OPEN ENDED LAB

FORECASTING NON-SHIFTABLE LOAD USING MACHINE LEARNING ON THE CITYLEARN DATASET

SUBMITTED BY :

SHAHID

UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR,
JALOZAI CAMPUS
DEPARTMENT OF ELECTRICAL ENGINEERING
SUBJECT: MACHINE LEARNING
OPEN ENDED LAB THEORY REPORT

SUBMITTED TO :

ENGR.IRSHAD ULLHA

DATE OF SUBMISSION: 15 JON 2025

Contents

1

INTRODUCTION

The global transition toward smarter, more sustainable energy systems has amplified the importance of accurate load forecasting in electrical grids. As electricity demand patterns become increasingly dynamic due to the integration of renewable energy sources, electric vehicles, and decentralized production, utilities are under greater pressure to ensure efficient, stable, and responsive energy distribution. A critical component in this complex landscape is the ability to predict non-shiftable electrical loadsâthose essential and inflexible demands that cannot be deferred or rescheduled without compromising user needs or safety.

Non-shiftable loads, such as lighting, heating, refrigeration, and life-support equipment, represent a foundational portion of daily energy usage. These loads are consistent and time-bound, making their accurate prediction vital for real-time balancing of supply and demand, preventing overloading, and optimizing energy resource allocation. While conventional forecasting methods have relied heavily on statistical models and historical averaging, they often fall short in capturing the complex and nonlinear relationships present in modern energy consumption patterns, especially when influenced by external variables such as weather conditions, market pricing, and human behavior.

The advancement of machine learning (ML) offers a transformative opportunity to improve the accuracy and adaptability of load forecasting systems. ML modelsâespecially those designed for time-series dataâcan learn intricate dependencies within sequential data and incorporate a wide range of contextual features to make future-aware predictions. In recent years, deep learning architectures such as Long Short-Term Memory (LSTM) networks have gained prominence due to their capacity to retain long-term temporal dependencies, making them particularly effective for forecasting tasks in dynamic environments.

In this context, the application of machine learning to forecast non-shiftable load presents both a promising solution and a significant challenge. It requires not only robust model selection but also careful data preprocessing, thoughtful feature engineering, and the ability to handle forecasted inputs without introducing data leakage. Moreover, given the operational implications of under- or overestimating non-shiftable demand, the reliability and interpretability of the forecasting model become equally critical.

This report explores a machine learning-based approach to this forecasting problem within a realistic, smart-grid-inspired setting. By focusing specifically on non-shiftable load, the project addresses a core need in energy management systemsâensuring that essential consumption is met predictably and efficiently. The integration of artificial intelligence into this domain reflects a broader shift toward data-driven decision-making in energy infrastructure and contributes to the larger goal of achieving resilient, low-carbon power systems in the face of growing global demand.

2 OBJECTIVES

The primary objective of this project is to develop an effective and reliable machine learning model capable of forecasting non-shiftable electrical load in a smart grid environment. By leveraging time-series data and contextual features such as weather conditions, electricity pricing, and carbon intensity, the model aims to enhance the predictability of essential, inflexible energy demand across urban buildings. The key objectives of this project are as follows:

- To investigate the characteristics of non-shiftable load and its role in modern energy systems.
- To explore and preprocess the CityLearn dataset, focusing on features relevant to non-shiftable load forecasting.
- To engineer temporal, environmental, and economic features that contribute significantly to the accuracy of load prediction models.
- To design and implement a deep learning-based forecasting model.
- To evaluate the performance of the model using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
- To identify the impact of predicted input variables (e.g., forecasted weather or pricing data) on model performance and forecasting robustness.
- To highlight challenges such as data leakage and dependency on the quality of forecasted inputs, and propose mitigation strategies.

These objectives align with the broader goal of integrating artificial intelligence into smart grid operations, improving energy efficiency, and supporting future-ready infrastructure in urban energy systems.

3 DATA OVERVIEW

The data used in this project is sourced from the CityLearn Challenge 2022 repository (Phase-All dataset) and consists of four distinct datasets: Building 2, Carbon Intensity, Pricing, and Weather. These datasets collectively provide a rich set of temporal, environmental, economic, and operational features essential for accurate forecasting of non-shiftable electrical load in smart buildings.

3.1 BUILDING 2 DATASET

This dataset contains detailed building-specific measurements and contextual features, including:

- Temporal variables: month, hour, day type (e.g., weekday/weekend), daylight savings status
- Indoor environmental conditions: indoor dry bulb temperature, indoor relative humidity
- Comfort-related features: average unmet cooling setpoint difference
- Energy demands: non-shiftable load (target variable), domestic hot water (DHW) demand, cooling demand, heating demand
- Renewable generation: solar generation data

3.2 CARBON INTENSITY DATASET

Contains time-series data for carbon intensity, representing the carbon emissions associated with electricity generation over time. This helps to understand the environmental impact linked to energy consumption.

3.3 PRICING DATASET

Includes electricity pricing data with both actual prices and short-term predictions, capturing economic signals that may influence consumption behavior. Variables include:

- Actual electricity pricing
- Predicted electricity pricing for the next 1, 2, and 3 time steps ahead

3.4 WEATHER DATASET

Contains observed and forecasted weather parameters influencing building energy consumption:

- Observed features: outdoor dry bulb temperature, outdoor relative humidity, diffuse solar irradiance, direct solar irradiance
- Forecasted features: predicted values for the above variables for 1, 2, and 3 time steps ahead

All datasets are time-synchronized to ensure consistency across features and enable the development of robust forecasting models. Preprocessing steps such as missing data imputation, normalization, and temporal feature engineering were applied to enhance model performance.

The integration of these datasets allows the model to capture complex interactions between building operations, environmental conditions, economic factors, and carbon emissions, facilitating more accurate and reliable forecasting of non-shiftable electrical load in urban smart grids.

4 DATA PREPROCESSING

This section outlines the preprocessing steps applied to prepare the raw datasets for model development. These steps ensured data consistency, minimized noise, and enhanced the predictive potential of input features.

The process began by loading and merging the four key datasets â **Building 2**, **Weather**, **Carbon Intensity**, and **Pricing** â using a common time index. This integration created a unified dataset encompassing temporal, environmental, and economic variables.

To reduce redundancy and focus on relevant predictors, the following non-essential columns were removed:

- `indoor_dry_bulb_temperature`
- `average_unmet_cooling_setpoint_difference`
- `indoor_relative_humidity`

Subsequently, **outliers** were detected and handled across key numerical features to ensure robust model performance. The columns were **reordered** to logically group features, enhancing readability and consistency during modeling.

Missing values were identified and addressed using appropriate imputation strategies to preserve data quality.

To capture the cyclical nature of temporal features such as hour and month, **cyclical encoding** was applied using sine and cosine transformations. This enabled the model to better understand periodic patterns in the data.

A **correlation matrix** was computed to analyze inter-feature relationships and support feature selection by identifying strongly correlated or irrelevant variables.

The final dataset was partitioned into three sets using a 70:20:10 split ratio:

- Training Set: (876, 26)
- Validation Set: (1752, 26)

- Test Set: (876, 26)

Datetime components were extracted where applicable to enhance temporal context.

For feature scaling, two normalization techniques were applied:

- **StandardScaler** was used for weather-related continuous variables.
- **MinMaxScaler** was applied to economic and target variables.

The scaled data was then saved as three separate sets â training, validation, and testing â for use during model training and evaluation. After model predictions, the target variable `non_shiftable_load` was **inverse transformed** to return results to their original scale, ensuring meaningful interpretation.

5 MODEL ARCHITECTURE

This section outlines the architecture and implementation of the deep learning model developed for forecasting non-shiftable load using the CityLearn dataset.

The modeling process begins by loading the preprocessed training, validation, and test datasets, each containing normalized values. Using a sliding window approach, the time series data is reshaped into sequences of 24 time steps to predict the next step (1-step ahead). This captures temporal patterns crucial for accurate load forecasting.

5.1 DATA PREPARATION AND SEQUENCING

The datasets are loaded from saved CSV files containing normalized data. A sliding window method is used to generate sequences of length 24 time steps, which are used as model inputs. The target for prediction is the immediate next step following each sequence, enabling the model to learn temporal dependencies.

5.2 MODEL STRUCTURE

The model architecture is based on a multi-layer Long Short-Term Memory (LSTM) network designed to handle sequential time-series data efficiently.

5.2.1 *Model Layers*

Layer	**Configuration**
Input Layer	Sequence length = 24, Feature dimension = 29
LSTM Layers	3 layers, 60 hidden units each, dropout = 0.2
Layer Normalization	Applied after LSTM output
Fully Connected (Dense)	Linear layer with 1 output

Table 1: Summary of model architecture layers

listings xcolor

5.2.2 *Model Code*

The following Python code defines the LSTM-based neural network architecture used for forecasting non-shiftable load. The model consists of a three-layer LSTM with dropout for regularization, followed by layer normalization and a fully connected layer for output. The weights are initialized using Xavier initialization for improved training stability.

```python
import torch
import torch.nn as nn

class MOSTLY_CITED(nn.Module):
    def __init__(self):
        super().__init__()
        self.device = ('cuda' if torch.cuda.is_available() else 'cpu')
        self.lstm = nn.LSTM(29, 60, 3, batch_first=True, dropout=0.2).to(
            self.device)  # Added dropout
        self.layer_norm = nn.LayerNorm(60).to(self.device)  # Added
            LayerNorm
        self.fc = nn.Linear(60, 1).to(self.device)

        for name, param in self.lstm.named_parameters():
            if 'weight_ih' in name or 'weight_hh' in name:
                torch.nn.init.xavier_normal_(param.data)
            elif 'bias' in name:
                nn.init.zeros_(param)

    def forward(self, x):
        h0 = torch.zeros(3, x.size(0), 60).to(self.device)
        c0 = torch.zeros(3, x.size(0), 60).to(self.device)
        out, _ = self.lstm(x, (h0, c0))
        out = self.layer_norm(out[:, -1, :])  # Apply LayerNorm
        out = self.fc(out)  # Removed sigmoid activation
        return out
```

Listing 1: PyTorch LSTM Model Class: `MOSTLY_CITED`

5.3 TRAINING SETUP

5.3.1 *Loss Function and Optimization*

The model is trained using Mean Squared Error (MSE) loss function optimized with the Adam optimizer. A learning rate scheduler reduces the learning rate if the validation loss plateaus to fine-tune training.

5.3.2 *Training Configuration*

Component	**Setting / Details**
Loss Function	Mean Squared Error (MSE)
Optimizer	Adam
Learning Rate	0.001
Scheduler	ReduceLROnPlateau (factor=0.5, patience=10)
Forecasting Type	Univariate, 1-step ahead
Input Shape	(batch size, 24, 29)
Target Variable	Non-shiftable load

Table 2: Training and optimization details

This architecture effectively captures temporal dependencies in energy consumption data, while techniques such as dropout and layer normalization improve model robustness and generalization.

5.3.3 *Training Procedure*

The training process was initialized with a starting epoch of 1 and set to run for 50 epochs. The model was trained using mini-batches with a batch size of 32 to efficiently manage memory and improve training speed.

Data loaders were created for the training, validation, and test datasets to facilitate batch-wise data feeding into the model.

Mean Squared Error (MSE) loss was used as the primary criterion for optimization, while Mean Absolute Error (MAE) was also prepared for evaluation.

A utility was implemented to save the best model based on validation performance during training. Additionally, training and validation loss curves were plotted to monitor the modelâs learning progress.

The total training time was recorded to understand the computational cost.

6 EVALUATION AND RESULTS

This section presents the evaluation of the trained model on the test dataset using standard performance metrics. The purpose is to quantify how well the model predicts the target variable compared to the actual observed values.

6.1 EVALUATION METRICS

To assess the model performance, three widely used error metrics were calculated:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values. It heavily penalizes large errors, providing a strong indication of overall prediction accuracy.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, giving an error measure in the same units as the target variable. It helps to interpret the magnitude of prediction errors.
- **Mean Absolute Percentage Error (MAPE)**: Expresses the prediction error as a percentage of the actual values, providing an intuitive measure of relative error.

The evaluation results are summarized in Table 3 below.

Table 3: Model Evaluation Metrics

Metric	**Value**
Mean Squared Error (MSE)	1,401,531.84
Root Mean Squared Error (RMSE)	1,183.86
Mean Absolute Percentage Error (MAPE)	2.98 %

7 FUTURE WORK AND LIMITATIONS

7.1 FUTURE WORK

There are several avenues to extend and improve the current model. First, integrating attention mechanisms such as self-attention or Transformer-based encoders may further enhance temporal dependency modeling, especially for long sequences. Additionally, exploring ensemble models or hybrid architectures that combine CNNs or GRUs with LSTM can be promising for improving forecasting accuracy.

Future work may also involve deploying the trained model in a real-time environment using IoT-based systems. This would enable dynamic load forecasting and timely energy distribution in smart grids. Moreover, using more granular datasets (hourly or 15-min intervals) and additional external factors like temperature, weather, and socio-economic variables could significantly improve prediction performance.

7.2 LIMITATIONS

Despite the promising results, the proposed model has certain limitations. Firstly, the model performance heavily depends on the quality and quantity of input features; insufficient or noisy data may lead to poor generalization. Secondly, although the LSTM model captures sequential dependencies well, it still struggles with long-range dependencies when compared to attention-based models.

Another limitation is computational cost. The training time on standard hardware (CPU/GPU) is relatively high, and hyperparameter tuning further increases the training duration. Lastly, the model was trained and evaluated on historical data only, without incorporating real-time changes in grid behavior or unexpected disruptions, which might limit its robustness in production environments.

8 CONCLUSION

This project focused on forecasting non-shiftable load using machine learning techniques applied to the CityLearn dataset. A deep learning-based approach was developed with a multi-layer LSTM architecture, enhanced by Layer Normalization and Xavier initialization, to effectively capture the temporal dependencies inherent in non-shiftable energy consumption patterns.

The dataset was carefully preprocessed, and the model was trained over 50 epochs using MSE as the primary loss function. Performance evaluation using MSE, RMSE, and MAPE indicated that the proposed model achieved high accuracy in predicting future non-shiftable loads.

The experimental results validate the effectiveness of LSTM-based architectures in handling sequential energy data, particularly for non-flexible, demand-side load components. The structured pipelineâfrom data loading to model evaluationâensures the approach is reproducible and scalable.

This work contributes to the growing field of energy demand forecasting by offering a reliable solution for non-shiftable load prediction, which plays a crucial role in planning, resource allocation, and demand-side energy management strategies.

9 REFERENCES

Vazquez-Canteli, J. R., & Nagy, Z. (2019). *Reinforcement learning for demand response: A review of algorithms and modeling techniques*. Applied Energy, 235, 1072â1089. https://doi.org/10.1016/j.apenergy.2018.11.002

CityLearn GitHub Repository. (2023). *CityLearn: An Open AI Gym Environment for the Implementation of Demand Response Using Reinforcement Learning for Building Energy Management*. Available at: https://github.com/intelligent-environments-lab/CityLearn

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation, 9(8), 1735â1780. https://doi.org/10.1162/neco.1997.9.8.1735

Armstrong, J. S. (1985). *Long-Range Forecasting: From Crystal Ball to Computer*. Wiley-Interscience.

Taieb, S. B., & Hyndman, R. J. (2014). *A gradient boosting approach to the Kaggle load forecasting competition*. International Journal of Forecasting, 30(2), 382â394. https://doi.org/10.1016/j.ijforecast.2013.07.005

Marino, D. L., Amarasinghe, K., & Manic, M. (2016). *Building Energy Load Forecasting using Deep Neural Networks*. 2016 IEEE IECON. https://doi.org/10.1109/IECON.2016.7793247