ECE 445: Senior Design Laboratory
Design Document

Electric Load Forecasting(ELF) System

Ao Zhao
aozhao2@illinois.edu
Liyang Qian
liyangq2@illinois.edu
Yihong Jin
yihongj3@illinois.edu
Ziwen Wang
ziwenw5@illinois.edu

Sponsor:
Ruisheng Diao

Zhejiang University
University of Illinois Urbana-Champaign Institute

April 21, 2023

Modification

1. Add histogram for weather condition data, Figure 2.5 to Figure 2.8, in tolerance analysis.

2. Update block diagram and visual aid pictures with corresponding headings.

3. Fixed some minor problems in English.

4. Add quantitative description to the HLR list in introduction sections.

5. Add detailed description for subsystems.

6. Update the requirement and verification table.

Contents

1 Introduction

1.1 Problem and Solution Overview

Electric load forecasting (ELF) is a method that takes into account unstable factors, such as weather conditions and electricity prices, to predict the demand for electricity. Many utility companies rely on manual forecasting techniques based on specific datasets, but these methods may lack accuracy when fine-grained time particle forecasting is required. To accurately predict expenses on electricity and construct reliable infrastructures that can withstand a certain electrical load, utility companies need more advanced and reliable forecasting methods.

As far as we know, the methods to forecast short-term load can be grouped into two types: multi-factor forecasting methods and time series forecasting methods. According to Hammad et al. [1], Multi-factor forecasting methods focus on finding the relationship between different influencing factors and forecasting values. Time series forecasting methods take care of the historical time serial load data. In our project, we utilize a deep learning algorithm called DeepAR [2], which uses multi-factor time series to forecast load values. Specifically, the DeepAR algorithm combines the characteristics of two distinct groups to improve the accuracy of forecasts. Additionally, we have developed an edge device by embedding the trained model into a Raspberry Pi. This device can be installed locally on the power grid and is able to accurately forecast electricity load with robustness to network conditions.

For our solution and demo, there will be a edge device which has a trained deepAR forecasting model inside it. The device is equipped with the Electrical Distribution Box inside ZJUI building. By detecting the current and voltage from the electricity line, it can calculate the load power for an area of a building and calculate the load in an hour. By setting the continuous 1-hour load history and corresponding weather condition, including temperature, humidity and wind speed, as the input, the forecasting model inside the device will forecast the 1-hour load in the following one month for this box.

1.2 Visual Aid

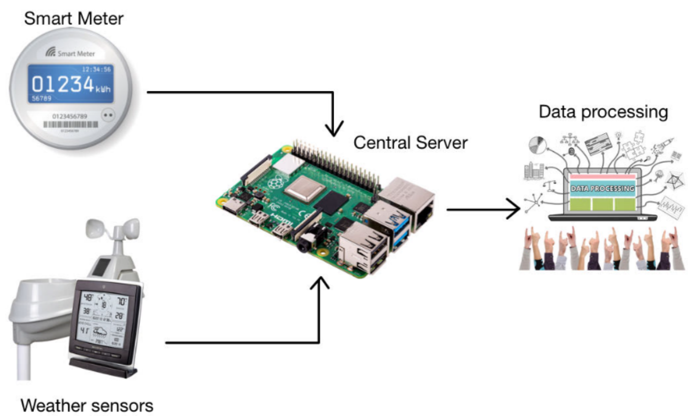

Figure 1.1: Visual Aid

Figure 1.1 shows the top-level view of our project. The project consists of smart meter, weather sensor, central server and data processing four parts. In this system, we will first use the smart meters we have selected and installed to obtain real-time power load data of the ZJUI building. With this and the weather forecast data we have obtained, a Raspberry PI or its updated version will serve as a central server to import our trained forecast model and finally obtain a forecast of the power demand of the ZJUI building.

1.3 High-level requirements list

1.3.1 Accuracy

The system will generate accurate predictions of future electric load usage. The accuracy of the predictions will be high enough to enable effective planning and optimization of electric power usage. **The RMSE, root mean squared error, will be less than 300. The MAPE, mean absolute percentage error, will be less than 0.5. The MAE, mean absolute error, will be less than 200.**

1.3.2 Scalability

The system will be capable of handling large volumes of data and generating predictions for a large number of electric load customers. The system will be able to scale up or down as the demand for electric power changes. **At the very least, it can process all the data of ZJUI's 1ABCDE and**

2ABCDE buildings and generate forecasts for them respectively. For each prediction request of a specific date, the memory space token by the subsystem is around 10MB.

1.3.3 Reliability

The system will be designed to be highly reliable and available. It will be able to handle failures gracefully and recover quickly from any disruptions in service. **The electricity usage data collector will be called per 30 minutes and the weather condition data will be called each day to keep track of the latest data. During the process of calling, both of these programs will check the data integrity in previous one month.**

1.3.4 Ease of Use

The system will be designed to be easy to use and accessible to a wide range of customers. The query API will be easy to understand and use, and the web page interface will be intuitive and user-friendly. **At least people can easily understand and use our user interface, such as choosing 1, 3 or 7 day power load forecasts. People can choose to make different predictions for different buildings.**

2 Design

2.1 Block Diagram

The system described consists of several interconnected subsystems responsible for data collection, aggregation, storage, training, and prediction. Process 1 is responsible for collecting and aggregating raw data from both the Cassandra database and the Data Measuring Subsystem. The aggregated data is then written to the Data Storage Subsystem, where it is periodically read by the Central Server Subsystem for use as a training dataset. Once the training is finished, the Central Server Subsystem triggers Process 2 with an extracted predictor, which is then deployed to the Edge Forecasting Subsystem. The Edge Forecasting Subsystem is enclosed in the Equipment Closure Subsystem and is responsible for predicting and facilitating human-computer interaction. Overall, the system is designed to efficiently and accurately process and predict large amounts of data.

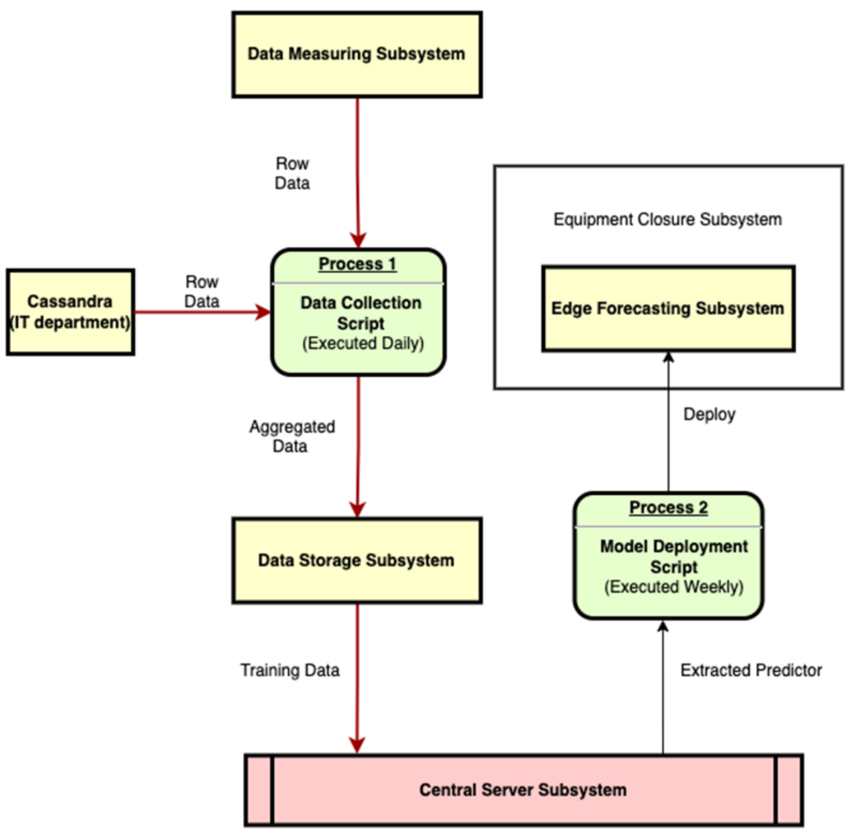

Figure 2.1: **Block diagram of project data pipline**

2.2 Physical Design

The physical design, shown **in Figure 2.2**, was designed to encapsulate the Raspberry PI. Mainly by the upper and lower two acrylic plates as the shell. A 4-inch display screen is required for the upper case. The size of the two boards is 126mm x 86.5mm, and the height will be supported by 35mm pillars.

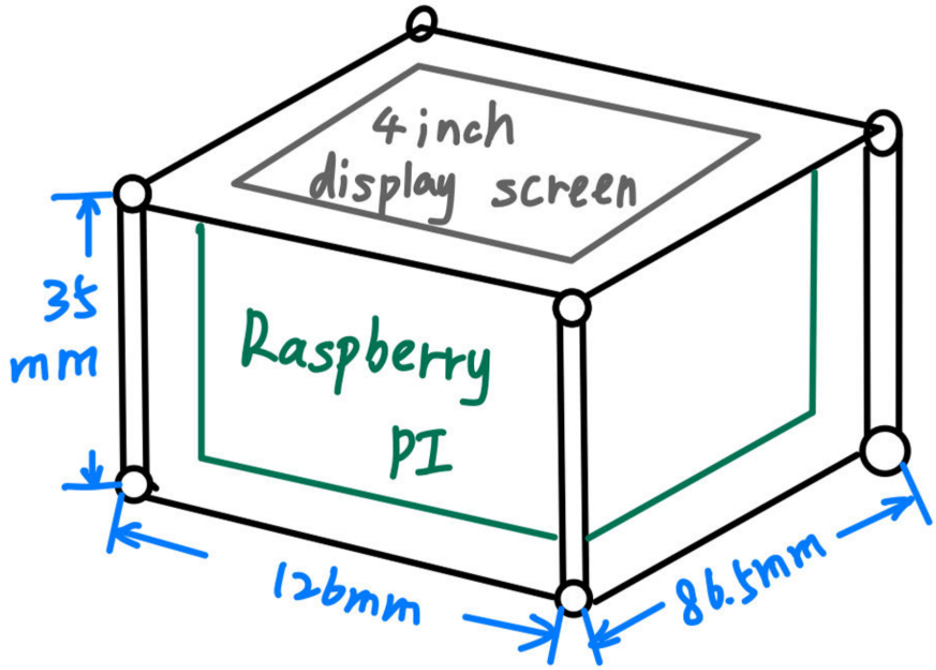

Figure 2.2: Physical Design

In the **Figure 2.3**, the left side is the upper plate and the right side is the lower plate. The upper board is equipped with a four-inch display screen, so it needs a center empty position. The lower plate needs four holes because at least four corner screws are required.

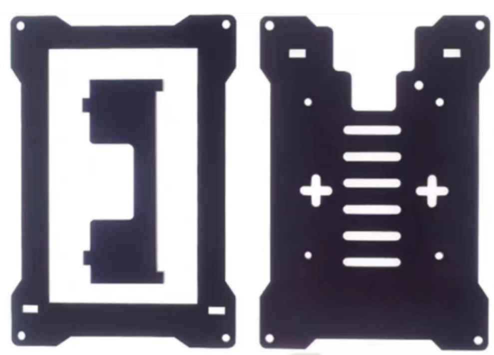

Figure 2.3: plate

The **Figure 2.4** shows the appropriate fan we selected to fit into the physical model, with an additional fan support.

Figure 2.4: fan and its support

2.3 Hardware module

2.3.1 Data Storage Subsystem

The subsystem is designed to interact with Raspberry PI, and there is a display to display the results. This subsystem consists of the above physical design model, plus Raspberry PI, fan, and display screen. **This subsystem will be employed to store the data from data collection script, which consists of the electricity usage data and weather condition data. After the historical data is collected, the data will be transferred to the central server for training the machine learning model.**

Table 2.1: Data Storage Subsystem

Requirement	Verification
The physical model can be properly encapsulated with Raspberry PI.	The size of the two boards is 126mm x 86.5mm, and the height will be supported by 35mm pillars. It's big enough to hold a raspberry PI.
After the data is entered, the final prediction needs to be presented.	We chose a 4.0-inch IPS full-view display with HDMI input and a refresh rate of 60FPS. Physical resolution 480×480, resistance touch screen. Compatible and can be plugged directly into all versions of Raspberry PI motherboards. It has a 3.5mm audio port, supports HDMI audio input, and has adjustable backlight. This will ensure that we have a good representation of the results on the raspberry PI.
The Raspberry PI needs to be kept at a safe temperature and avoid damage to the device.	Adding a fan is a good option, as it can effectively reduce the temperature. At the same time, we purchased 4 pairs of screws as the support of the fan.

2.3.2 Data Measuring Subsystem [Optional]

This subsystem is responsible for collecting real-time data on electric load usage. In this module, our basic solution is CT + meter + 4g gateway and then transfer the data to the cloud. This part can be used for the whole process to get the input data for the whole prediction process. **The Data Measuring Subsystem will be installed on the motors in ZJUI buildings to obtain the voltage and current data for a specific motor or a group of motors in a particular area. This data will be used to calculate the power consumption of these devices, serving as the historical electricity usage data for the prediction input data set.**

Table 2.2: Data Measuring Subsystem

Requirement	Verification
Accuracy is important in a data measuring subsystem to ensure the data collected and processed is reliable and trustworthy.	Use OPCT24AGL current transformers to detect the current and voltage: **Specification**: 1200A/5A **Frequency**: 50Hz **Error**: 0.5% **Load**: 5VA
The subsystem will be able to collect data in a timely manner, and provide real-time or near-real-time information that can be used for decision-making and analysis.	Write a python script to read from ZJUI's Cassandra database per 30 minutes to guarantee the real-time property of electricity. Similarly, a python script is required to crawl weather data from wunderground website [11].

2.4 Software module

2.4.1 Database Subsystem

This subsystem is mainly responsible for storing the collected real-time power load data of ZJUI building, to provide future calls to the data. **This subsystem is a key component of the Data Storage system and will be responsible for implementing various data cleaning tasks. This includes removing dirty code units from the weather condition data, cleaning up dirty electricity usage data, such as data points where the electricity usage is less than zero, and aligning the weather condition data with the electricity usage data for each building. Once the data cleaning task is complete, the subsystem will organize the clean data for each building into the training data set.**

Table 2.3: Database Subsystem

Requirement	Verification
Database design in a database system is important to ensure efficient data storage, retrieval, and management, and to prevent data inconsistencies and errors.	The design of the database will follow the paradigms of database design. The database will ensure that data is accurate, consistent, and valid. This can be achieved through the use of constraints. Besides, **normalization** ensures that each table has a unique purpose, and eliminates data duplication. It will also satisfy the ability of efficiency, flexibility, flexibility and scalability.
Hardware configuration in a database system is important for optimal performance, scalability, and reliability of the system, and to meet workload demands. The database requires a significant amount of memory to perform data operations effectively.	Hardware configure: **process**: AMD Ryzen Threaddripper PRO 3995WX 64-Cores **GPU**: NVIDIA RTX 3080*2 **Memory**: 64G*2 DDR4
Distributed systems in a database system are important for scalability, fault-tolerance, and high availability, and to enable data sharing and collaboration.	The storage of large data requires a distributed storage system that can store data across multiple machines in a network. Distributed storage systems such as Hadoop Distributed File System (HDFS) and Amazon S3 can provide distributed storage that is fault-tolerant and scalable.

2.4.2 Central Server Subsystem

The deployment of a trained machine learning model for making predictions on new data is typically handled by this subsystem. This subsystem is primarily responsible for training the final machine learning model that has been developed, and for transmitting the weather condition data from the weather detection website to the edge device. **The training API is called daily to train the model on the new data obtained from the building devices and weather website for that day. Once the training is completed, the central server subsystem will generate a validation plot for the training dataset and transfer the trained model to the edge device for use in the prediction function.**

Table 2.4: Central Server Subsystem

Requirement	Verification
It will have the ability to handle increasing amounts of data and computational demand without a significant degradation in performance.	The server access to wunderground weather [11] website per 30 minutes and crawls the weather condition form. It compares the weather condition form with the previous one, if it is different, then the newly crawled data will be transmit to the device.
Low latency in computing is important for fast response time and better user experience in real-time applications and services.	The above action guarantees that when the number of edge device increases, the transmit action does not take up too much network bandwidth.
Compatibility in computing is important for seamless integration, interoperability, and accessibility of software and hardware across different platforms and devices.	The subsystem will be designed to work with a variety of different machine learning models and frameworks, so that it can support a wide range of use cases and applications

2.4.3 Edge Forecasting Subsystem

The subsystem includes an edge device that is equipped with a predictor running on it. The predictor is generated from a previously trained machine learning model, and is responsible for processing input data received by the edge device in real-time.The edge device acts as a gateway for input data, receiving it from various sources and feeding it into the predictor for processing. The predictor then produces output that can be used for further decision-making or action by the overall system.

Table 2.5: Edge Forecasting Subsystem

Requirement	Verification
The Edge Forecasting Subsystem will have a predictor that can process input data in real-time with a latency of less than 500 milliseconds.	The data storage subsystem and forecasting subsystem is embedded on the same Raspberry Pi with 8G memory with 4G gateway. **The size of input data is around 1MB and the ML model has around 10K parameters. With the 8G memory space and CPU cortex-A53 that is enough for loading the trained machine learning model and running the prediction of ML model.**
The predictor will have an accuracy of at least 90% when predicting the occurrence of an event. The MAE metrics is employed as the statistical analysis method for the accuracy of the prediction for 1 month prediction.	The train dataset consists of the hourly weather data and electricity load data from 07-31-2020 17:00 UTC to 03-08-2023 17:00 UTC. DeepAR pytorch model with hyper-parameter: **learning_rate**: 0.1, **hidden_size**: 30, **rnn_layers**: 2, **prediction_length**: 720, **encoder_length**: 168. **Prediction metrics, consisting of RMSE, MAPE, MAE, is choosen to evaluate the accuracy of our machine learning model.**
The edge device will be able to handle input data from multiple sources, including sensors and other devices, with a minimum throughput of 100 messages per second.	The throughput of the edge device will be measured using a simulated workload. **Users are allowed to choose different time interval, random date in one month, as the custom weather condition to see the effect of weather condition on electricity usage.** The results will be compared against the **original record electricity usage** using **the evaluation metrics mentioned above**.
The subsystem shall have a robust error handling mechanism that can detect and handle errors in the input data or predictor output.	The error handling mechanism will be tested using different types of errors in the input data and predictor output, and the results will be evaluated by a team of software engineers.

Table 2.6: Average difference table for temperature, dew point, humidity and pressure

	Temperature (F)	Dew Point (F)	Humidity (%)	Pressure (in)
Average Δ/hour	-8.4e-4	-1.1e-3	-7.5e-4	1.2e-5
Average value	64.968	54.929	72.623	29.994

2.5 Tolerance Analysis

One important tolerance we must maintain is the error of weather condition data. The electrical load of the system varies with the weather conditions, and there are some scenarios in which the hourly weather condition is unstable. Optimally, to forecast the load difference per hour, a fine-grid continuous weather condition data is required, which causes large computation complexity and makes the train dataset hard to organize. Thus, we choose the 30th minute of each hour to be treated as the weather condition. For example, 1:30am's weather condition standards for that between 1:00am and 2:00am. According to the weather data we crawl, the average factor difference per hour is shown in the **Table 2.6**.

For these factors **in Table 2.6**, the average factor difference is trivial compared with the value of equipment itself. This suggests that, in the majority of cases, the weather condition is stable and the discrete weather condition data can be treated as the continuous weather data in one-hour time intervals.

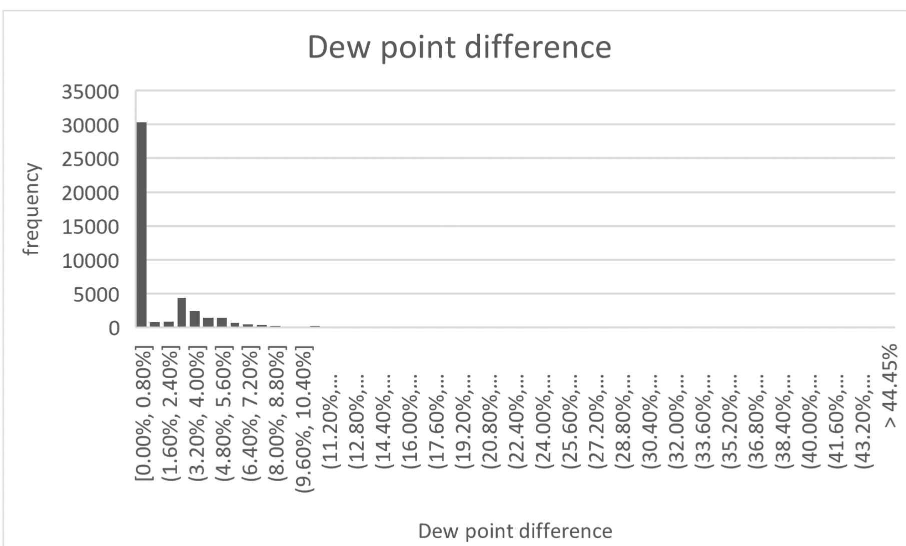

Figure 2.5: Dew point histogram

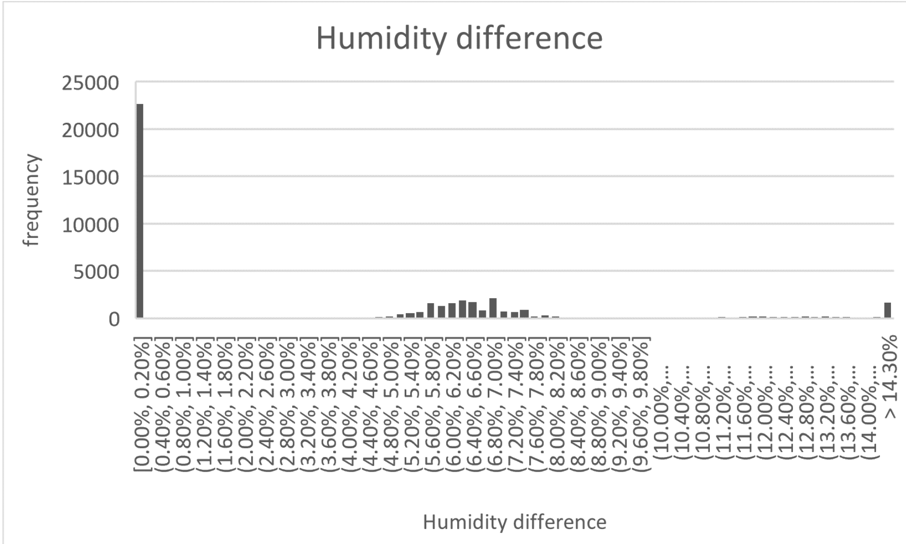

Figure 2.6: Humidity histogram

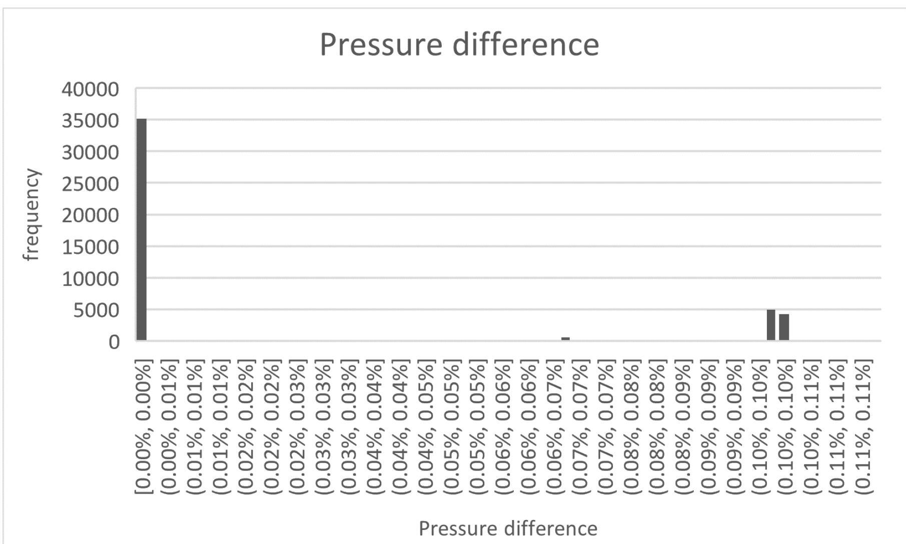

Figure 2.7: Pressure histogram

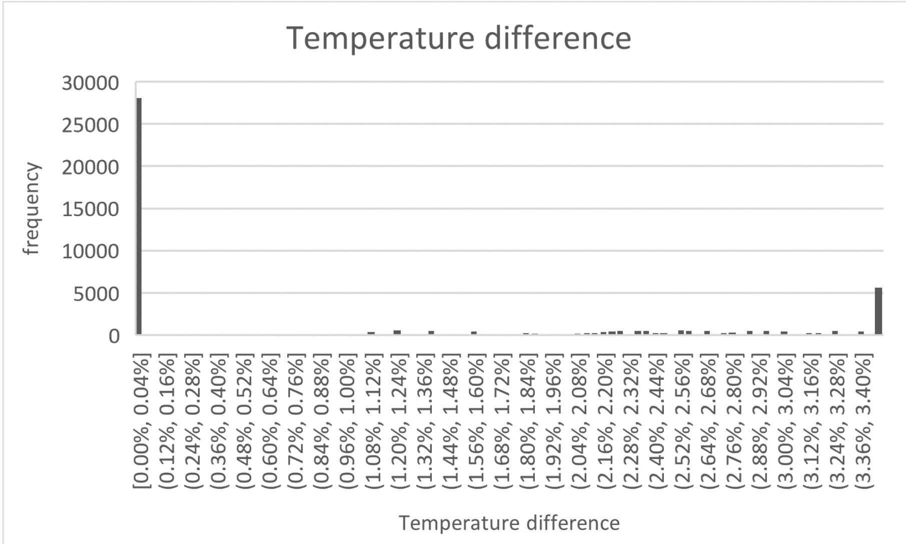

Figure 2.8: Temperature histogram

Figure 2.5 to Figure 2.8 show the frequency for the varying in each hour for the training data set we crawled. These figures indicate that more than 90% weather condition data varying less than 3% for one hour. Thus, it can be concluded that the weather condition we choose is stable in the one hour time interval. The error from using constant weather condition data in each hour instead of continuous weather condition data can be tolerated.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

Our fixed development cost is estimated at 40 yuan per hour for four people working 8 hours per week. We considered about 60% of the final design during this semester (14 weeks)
One member: (40 yuan/hour) x (8 hours/week) x (14 weeks x 0.6) x 2.5 = 6720 yuan
Total labor: 4 x 6720 yuan = 26880 yuan

3.1.2 Mechanical Parts

Table 3.1: Cost and Schedule: Mechanical Parts

Part Name	**description, manufacturer, part #, quantity**	**Cost**
Raspberry PI4 4B 8GB	It is used as a concentrator to obtain input data and obtain prediction as output after importing the model; Premier Farnell PLC; 1	1438 yuan
Display screen	It is used to display the forecast results for easy use; Shenzhen Shengchengwei Technology Co., LTD; 4.0 "IPS full viewing Angle 800 x 480 Raspberry PI display; 1	121 yuan
Fan	Ensure that Raspberry PI is operating at a normal temperature and within a safe range; Shenzhen Shengchengwei Technology Co., LTD; 5V all-in-one end Raspberry PI cooling fan ; 1	11.5 yuan
Shell board and screws	Provide an enclosed case for better installation and protection; Shenzhen Shengchengwei Technology Co., LTD; Acrylic sheet as well as screws, posts; 1	12.5 yuan
type current transformer three-phase opening and closing buckle	Used to ensure the safety of current, convenient measurement; Shandong People Network Co., LTD; OPCT24AL35AL-100A/5A; 1	56 yuan(optional)

Table 3.2: Cost and Schedule: Mechanical Parts (cont)

Part Name	**description, manufacturer, part #, quantity**	**Cost**
4G gateway, 220V AC power supply	To connect the meter data to the Internet, so we can get real-time data; Shandong People Network Co., LTD; USR-DR512; 1 (optional)	168 yuan(optional)
4G wireless three-phase four-wire multifunctional smart meter	It is used to measure real-time data of electric load; Changsha Shewei Meter Information Technology Co., LTD; [4G] 380V*1.5(6)A 0.5s class; 1 (optional)	638 yuan(optional)
Total	-	1583 yuan (2445 yuan if contain optional parts)

3.1.3 Sum of costs into a grand total

Our labor cost is 26,880 yuan, adding 1583 yuan for different parts, the total comes to 28,463 yuan. If we add a whole smart meter component, the total is 29,325 yuan.

3.2 Schedule

3.2.1 Schedule of Ao Zhao

2/17/23: Understand and determine the power load history and how to obtain weather data
2/24/23: Get campus data with team members
3/03/23: Contact the school staff (Jiang) to seek access to the data
3/10/23: The available historical power load data of ZJUI campus was successfully obtained
3/17/23: Field study to see if we can add meters to get real-time power load data, and buy a transformer, electric meter, gateway
3/24/23: Design reasonable hardware to ensure the normal operation of Raspberry PI, and purchased Raspberry PI, fan, shell, etc
3/31/23: Assemble hand Raspberry PI, fan, case, screws, monitor, etc.

4/07/23: Write the code to store the load data from the two buildings that we pulled from the school's master database
4/14/23: Solve any problems existing in the overall hardware device
4/21/23: Try to combine the real-time data in the above library with the algorithm part, to achieve input and output
4/28/23: Connecting the whole system, putting it together
5/05/23: Test the system and make improvements
5/08/23: Mock **demo**
5/12/23: Prepare **final report** draft
5/23/23: Complete the **final report and functionality demonstration video**

3.2.2 Schedule of Yihong Jin

2/17/23: Understand and determine the data flow and hardware platform
2/24/23: Get campus data with team members
3/03/23: Write script to replicate power load data from ZJUI campus
3/10/23: Setup data storage subsystem
3/17/23: Setup and design the HCI functionality of the edge forecasting subsystem
3/24/23: Write the script to deploy the model to the edge forecasting subsystem
3/31/23: Test the Raspberry PI with designed HCI functionalities
4/07/23: Train the model to get best performance
4/14/23: Train the model to get best performance
4/21/23: Build load and predict API
4/28/23: Embedding model
5/05/23: Test integrated system
5/08/23: Mock **demo**
5/12/23: Prepare **final report** draft
5/23/23: Complete the **final report and functionality demonstration video**

3.2.3 Schedule of Liyang Qian

2/17/23: Choose ML algorithm
2/24/23: Get campus data with team members
3/03/23: Crawl weather data from website
3/10/23: Organize weather data from Wunderground weather website [11]
3/17/23: Find the packet code for algorithm
3/24/23: Run the deepAR notebook with example data
3/31/23: Use the custom Data

4/07/23: Train the model to get best performance

4/14/23: Train the model to get best performance

4/21/23: Build load and predict API

4/28/23: Embedding model

5/05/23: Test integrated system

5/08/23: Mock **demo**

5/12/23: Prepare **final report** draft

5/23/23: Complete the **final report and functionality demonstration video**

3.2.4 Schedule of Ziwen Wang

2/17/23: Get familiar with campus measurement and data storage system

2/24/23: Get clear about what kinds of data we desire to have as input of machine learning model

3/03/23: Reach out to campus staff (Jiang) in charge to have deeper information

3/10/23: Align with Jiang about data structure and authentication of data

3/17/23: Develop code to select targetted data and get the historical data from Jiang

3/24/23: Try to have the access to real time data from campus measurement system (fail)

3/31/23: Align with Jiang about what and which extend of access can we have

4/07/23: Develop codes based on the access we can have to select data in best effort

4/14/23: Design enclosure system of main calculation system (Raspberry Pi)

4/21/23: Manufacture the enclosure system and test its usability

4/28/23: Assemble the overall system together

5/05/23: Test and improve performance of the system

5/08/23: Mock **demo**

5/12/23: Prepare **final report** draft

5/23/23: Complete the **final report and functionality demonstration video**

4 Discussion of Ethics and Safety

4.1 Ethical Standards

4.1.1 Privacy and Security

ELF systems collect data on energy consumption patterns, weather data, and other personal information. Developers must ensure that the system complies with data protection regulations such as the General Data Protection Regulation (GDPR) [4] and the California Consumer Privacy Act (CCPA) [5].

Additionally, developers must ensure that the data collected is not misused, abused, or sold to third parties without the users' consent. The ACM Code of Ethics [6] states that developers will respect privacy and protect confidential information, and the IEEE Code of Ethics [7] states that engineers should respect the privacy of others and protect the confidentiality of data.

4.1.2 Impact on Vulnerable Populations

ELF systems' predictions may lead to price hikes, making electricity more expensive for low-income households. The IEEE Code of Ethics [7] states that engineers should consider the social and environmental impact of their work and seek to minimize any negative consequences. Developers must prioritize the well-being of all stakeholders, including vulnerable populations.

4.1.3 Bias

ELF systems' algorithms can be influenced by underlying biases, leading to inaccurate predictions. Developers must ensure that the system's algorithms are designed to minimize any bias that may impact the accuracy of the system. The ACM Code of Ethics [6] states that developers should not discriminate against individuals or groups and should ensure that their work is free from bias.

4.2 Safety Standards

4.2.1 Electrical Safety

ELF systems are part of the electrical grid, and as such, they must comply with electrical safety standards such as the National Electrical Code (NEC) [8] . The NEC provides guidelines for the design, installation, and operation of electrical systems, including ELF systems.

The International Electrotechnical Commission (IEC) [9] also provides safety standards such as IEC 61508 and IEC 61511 that provide guidelines for the development and implementation of safety-critical systems such as ELF systems.

4.2.2 Cybersecurity

ELF systems are connected to the internet, which exposes them to cybersecurity risks such as hacking and data breaches. Developers must ensure that the system is secure and protected from unauthorized access or misuse. The IEC [9] provides cybersecurity standards such as IEC 62443 that provide guidelines for securing industrial control systems, including ELF systems.

4.2.3 System Reliability

ELF systems must be reliable to ensure that the predictions are accurate and that the system functions as intended. The IEC [9] provides reliability standards such as IEC 62278 that provide guidelines for the development of software-based safety systems, including ELF systems.

4.2.4 Worker Safety

Workers who operate and maintain the ELF system must be protected from electrical shocks or burns. Developers must ensure that the system is designed to minimize the risk of injury to workers who operate and maintain the system. The Occupational Safety and Health (OSH) [10] provides guidelines for worker safety, including electrical safety, that must be followed when designing and operating ELF systems.

In conclusion, referencing relevant ethical and safety standards can help developers address the ethical and safety challenges involved in developing and using Electric Load Forecasting (ELF) systems. Developers must comply with data protection regulations, prioritize the well-being of all stakeholders, minimize bias in the system's algorithms, ensure transparency, and comply with electrical safety standards. Additionally, developers must ensure that the system is secure, reliable, and that workers who operate and maintain the system are protected from harm.

4.3 Safety concerns of the project.

Developers of ELF systems should consider the following steps to mitigate safety concerns:

1. Identify potential hazards: Developers should conduct a hazard analysis to identify potential hazards associated with the ELF system, including electrical hazards, cybersecurity risks, and worker safety concerns.

2. Develop safety procedures: Based on the hazard analysis, developers should develop safety procedures that address each identified hazard. Safety procedures should include lab safety documents for batteries, interfaces, devices, etc. Safety procedures should be comprehensive, detailed, and easy to understand.

3. Train personnel: Developers should provide training to personnel who will be operating, maintaining, or otherwise interacting with the ELF system. Training should cover the safety procedures, potential hazards, and how to respond in the event of an emergency.

4. Monitor and evaluate: Developers should regularly monitor and evaluate the effectiveness of the safety procedures to ensure that they remain up to date and effective in addressing potential hazards.

In terms of justifying design decisions that sufficiently protect both users and developers from unsafe conditions caused by the project, developers should be able to demonstrate that they have conducted a comprehensive hazard analysis, developed and implemented appropriate safety procedures, and provided personnel with appropriate training and PPE. Additionally, developers should regularly monitor and evaluate the safety procedures to ensure that they remain effective in addressing potential hazards.

Overall, safety should be a top priority for developers of ELF systems, and safety procedures should be regularly reviewed and updated to ensure that they continue to provide effective protection from potential hazards.

5 Citations

[1] M. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and Models for Electric Load Forecasting: A Comprehensive Review," *Logistics & Sustainable Transport*, vol. 11, no. 1, pp. 51–76, Feb. 2020, doi: 10.2478/jlst-2020-0004.

[2] D. Salinas, V. Flunkert, and J. Gasthaus, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks," *arXiv (Cornell University)*, Apr. 2017, [Online]. Available: `https://arxiv.org/pdf/1704.04110.pdf`

[3]"DeepAR — pytorch-forecasting documentation." `https://pytorch-forecasting.readthedocs.io/en/stable/api/pytorch_forecasting.models.deepar.DeepAR.html#pytorch_forecasting.models.deepar.DeepAR`

[4] P. Voigt and A. Von Dem Bussche, "The EU General Data Protection Regulation (GDPR)," *Springer eBooks*, Jan. 2017, doi: 10.1007/978-3-319-57959-7.

[5]P. Bukaty, *The California Consumer Privacy Act (CCPA): An implementation guide*. IT Governance Ltd, 2019.

[6] D. Gotterbarn, A. Bruckman, C. Flick, K. W. Miller, and M. J. Wolf, "ACM code of ethics," *Communications of the ACM*, vol. 61, no. 1, pp. 121–128, Dec. 2017, doi: 10.1145/3173016.

[7] "IEEE Code of Ethics," *IEEE Potentials*, vol. 23, no. 2, pp. 32–33, Apr. 2004, doi: 10.1109-/mp.2004.1301956.

[8] N. F. P. Association, *National Electrical Code*. Cengage Learning, 2010.

[9] I. E. Commission, *The International Electrotechnical Commission (IEC) Structure and Operations: General Notes*. 1995.

[10] C. D. Reese, *Occupational Safety and Health: Fundamental Principles and Philosophies*. CRC Press, 2017.

[11] "Hangzhou, People's Republic Of China Weather History | Weather Underground." https://www.-wunderground.com/history/daily/ZSHC/date/2022-3-27