Assignment 2 - Group: 82

Group members:

1. **540919152**
2. **540323234**

Abstract

The objective of the report is to predict the future electricity consumption of users in order to rationally allocate electricity resources and solve the problem of electricity load. The models used in this project include Multi-layer Perceptron (MLP), Long short-term memory Network (LSTM), and Gated Recurrent Unit (GRU). We used the "Electricity Load Diagrams" dataset from 2012 to 2015 [2], which records the hourly electricity consumption of 100 customers. To solve this time-series regression problem, we standardized the data and trained the MLP, LSTM, and GRU models separately. Through the modality training of the model, the experimental results show that although LSTM and GRU have advantages in dealing with time dependence, the MLP model performs best in this task, with the lowest mean square error (MSE) and the shortest training and prediction time. As for future studies, we hope to further explore how to optimize the model to capture more time series characteristics and optimize model performance in more complex power demand scenarios.

Introduction

In recent years, with population growth and rising living standards, global energy demand has continued to increase, power load forecasting becomes critical in modern smart grids [1]. Accurately predicting future power consumption can help power companies optimize resource scheduling, reduce operating costs, and improve system stability and efficiency. The project used the "Electricity Load Diagrams 2012-2015" dataset [2], which records the hourly electricity consumption of 100 customers from 2012 to 2015. The data set has obvious time dependence and seasonal fluctuations.

To deal with this time series regression problem, we employ three deep learning models: Multi-layer Perceptron (MLP), Long short-term memory Network (LSTM), and Gated Recurrent Unit (GRU). Each of these models has its advantages, with MLP models generally suitable for dealing with nonlinear relationships, while LSTM and GRU are better at capturing time dependencies. The report evaluates the performance of the three models by preprocessing the data, adjusting the hyperparameters, and training the models, and compares them based on the mean square error (MSE).

The experimental results show that the MLP model has the best performance and the lowest prediction error. Although LSTM and GRU are excellent at handling time series, MLP achieved the highest prediction accuracy in this mission. This finding provides an important reference for selecting data-driven models suitable for specific tasks.

Data

1. Data Description and exploration

This dataset covers the electricity consumption per hour of 100 users (MT_001-MT_100) from 2012 to 2015, where the electricity consumption is measured in kilowatt-hours (kWh).

```
[5 rows x 101 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26304 entries, 0 to 26303
Columns: 101 entries, time to MT_100
dtypes: datetime64[ns](1), float64(100)
memory usage: 20.3 MB
None
```

Figure 1. Information of the dataset

Figure 1 shows that the dataset has 101 columns and 26304 rows of data. Since some users were counted after 2011, the previous electricity consumption data was 0 [2]. Due to the existence of summer time and winter time, the electricity consumption of 1 hour when the winter time is converted to daylight saving time is 0, and the electricity consumption of 1 hour when the summer time is converted to winter time is actually 2 hours.

At first, we consider to explore the electricity distribution of the users. It is difficult to display all of the 100 users completely in one figure, so we just select the first three users to do visualization by KDE (Kernel Density Estimation) plot.

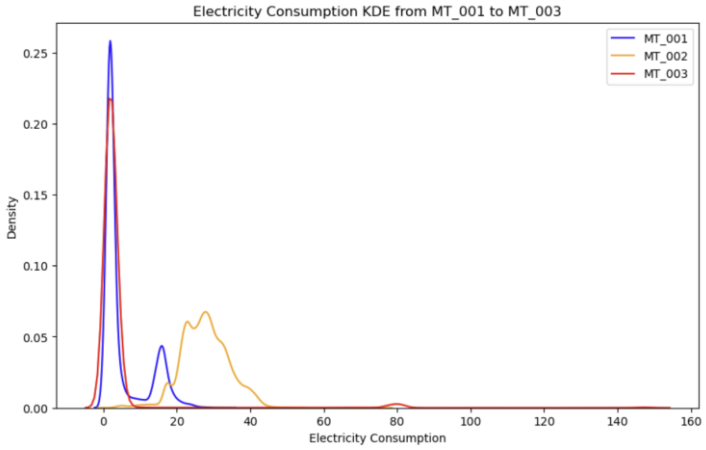

Figure 2. Electricity consumption KDE of the first three users

Figure 2 shows that MT_001 and MT_003 show very similar consumption patterns, with both users having low electricity usage ranging from 0 to 5. In contrast, MT_002 demonstrates a wider range of consumption, which suggests that MT_002 consumes electricity frequently from 2012 to 2015.

Moreover, we want to explore the dataset from the dimension of time. Thus, we randomly choose the January of 2013 to explore whether the dataset has some rules.

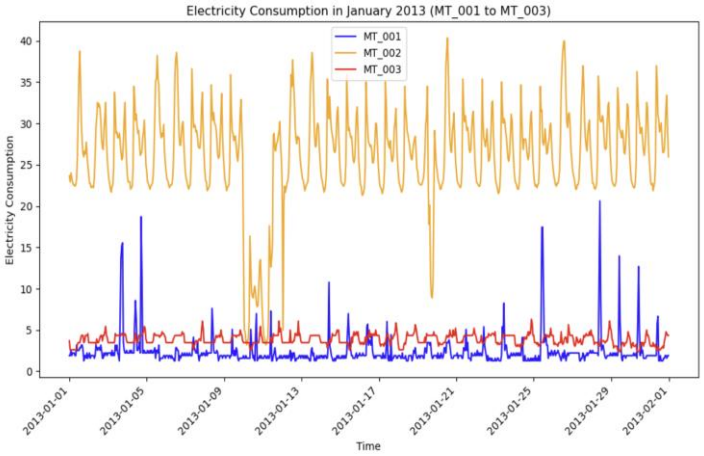

Figure 3. Electricity consumption in January 2013

Figure 3 illustrates that the electricity usage of all of the first three users exhibits a periodic pattern, which means that they tend to use electricity more frequently during certain periods, such as for work and daily activities, and use less electricity during other periods, such as for rest and sleep. MT_002 shows the highest and most fluctuating electricity usage throughout the month, while MT_001 and MT_003 display significantly lower and more stable consumption.

2. Pre-processing

Retain outliers:

Since there are many factors affecting the electricity consumption, such as whether people live in the property, seasonal changes, the impact of winter and summer time, etc., though we found outliers when we looked at the data, we could not directly fill in the mean or delete these outliers. Therefore, we retain these outliers, which is conducive to the training of time series regression models and improve their accuracy.

Standardization:

Before training the models, we used 'StandardScaler()' to standardize the input features, which means transform them with mean equals to 0 and standard deviation equals to 1. When we applied the models such as MLP, LSTM, and GRU, it helps to accelerates the convergence of models. Moreover, in this experiment, standardization ensured that both the training and test sets were scaled consistently.

Data Split

In order to ensure the validity of the model in time series prediction tasks, we use 'TimeSeriesSplit()' to split the data. We divide the dataset into training sets and test sets in time series, and after each split, the subsequent test_set portions are not included in the previous training_set, ensuring that the model does not see the data for future time steps.

Test data

The dataset contains electricity consumption statistics of 100 users and it is relatively independent. We use all the data to train the model, but if all the data is selected for verification and visual evaluation of the model, the workload is huge. Due to time and other factors, we decide to select three representative users for verification, namely MT_001-MT_003. The reason for selecting these three users is that when we checked the data, we found that some users had accounts opened after 2012 or were away from home for a

long time due to work or travel, and the electricity statistics were all 0 for several months, so we selected the data of 3 households with normal electricity consumption.

Methodology

1. MLP:

The first model we chosen is Multi-layer Perceptron (MLP) because it is simple and efficient, and it is especially suitable for dealing with regression problems with nonlinear relationships. And because of its inherent capability of arbitrary input-output mapping, the MLP networks are used in a variety of problems especially in forecasting [3].

The MLP usually consists of several layers of nodes. The first or lowest layer is the input layer that receives the external information [3], for this study, it is the electricity consumption data (after standardized) and the time feature (year, date and hours) column. The last or highest layer is the output layer to obtain the solution to the problem [3], which is the predicted value of the electricity consumption. The input and output layers are separated by one or more intermediate layers called hidden layers [3]. The MLP model built by this project has two hidden layers, both of them contain 64 and 32 neurons, respectively. And we use ReLU activation functions to capture nonlinear relationships in the data which is beneficial to improve the computational efficiency, reduce the gradient disappearance problem, improve the generalization ability of the model and reduce the risk of overfitting. After each hidden layer, 20% Dropout (0.2) is also applied to reduce overfitting. Nodes in adjacent layers are completely connected by a ring arc that runs from lower to higher layers.

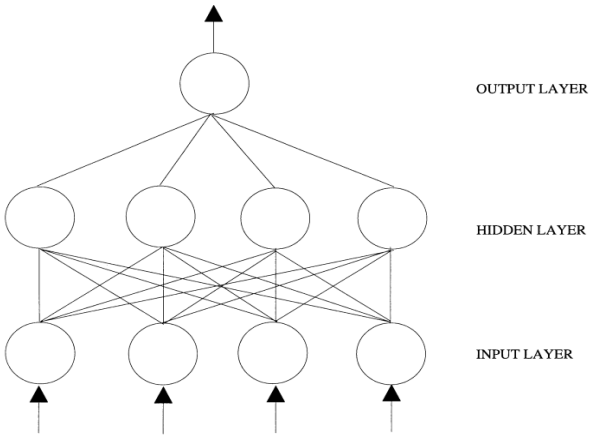

Figure 4. MLP simplified diagram [3]

However, the MLP model also has some limitations. MLP models cannot handle long-term dependencies in time series. Therefore, despite the introduction of temporal features through feature extraction, MLP may still not be able to capture temporal dependencies and periodic changes in power consumption as naturally as LSTM or GRU. For this reason, MLP relies heavily on input data preprocessing, as it must rely on a large amount of feature engineering (such as extracting time features) to help the model understand the time laws in the data. If feature extraction is inadequate, the performance of the model may be greatly affected. MLP is prone to gradient disappearance in time series data, especially when long-term dependence is

involved, and it is difficult to capture temporal correlation effectively [4]. Moreover, MLP has a limited number of layers and neurons, and can only capture simple nonlinear relationships.

2. LSTM:

We chose Long short-term memory network (LSTM) because it is good at handling long-term dependencies in time series data. Through its unique gating mechanism, LSTM can effectively solve the problem of gradient disappearance encountered by traditional neural networks when dealing with long time series. LSTM is a special kind of RNN, which has additional features for memorizing the sequence of data, namely, remembering the trend of the data until some point in time is rendered possible through some gates along with a memory line [5].

As for the forecast of future electricity consumption project, and the electricity consumption is recorded on an hourly basis, we set the characteristic number of the input layer, that is, the power consumption value, to 1. The LSTM model consists of two LSTM layers, each of which contains 50 neurons. Setting 'return_sequences=True' in the first layer means that the layer will return the output of each time step, not just the last time step. This setup ensures that the output of that layer can be used as input to the LSTM of the next layer, continuing to capture complex dependencies in the time series. The second layer sets 'return_sequences=False' to return only the output of the last time step. After the LSTM layer, we added a fully connected layer with 25 neurons. The role of this layer is to further process the features of the output of the LSTM layer in order to generate valid inputs for the final regression task. The output layer is the last layer of the model and contains a neuron that outputs the final predicted value of power consumption. Since this task is a regression problem, the output layer does not use the activation function, and directly outputs a continuous numerical result representing a prediction of power consumption at a certain time in the future.

To better explain this model, we refer to the following functions [6]:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i)$$
$$f_t = \sigma(W_{x_f} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f)$$
$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o$$
$$H_t = o_t \circ \tanh(C_t)$$

In these functions, X_t represents electricity consumption value in the current hour, H_{t-1} means the hidden state of the previous time step, that is, when the model processes electricity consumption at 14:00, it represents the hidden state at 13:00.

Although LSTM model is very suitable for processing time series data, it still has disadvantages. The gate mechanism of the LSTM model (input gate, forget gate, output gate) requires a lot of computation at each time step. This makes LSTM training inefficient when dealing with long time series or large datasets, especially when using multi-layer LSTM. And because of the dependency between LSTM time steps, the calculation of each time step must depend on the result of the previous time step, so it is difficult to parallel processing. This feature of sequential computation makes LSTM less efficient when dealing with long sequences. The LSTM model has a lot of parameters and requires a lot of training data to effectively learn complex dependencies in time series. If the amount of data is insufficient, the LSTM model is easy to overfit, resulting in poor generalization ability.

3. GRU:

Due to the limitations of LSTM model, we chose Gated Recurrent Unit (GRU) model for training. As a simplified version of LSTM, it has similar long and short-term dependency capture capabilities, while being more computationally efficient. GRU is able to handle long-term dependencies in time series well by reducing the redundant gate mechanism, which incorporates forget gates and input gates [7]. As a result, GRU models may have greater advantages in time-series tasks such as predicting electricity consumption, especially in cases where rapid training and prediction are required. In addition, because GRU has fewer parameters, compared with LSTM model, it can reduce the calculation cost while ensuring the accuracy.
In order to visually demonstrate the difference between GRU and LSTM, we quote the following functions [7]:

$$z_t = \sigma(W_z * [h_{t-1,} x_t] + b_z)$$
$$r_t = \sigma(W_r * [h_{t-1,} x_t] + b_r)$$
$$h\tilde{}_t = tanh(W * [r_t * h_{t-1,} x_t] + b)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * h\tilde{}_t$$

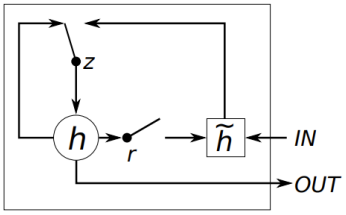

Figure 5. GRU diagram (where r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.) [7]

Similar to the LSTM model, the hourly electricity consumption value is used as the input to the model. 'input_shape, 1', where 'input_shape' represents the number of steps in the time series, and '1' indicates that each time step has only one feature, the electricity consumption value. The GRU model consists of two GRU layers. The first layer contains 50 neurons and set 'return_sequences=True' to output the results of each time step of the sequence, ensuring that the output from this layer can be passed to the second layer GRU. The second layer GRU also contains 50 neurons, but 'return_sequences=False' is set to return only the output of the last time step for transmission to subsequent fully connected layers. After the GRU layer, we also added a fully connected layer of 25 neurons to further process the features extracted by the GRU layer and generate the predicted inputs. Similar to the LSTM architecture, the fully connected layer of the GRU does not use activation functions in order to maintain linear transfer and stability of the model. Finally, there is the output layer, which contains a neuron that outputs the final predicted value of electricity consumption.
Although GRU has fewer parameters and can calculate faster than LSTM, it has some limitations as well. In the GRU structure, the forget and input gates are combined into one update gate. In addition, GRU does not have a separate Cell State. The Cell State of LSTM model provides a specialized pathway for the long-term storage of information, while the GRU relies only on Hidden State, so in some cases GRU may not retain long-term information as effectively as an LSTM. Due to this limitation, GRU may not be as flexible and efficient as LSTM when dealing with complex or long-term tasks. Moreover, though GRU has been

widely used, the research literature and application cases of GRU are relatively few compared to the maturity of LSTM. This may mean that the GRU is relatively under-optimized and improved in some specific areas and needs more practice to perfect.

Experimental Results

The dataset used for this experiment includes electricity consumption data of 100 users, with each user representing a unique time series. The object of this experiment is to predict the electricity consumption of each user using MLP, LSTM and GLU. However, it is hard to take all of the 100 users as target users because of the huge workload. We just took the first 3 users in this experiment to do training and evaluating. They are independent of each other, so the three were selected separately as target users. This allows us to observe how well the models adapt to users with different characteristics. Each model used hyperparameter tuning to optimize the number of neurons and epochs by 'RandomizedSearchCV()'. The experimental setting and model performance were evaluated using Mean Squared Error (MSE), training time, and prediction time. As our target users are MT_001, MT_002 and MT_003, we intended to set the average value of MSE, training time, and prediction time as our final metrics to find the best model.

Table 1. Comparison of Best Parameters, MSE, Training Time, and Prediction Time for MLP, LSTM, and GRU Models Across MT_001, MT_002 and MT_003 after the third split

User	Model	Best Parameters	MSE	Training Time (s)	Prediction Time (s)
MT_001	MLP	{'model__neurons': 64, 'epochs': 40}	36.94	80.98	0.74
	LSTM	{'model__neurons': 64, 'epochs': 20}	35.52	1188.45	10.88
	GRU	{'model__neurons': 64, 'epochs': 20}	42.95	1085.03	7.84
MT_002	MLP	{'model__neurons': 32, 'epochs': 20}	13.95	27.40	0.64
	LSTM	{'model__neurons': 32, 'epochs': 20}	15.53	706.63	6.11
	GRU	{'model__neurons': 32, 'epochs': 20}	15.28	813.59	5.56
MT_003	MLP	{'model__neurons': 64, 'epochs': 20}	0.14	47.86	1.43
	LSTM	{'model__neurons': 32, 'epochs': 20}	0.97	1109.28	10.48
	GRU	{'model__neurons': 64, 'epochs': 40}	7.62	2862.07	9.08
Average	MLP	/	17.03	52.08	0.94
	LSTM	/	17.43	1001.45	8.89
	GRU	/	24.46	1586.90	7.49

From table 1, we could summary that when predicting the electricity consumption for users MT_001, MT_002, and MT_003, the MLP, LSTM, and GRU models demonstrated different performance. Among

all users, the MLP model had the lowest average MSE of 17.03, the shortest average training time of 52.08 seconds, and the fastest prediction time of 0.94 second. Additionally, LSTM and GRU had longer average training and predicting time overall. LSTM is up to 1001.45 seconds in training, 8.89 seconds in predicting and GRU is up to 1586.90 seconds, 7.49 seconds in predicting. It reflects the greater complexity and accuracy of LSTM and GRU in time series prediction tasks.

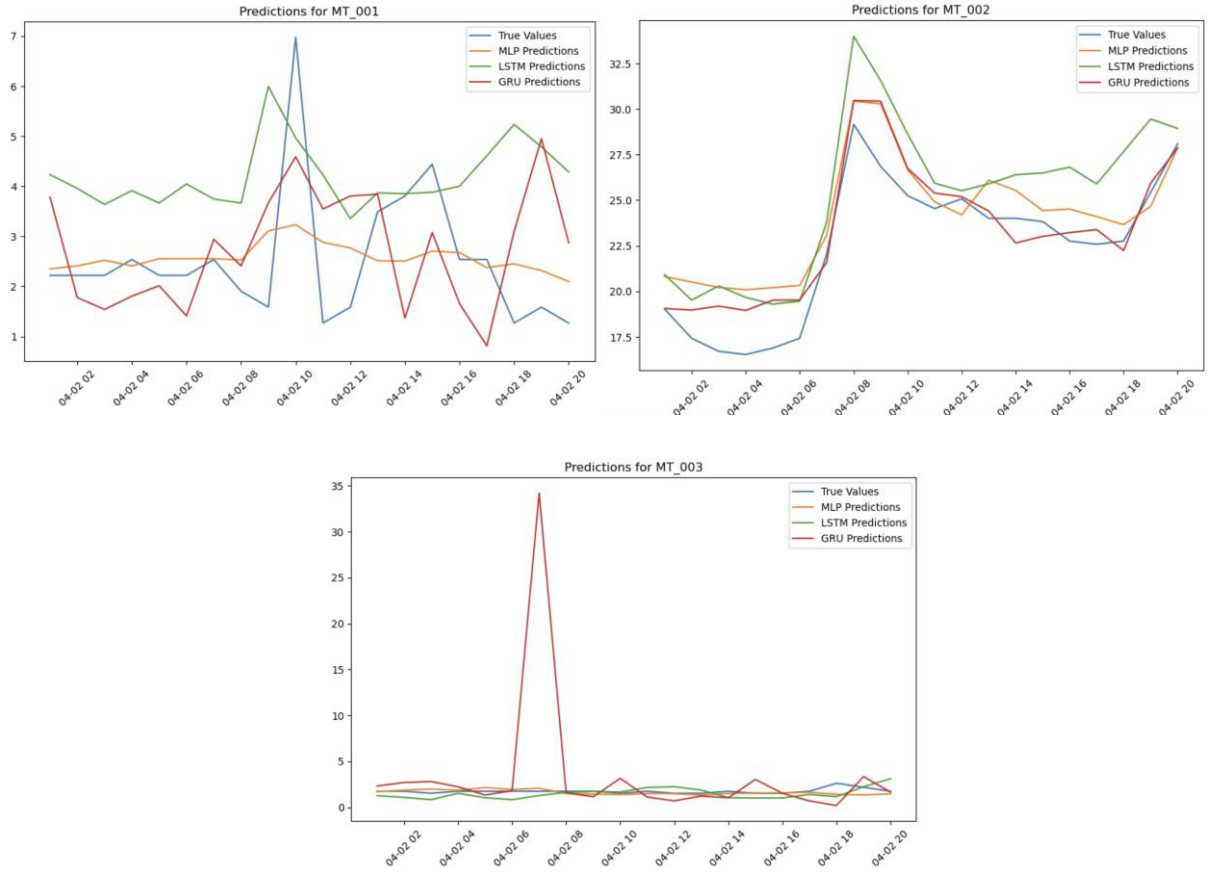

Figure 6. Comparison of MLP, LSTM, and GRU Predictions vs. True Values for Users MT_001, MT_002, and MT_003

We chose the top 20 values to do visualization in order for simplicity.

From the three pictures in Figure 6 above, we could find that the model of LSTM and GLU express stronger capability of response, which could capture the trend of the fluctuation of true value, especially periods of significant fluctuation such as the peak at 8:00 a.m. in the picture of MT_002. Their predictions are relatively close to each other, although there are occasional overpredictions or underpredictions. In contrast, the MLP model produces much smoother predictions with minimal variation, which works well when the true values fluctuate smoothly (e.g., MT_003). However, in cases with greater fluctuation (e.g., MT_001 and MT_002), MLP's predictions differ with true values. Additionally, LSTM tends to exaggerate some of the fluctuations more than GRU, while GLU shows the unstable prediction results especially in MT_003, but both models perform better than MLP in capturing complex time series patterns.

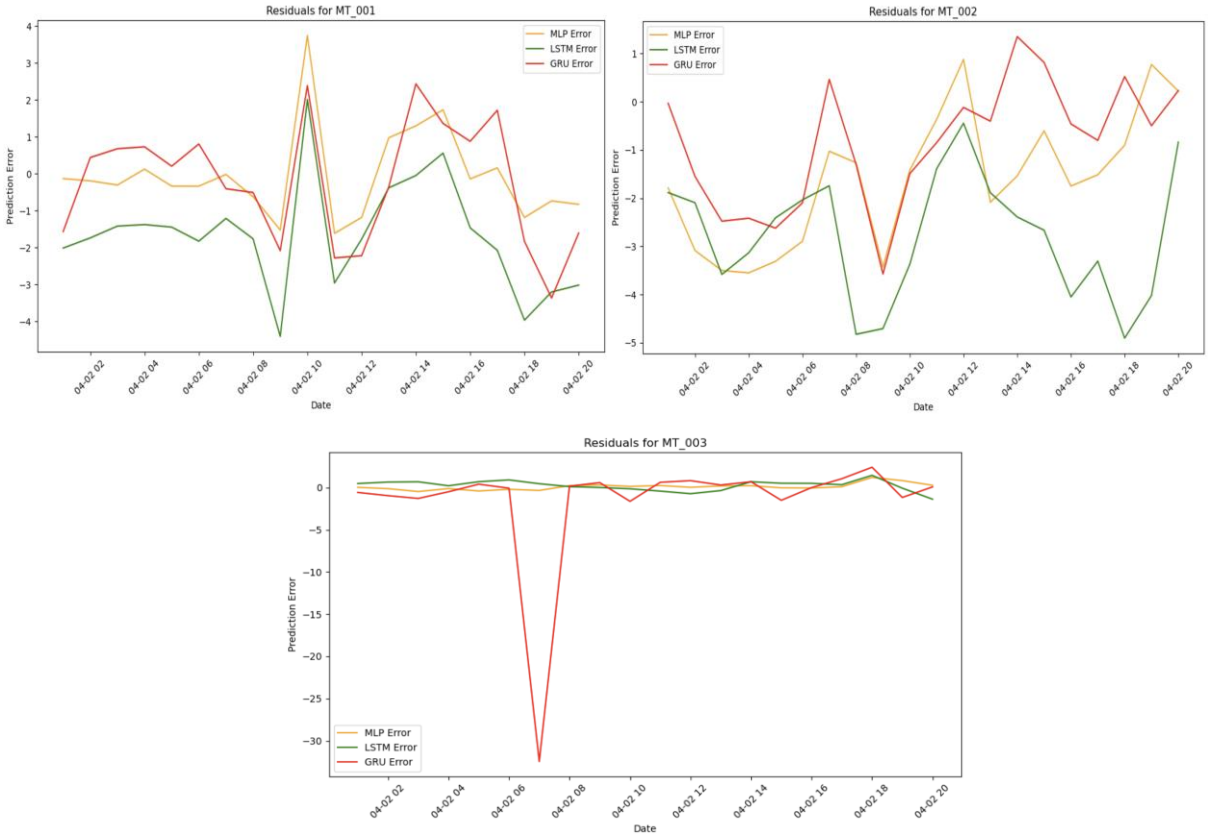

Figure 7. Residuals for Users MT_001, MT_002, and MT_003

From the three residuals pictures in Figure 7 above, we can observe that for MT_001 and MT_002, the LSTM and GRU models show similar error patterns, while the MLP model generally has smaller and more stable errors with less fluctuation. However, in the MT_003 plot, the GRU model shows a significant fluctuation. Overall, the LSTM and GRU models express close performance in terms of error for most of the time, while the MLP model is relatively stable and provides lower prediction.

Above all, we can summarize the relevant regularities as follows:
When the user's electricity consumption data is relatively stable and exhibits minor fluctuations, MLP is better suited to adapt to the data, while LSTM and GRU tend to overpredict. However, when the electricity usage data has more significant fluctuations and peaks, LSTM and GRU, with their more complex architectures, are more sensitive in capturing these peaks, whereas MLP is less responsive to such values. From a time perspective, MLP's simpler architecture allows it to achieve the desired results with less time and greater efficiency. From the perspective of the dataset, users' electricity consumption generally follows a cyclical pattern, which means that daily variations are not substantial, and electricity consumption in fixed time periods doesn't fluctuate drastically. For example, when users are away for work, household electricity usage is lower and more stable, whereas consumption increases when they are at home. It benefits MLP in training and prediction. In terms of performance metrics, MLP shows significantly better average MSE, training time, and prediction time compared to LSTM and GRU. Therefore, we conclude that MLP can be considered the optimal model to deal with this dataset.

Conclusion

By comparing the MLP, LSTM and GRU models, it is found that the MLP model has the best performance in the electricity consumption prediction task, the lowest mean square error and the fastest training and prediction time. However, LSTM and GRU show better wave-catching capabilities when dealing with users with high data volatility or strong cyclical changes. Therefore, in future studies, we can further explore how to combine the advantages of MLP and LSTM/GRU to better capture time series features and improve prediction accuracy in more complex electricity consumption scenarios. In addition, one of the limitations of this experiment is that although all consumer electricity consumption data is used for modeling and model training, only the data of MT_001-MT_003 is used for evaluation and prediction, and the experimental results may have certain limitations. In the future, we can consider increasing the user range of validation and prediction in order to better evaluate the model.

References

[1] W.-C. Hong, 'Electric load forecasting by support vector model', *Applied Mathematical Modelling*, vol. 33, no. 5, pp. 2444–2454, May 2009, doi: 10.1016/j.apm.2008.07.010.
[2] A. Trindade. "ElectricityLoadDiagrams20112014," UCI Machine Learning Repository, 2015. [Online]. Available: https://doi.org/10.24432/C58C86.
[3] G. Zhang, B. E. Patuwo, and M. Y. Hu, 'Forecasting with artificial neural networks:: The state of the art', *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998, doi: https://doi.org/10.1016/S0169-2070(97)00044-7.
[4] Y. Bengio, P. Simard, and P. Frasconi, 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.
[5] V. I. Kontopoulou, A. D. Panagopoulos, I. Kakkos, and G. K. Matsopoulos, 'A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks', *Future Internet*, vol. 15, no. 8, Art. no. 8, Aug. 2023, doi: 10.3390/fi15080255.
[6] A. Graves, 'Generating Sequences With Recurrent Neural Networks', Jun. 05, 2014, *arXiv*: arXiv:1308.0850. Accessed: Oct. 11, 2024. [Online]. Available: http://arxiv.org/abs/1308.0850.
[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling', Dec. 11, 2014, *arXiv*: arXiv:1412.3555. Accessed: Oct. 12, 2024. [Online]. Available: http://arxiv.org/abs/1412.3555.

Appendix

The libraries installed and imported：

```
import numpy as np
import pandas as pd
import tensorflow as tf
import random
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import TimeSeriesSplit
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, GRU, Dropout, Input
from scikeras.wrappers import KerasRegressor
import time
```

Environment to run code: Jupyter Notebook
Version of Python: 3.11.7
Version of numpy: 1.26.4
Version of pandas: 2.1.4
Version of tensorflow: 2.17.0
Version of matplotlib: 3.8.0
Version of seaborn: 0.12.2
Version of scikit-learn: 1.2.2