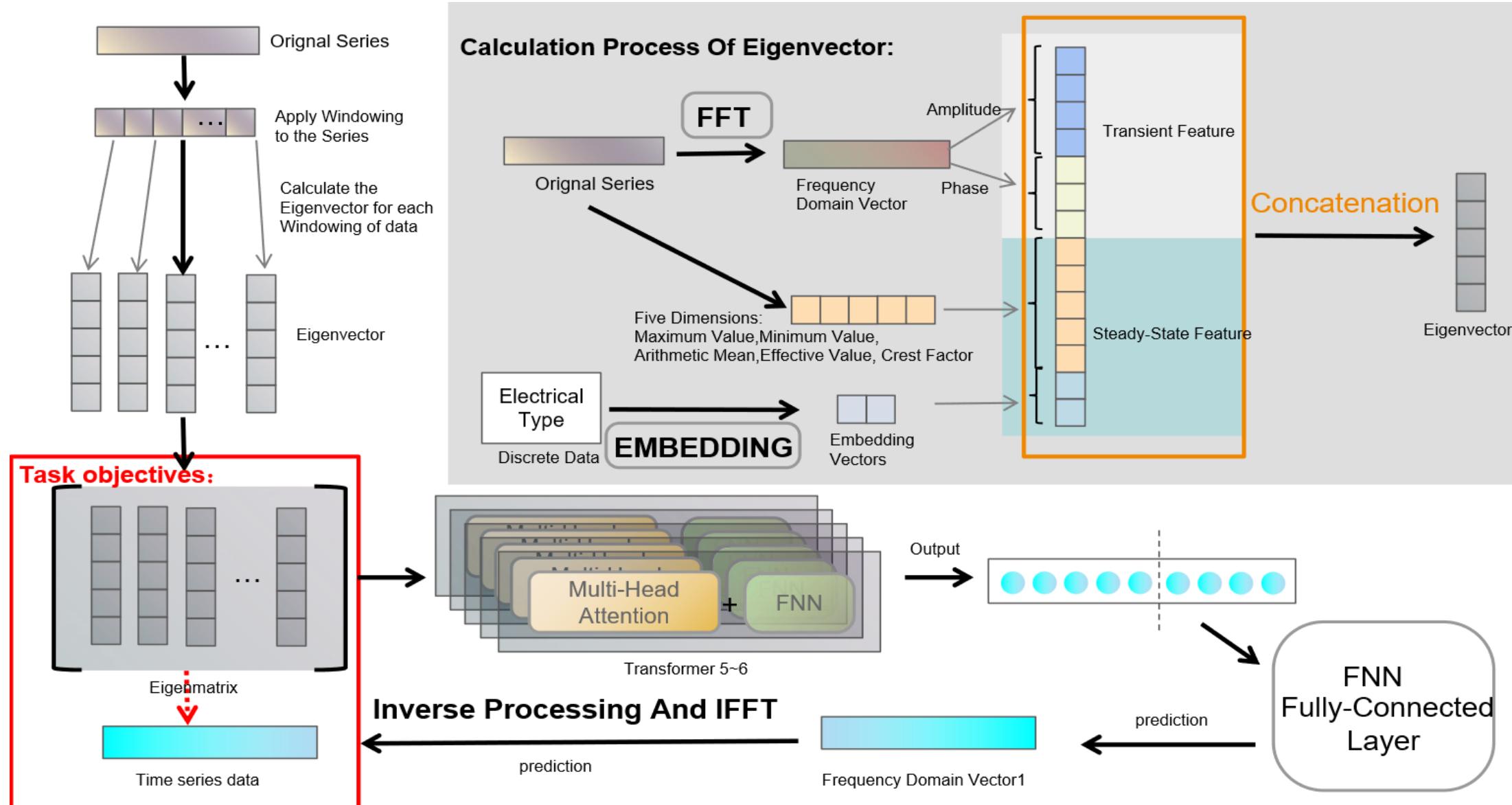
Power Load Data Prediction NILM Algorithm

上图展示的是某家庭电力负荷总数据，时间粒度为1分钟，时间跨度为8700分钟。

下图展示的是同一家庭电力负荷总数据。时间粒度为1分钟，时间跨度为500分钟。

我们可以很直观的感受，电力负载时序数据具有的特点是，当用电器种类与状态不变时，**宏观上非常具有规律性，极易预测；而当时间跨度缩小时，序列在微观上就没有了规律可言，基于短时间检测值的功率预测将变得非常困难。**究其原因，不外乎短时功耗受外部不确定性因素影响太大，我们能做的，就是从万变中找不变，利用人工智能与数学的算法找出数据中的“主要矛盾”。







电力负载数据预测算法设计

Power Load Data Prediction NILM Algorithm

首先进行FFT计算，将时间窗内的时间序列信号数据转化为频域表示。转化后的信号长度与原始时序信号长度相等。其中第一个维度通常表示信号的直流分量(DC)，由于其不包含信息，故我们将其去除。该计算过程可以表示为：

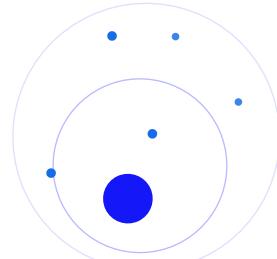
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} kn}$$

其中N表示信号长度，k表示频率序数，x[n]为原始时序信号。

事实上，上述表达式为离散傅里叶变换(DFT)的计算过程。在数学上，该表达式可以进一步优化，以便于计算机程序的执行。

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j \cdot 2\pi f t} dt$$

上述过程为快速傅里叶变换(FFT)。计算该函数时，首先进行信号长度的填充，使其长度为2的幂。然后使用分治策略和蝶形运算充分优化，此处不过多讨论。





电力负载数据预测算法设计

Power Load Data Prediction NILM Algorithm

经过计算得到的频域向量不能直接用作暂态特征向量，原因是未去除不包含有效信息的DC分量，并且通过FFT计算出的频域向量存在对称性的共轭冗余。**我们只需要得到除去了DC的一半向量即可获得该时间窗的所有信息。**由于直流分量等于窗内数据的算术平均值，故以上操作等效于：

$$X_{RDC} = X - \frac{1}{N_X} \sum X$$

$$X_{T\text{-feature}} = \text{Concat}(\text{Real}(X_{RDC}), \text{Imag}(X_{RDC}))$$

特征向量即为暂态特征与稳态特征的合并：

$$X_{\text{feature}} = \text{Concat}(X_{T\text{-feature}}, X_{S\text{-feature}})$$

稳态特征是指我们**假定若电器的种类不变与运行工况没有出现大的变化时**，该时间窗具有的特征，我们设计了窗内数据最大值、最小值、算术平均值、有效能量值以及波峰系数共5个固定维度以及可以根据数据规模更改的n维电器类型嵌入维度，共n+5个维度组成。以下是波峰系数计算方法和电器种类的映射示意：(其他从略)

$$\text{Peak_Factor} = \frac{\max(X_i)}{\max(0, \sqrt{\frac{1}{ws} \sum X_i^2})}$$

$$Type_Vector = \text{Embed}_{Z \rightarrow R^n} (\text{Type}_i)$$

电力负载数据预测算法设计

Power Load Data Prediction NILM Algorithm

短时电能功率数据受当前时刻及其之前相近时间段的功率负荷变化的影响较大，而宏观上的数据变化趋势对微观时刻数据变化的影响微乎其微，因而我们决定使用自然语言处理领域中的**注意力机制的算法，捕捉近距离若干时间窗的数据中，哪些数据对接下来的“影响”比较大，“受影响”的权重是多少，进而预测新的数据。**就好比自然语言处理中的生成式语言模型一样，通过提示词的词嵌入矩阵来预测下一个token的可能的词嵌入向量，从而生成下一个token。可以表达为：

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

“token”的特征向量构成的矩阵K也可用其本身表示，再设ws为频域向量长度（即为数据窗长度），稳态特征向量长度为ss，故其注意力权重向量可以表示为：

$$\text{Weight}(X) = \text{softmax} \left(\frac{XX^T}{\sqrt{ws + ss}} \right)$$

其中激活函数softmax的表达式为：

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

电力负载数据预测算法设计

Power Load Data Prediction NILM Algorithm

算法预测下一时间窗的频率值的“依据”仍然为其自身，因此权重矩阵应与其作点乘，得到表示各频率分量以及其相位在各个时间窗内重要程度的加权后的向量。我们将这些向量在各个维度上的加权值求平均，得到预测的值。其中 f_s 为数据帧的长度：

$$Y = Pred(X) = \frac{1}{f_s} \sum \text{Weight}(X)X$$

以上内容是一个自注意力层的预测输出，我们希望使用多个注意力层，结合它们的判断，做一个综合的输出。其中 Z 为多头注意力的头数。

$$Z = \text{Mutihead}(Y) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_Z)W^o$$

$$\text{head}_i = \text{Attention}(XW_i, XW_i, XW_i)$$

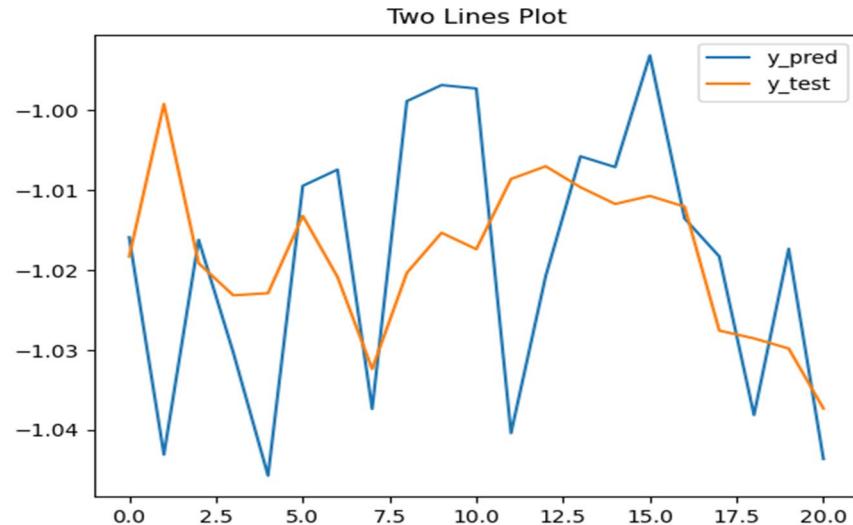
最终的输出值 Z 即为注意力层根据时序数据预测的值。



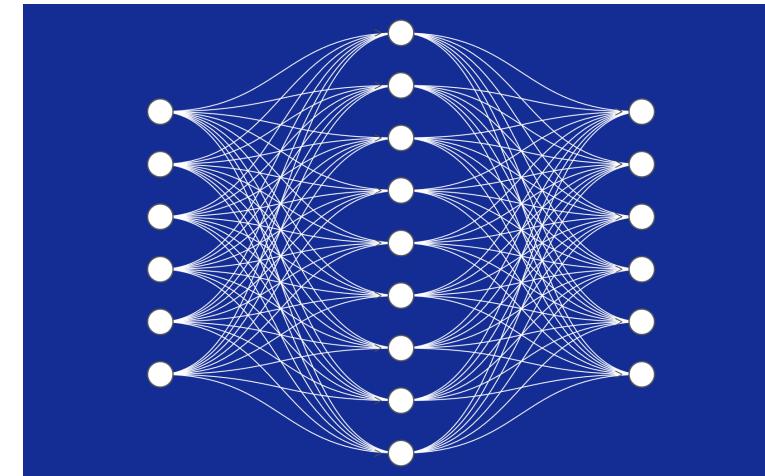
电力负载数据预测算法设计

Power Load Data Prediction NILM Algorithm

但是经过训练试验发现，模型的损失值居高不下，并且它生成的频域向量补充DC后经过IFFT得到的时域波形的形状与测试集中的真实波形相差较大。(暂不考虑DC带来的波形偏移)



由Multihead多头注意力输出的频域向量直接转换为时间序列的效果图，很显然这个效果并不是很好的，误差很大。

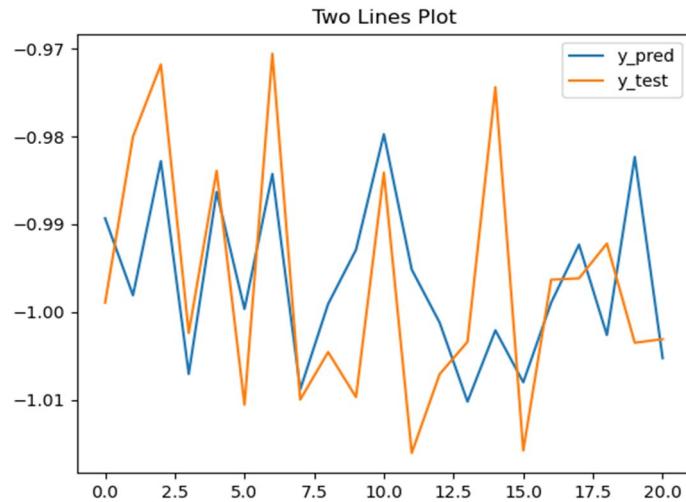
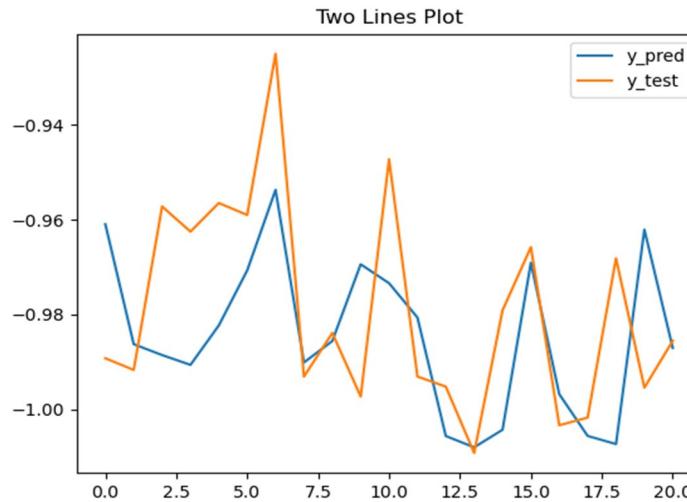
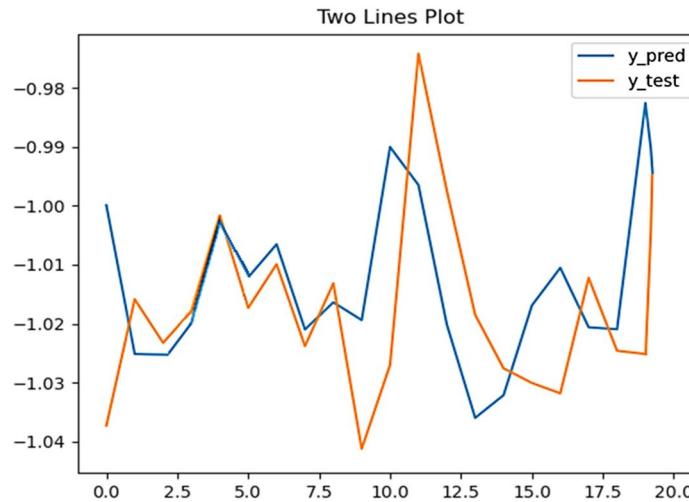
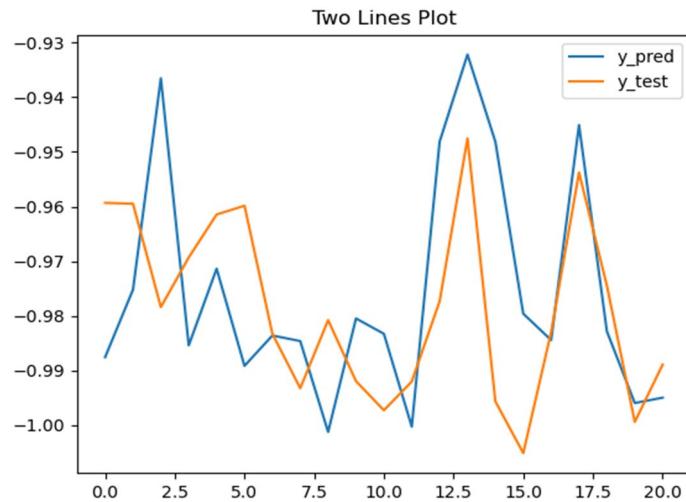
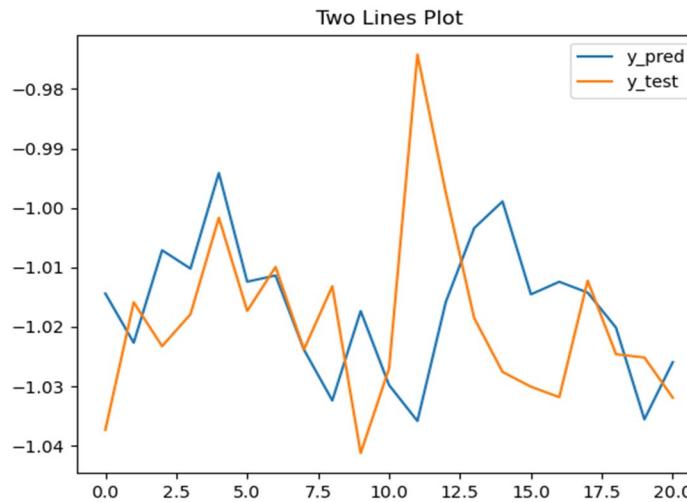
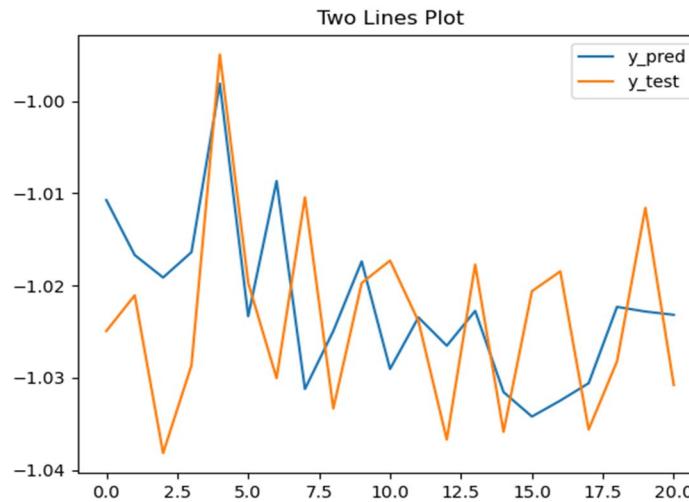


$$FNN(x) = \text{ReLU}(\text{ReLU}(xW_1 + b_1)W_2 + b_2)W_3 + b_3$$

我们考虑出现这个问题的原因在于，注意力机制只会根据每个时间步的特征的各个维度来计算出加权向量，**它不会考虑一个时间步内部特征与特征各维度之间的关系**，而全连接神经网络却可以很好的解决这个问题。因而我们的解决方案是在多头注意力层后面**再添加一个全连接层**，它的输入层与输出层均为每个时间窗的特征向量长度。

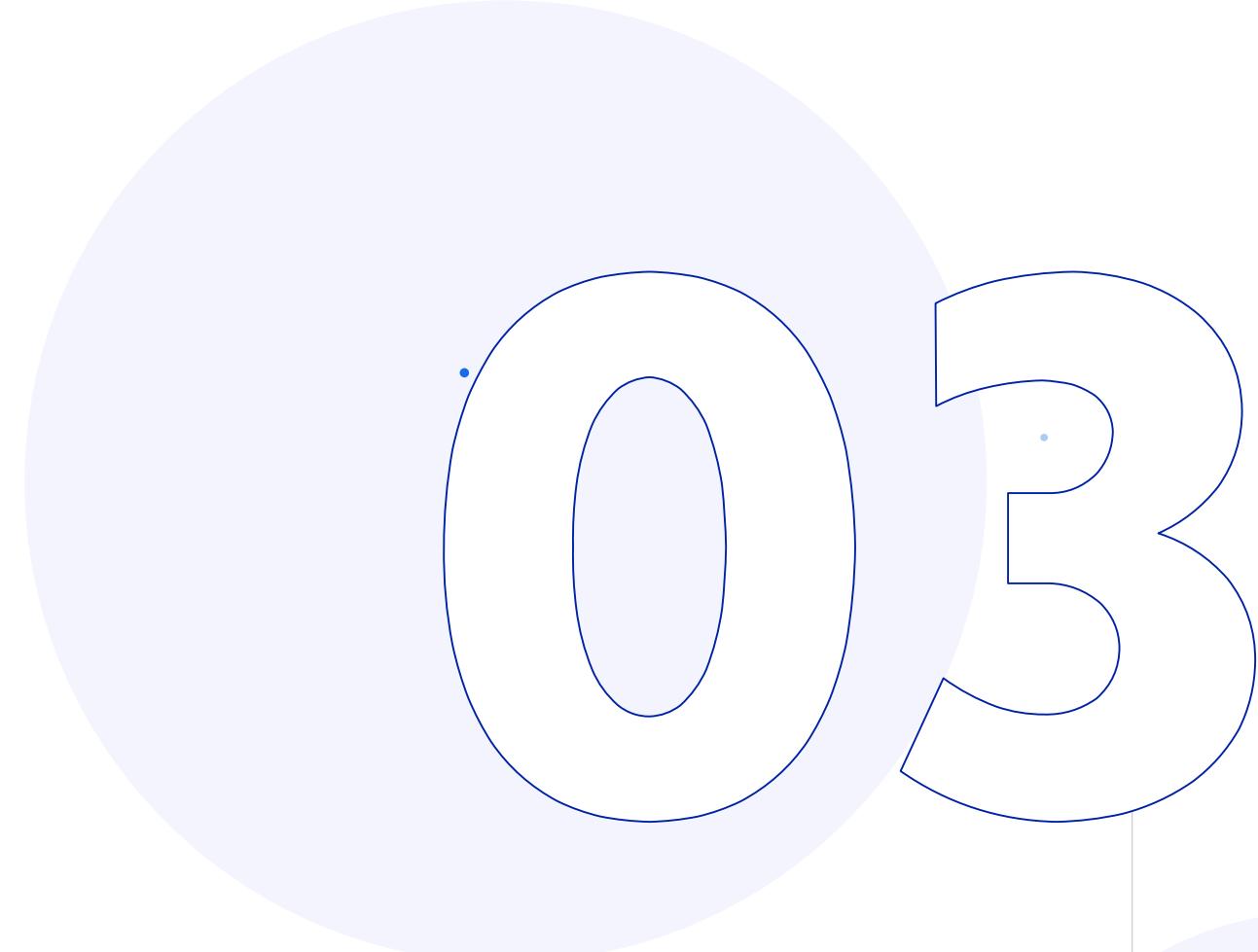
经过修改后的模型在测试集上的预测值与真实值对照

The Predicted Value of the Modified Model on the Test Set is Compared With the True Value



第三部分” 项目部署

System Deployment



项目的部署采用了适于现代化超算服务平台的“**应用分布，算力集中**”云端协同架构。

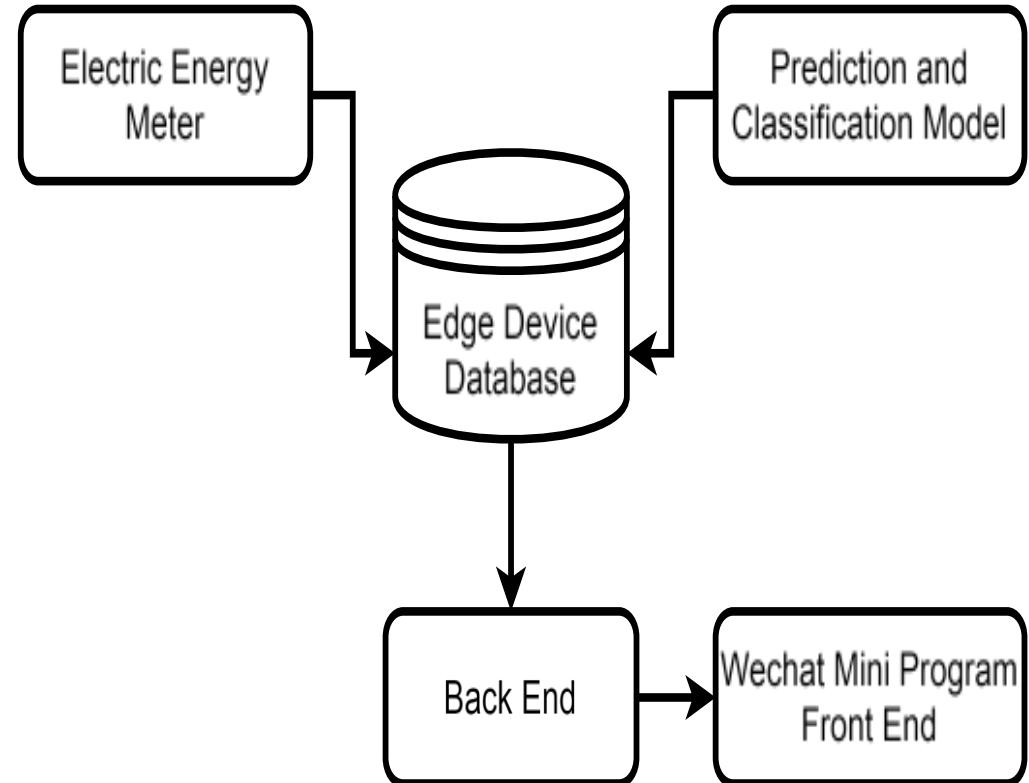
其中电能表属于边缘设备，用于采集各用电单位的耗电数据，将采集结果写入数据库。数据库采用**SQLite轻量级数据库**。

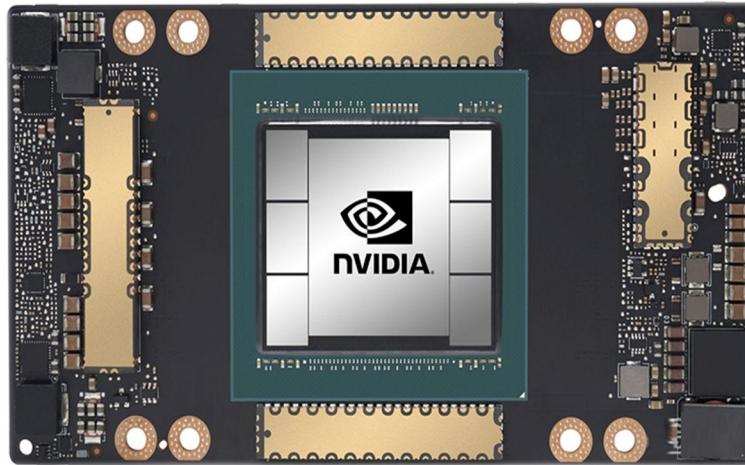
MQ的设计则削减了来自分布式后端的请求洪峰，减轻了GPU集群的压力，提升系统整体性能。系统后端部署于小规模用电单位构成的集合中，比如：一个居民小区、一家公司等，后台为集合内的每个用电单位用户的微信小程序提供分类与预测数据转发服务。

分类分解与预测AI模型部署于依托国家超算互联网强大算力的服务器集群。模型每隔一定的时间“主动”读取数据库中由边缘设备采集到的数据，并进行推理计算。完成后将计算结果写入对应数据库。**后台处理客户端请求与向模型请求数据的两个过程是不相关的，两者互不影响，互不牵制。**其响应客户端时只需读取数据库中的各项结果，不与AI模型端直接关联，降低了耦合度，且易于拓展。

系统宏观架构

Macro Architecture of System





```
NVIDIA-SMI 470.182.03  Driver Version: 470.182.03  CUDA Version: 11.4
+-----+
| GPU Name Persistence-M| Bus-Id Disp.A Volatile Uncorr. ECC
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage GPU-Util Compute M.
|                            |          MIG M.
+-----+
| 0 NVIDIA A100-SXM... Off 00000000:00:09.0 Off 0MiB / 40536MiB 22% Default
| N/A 27C P0 47W / 400W          |          Disabled
+-----+
Processes:
+-----+
| GPU GI CI PID Type Process name GPU Memory
| ID ID          Usage
+-----+
No running processes found
```



```
Log: /tmp/tensorboard_1594823856
- Batch: 1
- Input layers: Not specified, inherited from the model
- Output layers: Not specified, inherited from the model
- Input shapes: Not specified, inherited from the model
- Mean values: Not specified
- Scale values: Not specified
- Scale factor: Not specified
- Precision of IR: FP32
- Enable fusing: True
- Enable grouped convolutions fusing: True
- Move mean values to preprocess section: False
- Reverse input channels: True
TensorFlow specific parameters:
- Input model in text protobuf format: False
- Path to model dump for TensorBoard: None
- List of shared libraries with TensorFlow custom layers implementation: None
- Update the configuration file with input/output node names: None
- Use configuration file used to generate the model with Object Detection API: None
- Use the config file: None
Model Optimizer version:
SUCCESS ] Generated IR version 10 model.
SUCCESS ] XML file: D:\Competition\Robot\Raspberry_Pi\Package\2IR.\frozen_darknet_yolov4_model.xml
SUCCESS ] BIN file: D:\Competition\Robot\Raspberry_Pi\Package\2IR.\frozen_darknet_yolov4_model.bin
SUCCESS ] Total execution time: 13.03 seconds.
It's been a while, check for a new version of Intel(R) Distribution of OpenVINO(TM) toolkit here https://software.intel.com/en-us/openvino-toolkit/choose-download?cid=&source=upgrade&content=2020.3.LTS or on the GitHub*
```

NVIDIA A100 Tensor Core GPU

Intel Neural Compute Stick 2



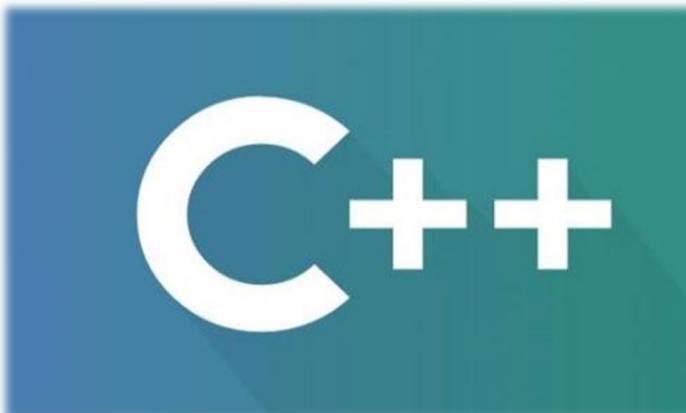
基于深度学习算法的
非入侵式电力负载分类与预测系统

系统部署生产环境

Production and Deployment Environment of System



微信小程序



label_data	
列 4	
label_id integer	
user_id integer	
time_unixstamp integer	
label_value TEXT	
prediction_data	
列 4	
pred_id integer	
user_id integer	
time_unixstamp integer	
pred_value integer	
realtime_data	
列 4	
data_id integer	
user_id integer	
time_unixstamp integer	
powerload_value integer	
sqlite_master	
列 5	
user	
列 5	
user_id integer	
user_password TEXT	

01. 用户登录

请求方法: POST
 请求 URL: /api/login
 请求参数:
 {
 phone_number: xxx; //用户的电话号码,
 password: xxx //用户输入的密码【注意密码传递的是 MD5 值】
 }
 响应参数:
 {
 code: xxx; //取值 0、1，分别表示成功、失败。
 token: xxx //用户令牌
 }

02. 用户注册

请求方法: POST
 请求 URL: /api/register
 请求参数:
 {
 phone_number: xxx; //注册的电话号码
 password: xxx; //用户设置的密码【MD5 值】
 room: xxx //用户的房间号
 }
 响应参数:
 {
 code: xxx; //取值 0、1，分别表示成功、失败。
 description: xxx //失败描述(比如“电话号码已注册”等情况)
 }

03. 读取负载数据

请求方法: POST
 请求 URL: /api/getdata
 请求参数:
 {
 phone_num: xxx; //电话号码
 token: xxx //令牌
 }
 响应参数:
 {
 code: xxx, //取值 0、1，分别表示成功、失败。【无 token 或与电话不匹配，是失败】
 description: xxx //失败描述(比如“电话号码已注册”等情况)
 real_time: [x, x, x, x……] //实时监测序列【返回该用户的实时序列最后 50 个值；若不足 50 个时间步，则前面补 0，返回足够 50 个数据】
 prediction: [x, x, x, x……] //预测数据【返回该用户的预测序列最后 20 个值，若不足 20 个时间步，则后面补 0，返回足够 20 个数据】
 label: xxx //用户正在使用的电器的标签【返回该用户标签序列的最后一个值，若标签序列为空，返回“正在计算”】
 }

description: xxx //失败描述(比如“电话号码已注册”等情况)
 real_time: [x, x, x, x……] //实时监测序列【返回该用户的实时序列最后 50 个值；若不足 50 个时间步，则前面补 0，返回足够 50 个数据】
 prediction: [x, x, x, x……] //预测数据【返回该用户的预测序列最后 20 个值，若不足 20 个时间步，则后面补 0，返回足够 20 个数据】
 label: xxx //用户正在使用的电器的标签【返回该用户标签序列的最后一个值，若标签序列为空，返回“正在计算”】
}

我们采用轻量级数据库SQLite。数据库是整个系统的“枢纽”，连接边缘设备、后端以及位于算力中心的AI模型。故数据库的运行效率直接决定了整个系统的运行效率。

数据库包含存储实时数据、预测数据、预测标签以及用户控制等表结构。其中实时数据由边缘设备写入，算力网络和后端读取。预测数据和预测标签由算力网络写入，后端读取。用户控制表则完全由后端读写。

2:40 100% WeChat WiFi

用户登录



请输入手机号码

请输入密码

登录

注册

2:47 100% WeChat WiFi

电器数据展示



用户: 15254102907

退出登录

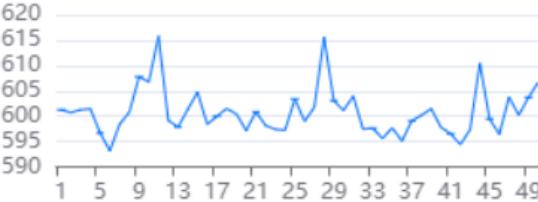
2:45 100% WeChat WiFi

Weixin

2024年04月21日 02:45:30

实时功率 预测功率

实时功率曲线 单位: 瓦(W)



620
615
610
605
600
595
590

1 5 9 13 17 21 25 29 33 37 41 45 49

您正在使用的电器: 热水壶

最近50分钟消耗电能: 1802 千焦

2:46 100% WeChat WiFi

Weixin

2024年04月21日 02:46:21

实时功率 预测功率

预测功率曲线 单位: 瓦(W)



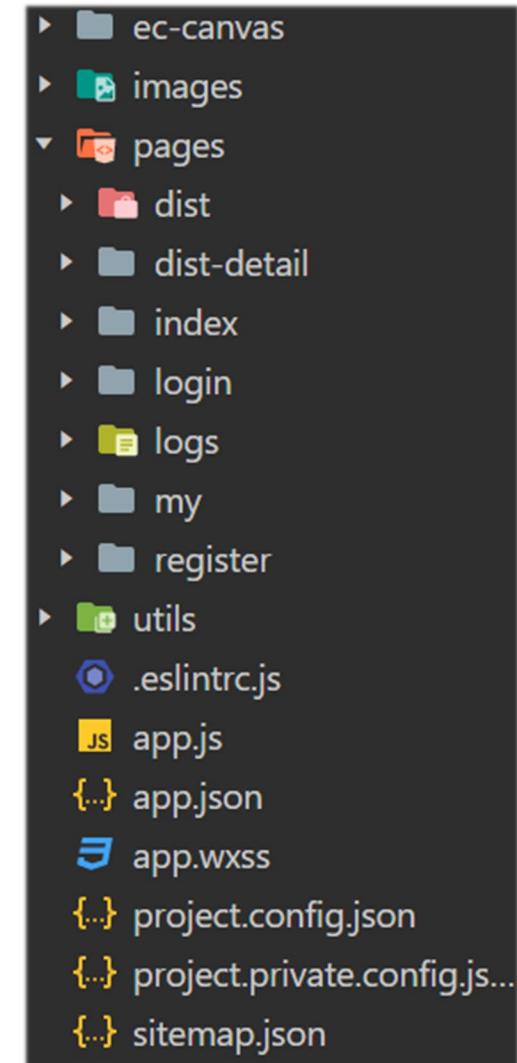
615
610
605
600
595
590

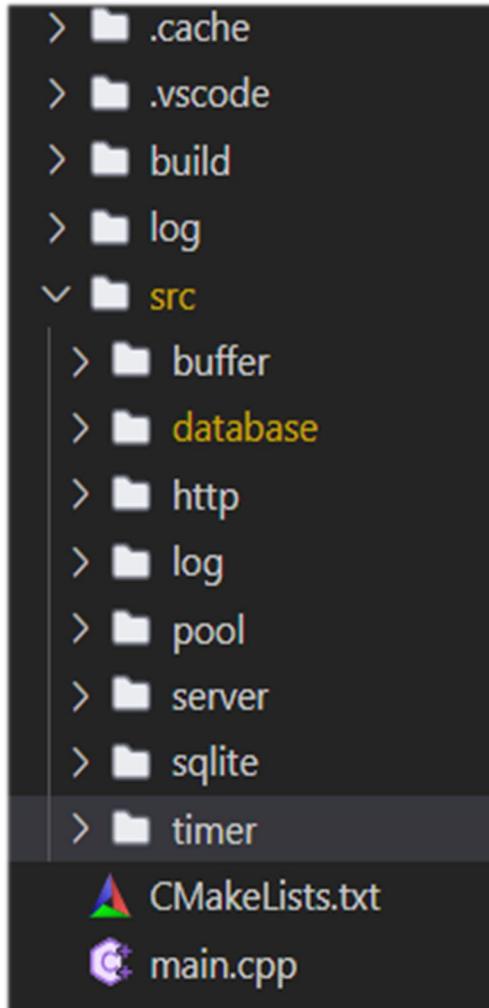
1 3 5 7 9 11 13 15 17 19



前端实现：前端以微信小程序的形式实现，使用微信官方平台的微信开发者工具进行开发。

图表展示采用了ECharts的折线图组件，每不到1分钟轮询请求一次图表数据（即给定长度数组）分别是实际和预测两种数据，用轮播图实现切换展示。用户信息添加到全局变量，在所有页面可共享，注销时清空。





总体框架采用的是单Reactor多线程模型，在主线程里面通过I/O多路复用监听多个文件描述符上的事件。主线程负责连接的建立和断开、把读写和逻辑处理函数加入线程池的任务队列，由线程池的子线程完成相应的读写操作，实现任务的高并发处理。在最底层实现了自动增长的Buffer缓冲区。对于HTTP请求报文，采用分散读进行读取，使用有限状态机和正则表达式进行解析；并通过集中写和内存映射的方式对响应报文进行传输。最后还加入了日志模块帮助工程项目开发和实现服务器的日常运行情况的记录。

首先是服务器的一个参数初始化操作。通过构造WebServer这个对象传递参数进行服务器相关参数的设定，主要参数有设置定时器超时时间、设置Epoll触发模式、设置日志系统的开启、日志路径以及异步开关、设置线程池的线程数量。然后是通过设定的参数对服务器的各个模块进行初始化。主要有日志、线程池、IO复用、HTTP对象、缓冲区、阻塞队列等模块。

日志模块的初始化是先建立一个日志文件，通过一个变量按照天数去划分不同的日志文件。线程池采用RAII手法，在构造时创建线程，析构时唤醒所有睡眠的线程使其退出循环。IO复用是对epoll函数调用的一个封装。

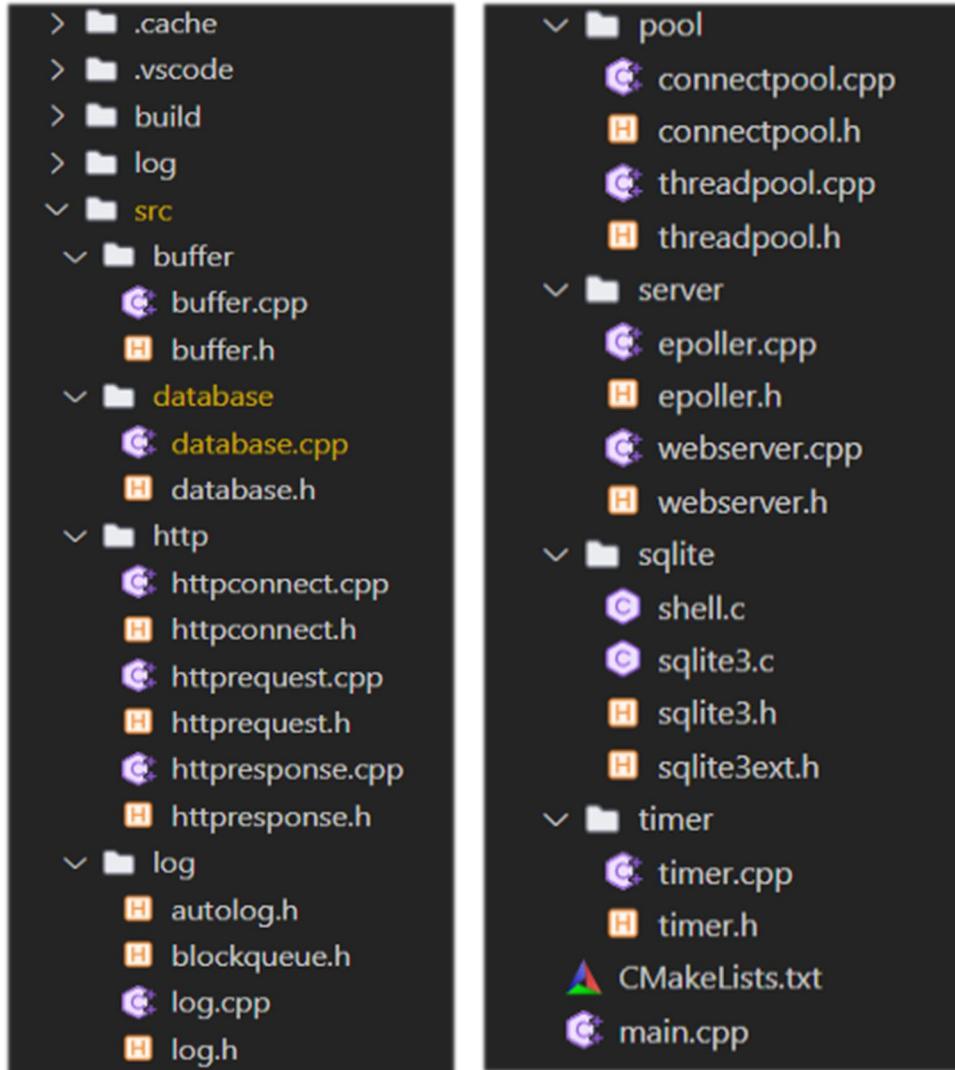
基于C++17的WebServer后端

Back End Based on C++ 17 Web Server

HTTP对象主要设置文件存放的相关路径。缓冲区和阻塞队列主要完成指定大小的参数设定。服务器各个模块初始化完成之后就是主线程里I/O复用监听事件的循环。监听事件有新连接到达、读事件和写事件和异常事件。根据不同的事件进行一个任务处理。

当新连接到达的时候，通过调用accept取出新连接（ET模式下需要循环操作），将新连接的文件描述符用来初始化HTTP连接（套接字地址和文件描述符绑定到一个HTTP对象），完成绑定定时器的初始化，同时添加监听读事件，设置其文件描述符为非阻塞。

当有异常事件发生的时候，关闭该连接同时移除监听事件和定时器。当触发读事件的时候，调整定时器的定时时间，将读任务放入线程池的任务队列当中去。这个时候线程池对象里的多个线程，处于一个睡眠或者竞争任务并执行的过程，任务加入到任务队列当中去时会发送一个唤醒信号，如果有睡眠的线程则会被唤醒，进入循环里探测任务队列是否为空，取出任务并执行或者队列为空继续睡眠。线程执行读任务函数主要是完成一个非阻塞读取调用直到读完，将数据缓存在用户缓冲区中，接着执行一个消息解析的操作，根据HTTP解析是否成功的判断来决定重新注册写事件还是读事件。如果解析失败那么重新注册读事件等待下次读取更多数据直到一个完整的HTTP请求。如果是解析成功的话就制作响应报文并且注册写事件，等待内核缓冲区可写触发事件时，将其写入内核缓冲区。



基于C++17的WebServer后端

Back End Based on C++ 17 Web Server

同时，基于**心跳机制**，每过一段时间会检测时间堆，去除掉长时间非活跃的连接。

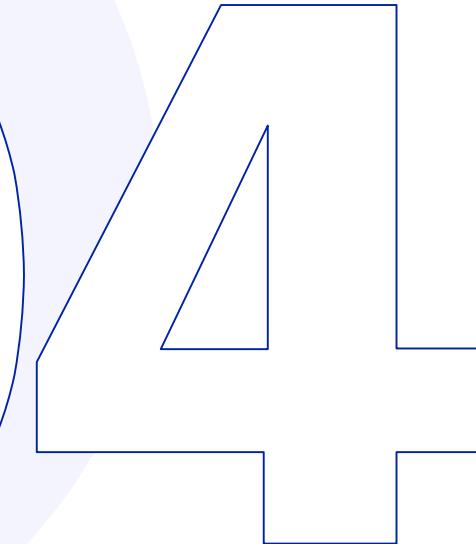
这部分的**重点是逻辑处理的过程，也就是HTTP解析和HTTP报文的制作。**

解析采用的是状态机和正则表达式，每次都读取以\r\n结尾的一段字符串，通过状态机来判定获取的字符串是属于HTTP请求的哪一部分，再跳转到相应的函数进行解析，如果读取的字符串没有以\r\n结尾则认为此次数据获取不完整，返回解析失败重新注册读事件。解析的时候判断是否是POST请求，如果是的话，还要解析POST的数据，里面包含了前端的请求信息，这部分是在数据体BODY部分，将BODY部分进行解析并存入到映射表里。**制作响应报文时，根据从请求报文中解析出来的POST来判断前端想要请求的什么信息，从而进行LOGIN, REGISTER, GETEDATA等操作**，并通过集中写将资源文件和响应报文分别发送回客户端。



第四部分” 项目总结

Summit of System Project



白驹过隙，时光荏苒，转眼间，基于深度学习的电力负载分类与预测系统的开发工作就到了收尾的阶段了。回顾这几个月的时间，我们收获颇丰。首先，在技能方面，我们更加熟练了人工智能算法设计与软件开发的相关知识和技术，积累了丰富而又宝贵的经验。我们以赛促学，通过这次大赛，我们学到了很多前沿的技术，比如基于超算互联网与边缘设备的架构，基于WeChat小程序的前端开发，以及使用Transformer结构的电力负载时间序列预测的人工智能算法设计等。此外，在这次大赛中，我们开阔了眼界，我们有幸能够直接接触到国家超级计算资源。

最重要的是，在这次大赛中，我们深深地体会到了合作的力量，而且我们三位成员都感受到了来自老师和同学们的无私帮助。我们三人通力协作，在指导老师的悉心指导下，共同努力了一百多个日日夜夜，终于完成了这个规模不小的电力负载分类与预测系统。我们会将这种齐心协力的精神发扬下去，传递给身边更多的同学，让拼搏与奋进的精神永不停息！

感谢各位评委老师的聆听！
Thank You For Your Listening!

Copyright © 2024 The research and development group for Power Load Classification and Prediction System Based on Deep Learning Algorithm, Faculty of Computer Science & Technology, Qilu University of Technology (Shandong Academy of Sciences).

Research and development team members:

- *Yu DU*(Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), No.202103180009)
- *Chuan JIANG*(Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), No.202103180020)
- *Xiaoyu LI*(Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), No.202103180001)
- *Qinglong LI*(Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), No.202103180027)
- *Yiwen ZHANG*(Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), No.202103180054)

Supervisors:

- *Ruixiang JIA*(Lecturer of Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences))
- *Jing CHEN*(Associate research fellow of Shandong Computer Science Center, National Supercomputing Center in Jinan)

WEBSITE: <https://github.com/duyu09/Powerload-Classification-and-Prediction-System>