Chapter 3

Short-Term Energy Forecasting in Smart Buildings

Efficient energy management is required to achieve optimal energy use in the building. The building sector utilizes 40% of worldwide energy production and is projected to reach 50% by 2050 [100]. With rising electricity prices, buildings require cost-effective and efficient energy management. Researchers in smart building management are exploring the use of AI and IoT to forecast energy usage [101]. This chapter offers a hybrid deep-learning model using CNN and RNN to forecast hourly energy usage in smart buildings. Experimental results demonstrate that the CNN-GRU model, with an accuracy of 97%, outperforms the state-of-the-art techniques.

3.1 Introduction

According to [102], energy efficiency refers to using energy in the most resource-efficient manner. The inefficient use of technological devices results in energy waste. Buildings currently account for 40% of global energy usage [102]. Lighting, air conditioning, fans, and other electronic devices increase a building's energy consumption[103]. As a result, buildings in these places produce more carbon than transportation. Predicting

a building's energy usage is crucial for efficient energy management and reducing the carbon footprint, which is now achievable with smart buildings [102]. The goal of smart building is to reduce energy consumption without compromising user comfort [104]. In 1984, researcher found that residential buildings in the United Kingdom lost 10 million pounds of energy per year owing to inaccurate energy predictions [105]. Between 2018 and 2050, buildings are expected to consume more than 65% of total energy [106]. Carbon Dioxide (CO_2) emissions are projected to reach 1.3 billion metric tonnes by 2030, posing a significant threat to the environment and public health [106]. Thus, reliable load forecasting is required for smart buildings. Smart buildings are becoming increasingly popular as the smart city sector grows. Precise energy forecasting is crucial for implementing smart building concepts [107].

A linear regression approach and statistical models like the Auto-Regressive Integrated Moving Average (ARIMA) are used for energy prediction [108]. Deep learning approaches are increasingly popular for anticipating energy demand in smart buildings. CNNs and RNNs are gaining popularity among deep learning methods for energy prediction. CNN is a deep neural network architecture in which the calculations are entirely based on convolutional operations. RNNs, recurrent neural networks, are the algorithms where the internal states of the RNNs find the dependent feature of cells concerning time [109]. This chapter proposes a strategy for smart building energy prediction using state-of-the-art techniques. The proposed model is compared to other machine learning, statistical, and deep learning models in terms of performance.

The major contributions of this chapter are as follows:

1. In this chapter, a novel hybrid deep learning-based CNN-RNN model has been proposed for forecasting the energy consumption of smart buildings.
2. The model is analyzed for short-term forecasting using two variants of RNN, i.e., LSTM and GRU.
3. The proposed model has been evaluated and compared on a publicly available

dataset with other alternative approaches in terms of Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 score.

3.2 Proposed Methodology

This chapter proposes an energy forecasting model for smart buildings. The proposed model combines two deep learning architectures: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNN extracts hidden characteristics from raw data using expanded convolutions and filters. On the other hand, the RNN assists in capturing sequential aspects of data and predicting comparable patterns in sequences, such as time series data. The proposed model integrates features from both CNN and RNN to provide an optimal prediction.

RNNs struggle with vanishing gradients, making model learning problematic since it fails to capture longer sequence dependencies. To address this issue, gating techniques are implemented in Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). LSTMs are a special type of RNN that can learn long-term associations between data points. The LSTM design contains three gates: input (i_t), forget (f_t), and output (o_t), allowing them to selectively learn, unlearn, and retain information from each unit over time. Furthermore, LSTM retains its state information, known as cell state (c_t). Figure 3.1. depicts the block diagram of the LSTM cell. The mathematical representation of input gate (i_t), forget gate (f_t) and output gate (o_t) of LSTM are as follows:

$$i_t = \sigma\left(v_i w_{i,m1} + h_{t-1} w_{i,m2} + b_i\right) \tag{3.1}$$

$$f_t = \sigma\left(v_i w_{f,m1} + h_{t-1} w_{f,m2} + b_f\right) \tag{3.2}$$

$$o_t = \sigma\left(v_i w_{o,m1} + h_{t-1} w_{o,m2} + b_o\right) \tag{3.3}$$

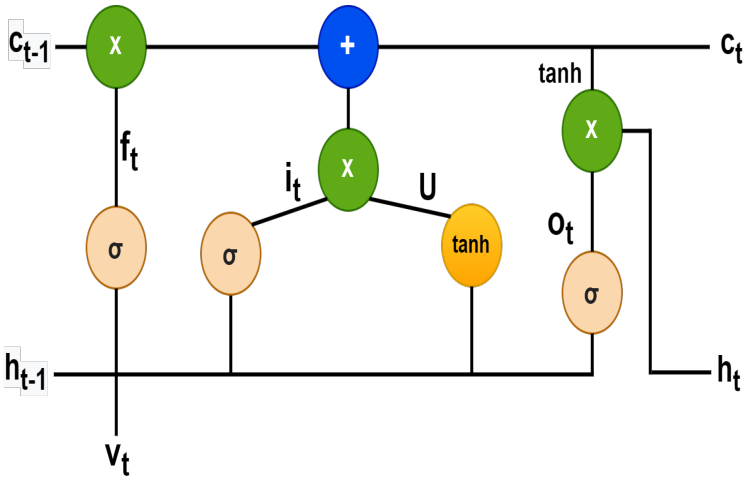


Figure 3.1: The block diagram of LSTM.

$$U = tanh(x_i w_{u,m1} + h_{t-1} w_{u,m2} + b_u) \tag{3.4}$$

$$c_t = f_t \times c_{t-1} + i_t \times U \tag{3.5}$$

$$h_t = o_t \times tanh(c_t) \tag{3.6}$$

where σ is the sigmoid activation function. The v_i is the input vector. $w_{i,m1}$, $w_{i,m2}$, $w_{f,m1}$, $w_{f,m2}$, $w_{o,m1}$, $w_{o,m2}$ are weight matrices for corresponding input, output and forget gates. v_i is the current input vector, and h_{t-1} is the previous hidden state. Bias for corresponding input, forget, and output gate are represented by b_i, b_f, b_o, and the U is an update signal. The c_t is the state value concerning time t, and h_t is the LSTM output cell. GRU consists of fewer gates than LSTMs since they only have one hidden state, and their straightforward architecture makes them easier to train. Figure 3.2 illustrates the overall structure of GRU. Compared to the LSTM cell, the GRU has two gates such as the reset and update gates. The mathematical model of GRU's reset and update gates are presented below.

$$z_t = \sigma(v_i w_{z,m1} + h_{t-1} w_{z,m2} + b_z) \tag{3.7}$$

$$r_t = \sigma(v_i w_{r,m1} + h_{t-1} w_{r,m2} + b_r) \tag{3.8}$$

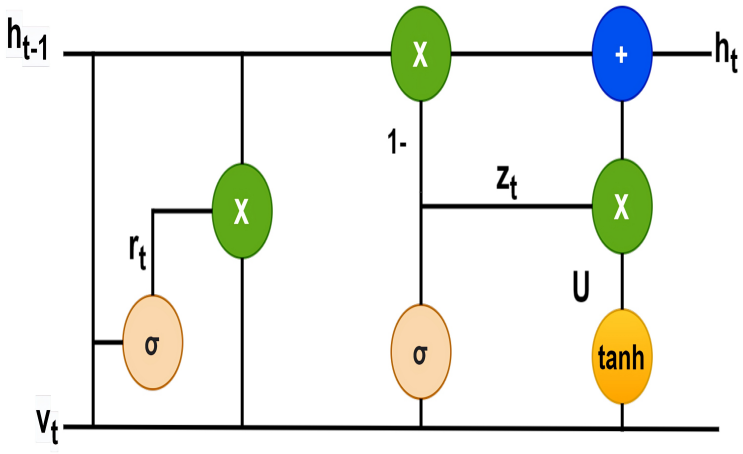


Figure 3.2: The block diagram of GRU.

$$U = tanh(v_i w_{u,m1} + [r_t \odot h_{t-1}]w_{u,m2} + b_u) \tag{3.9}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot U \tag{3.10}$$

where σ denotes the sigmoid activation function, v_i is the input vector. The $w_{z,m1}$, $w_{z,m1}$, $w_{z,m1}$, $w_{z,m1}$, $w_{z,m1}$ and $w_{z,m1}$ are weight matrices and b_z, b_r and b_u are bias vectors. h_t and U are output vectors and update signals. The GRU update gate is comparable to the LSTM's input and forget gate. Figure 3.3 depicts the architecture of the proposed CNN-RNN model. It begins with an input layer that receives a sequence of inputs, followed by a CNN layer that extracts local properties from the time series. Convolutions automatically extract hidden data properties in the time direction, making one-dimensional CNN (1D-CNN) useful for time series applications. The flatten layer converts multidimensional data to a single-dimensional vector, which is then sent to the RNN and dense layers. The CNN's extracted features and fed into the RNN architecture to assist the RNN models in determining the connections between the input and output sequences. This study combines CNN and RNN to build a robust model, and univariate and multivariate data are inputted into this model to observe the changes in prediction results. Mathematically, the CNN layer of the proposed CNN-RNN model can be expressed as follows:

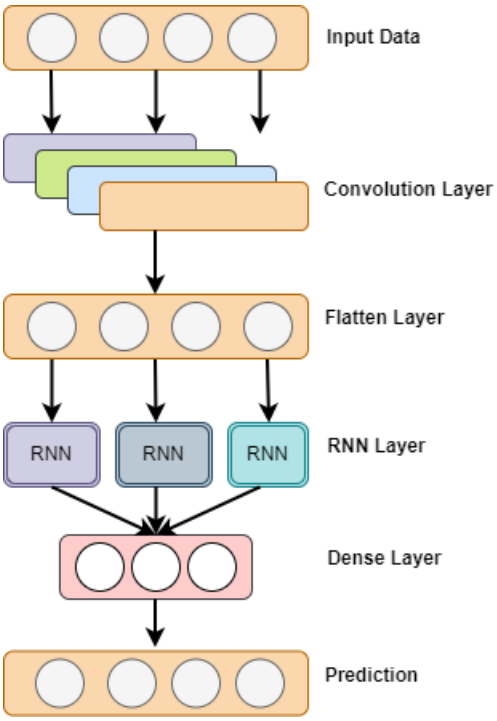


Figure 3.3: Proposed CNN-RNN Model.

$$x_p{}^l = b_p^l + \sum_{i=1}^{N_{l-1}} Conv1D(\alpha_{ip}^{l-1}, \beta_i^{l-1}) \tag{3.11}$$

where $x_p{}^l$ is input of pth neuron of layer l, b_p^l is bias of pth neuron of layer l, N is total neuron, α_{ip}^{l-1} is the kernel from the i^{th} neuron at layer l-1 and pth neuron at layer l, β_i^{l-1} is the output of the i^{th} neuron at layer l-1, Conv1D is Convolution neural network with zero padding.

This study suggests two variations of the CNN-RNN model such as CNN-LSTM and CNN-GRU. The CNN-LSTM model starts with 1D CNN layers and progresses to LSTM and dense layers. Defining 1D CNN layers first is beneficial since they learn quickly and extract high-level characteristics. These features are fed into the LSTM layer before predicting the next values. The fully connected layers use the output of the LSTM layer as input to create the final prediction. This proposed hybrid model makes use of filters and kernels. The filters determine how many times the input is processed,

while the kernel size determines how many time steps are taken. Both the filter and kernels are set to two. The LSTM layer contains 100 neurons and uses Relu as an activation function. GRU performs a similar operation. Finally, the results of energy forecasting are projected. Adam is an optimizer, while mean absolute error is the loss function. Once the predictions are made, several error metrics are calculated between the actual and predicted values to assess how effectively the model works. Figure 3.4 describes the model's entire operating procedure. The provided data is preprocessed, which is essentially a normalization step. The data is then separated into two sets such as training (90%) and testing (10%). The proposed model is applied to training data and compared with the testing data. Finally, it makes predictions and is evaluated on the test set using various performance metrics.

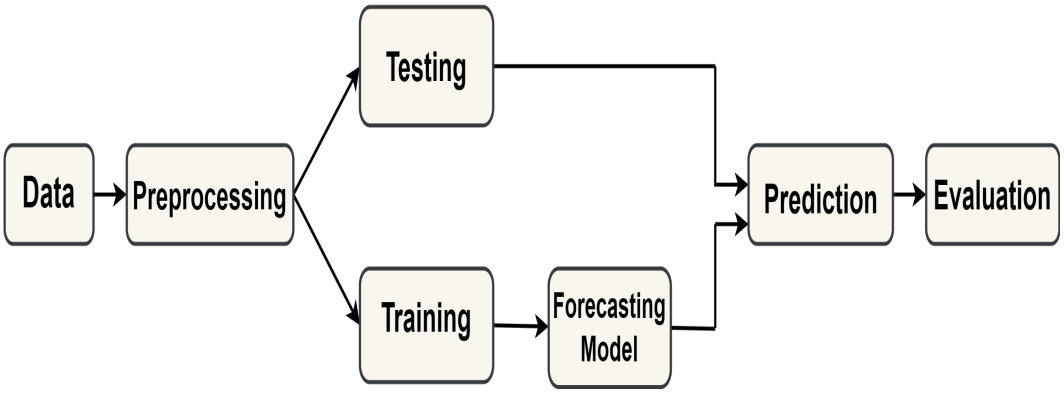


Figure 3.4: Workflow of data with models.

3.3 Experimental Setup

This section describes the datasets, data preparation, performance indicators, and alternative models compared with the proposed model. Machine Learning (ML) model Linear Regression (LR) is executed by fitting them using a variety of lags and four different types of input data. Input is generated at varying lags, and output is viewed and recorded. Statistical model ARIMA is used for the first data type, with lag measured in days for energy prediction.

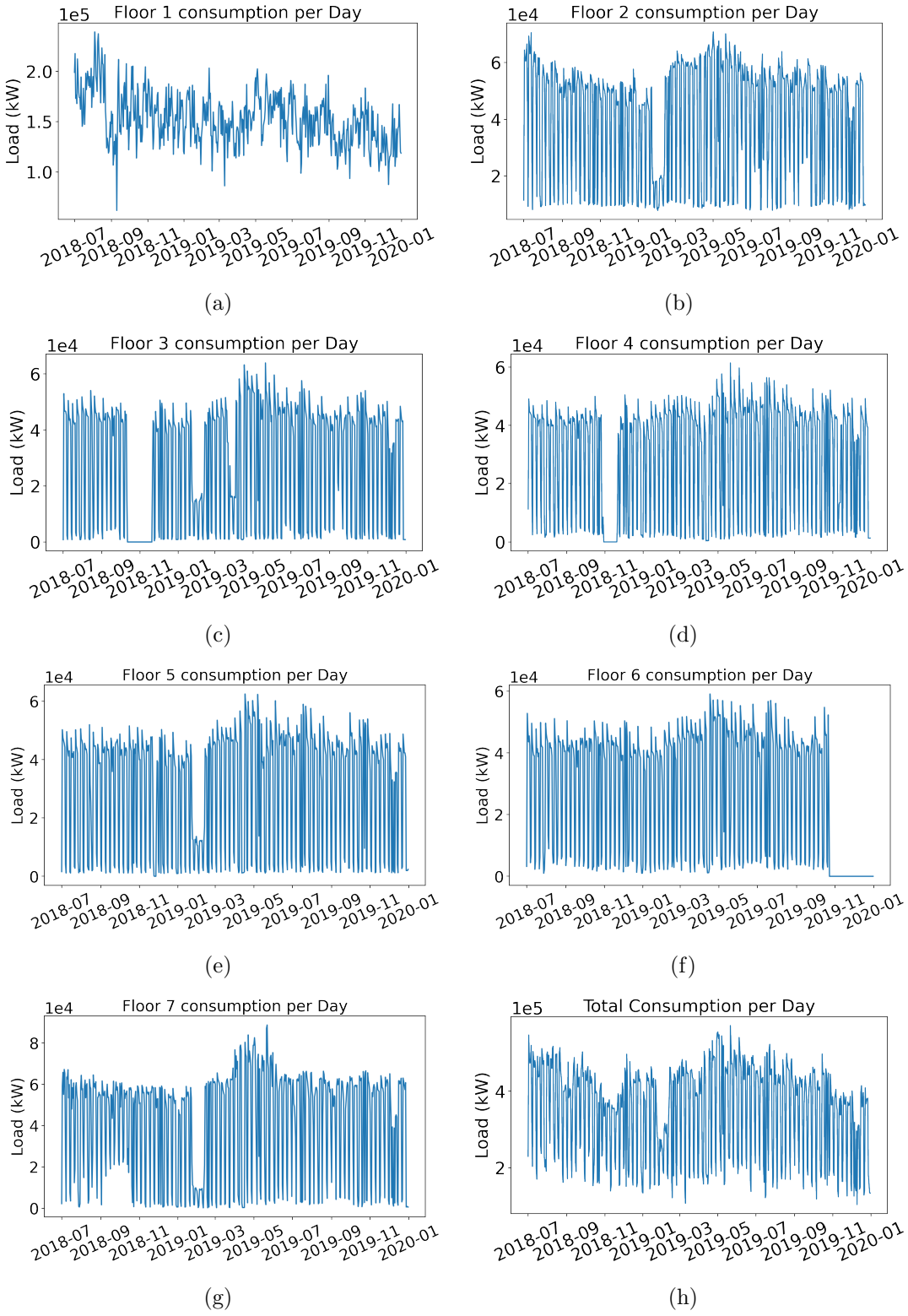
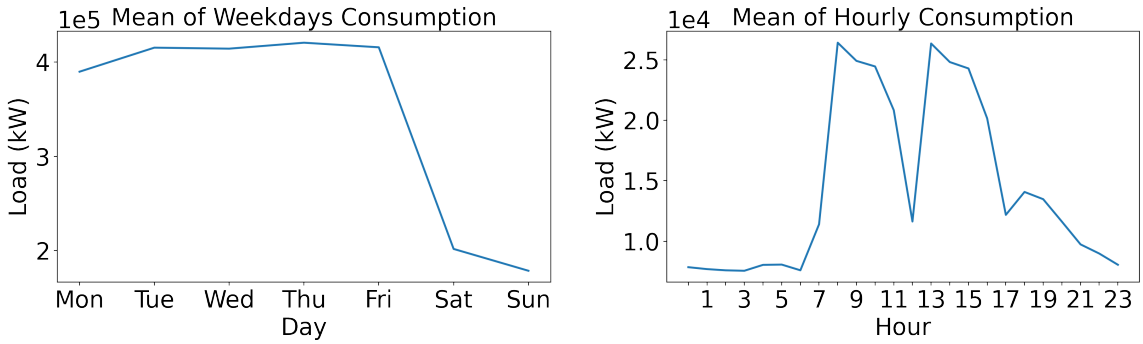


Figure 3.5: Floor wise and overall energy consumption of a building.



(a) (b)

Figure 3.6: Energy consumption in a day and hour wise.

3.3.1 Dataset Description

The data was collected by Chulalongkorn University-Building Energy Management Systems (CU-BEMS) [110], a seven-story academic office building in Bangkok, Thailand. The electricity usage data (kW) includes separate air conditioning systems, lights, and plug loads for the building's 33 zones. The first two floors each have four zones, and the next five floors have five zones each. The building has 24 multi-sensors, 21 Energy Monitoring Units (EMUs), 30 digital meters, and seven gateways. EMU measures the consumption of 1-2kW of wall-mounted AC units and lighting and plug load usage. The digital meter measures 20–40 kW of a large AC compressor. Multisensors measure temperature, humidity, and ambient light conditions. In this dataset (CU-BEMS), sensors were taken down for a maintenance period of 6 out of 18 months; therefore, the temperature couldn't used as a good and reliable feature. Each of the 2018 data files has one-minute interval data of 1,440 data points/day for 184 days from July 2018 to December 2019.

Figure 3.5(h) depicts the energy forecast for a smart building over an 18-month period. This is the data we utilized to test our models. So, the total energy consumption for CU-BEMS data for all floors combined is resampled by adding up each minute for that day. Let's look at floor-wise energy usage, which is resampled per day from per

minute data as shown in Figure 3.5(a-g). From there, it was discovered that there was a rapid fall in consumption at certain periods due to a lack of data. This makes the data preprocessing essential. Now it is seen that the pattern of weekly consumption is resampled by the mean per day consumption shown in Figure 3.6. There is a noticeable fall in consumption on weekends; thus, taking weekdays into account is a good idea, following the pattern of hourly consumption resampled by the mean per-hour consumption. Part b of Figure 3.6 shows a dramatic increase in consumption during working hours; thus, using the present hour as a feature will result in an accurate estimate.

3.3.2 Data Prepossessing

The data comes from the CU-BEMS dataset, which is preprocessed with imputation techniques to fill the null values. Z-score normalization is performed since energy usage ranges from a few kW to hundreds of kW per minute. The dataset is resampled for hourly consumption. Additional data is separated for training and testing purposes. For training, 16 months of data (about 11,500 data points) were collected and resampled to hours, while testing took two months (roughly 1400 data points). To ensure a high-quality dataset, cells with missing values should be preprocessed. The first phase in data processing is data cleaning, conducted on datasets containing missing values for some attributes. The next step in data preprocessing is data transformation. For better predictions, data from a specific range is required to train the models. However, the energy usage of the entire building ranges from a few kW to 700 kW. As a result, data transformation requires standardization. The mean and variance of the attribute (energy usage) are standardized by scaling the data by Equation 3.12.

$$\bar{X} = \frac{x - \mu}{\sigma} \tag{3.12}$$

where x represents the original value, $\bar{X}$ denotes the value after normalization, μ is the mean, and σ is the standard deviation. These normalize every value in the dataset

such that the mean and the standard deviation become 0 and 1, respectively. Once the feature scaling is performed, the dataset can be fed into the model.

3.3.3 Performance Metrics

Different types of machine learning, deep learning, and statistical models are used for energy forecasting in smart buildings. Three metrics have been used to compare their performance, which are listed below. (Let $\hat{y}_i$ and y_i represent the predicted and true values of the i^{th} sample in the dataset, and n is the total number of data points utilized in the experiment).

1. Mean Absolute Error (MAE): It is the absolute difference between actual values and the values predicted by the model.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.13}$$

2. Root Mean Square Error (RMSE): RMSE is used to measure data deviation in which the square root of the mean of the squared errors is calculated.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{yi}\right)^2}{n}} \tag{3.14}$$

3. R^2 Score: It shows how well the data fit the regression model.

$$R^2 = 1 - \frac{\sum\left(y_i - \hat{yi}\right)^2}{\sum\left(y_i - \bar{y}\right)^2} \tag{3.15}$$

   where $\bar{y}$ is the mean of predicted values.

The following attributes were used in the prediction process:

- **Energy Usage:** This is given in terms of kW per minute, which is resampled to per hour and day by addition as per the lag requirement.
- **Hour of the day:** As the electricity consumption of a building is different

throughout the day, one of the essential aspects for getting accurate energy prediction is considering the current hour of the day. Hence, for 12 am, it is considered as 0, and so on up to 23 for 11 pm.

- **Day type:** Day type (from Monday to Sunday) is also helpful for predicting energy because the building can consume less energy on the weekend. So, mapping each weekday with an integer is done by Monday as 0 to Sunday as 6.

Various models, such as Linear Regression (LR), ARIMA, LSTM, and GRU, are described in Table.3.1, which are compared with the proposed models, such as CNN-GRU and CNN-LSTM, with their parameters.

Table 3.1: Model Names and their Parameters

Model Name	**Parameters**
Linear Regression	Input shape: lag x 2 dimension
ARIMA	Lag p=7, difference d=0, Past Error q=7
LSTM	Nodes = 100, Activation function = ReLU, Loss= MAE , Optimizer = adam.
GRU	Nodes = 100, Activation function = ReLU, Loss=MAE Optimizer = adam
CNN-LSTM	CNN Filters = 2, Flatten Layer, One layer LSTM, Number of Nodes = 100 Activation function = ReLU, Dense, Loss=MAE, Optimizer = adam
CNN-GRU	CNN Filters = 2, Flatten Layer, Two layer GRU, Nodes = 100 Activation function = ReLU, Dense layer Loss=MAE, Optimizer = adam.

3.3.4 Comparing with other machine learning techniques

In this section, statistical learning and machine learning models such as LR and ARIMA are described and compared with the proposed hybrid CNN-RNN model.

- **Linear Regression (LR):**

  It is a statistical method for predicting one variable value based on another. The slope and intercept of X determine the output of linear regression h(X) as follows:.

$$h(X) = \alpha_0 + \alpha_1 X \tag{3.16}$$

  The linear regression technique minimizes the sum of the squares of differences

between the observed and predicted values. There are two types of linear regression models such as univariate and multivariate. Univariate indicates the value based on some of the past values, called lag, so whenever lag varies, results are noted to find the optimal lag. In multivariate data, past energy consumption values, temperature, weekday, or current hour are provided as input, and various lags and results are compared.

- **Auto Regressive Integrated Moving Average (ARIMA):** Statistical analysis models such as ARIMA are used for time series forecasting [108]. If a statistical model forecasts future values using data from the past, it is said to be autoregressive. Autoregressive models implicitly assume that the future will be similar to the past. In this study, the ARIMA model is observed by adjusting three hyperparameters, p, d, and q, where p represents the number of lag observations, which is eight. D is the number of times the raw observations are differenced to make the data stationary, equal to zero. Q is the moving average window size. The Q value is set to eight, demonstrating the relationship between data and previous errors. The ARIMA model is used for univariate data. The calculations used in ARIMA are as follows:

$$Y_t = c + \delta_t + \sum_{i=1}^{p} \varphi_i Y_{t-1} + \sum_{i=1}^{q} \Theta^i \delta_{t-i} \tag{3.17}$$

The p is the prior value of the variable in the linear combination of the autoregressive part, and q is the prior prediction error of the linear combination of the moving average.

3.4 Results and Discussion

The proposed model performs time series analysis to predict new data from previous data. In this study, univariate and multivariate data are subjected to time series anal-

ysis to extract features and model the nonlinear relationships present in the CU-BEMS dataset. The proposed CNN-RNN model predicts the hourly energy consumption of smart buildings and is compared to other machine learning, statistical, and deep learning models such as Linear Regression, ARIMA, LSTM, and GRU, as discussed in this section. The results of the experiments are performed on a computer node with 2.4GHz Intel-Xeon Skylake 6148 CPU cores and 192GB RAM. In this research, 90% of the data is used for training, while the remaining is used for testing. The original minute-by-minute data are resampled to hour-by-hour data, with various lags taken into account to produce the result. Different scenarios or types are considered for forecasting the required energy for smart buildings using the proposed hybrid CNN-RNN model.

The scenarios are discussed below: In scenario 1 (or type 1), the univariate model with NULL cells filled from the preceding cell is utilized, whereas scenario 2 (or type 2) uses the multivariate model with weekday/hour as a feature using the past data. Various combinations of lags are used in scenario 1 as an input. As they are univariate models, input is given as an array of past energy consumption based on lag, and the next value for energy consumption is predicted as output. In scenario 2 of the multivariate model, input is given as past consumption and the present weekday or hour of the day in a 2D array. So, scenario 2 of the multivariate model uses features to make more accurate predictions than scenario 1.

- **Scenario 1:** In this, it is observed that the model's lag value performs better in identifying when the error value becomes minimum. These are the results of scenario 1 univariate models, in which the data is preprocessed and only the past energy consumption is used for prediction. Table.3.2 shows that the lag values of all models, the MAE and RMSE of CNN-LSTM and CNN-GRU, have lower values than other models. Similarly, metrics like R^2 of CNN-LSTM and CNN-GRU in 8 hours are 91% and 88%, respectively. The proposed model's R^2 score gives better results than all other models in case of an 8-hour lag. The error

Table 3.2: Scenario 1 lag values of all models with error metrics

Error Metric	Models	Lag			
		2hr	4hr	**8hr**	10hr
MAE	Linear Regression	2769.82	2750.1	2262.46	2041.94
	ARIMA	2728.1	2381.43	1991.3	2164.17
	LSTM	2228.6	1914.2	1377.2	1404.6
	GRU	2409.36	1801.48	1165.67	1173.26
	CNN-LSTM	2391.7	2025.4	**1129.21**	1287.8
	CNN-GRU	2332.44	1758.21	1499.78	1271.39
RMSE	Linear Regression	5278.14	5282.98	4210.88	4025.59
	ARIMA	4963.81	4494.39	3562.87	3874.07
	LSTM	5138.76	4637.65	3361.66	3416.55
	GRU	4641.04	3625.87	2657.77	2721.22
	CNN-LSTM	4629.96	3653.8	**2395.86**	2882.23
	CNN-GRU	4418.02	4311.14	3214.45	3448.07
R^2	Linear Regression	0.64	0.64	0.77	0.79
	ARIMA	0.68	0.75	0.83	0.81
	LSTM	0.66	0.72	0.83	0.82
	GRU	0.72	0.82	0.86	0.9
	CNN-LSTM	0.75	0.76	**0.91**	0.82
	CNN-GRU	0.72	0.83	0.88	0.9

Table 3.3: Scenario 2 Lag values of all the models with error metrics

Error Metric	Models	Lag			
		2hr	4hr	**8hr**	10hr
MAE	Linear Regression	2756.71	2889.57	2590.78	1893.93
	ARIMA	2695.32	2965.15	2423.12	2137.84
	LSTM	1240.87	1089.21	1052.44	1542.84
	GRU	1059.74	989.16	968.21	931.47
	CNN-LSTM	2548.72	1364.64	977.66	1100.12
	CNN-GRU	2654.05	1489.21	**854.42**	961.64
RMSE	Linear Regression	5227.53	5176.77	3968.77	3024.49
	ARIMA	5163.51	5621.64	4504.69	3875.27
	LSTM	2208.39	1933.57	1671.06	2132.81
	GRU	1729.25	1648.65	1615.66	1511.42
	CNN-LSTM	4547.33	3331.28	1683.37	1841.73
	CNN-GRU	5252.64	3979.95	**1374.47**	1478.63
R^2	Linear Regression	0.65	0.65	0.8	0.88
	ARIMA	0.64	0.62	0.77	0.83
	LSTM	0.94	0.95	0.96	0.94
	GRU	0.92	0.93	0.96	0.94
	CNN-LSTM	0.73	0.86	0.93	0.97
	CNN-GRU	0.64	0.8	**0.97**	0.96

metrics of the scenario 1 model to visualize the comparison are shown in Figure 3.7(a-c). It is observed that the model performs better as the lag increases up to

a certain limit and reaches its optimal value of 8 hours lag. Also, by the values of error metrics, it can be inferred that CNN-LSTM performs better overall.

- **Scenario 2:** In this, it is observed that the model's lag value helps better identify when the error value becomes minimum. These are the results of scenario 2 multivariate models, in which data is preprocessed and historical energy use is combined with the current hour/weekday as an additional feature for prediction. Table.3.3 shows the entire results for scenario 2.

Table 3.3 shows the lag values of all models, as well as error measures such as MAE and RMSE for CNN-LSTM and CNN-GRU, are achieved with a decreasing trend in error compared to other models. Other measures, such as R^2 of CNN-LSTM and CNN-GRU for 8 hours lag, are 93% and 97%, respectively. The proposed model's R^2 score outperforms previous models at 8 and 10 hours lag. Tables 3.2 and 3.3 show that CNN-LSTM and CNN-GRU models have the lowest MAE and RMSE errors, as well as the highest R^2 accuracy. The error metrics of the scenario 2 model are shown in Figure 3.7(d-f). The model performs better as the lag grows from 2 to 10 hours, with 8 hours being the optimal lag. The values of error measures show that CNN-GRU outperforms all other models. The models perform better for a lag of 8 hours for the given CU-BEMS dataset, as shown in Tables 3.2 and 3.3. For this lag duration, the proposed CNN-RNN model performs better for univariate and multivariate data. For univariate data, CNN-LSTM gives optimal results with an MAE value of 1129.21 kW, RMSE value of 2395.86 kW, and R^2 Score of 0.91. For multivariate data where an hour of the day is considered as an extra feature along with the past energy consumption, CNN-GRU is giving optimal results for error metrics with MAE value as 854.42 kW, RMSE value as 1374.47 kW and R^2 Score as 0.97. Tables 3.2 and 3.3 show that the prediction error is decreasing in CNN-LSTM and CNN-GRU compared to other models for all lag duration from 2 to 10 hours. The actual versus predicted values of the proposed CNN-LSTM and CNN-GRU models give a minor error and high accuracy, as shown in Figure 3.7.

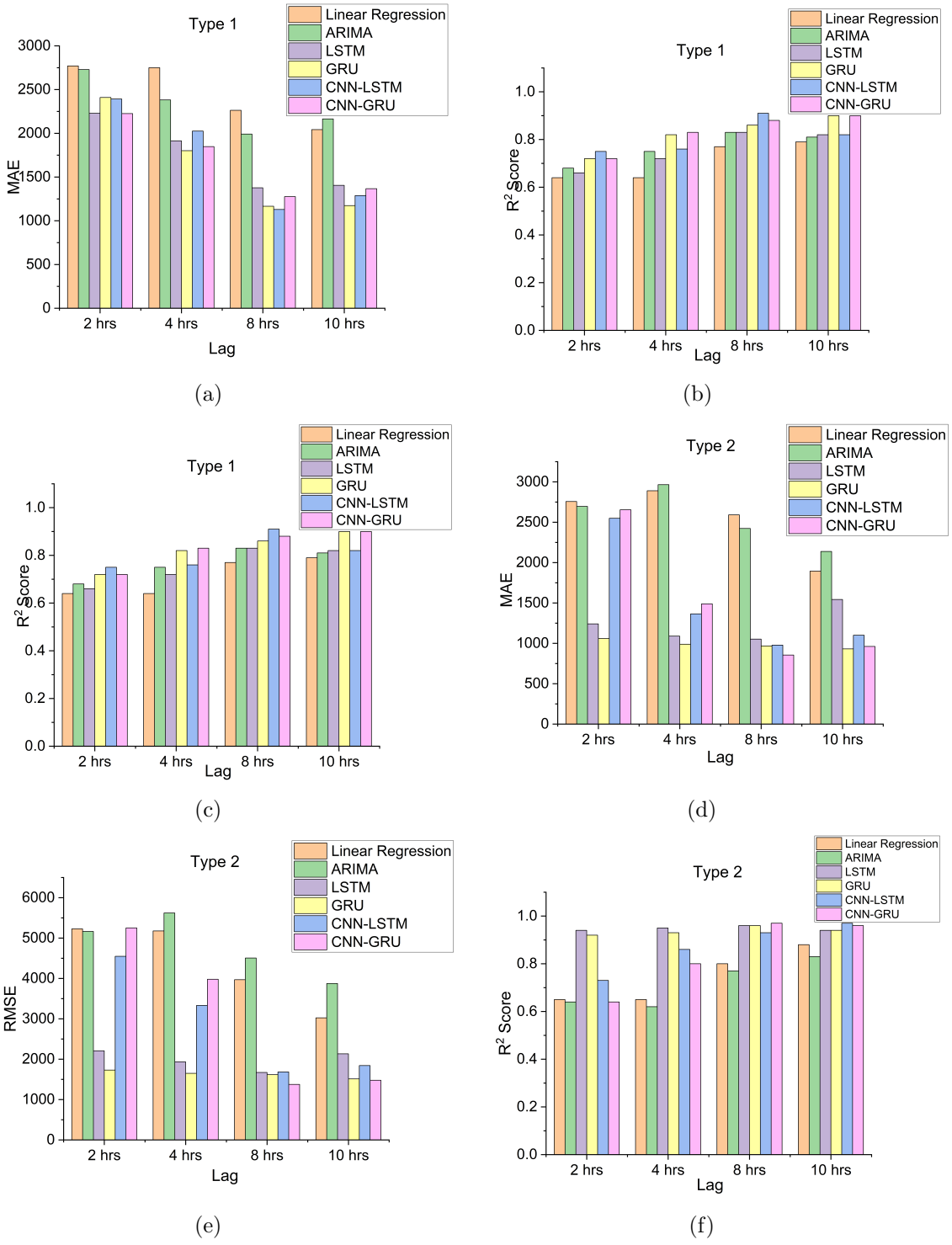
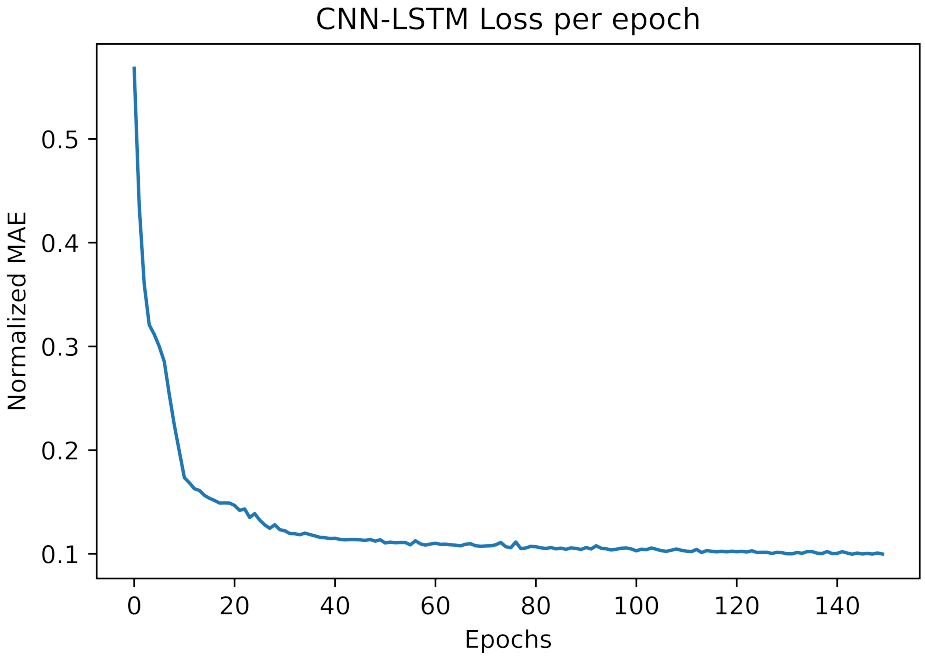
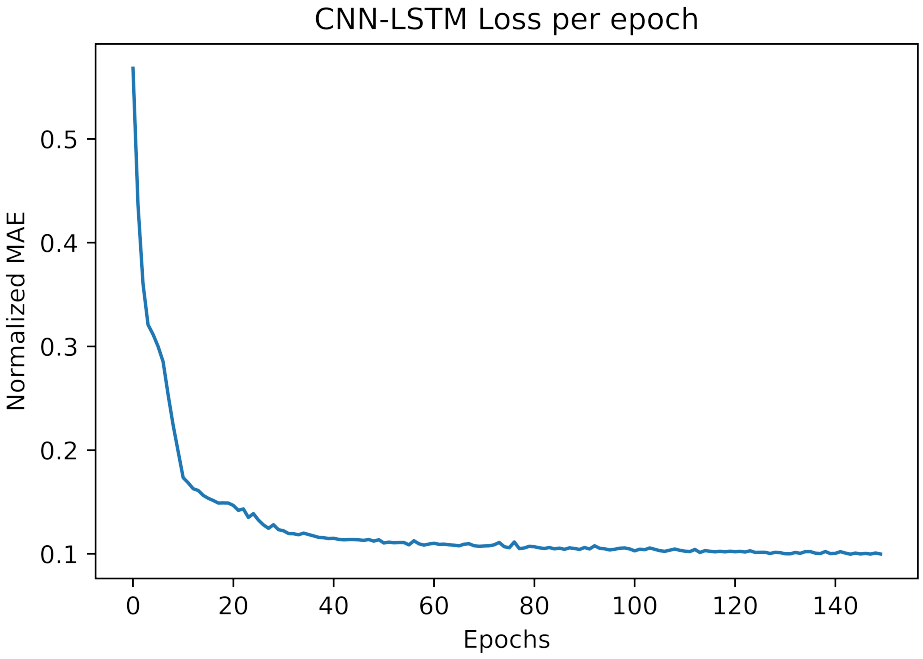


Figure 3.7: (a-c)MAE, RMSE, R^2 of scenario 1 model, (d-f) MAE, RMSE, R^2 of scenario 2 model.



(a)



(b)

Figure 3.8: Loss per epoch values for (a) CNN-LSTM model (b) CNN-GRU model.

Figure 3.8 shows that the proposed CNN-LSTM and CNN-GRU models have low errors and good accuracy. The hours are shown on the y-axis as an independent feature, while the load in kW is shown on the x-axis as a dependent feature. The predicted values are found to be very close to the actual ones. It shows that both CNN-LSTM and CNN-GRU give good accuracy. Figure 3.8(a) depicts the loss per epoch values plot of the CNN-LSTM model for scenario 1 for the lag in hours, where loss is in terms of normalized MAE values. Figure 3.8(b) depicts the loss per epoch plot for the CNN-GRU scenario 2 multivariate model with the lag in hours, where loss is expressed as normalized MAE. Figure 3.8(a) and Figure 3.8(b) indicate that the predicted value of energy consumption is similar to the actual value, even when there is a sudden surge or drop in consumption. This demonstrates that the proposed model outperforms the other models.

3.5 Conclusion

This study showed a combination of various methodologies for time-series energy forecasting and observations from the results by varying the types of input data. It is discovered that the best-case scenario for energy prediction involves using the current day/hour as a feature in multivariate modeling. While testing the models for various lags, it is found that the previous 8 hours' energy usage shows the best value for the 9^{th} hour. When the lag is measured in hours, CNN-LSTM and CNN-GRU models perform better in scenarios 1 and 2 for univariate and multivariate models, respectively. Building occupants have an impact on energy usage. Demand Flexibility Control (DFC) and Model Predictive Control (MPC) significantly increase building energy efficiency when accurately predicted occupant counts. Therefore, finding short-term energy forecasting in the presence of occupancy is necessary. The next chapter proposes occupancy-based energy forecasting for smart buildings.